UNIVERSITY OF CALABRIA

Abstract

Department of Computer Science, Modeling, Electronics and Systems Engineering (DIMES)

Doctor of Philosophy

Big Data Analysis: Methodologies, Frameworks and Real-World Applications

by Francesco Branda

In the last years, the capacity to produce and collect data has increased exponentially. The huge amount of data generated, commonly referred to as Big Data, the speed at which it is produced, and its heterogeneity in terms of format represent a challenge to current storage, processing, and analysis capabilities. This scenario requires the design and implementation of new architectures and analytical platform solutions that must process Big Data to extract complex predictive and descriptive models. Today, high-performance computing (HPC) infrastructures such as highly parallel clusters, supercomputers, and clouds can be used for processing and analyzing massive sources of real-world data in various fields, including genomic sequencing and medical research, fraud detection, and weather forecasting. Following these preliminary observations, the goal of this thesis is twofold. First, the main challenges to be solved for implementing innovative data analysis applications on HPC systems are investigated. In particular, the main key research topics addressed include: (i) studies of software systems for Big Data storing, processing, and analysis; (ii) methods, techniques, and prototypes designed and used to implement Big Data solutions on massive data sources requiring the use of high-performance computing systems; and (iii) design and programming issues for Big Data analysis in Exascale systems, which will represent the next computing step. Second, several innovative applications and use cases of Big Data analytics that can be implemented in large-scale parallel systems are proposed. These research contributions provide new insights and solutions for extracting useful knowledge from large volumes of data, describing methods and mechanisms to support users, practitioners, and scientists working in the area of Big Data in the design and execution of data analysis techniques in different application domains.

"*You can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future. You have to trust in something - your gut, destiny, life, karma, whatever. This approach has never let me down, and it has made all the difference in my life.*"

S. Jobs

Contents

List of Figures

List of Tables

List of Publications

1. Branda F, Marozzo F, Talia D. Discovering Travelers' Purchasing Behavior from Public Transport Data. InInternational Conference on Machine Learning, Optimization, and Data Science 2020 Jul 19 (pp. 725-736). Springer, Cham.
2. Branda F, Marozzo F, Talia D. Ticket sales prediction and dynamic pricing strategies in public transport. Big Data and Cognitive Computing. 2020 Nov 27;4(4):36.
3. Belcastro L, Marozzo F, Talia D, Trunfio P, Branda F, Palpanas T, Imran M. Using social media for sub-event detection during disasters. Journal of big data. 2021 Dec;8(1):1-22.
4. Belcastro L, Branda F, Cantini R, Marozzo F, Talia D, Trunfio P. Analyzing voter behavior on social media during the 2020 US presidential election campaign. Social Network Analysis and Mining. 2022 Dec;12(1):1-6.
5. Branda F. Impact of the additional/booster dose of COVID-19 vaccine against severe disease during the epidemic phase characterized by the predominance of the Omicron variant in Italy, December 2021-May 2022. medRxiv.
6. Branda F, Abenavoli L, Pierini M, Mazzoli S. Predicting the Spread of SARS-CoV-2 in Italian Regions: The Calabria Case Study, February 2020–March 2022. Diseases. 2022 Jun 30;10(3):38.
7. Branda F, Tosi D. Comparing Worldwide, National, and Independent Notifications about Adverse Drug Reactions Due to COVID-19 Vaccines. Information. 2022 Jul 8;13(7):329.
8. Branda F, Pierini M, Mazzoli S. Hepatitis of unknown origin in children: why and how to create an open access database. Journal of Clinical Virology Plus. 2022 Jul 30:100102.
9. Branda F, Pierini M, Mazzoli S. Monkeypox: EpiMPX surveillance system and open-data with a special focus on European and Italian epidemic. Journal of clinical virology plus. 2022 Oct 4:100114.
10. Branda F, Pierini M, Mazzoli S. Monkeypox: Early Estimation of Basic Reproduction Number R0 in Europe. J Med Virol. 2022 Nov 1. doi: 10.1002/jmv.28270.
11. Branda F, Maruotti A. 2022 Uganda Ebola outbreak: early descriptions and open data. Journal of Medical Virology. 2022 Nov 24.
12. Branda F, Mahal A, Maruotti A, Pierini M and Mazzoli S (2023), The challenges of open data for future epidemic preparedness: The experience of the 2022 Ebolavirus outbreak in Uganda. Front. Pharmacol. 14:1101894. doi: 10.3389/fphar.2023.1101894
13. Branda F, Lodi G. Challenges and Perspectives of Open Data in Modelling Infectious Diseases. Data. 2023 Feb;8(2):27.
14. Piconi S, Pontiggia S, Franzetti M, Branda F, Tosi D. Statistical models to predict clinical outcomes with anakinra vs. tocilizumab treatments for severe pneumonia in COVID19 patients. European Journal of Internal Medicine.
15. Branda F, Scarpa F, Ciccozzi M, Maruotti A. Is a new COVID-19 wave coming from China due to an unknown variant of concern? Keep calm and look at the data. J Med Virol. 2023 Feb 23. doi: 10.1002/jmv.28601. Epub ahead of print. PMID: 36815498.

Technical and scientific reports

16. Paradigms and Models at Run-time – Final Report. Deliverable D2.6, ASPIDE Project: Exascale programing models for extreme data processing No 801091.
17. Integrazione e standardizzazione dati, modalità di storage, protocolli di comunicazione e interfacce fra i moduli e con SMARTBUS. Deliverable WP1. Moving+: Dati in movimento (MIUR rif.: J28C17000300006).
18. Analisi di dati, trajectory mining e previsione della domanda. Deliverable WP3.D1. Moving+: Dati in movimento (MIUR rif.: J28C17000300006).
19. Rapporto sul mercato delle Autolinee Federico S.p.A. Deliverable WP3.D1. Moving+: Dati in movimento (MIUR rif.: J28C17000300006).
20. Il dimostratore – funzionalità del prototipo. Deliverable WP3.D3. Moving+: Dati in movimento (MIUR rif.: J28C17000300006).

Chapter 1

Introduction

The World Wide Web, commonly referred to as the Web, is one of the greatest inventions of the 20th century. It revolutionized the way we access and consume information, from looking up encyclopedic information, to sharing videos and photos with our friends, to online shopping. Today, the ability to communicate instantly, to establish relationships, and to sell and buy has increased exponentially, and playing an increasingly predominant role are search engines, social media, and applications that work precisely because of the Web. Its intensive use leads to the generation of a huge amount of data, commonly called Big Data, can be exploited to extract useful information and produce valuable knowledge for economy [1, 2], health [3], science [4], and society [5]. Nowadays, scientific progress and the development of new technologies promote the availability of an unprecedented amount of data in many application areas. Computational science is one of the most popular. The Large Synoptic Survey Telescope (LSST)[1] generates several petabytes of new images and catalog data each year. The Square Kilometer Array (SKA)[2] produces about 200 Gbytes of raw data per second that require petaflops (or perhaps exaflops) of processing to produce detailed radio maps of the sky. Bioinformatics applications mine databases that can hold terabytes of data. Earthquake simulators process huge amounts of data, produced as a result of recording earth vibrations around the world. In addition to scientific computing, several sectors of the information technology industry require support for data-intensive computations. A telecommunications company's customer data easily hovers around 10,100 terabytes. This amount of information is not only processed to generate statements, but is also mined to identify scenarios, trends, and patterns that help these companies provide better service. The petabyte scale is even more common when we consider IT giants such as Google, which processes about 24 petabytes of information per day and sorts petabytes of data in a matter of hours.

With the increasingly widespread availability of Big Data, new discoveries and more accurate investigations will be made. However, to take advantage of these large volumes of data, users and researchers need advanced data analysis models and programming tools, coupled with scalable, reliable and high-performance architectures. Current Cloud Computing platforms and parallel computing systems represent two different technological solutions to meet the computational and data storage needs of Big Data mining and parallel knowledge discovery applications. The former implements a computing model in which virtualized and dynamically scalable resources are provided to users and developers as a service over the Internet, while the latter offer high-end processors with the main goal of supporting high performance computing (HPC) applications. No other single technology in the history of humanity has experienced a similar rate of growth such as HPC systems, even in its relatively short

[1] www.lsst.org
[2] www.skatelescope.org

existence. From less than a 1000 basic operations per second in the late 1940s to today's performance in excess of a 100 quadrillion floating-point operations per second (over 100 petaflops), HPC speed has steadily improved by about a factor of 200 times every decade through a series of advances in technology, architecture, programming methods, algorithms, and system software [6].

Now that data sources became very big and pervasive, these computing architectures are needed to run data analysis because complex data mining tasks involve data- and compute-intensive algorithms that require large, reliable and effective storage facilities together with high performance processors to get results in appropriate times. Data-intensive applications face two major challenges: (i) managing and processing exponentially growing data volumes, often arriving in time-sensitive streams from arrays of sensors and instruments, or as the outputs from simulations; and (ii) significantly reducing data analysis cycles so that researchers can make timely decisions. Figure 1.1 shows a simple diagram that can be used to classify the application space between data- and compute-intensive problems. Purely data-intensive applications process datasets ranging in size from multiterabytes to petabytes. These data are often available in multiple formats and are often distributed across multiple locations. Processing of these datasets typically occurs in multistep analytic pipelines that include transformation and fusion steps. Processing requirements usually scale almost linearly with data size and are often amenable to simple parallelization. Key research topics include data management, filtering and fusion techniques, and efficient querying and distribution. Data/compute-intensive problems, on the other hand, combine the need to process very large data sets with increased computational complexity. Processing requirements typically scale superlinearly with data size and require complex searches and complex merges to produce key insights from the data. Application requirements can also place time constraints on the production of useful results. Key research topics include new algorithms, signature generation, and specialized processing platforms such as hardware accelerators.

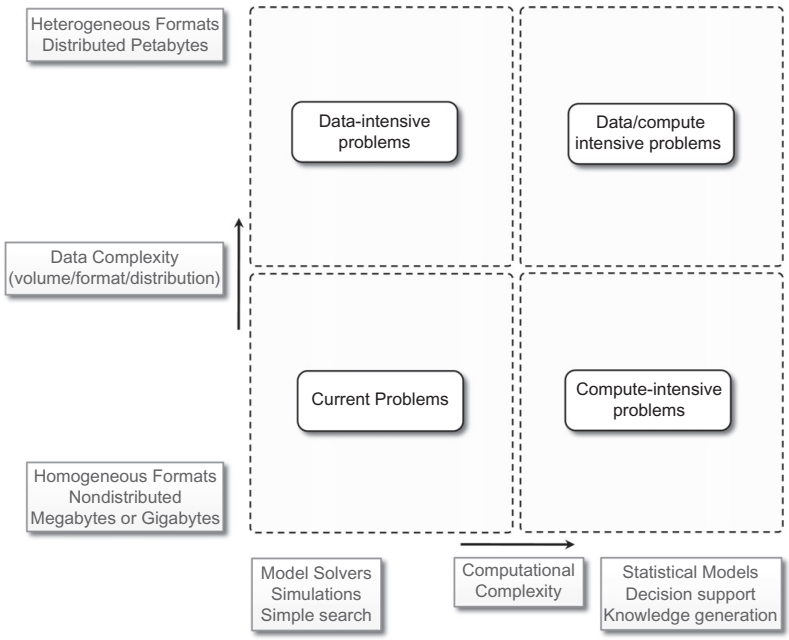


FIGURE 1.1: Data-intensive research issues [7].

Open challenges in data-intensive computing given by Ian Gorton et al. [7] are:

- scalable algorithms that can search and process massive datasets;
- new metadata management technologies that can scale to handle complex, heterogeneous, and distributed data sources;
- advances in high-performance computing platforms aimed at providing a better support for accessing in-memory multiterabyte data structures;
- new approaches to software mobility for delivering algorithms that are able to move the computation to where the data are located; and
- specialized hybrid interconnect architectures to process and filter multigigabyte data streams coming from high-speed networks and scientific instruments and simulations.

To reach these goals, models and technologies enabling Cloud Computing systems and HPC architectures must be extended/adapted or completely changed to be reliable and scalable on the very large number of processors/cores that compose extreme scale platforms and for supporting the implementation of clever data analysis algorithms that ought to be scalable and dynamic in resource usage. Starting from this scenario, Exascale computing systems will represent the next computing step [8, 9], and will play the role of an extraordinary platform for addressing both the computational and data storage needs of Big Data analysis applications.

In this thesis we first provide a comprehensive introduction to the field of HPC and an in-depth overview of high-performance Big Data computing and the associated technical issues, approaches, and solutions. Then, we introduce some already existing tools and libraries for supporting extreme data processing from the state-of-the-art. Additionally, we examine the main topics and trends in Big Data analytics, with a focus on the research issues that must be addressed for implementing massively parallel data mining applications on Exascale computing systems. Chapter 3 summarizes the real-world cases to show pratical examples in several domains where data analytics models can provide effectively valuable support (i) to improve urban transportation and citizens' lives; (ii) to exploit a huge amount of user-generated data in social networks to extract valuable information concerning human dynamics and behaviors; and (iii) to develop new surveillance systems for the planning, implementation and evaluation of public health programmes. Finally, Chapter 4 completes the thesis and outlines perspectives for future research opportunities.

Research questions

This thesis examines the need for HPC-based data analytics in scientific research and proposes a systematic way to address it, with a thorough investigation of the specific requirements imposed by particular research problems. The main purpose is to identify efficient and easily applicable solutions to data analysis problems in the different domains covered in this work. To reach this goal, the thesis includes general frameworks that can be used for processing and analyzing data gathered, for instance, from social media for finding different types of information (e.g., user sentiments, topics trends, detection of catastrophic event experiences) or for building complex surveillance systems for detecting disease outbreaks quickly

before they spread, cost lives and become difficult to control. In this way, data scientists and analysts having limited programming skills, especially with regard to parallel programming, can efficiently design and execute data analysis applications dealing with large amounts and different types of data. Specifically, the following set of research questions (RQ) are addressed:

RQ 1. **"How can we exploit modern massively parallel HPC systems in advanced data analysis pipelines?"**

RQ 2. **"Is it possible to address urban transportation challenges with Big Data?"**

RQ 3. **"How can the implementation of artificial intelligence into Big Data analytics help analyze users on social media?"**

RQ 4. **"Is it possible to use advanced analytics solutions to improve epidemic control?"**

An overview of the relation between the research questions and the publications can be found in Table 1.

Paper / *RQ*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RQ 1																X
RQ 2	X	X														
RQ 3			X	X												
RQ 4					X	X	X	X	X	X	X	X	X	X	X	

Chapter 2

Background

The first use of the term "Big Data" in the literature dates back to 1997, when Michael Cox and David Ellsworth, in their paper "Application-controlled demand paging for out-of-core visualization," used it to refer to the phenomenon of datasets that do not find space in local memory or disk, especially in the context of visualization [10]. However, Big Data has attracted worldwide attention thanks to two key developments, namely Google File System in 2003 [11] and MapReduce in 2004 [12]. The implications of sifting through a multitude of logs, which until then had only been methods for debugging errors in the system, profiling visitors, and targeted advertising, brought data-centric computing into vogue. Web servers, with their logs and browser cookies, allow organizations to collect detailed information about visitors (e.g, device, location, and time spent on Web pages). By processing this information through data mining and analytics, the visitor's age and gender can be inferred, if not already provided through social media or other account information. Over time, the data collected also lead to socioeconomic and demographic inferences. All of this helps services like Google or Facebook perfectly target advertising to their visitors, making their platform the perfect channel for other companies to engage their target market. Today, the extraction of useful knowledge from the big digital datasets requires smart and scalable analytics algorithms, services, programming tools, and applications. This chapter presents the major concepts and detailed skills that together give a meaningful understanding of HPC, including a formal definition of "high performance computing" that applies to the entire treatment of the field, the end-user application problems that are the purpose of HPC in a wide range of scientific, engineering, social, and security domains, the central concept of performance that is the feature that distinguishes HPC from other forms of computing, the hardware and software components that make up an HPC system, the environments, tools, application programming, and interfaces used. Each of these aspects is presented in detail in the following sections and together they form an important part of the concepts, knowledge content and skills that make up this thesis.

2.1 Big Compute Versus Big Data

Simply processing large datasets is generally not considered a Big Data application. Groups such as the Conseil Européen pour la Recherche Nucléaire (CERN) and Transnational Research In Oncology (TRIO) have used high performance computing systems and scalable software to analyze very large datasets. However, this is considered computationally centered processing. Table 2.1 illustrates the main differences between big compute and Big Data systems.

Computers, large and small, are very efficient and fast in processing decimal numbers and mathematical equations. Although not always true, typically compute-centric processing takes small amounts of data as input and generates large amounts

of data during processing or as output. As the thermal and power limits of silicon have approached, multiple processors have been clustered to parallelize and speed up processing. Compute-centric applications typically parallelize processing by distributing data and instructions to multiple processors, which may be nearby or located on the same chip, up to being geographically remote and accessible only via the Internet.

Taking traditional high performance computing as an example, the processing elements (PEs) are typically located in a tightly coupled local area network operating at network speeds between 1 and 100 Gbps. The main bottleneck of a parallel program focused on computation on an HPC system is the network. Complex programs are unable to scale to larger systems because of the large global communication operations. Therefore, input data cannot be distributed in the same way as the instruction set, from one system to the other parallel machine. A fast, shared storage device is used to make the data available to all processing cores. Since all data is grouped together, even if the data domain is partitioned within the algorithm, each PE must still traverse the entire dataset to find the required subset. This results, first, in wasted computation cycles and, second, in an unsustainable load on the storage device (e.g., 50-100 PEs synchronously accessing a storage server to find 1 GB of data to process). When designing a typical HPC system, the key metrics in order of importance are processor clock speed, processor density, network interconnection, processing power and network capacity of centralized storage, and finally memory per PE.

Component	Big compute systems	Big Data systems
Data network	Centralized shared storage High bandwidth and low latency required for scalability	Local storage managed centrally Latency plays minimal role, most bandwidth required during data loading and unloading phases
System Design	Tightly coupled systems	Run on almost anything approach – PEs can be heterogeneous with consumer grade networks
Task Management	Task concurrency required. System can be divided among jobs but each job must run as a single unit	Task and data replication allows for out of order execution of tasks and jobs can also be split
Job Resilience	If a PE involved in a job fails, the job fails and needs to be restarted	Replication of tasks allows for the failure of PE without affecting the job
System Resilience	Systems can cope with loss of PE. Single points of failure: Centralized Storage, Controller Node (redundancy possible), Internal Networking	Built-in redundancy in nearly all components. Default operation includes replication. Single Point of Failure: Internal Networking
Programming Model	Message Passing Interface (MPI), Symmetric multiprocessing (SMP), Parallel Virtual Machine (PVM)	MapReduce, Bulk Synchronous Parallel (BSP)

Table 2.1: Big compute systems vs Big Data systems

The main distinction of the Big Data computing paradigm is that algorithms and processing systems are usually designed with data-centricity in mind. Data-centric systems spend most of their execution time on data manipulation and I/O as opposed to numerical operations. Algorithm optimization is secondary to data handling. Within the MapReduce ecosystem (discussed below), data locality is an important component of the data-centric concept. Unlike the HPC systems mentioned above, data is not stored in a single location. In fact, upon ingestion, it is divided and replicated in a distributed computing system and stored locally to the processing elements. In this way, each PE only has to deal with its local subset of data, without having to search for it. In addition, PEs do not have to compete with each other for network bandwidth. This creative distribution of data ensures that processing instructions reach only those PEs that have direct access to the relevant

data. Replication of subsets of data ensures further and better parallelization. If two instructions require access to the same subset of data, replication ensures that two separate PEs can execute both instructions simultaneously without any contention for network or disk. When designing a Big Data system, the metrics to keep in mind are PE-level storage size, PE-level storage speed, memory per PE, processor clock speed, and network interconnect. This is another reason for the penetration of Big Data in the commercial sector. HPC systems have high power consumption, resulting in high process density, and require highly specialized network components to ensure performance. In contrast, Big Data systems can be very efficient using high-quality workstations loosely coupled with commonly used network equipment. This makes Big Data systems easily deployable on Cloud Computing infrastructures, reducing the financial barrier to entry.

2.2 Toward High Performance Computing and Big Data Analytics Convergence

The convergence of high performance computing and Big Data analytics (BDA) is now an established research area that has spawned new research topics such as data-intensive scientific computing, high-performance data analytics, and hybrid platforms and infrastructures based on virtualization techniques and new storage hierarchies. HPC-BDA convergence became a hot topic when applications and their associated data evolved outside their original ecosystems. At the time, the issue for HPC was, "How can we cope with the increase in datasets?" while BDA was asking, "How can we run analytics faster?" Since then, the HPC [13] and BDA [14] communities have recognized new opportunities in unifying the platform layer and data abstractions for both HPC and BDA [15].

The divergence between the HPC and BDA software ecosystems emerged at the beginning of this century, when software infrastructure and data analysis tools developed by online service providers became available and adopted by various scientific communities to solve their own data analysis challenges [16]. The main technical differences between HPC and BDA ecosystems include software development paradigms and tools, virtualization and scheduling strategies, storage and network models, resource allocation policies, and redundancy and fault-tolerance strategies. Many issues remain unresolved, and the situation has been exacerbated by the emergence of new application domains that are completely hybrid in nature, such as autonomous vehicles, surveillance, e-science with Big Data sources, large-scale infrastructure monitoring, and smart cities.

These domains have in common the need to support simulation of complex models, assimilating voluminous and variable data in real time in order to generate refined models for better understanding of the domain, to prescribe model-based control actions or to predict future behavior. Under these circumstances, borrowing features from the other paradigm proves insufficient and deeper convergence becomes necessary to cope with mixed requirements, new infrastructure, and impending performance expectations. The main technical requirements involved in this process include highly scalable performance, high memory bandwidth, low power consumption, and excellent short arithmetic performance. As a result, BDA and HPC platforms remain largely incompatible today.

2.2.1 Big Data Analytics Ecosystem

Big Data affects many different ecosystems and sectors of research and business; therefore, it has no single definition and its scope is still a controversial topic in these communities. From a data analysis perspective, the multi-V model reflects one way to define Big Data by describing different characteristics, and it continues to evolve over time by adding more attributes as needed [17]. The main features included in this model are as follows:

- *Volume of data.* Volume is necessary in order to get valuable insight from analytics tools. It is usual to find volumes in the order of peta or terabytes at the enterprise level. These volumes can also be quantified in the order of billions of records, tables, files or transactions depending on the platform used by the system. In order to handle these volumes, Big Data systems and applications must be designed to maximize throughput and resilience.

- *Velocity of data production and processing.* Data can be produced and consumed at different rates. Big Data systems can even incorporate different and processing speeds, including batch processing, streaming, near and real-time speed.

- *Variety of data types.* Nowadays a data source can be anything: sensors, third parties, Web applications, etc. As a result, data can be highly heterogeneous and unstructured, and platforms must be able to understand and integrate these different data to aggregate knowledge from several sources. Moreover, the types of data depend greatly on the application and its domain: we find structured statistical data in business intelligence, and time series and geospatial data in the Internet-of-Things (IoT).

- *Veracity of data.* The volume of data is key to obtain knowledge, but the derived information would be flawed if the quality of data is low. High-quality Big Data must be reliable in terms of trust and integrity.

- *Value in business terms.* The model or analysis that results from processing Big Data must provide enterprise value to make up for the investment expenses necessary to collect and analyse data.

These definitions have one key aspect in common: Big Data focuses on data perceived as large in volume. This paradigm shift has affected all areas of computing, from data transfer and storage to data analysis and visualization. This shift has been reflected in traditional areas of business and science, such as genomics, climatology, finance, and business intelligence, which have been able to gain better insights with existing methods, but have also fostered new areas of research to leverage the inherent value of data and demonstrate the system's ability to cope with data processing and storage requirements. Areas such as IoT and BDA have developed significantly due to advances in Big Data.

Big Data analytics is one of the best examples of how Big Data has disrupted an established area such as business intelligence, leveraging advanced analytical techniques that operate on Big Data to evolve from descriptive tools to predictive and prescriptive models. Today, enterprises are exploring Big Data to incorporate knowledge discovery into their operations to identify interrelationships among seemingly unrelated attributes of datasets. Enterprises can now understand the current state of business and customer behavior through complex techniques such as predictive analytics, data mining, statistical analysis, data visualization, artificial intelligence and natural language processing, coupled with supporting platforms such as Map-Reduce,

in-database analytics, in-memory databases and columnar data stores. Some of these techniques have existed for years and have been revamped for their good adaptability to very large datasets with minimal data preparation. In addition, infrastructures such as Cloud Computing offer opportunities to reduce the economic costs of implementing BDA and to build analytical workflows at different levels of abstraction.

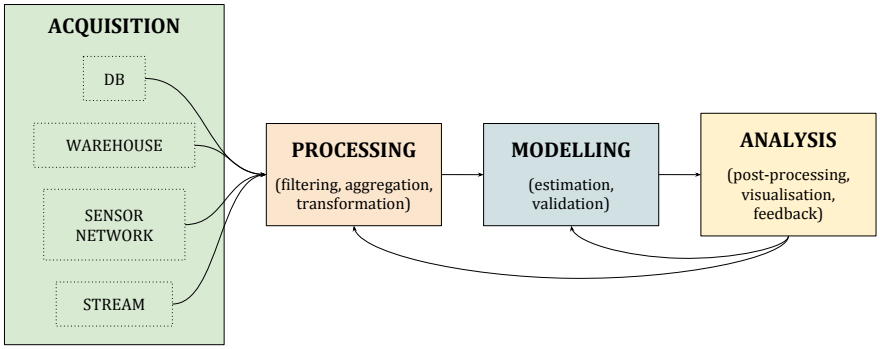


FIGURE 2.1: Typical workflow of a BDA application. Source image: https://www.aspide-project.eu/wp-content/uploads/sites/62/2019/07/D2-2.pdf

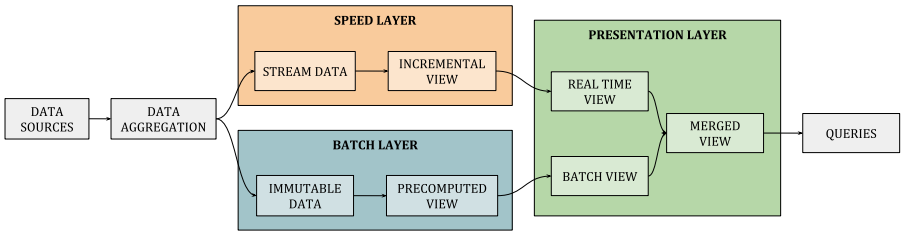


FIGURE 2.2: Representation of the lambda architecture. Source image: https://www.aspide-project.eu/wp-content/uploads/sites/62/2019/07/D2-2.pdf

Figure 2.1 represents the traditional knowledge discovery workflow for BDA, which includes acquiring data from different sources, processing and combining the data in many ways to build a model that can be used for analysis and visualization, and incorporating feed-back mechanisms to refine the data processing and modeling steps. This workflow has usually been combined with the lambda architecture [18] to provide scalable integration and interoperability between different datasets through real-time analysis. This architecture was proposed with the goal of providing a generic platform to serve different applications with different latency requirements in a streaming environment. As illustrated in Figure 2.2, the lambda architecture includes a velocity layer for processing pure streams in real time, a batch layer for storing raw data and processing higher-quality views of long-term data, and a presentation layer that handles queries and visualization of results.

2.2.2 Programming models

This section discusses the main requirements of programming systems for Big Data and describes the four classification criteria used to classify them [19]. To cope with the need to process large amounts of data, a programming system must meet the following requirements:

- *Efficient data management and exchange.* Data management is the practice of collecting, keeping, and using data securely, efficiently, and cost-effectively. In this context, programming systems must support efficient protocols for data transfers and for communications as well as they have to enable local computation of data sources and fusion mechanisms to compose the results produced in distributed nodes.

- *Interoperability.* It is a major problem in large-scale applications using resources such as data and compute nodes, as programming systems for Big Data must allow the use of different data formats and tools.

- *Efficient parallel computation.* An effective approach to analyzing large volumes of data and obtaining results in reasonable time relies on exploiting the inherent parallelism of most data mining and analysis algorithms. Therefore, programming systems for Big Data analysis must allow parallel processing of data and provide a way to easily monitor and adjust the degree of parallelism.

- *Scalability.* With the exponential increase in the volume of data to be processed, programming systems for Big Data analytics must adapt to the rapid changes in data growth, both in terms of traffic and volume, efficiently taking advantage of the increase in computing or storage resources.

Based on these requirements, we can classify the programming systems for Big Data analysis according to the following four criteria:

1. **Level of abstraction**. It refers to its programming capabilities to hide the low-level details of a solution (e.g., a function, a data structure, a communication protocol). In this way, developers can focus on the logic of the problem without having to implement it from scratch. A high level of abstraction makes it easy to create applications, but difficult to compile them into efficient code. While a low level of abstraction makes it difficult to build applications, but easy to implement them efficiently [20]. We use these requirements as a parameter to classify and evaluate programming systems for Big Data analysis. For comparison purposes, we distinguish three levels of abstraction:

   – *Low*, when the programmer can take advantage of low-level APIs, mechanisms, and instructions that are powerful but not trivial to use. More development effort is required than for systems that provide a higher level of abstraction, but code efficiency is very high because it can be fully tuned. It also requires a low-level understanding of the system, including working with files on distributed environments [21]. At this level, programmer productivity is low, while program performance can be effective.

   – *Medium*, when a programmer defines an application as a script or visual representation of program code, hiding low-level details that are not critical to the design of the application. It requires average development effort and code tuning skills.

   – *High*, when developers, even with low programming skills, can build applications using high-level interfaces, such as visual IDEs or abstract models with high-level constructs unrelated to the running architecture. At this level, program development effort is low, as is run-time code efficiency, because executable generation is more difficult and code mapping is not direct [22].

2. **Type of parallelism**. The type of parallelism describes how a programming model or system expresses parallel operations and how its runtime supports the execution of concurrent operations on multiple nodes or processors. For comparison purposes, we distinguish three types of parallelism:
   - *Data parallelism*: it is achieved when the same code is executed in parallel on different data elements. Single Instruction Multiple Data (SIMD) architecture is the architecture where data parallelism can be implemented.
   - *Task parallelism*: it is the mode of parallelism in which tasks are divided among processors to be processed simultaneously. Such parallelism can be defined in two ways: (i) *explicit*, i.e., the programmer defines the dependencies between tasks through explicit instructions; (ii) *implicit*, i.e., the system analyzes the input/output of tasks to understand the dependencies between them.
   - *Pipeline parallelism*: it is achieved when data are processed in parallel in different stages, so that the (partial) output of one task is passed to the next task to be processed. Because of its characteristics, pipeline parallelism can be considered a specialization of both data parallelism and task parallelism [23].
3. **Infrastructure scale**. This feature refers to the ability of a scheduling system to efficiently run applications on an infrastructure of a given size (i.e., the number of compute nodes). Some systems are designed to be used on infrastructures with a small number of nodes, while others scale to a large number of nodes. For comparison purposes, we distinguish three scales of infrastructure:
   - *Small*, i.e., a small enterprise cluster or cloud platforms with up to hundreds of computational nodes.
   - *Medium*, i.e., a medium enterprise cluster consisting of up to thousands of nodes.
   - *Large*, i.e., a large HPC environments or high-level cloud services with up to ten thousands of nodes.
4. **Classes of applications**. Choosing the right programming solution for the development of a Big Data analytics application is not easy, as several different systems, libraries, and programming languages are available today. Some solutions can be used efficiently in a specific field (e.g., streaming data processing, data querying, machine learning), while others are more general and can be used for different classes of applications.

The following most popular programming models for Big Data analytics and their languages and libraries are outlined and discussed. The goal is to highlight the features, issues, and benefits of each programming model and the systems that implement it according to the four criteria defined in the previous section.

2.2.2.1 MapReduce

MapReduce is a simple programming model for performing distributed computations, including data processing on very large input sets, in a highly scalable and fault-tolerant manner. Although the concept of MapReduce was initially motivated by functional programming languages such as *LISP* with its map and reduce primitives, it is also closely related to the message-passing interface (MPI) concepts of scatter and

reduce for distributed memory architectures. However, unlike MPI programming, the details of the parallelization underlying MapReduce are hidden from the programmer, making it easier to use. MapReduce algorithms have been shown to scale from single servers up to hundreds of thousands of cores, while offering transparent fault tolerance to the end user.

MapReduce was developed by Google [24], and the programming model has since been adopted by many software frameworks, libraries, and end users. Apache's open-source Hadoop framework is one of several libraries which support MapReduce, and is used for the examples in this section.

In general, the whole transformation process performed in a MapReduce application can be described through the following steps (see Figure 2.3):

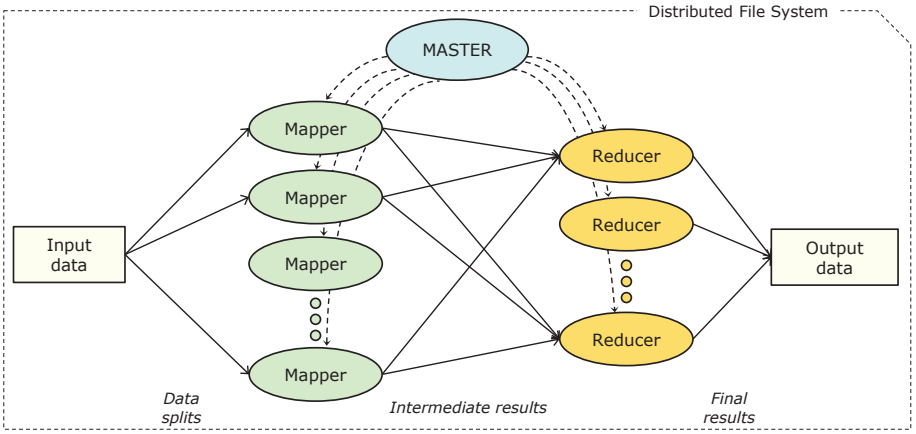


FIGURE 2.3: MapReduce execution flow [19].

1. A master process receives a job descriptor that specifies the MapReduce job to be executed. The job descriptor contains, among other information, the location of the input data, which can be accessed using a distributed file system.

2. Based on the job descriptor, the master starts a series of mapper and reducer processes on different machines. At the same time, it starts a process that reads input data from its location, splits it into a series of splits, and distributes it to the various mappers.

3. After receiving its data partition, each mapping process runs the map function (provided as part of the job descriptor) to generate a list of intermediate key/value pairs. These pairs are then grouped according to their keys.

4. All pairs with the same keys are assigned to the same reduction process. Then, each reducer process performs the reduction function (defined by the job descriptor) that combines all values associated with the same key to generate a possibly smaller set of values.

5. The results generated by each reduction process are then collected and delivered to a location specified by the job descriptor to form the final output data.

2.2.2.2 Apache Hadoop

Apache Hadoop[1] is an open-source project first developed by researchers at Yahoo. It was designed to replicate the functionality of both Google's File System (GFS) [11] and MapReduce system. Since the code was first made publicly available in 2007, Hadoop has grown to become the de facto standard for Big Data processing, relieving developers from having to deal with classic distributed computing problems such as load balancing, fault tolerance, data locality and network bandwidth savings. An overview of the Hadoop software stack is shown in Figure 2.4. The Hadoop project is not only about the MapReduce programming model (*Hadoop MapReduce* module), as it includes other modules such as:

- *Hadoop Distributed File System* (HDFS): a distributed file system providing fault tolerance with automatic recovery, portability across heterogeneous commodity hardware and operating systems, high-throughput access and data reliability.
- *Hadoop YARN*: a framework for cluster resource management and job scheduling.
- *Hadoop Common*: common utilities that support the other Hadoop modules.

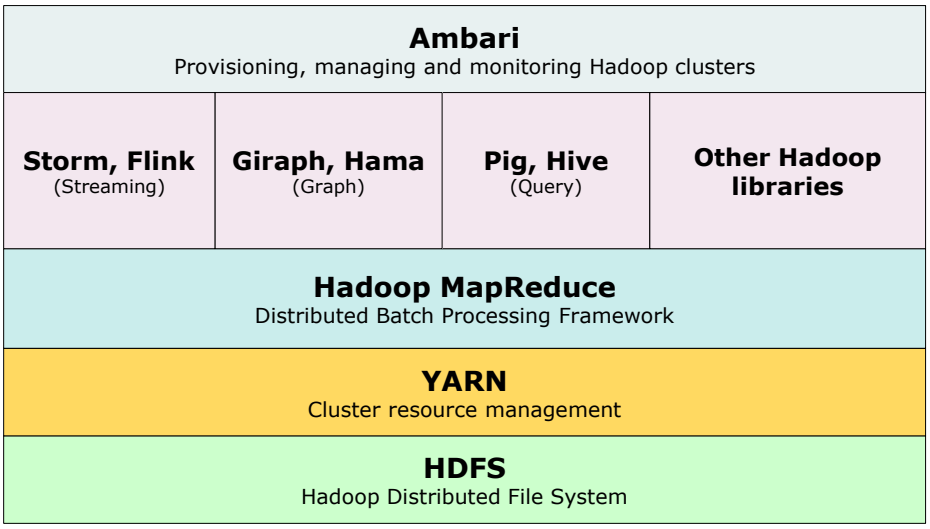


FIGURE 2.4: Hadoop software stack [19].

In particular, thanks to the introduction of YARN in 2013, Hadoop turns from a batch processing solution into a platform for running a large variety of data applications, such as streaming, in-memory, and graphs analysis. As a result, Hadoop has become a reference for several other programming systems, such as: *Storm* and *Flink* for streaming data analysis; *Giraph* and *Hama* for graph analysis; *Pig* and *Hive* for querying large datasets; *Oozie*, for managing Hadoop jobs; *Ambari* for provisioning, managing, and monitoring Apache Hadoop clusters.

Hadoop is designed to run on a distributed shared nothing cluster of commodity machines. The original Hadoop implementation runs as a series of daemon processes, with nodes in the cluster taking one of two roles, master or slave. Each of the four separate daemon processes runs within a Java Virtual Machine (JVM). These daemon processes shown in Figure 2.5 are:

[1] https://hadoop.apache.org/

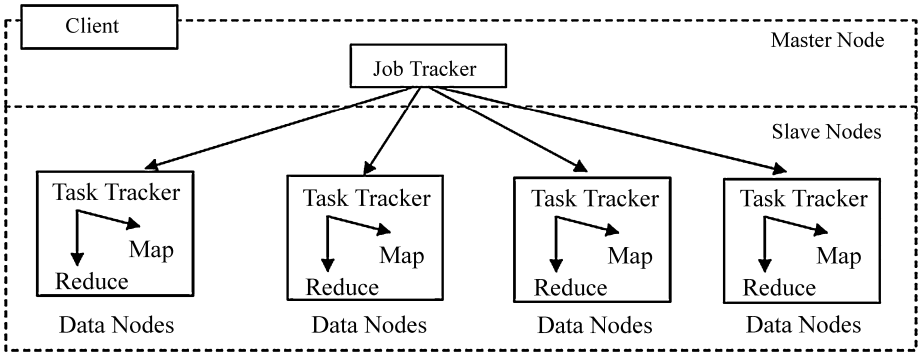


FIGURE 2.5: Interaction between various Hadoop processes [25].

- *JobTracker*. Master process for the MapReduce component that controls the submission and scheduling of jobs on the cluster. Runs on the master node.
- *NameNode*. Master process for the HDFS that keeps track of how data has been distributed across the available DataNode. Runs on the master node.
- *TaskTracker*. These processes are the worker components of the MapReduce system. The TaskTracker demons themselves do not perform the computation; instead they control the spawning of separate JVMs for each MapReduce job. These are run upon the slave nodes.
- *DataNode*. These processes control the data stored in the HDFS. These are also run upon the slave nodes.

Apache Hadoop provides a low-level of abstraction, in that a programmer can define an application using APIs that are powerful but not easy to use because they are tied to the computing infrastructure. It also requires a low-level understanding of the system and execution environment to address issues related to file systems, networked computers, and distributed programming [21]. Developing a Hadoop-based application requires more lines of code and development effort than systems that offer a higher level of abstraction (e.g., Hive or Pig), but the efficiency of the code is greater because it can be fully tuned. Hadoop is designed to take advantage of data parallelism during map/reduce phases. Incoming data is split into chunks and processed in parallel by different compute nodes. It is designed to process large amounts of data in large-scale infrastructures with tens of thousands of machines. In fact, Hadoop is used by most leading IT companies, such as Yahoo!, IBM and Amazon. Hadoop is a generic system that enables large-scale data processing on a set of distributed nodes. It is widely used to develop iterative batch applications using many programming languages, such as Java, C, C++, Ruby, Groovy, Perl and Python.

2.2.2.3 Apache Spark

Spark is a general-purpose parallel computing framework designed for computations of increased complexity to be performed upon massive datasets [26]. The Spark computing framework has quickly gained momentum within the data intensive computing community due to its performance and flexibility. The speed increase is due, in part, to the Resilient Distributed Dataset (RDD) abstraction that allows working data be cached in memory, eliminating the need for costly intermediate stage disk writes [27]. At its core, Apache Spark is a cluster-computing platform that provides an API allowing users to create distributed applications and it was designed to resolve some

of the constraints of the MapReduce programming model, specifically its poor performance when utilizing the same dataset for iterative compute processes due to its lack of an abstraction for distributed memory access [27]. Spark has been optimized for compute tasks that require the reuse of the same working dataset across multiple parallel operations, especially iterative machine learning and data analytic tasks, while maintaining scalability and fault tolerance [28]. Modern data analytic tasks that require the reuse of data are increasingly common in iterative machine learning and graph computation algorithms. Examples of such algorithms include PageRank, used to create a rank of the popularity of web pages and other linked data sources and K-means clustering, used to group common members of a dataset together.

As shown in Figure 2.6, different libraries have been built on top of Spark: *Spark SQL* for dealing with SQL and Data Frames, *MLlib* for machine learning, *GraphX* for graph-parallel computation, *Spark Streaming* for building streaming applications. The execution of a generic Spark application on a cluster is driven by a central coordinator (i.e., the main process of the application), which can connect with different cluster managers, such as *Apache Mesos*, *YARN*, or *Spark Standalone* (i.e., a cluster manager available as part of the Spark distribution). *Ambari* can be used for provisioning, managing, and monitoring Spark clusters.

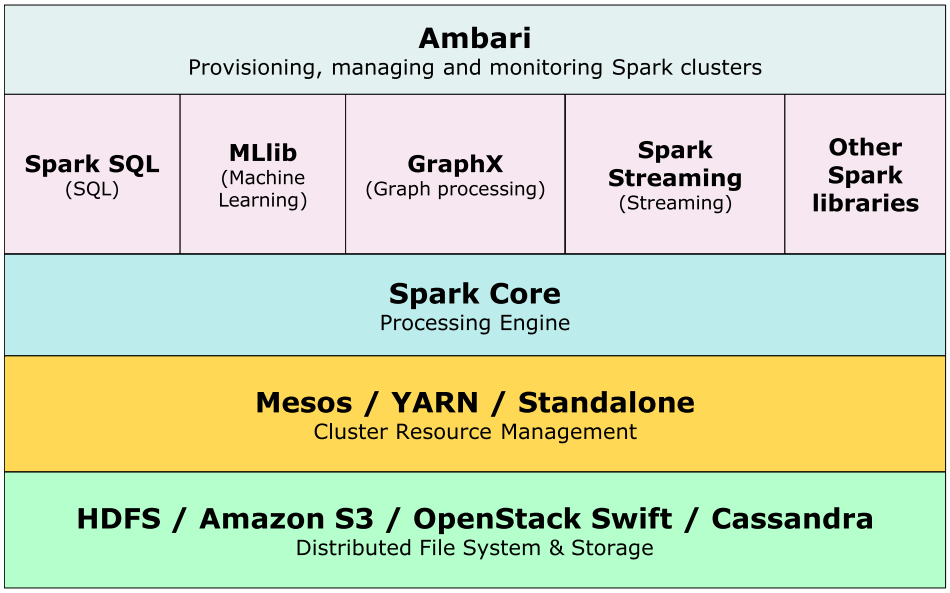


FIGURE 2.6: Spark software stack [19].

From the application developer's perspective, Spark provides a low-level of abstraction allowing for the creation of standalone programs in Java, Scala, R, and Python, but require high programming skills. Interestingly Spark also offers users the ability to utilize an interactive shell that runs a top of the cluster, behaving much like an interactive Python interpreter would. Spark has three key advantages for end users over MapReduce. Firstly, user's programs are less complicated to create and often require fewer lines of code. Secondly, users can create more complicated algorithms owing to the increased functionality of the API. In fact, Spark provides some built-in operators (e.g., *flat*, *flatMap*, *saveAsTextFile*, *reduceByKey*, *groupByKey*) that make easier to code a parallel application. Thirdly, it is designed to process very large amounts of data in large-scale infrastructures with up to tens of thousands of nodes, exploiting data parallelism into stages. Input data is divided in chunks and processed

in parallel by different computing nodes. It supports also task parallelism, when independent stages of the same application are executed in parallel. Due to Spark's speed advantage, user's computation will be completed in less time for the Spark's runtime performance, which has been shown to be superior to that of MapReduce [29], 20 times faster in K-means clustering [29] and 8 times faster in PageRank [30].

From an administrator's perspective, Spark is highly portable and can run on systems ranging from laptops to supercomputers to the cloud. Spark also integrates well with existing Hadoop distributions, as it can be managed through YARN. In addition, Spark can be controlled through Mesos, an alternative cluster management framework developed by the same team as Spark [31]. Spark can also be run in standalone mode, managing its own resources and scheduling. This mode can be configured to run on a single machine for testing and development work. For cloud deployment, each Spark release contains scripts to manage a standalone deployment on Amazon's EC2 cloud. Spark's popularity has resulted in all major cloud providers (Amazon's EC2, Microsoft's Azure, and Google's Compute Engine) offering prepackaged images containing Spark and other data-intensive frameworks.

2.2.3 High Performance Computing Ecosystem

The term High Performance Computing generally refers to the practice of aggregating computing power in an approach that provides much higher performance than could be obtained from a typical desktop computer or workstation in order to solve complex problems in science, engineering, weather forecasting, simulation, or any other field of interest. The action of running an application on a supercomputer is called "supercomputing" and is synonymous with HPC. The term HPC has evolved over the years: what were once large composite mainframes have been replaced by thousands of processors clustered together to harness their processing power. The synergy of large numbers of computing nodes has improved the performance of computing systems, enabling them to do more in less time and at lower cost.

The purpose of HPC is to obtain answers to questions that cannot be adequately addressed on their own through empiricism, theory, or even widely available or accessible commercial computers (e.g., enterprise servers). Historically, supercomputers have been applied to science and engineering, and the methodology has been described as the *third pillar of science*, alongside and complementing experimentation (empiricism) and mathematics (theory). But the range of problems that supercomputers can address goes far beyond classical science and engineering studies, including challenges in socioeconomics, process control, and national security. An application, then, is both the problem to be solved and the body of *code* or collection of ordered computational instructions that represent the means to solve the problem. The code is the means by which the user tells the supercomputer how it should perform the calculations necessary to achieve the goals of the problem. The set of code used is a *computer program* or simply *program*, and the person who develops the application code is the *programmer*.

To take advantage of the scalability and performance of supercomputers, HPC applications rely heavily on parallelism techniques to maximize resource utilization. Supported by advanced execution engines, these applications coordinate parallel processing across many cores and nodes with network-intensive data transfers between compute and storage nodes. In addition, some applications must iterate to refine results, modify the underlying model, or incorporate new data. Figure 2.7 illustrates these relationships, which form the structure of many HPC applications. The basic simulated models are typically initialized with a combination of input data and basic

environmental conditions as parameters, and the simulation domain is distributed so that kernel calculations can be conducted in parallel. Ideally, these simulations are nicely parallel, and calculations can be performed independently while incorporating new partial data. Once the kernels converge, the resulting data are merged with results from the other processing units to update the model, which typically leads to a communication-intensive process that results in the input that will be provided at the next stage. As a measure of fault tolerance, most simulations include checkpoint procedures to store the inter-averaged models and restore the simulation from them in case of failure. The simulation results are then written to memory.

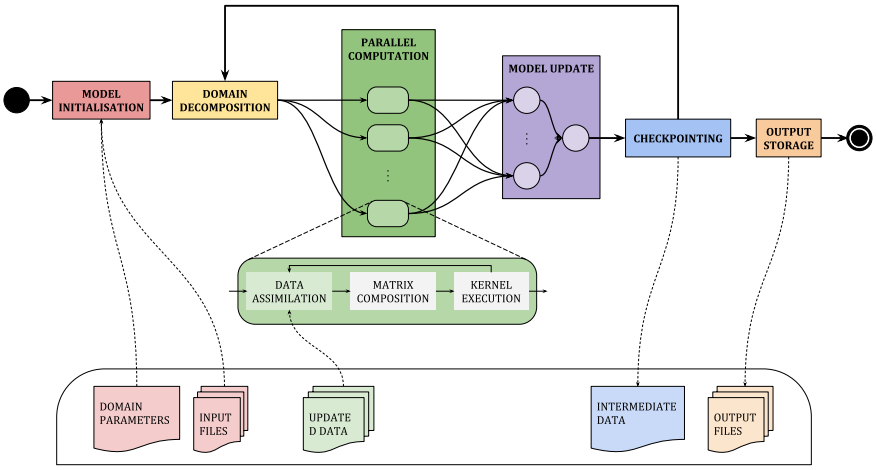


FIGURE 2.7: Application model for the typical HPC scientific application. Source image: https://www.aspide-project.eu/wp-content/uploads/sites/62/2019/07/D2-2.pdf

2.2.3.1 Performance and Metrics

The concept of performance may be intuitive, but it is not simple. There is no single measure of performance that fully reflects all aspects of the quality of computer operation. A metric is an observable and quantifiable operational parameter of a supercomputer. Two fundamental measures used singly or in combination and in different contexts to characterize the behavioral properties and capabilities of an HPC system are the *time* and *number of operations* performed. However, for HPC the most commonly used measure is floating-point operations per second or *flops*. A floating-point operation is an addition or multiplication of two real numbers represented in a machine-readable and manipulatable form. The field adopts the same notation system as science and engineering, using the Greek prefixes *kilo*, *mega*, *giga*, *tera* and *peta* to represent 1000, 1 million, 1 billion, 1 trillion and 1 quadrillion, respectively.

A better measure of flops is the time it takes to complete a given problem, but since there are literally thousands of such problems, this measure is not particularly useful in general. Therefore, the HPC community selects specific problems around which to standardize. Such standardized application programs are the *benchmarks*. A benchmark is a method used to provide a measurement that allows two independent systems to be compared. HPC benchmarks are typically one or more program and defined input that must be run correctly; the measurement is usually either the time or the rate (operations/second). Thus, a second way to measure performance is the completion time of a fixed problem. A particularly popular supercomputer benchmark is *Linpack*, or more precisely *Highly Parallel Linpack* (HPL), which solves a set of linear

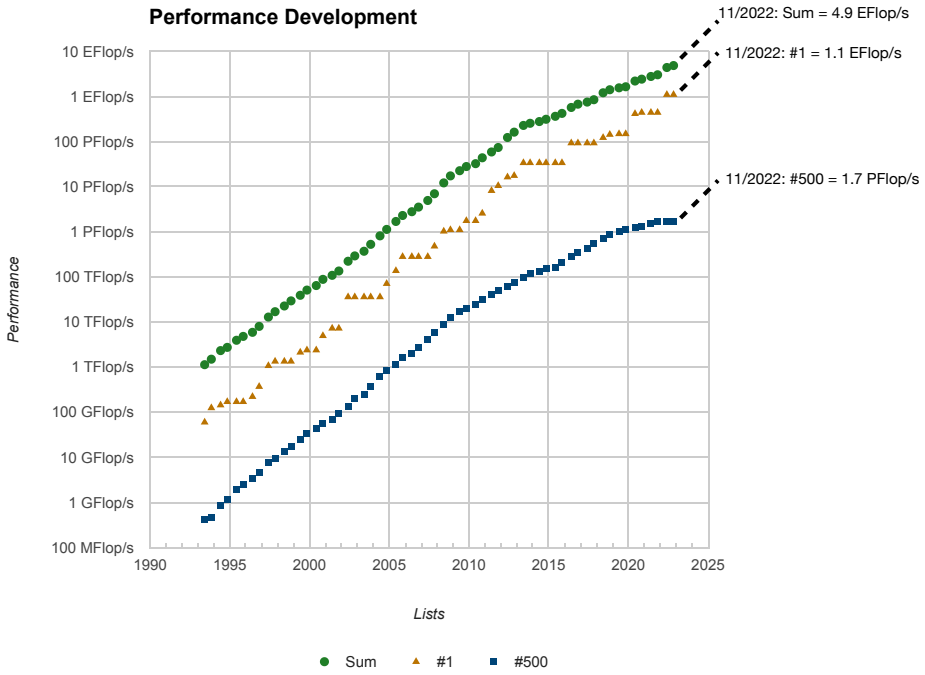


FIGURE 2.8: The evolution of the R_{max} from the HPL benchmark for supercomputing systems in the Top 500 list since the list began in 1993. The top line indicates the cumulative performance of all the computers in the list. The middle line shows the performance of the number one computer in the list. The bottom line shows the performance of the last computer in the list (number 500). Source image: https://www.top500.org/statistics/perfdevel/

equations in the form of a dense matrix [32]. It is used as a benchmark to provide data for the Top500 list[2] and rank the world's supercomputers, publishing every two years a list of the 500 most powerful supercomputers in the world based on R_{max} and R_{peak} scores (Figure 2.8). The R_{max} score of a system describes its maximum performance achieved; the R_{peak} score describes its theoretical peak performance.

The current fastest supercomputer, Frontier [33] located at Oak Ridge National Laboratory (ORNL), has peak performance on the order of 1.102 EFlop/s and has been able to reach 2.5 times the speed of the Fugaku [34] system from the Riken Center for Computational Science (R-CCS) in Kobe, Japan, which is currently in second place, followed by the LUMI system (0.309 EFlop/s) installed at the CSC EuroHPC center in Finland and LEONARDO (0.174 EFlop/s) installed at EuroHPC/CINECA in Bologna, Italy.

2.2.3.2 Anatomy of an HPC system

The most visible aspect of the HPC field is high-performance computers, or simply supercomputers. Today these machines occupy thousands of square meters and consume potentially several megawatts of electricity. Implementing a state-of-the-art supercomputer is a major engineering feat that requires time, expense, and expertise, as well as responsible management and maintenance over the life of the system. However, the visible, audible and sensory experience barely reflects the true nature of the realization embodied by these machines. At the heart of the HPC system is the

[2] www.TOP500.org

structure and organization of the various components and the semantics or rules by which the applications offered to users operate and execute. More even than hardware, the HPC system consists of a wide range of software components that control the hierarchy of physical components and manage user workloads. If the physical hardware, racks and all, is what the visitor experiences, the system software, its interfaces and functionality are what the user experiences (usually in a location away from the physical machine) when developing and running applications and analyzing the results.

The block diagram of the system stack (Figure 2.9) shows the layered hierarchy of the many physical and logical components contributing to a general-purpose supercomputer [6, 35]. The bottom layer, the system hardware, represents the physical resources that are the most visible (and perceptible) aspect of a supercomputer. The processors that perform the calculations and the memory that stores both the data and the program codes that operate on them are shown, as well as the interconnection network that potentially integrates many thousands and eventually millions of these processor/memory nodes into a single supercomputer. Another family of hardware enables long-term storage of data and programs. Hard drives and storage tapes can store user data indefinitely and have much greater capacity than ephemeral main memory, but at the cost of significant access times.

The first layers of software that control the hardware and manage these resources are associated with the operating system, which is much more complex than the two layers shown imply. Each node has a local instance of an operating system that controls the node's physical memory and processor resources, as well as the interface to the system network external (to the node). An additional operating system layer, sometimes called middleware, logically integrates the many nodes and their local operating systems into a single system image to which users can send their application programs and access standard I/O channels. In some supercomputers, a separate front-end software environment running on a dedicated computer, known as a host, provides most of the user interfaces and services other than the scalable processing itself. The layered operating system reflects an abstract or virtual machine for the higher layers of the system, including the user programming interface. It ensures a standardized set of user services on which the application can depend, with common interface protocols independent of the specific system on which it runs. Among these services is the file system, which handles much of the persistent storage and presents a structured organization of the various forms of user data.

Above the layers dedicated to resource management are those associated with the means of development and execution of user applications and workloads. In this case, *workloads* are defined as a set of loosely coupled applications, each performing a different aspect of the set of tasks to be computed. For example, one might have three applications, a simulator, a data analyzer, and a visualization package, with each of these three applications transmitting data to the next function. The general job management layer includes several supporting features, including programming languages (e.g., Fortran, C), additional libraries often for parallelism (e.g., MPI, OpenMP), and compilers that provide machine-readable code for processor cores translated and optimized by user code. Higher-level environments and tools help build more complex workflows and manage them. An important part of these environments are the many sophisticated libraries of previously developed and highly optimized functions that many programs can use; these improve programmer efficiency through code reuse.

A final layer of the supercomputer system is the runtime system software. While this layer tends to be a rather thin layer in conventional HPC practices, in programming models such as Java it can be much more substantial, as in the JVM (Java

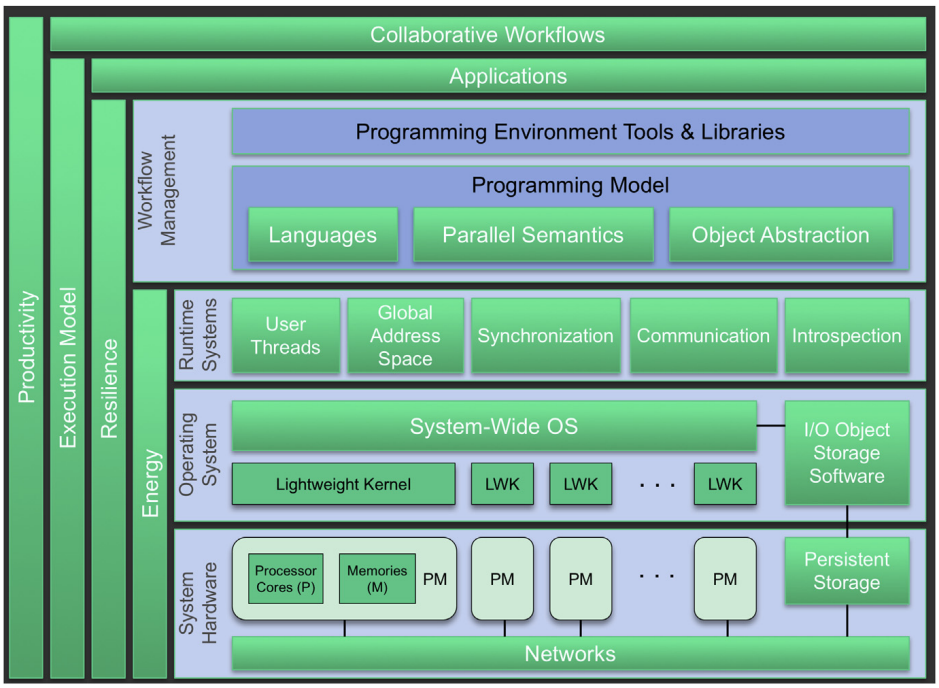


FIGURE 2.9: The system stack of a generic supercomputer consists of a hardware layer and several software layers. The first software layer is the operating system, which includes both resource management and middleware for accessing input/output (I/O) channels. The higher software layers include runtime systems and workflow management [36].

Virtual Machine). Runtime systems for HPC handle some aspects of resource management and task scheduling, as well as communication. It is possible that in future supercomputers runtime systems will play a much more significant role, but such claims are currently speculative and await confirmation through experimentation.

A high performance computer system has basic functionality and subsystems in common with a laptop personal computer, as described in the following:

- The operational functions that transform input data values into output results.
- The internal memory that stores the data on which the system operates.
- The communication channels through which intermediate data are transferred between different components and subsystems during application execution.
- The control hardware that coordinates interoperability among constituent components and subsystems.
- The mass storage that organizes and contains persistent data, system software, and application programs.
- The input/output (I/O) channels and interfaces (such as keyboard, screen, etc.) that connect users to the system.

Similarly, the software of an HPC system has much in common with a department's desktop workstation or enterprise server. Like these more pervasive though more modest computers, the supercomputer has a software framework that serves many of the same interface, control, and functional purposes, including, but not limited to, the following:

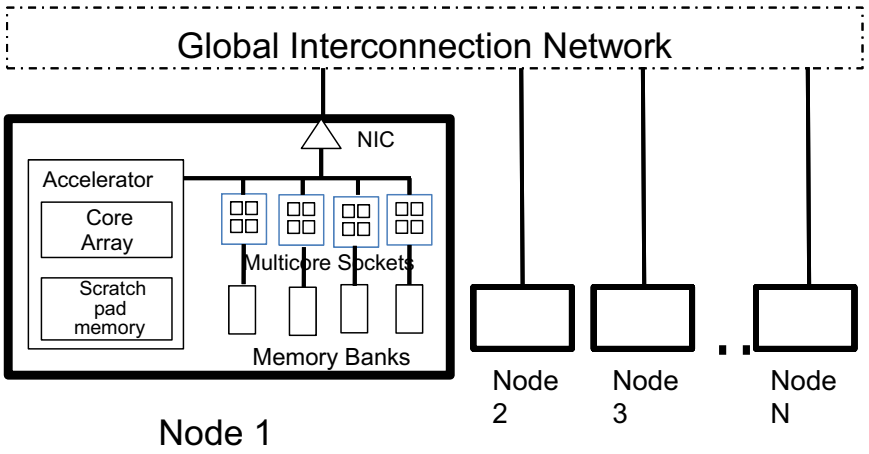


FIGURE 2.10: HPC systems are distinguished from a conventional computer by the organization, interconnectivity, and scale of the many component resources illustrated here. A *node* incorporates all the functional elements required for computation, and is highly replicated to achieve large scale [36].

- Operating system that manages all machine operations.
- Compilers that translate application programs written in human-readable syntactic languages into machine-readable binary code.
- File systems are a logical abstraction of mass storage and organize data on mass storage devices such as hard drives.
- The various software drivers of I/O devices with which the computer communicates with the outside world and users.

An HPC system differs from a conventional computer in its organization, interconnectivity, and scale of component resources and the ability of the supporting software to manage the operation of the system at that scale (Figure 2.10).

Scale refers to the degree of physical and logical parallelism, that is, the replication of key physical components such as processors and memory banks and the delineation of a number of tasks to be performed simultaneously. While even a single-socket laptop incorporates some parallelism, an HPC system is structured in multiple layers, each of which is usually much more consistent. It is this parallel organization, the methods by which the constituent subsystems are coordinated to solve a shared problem, and the additional capabilities of the system software and programming models that provide such management that differentiate the supercomputer from its smaller counterparts. From the programmer's perspective, it is the need to think in parallel (i.e., many things happening at the same time) and in a distributed manner (i.e., things happening in different locations separated by distance) that differentiates the supercomputer from the everyday computer. This requires knowledge and skill in using programming interfaces that expose and exploit application parallelism and algorithms that allow many parts of the computation that contribute to the final answer to operate simultaneously.

2.2.3.3 Key Properties of HPC Architecture

The HPC architecture extracts performance from the underlying enabling technologies for the range of applications deemed important in the context of user goals. The organization of the architecture incorporates the component structures that make best use of the devices and the data flow patterns that move information between them. Three key properties of an architecture determine the performance provided: the speed of the components that make up the system, the parallelism or number of components that can operate simultaneously doing many things at once, and the efficiency of the use of those components in the degree of utilization achieved. A simple relationship of these key factors shows their contributions to delivered performance in 2.1

$$P = e \cdot S \cdot a(R) \cdot \mu(E) \tag{2.1}$$

where P is average performance, S is scaling (the number of units that can operate at the same time), a (which is a function of reliability, R) is availability, which is the total fraction of time the system is capable of performing a computation, and μ is the instruction retirement rate (usually the clock rate) of the processor core, which is a function of the power, E. Average performance is normally reported in terms of clock rate, while S and a are generally reported without units. The main properties of an HPC architecture are:

1. **Speed**. The performance of an HPC system is directly related to the speed of its components. A key parameter is the clock speed of its component processor cores, i.e., the speed at which each one withdraws instructions, but the technologies used for different features have very different speeds or cycle times. Much of the architecture involves devising structures and methods that accommodate these different speeds. For example, a major concern is processor clock speed, again, with main memory cycle time. Processor clock speeds can range from just under 1 GHz to nearly 3 GHz. Memory cycle times presented by dynamic random access memory (DRAM) devices to processors are on the order of 100 times longer. However, there are other forms of memory, notably static random access memory (SRAM) technology, which depending on size and power consumption, can operate at or near the speed of processor core logic. A modern architecture will include a memory hierarchy consisting of a combination of slower, high-density DRAM for capacity and faster, low-density SRAM, called *cache*, to achieve speed. A third aspect of speed is the speed at which data can be transferred or communicated between any two points in the system. Two measures apply to this communication speed. Bandwidth determines the amount of information that can be transferred between two points in the unit of time, or the speed at which data moves. Latency measures the time it takes to move data between the two points. These values, too, can vary widely depending on the distances between the source and destination, as well as the type of technology used and the amount used. Architecture is, among other things, the ability to balance these different time constants in structures and methods that produce the best overall performance for a user workload (e.g., an application) while meeting normalizing cost factors such as implementation, power, and user throughput.

2. **Parallelism**. However fast the component technology may be, it will never be fast enough on its own to provide the performance needed for major application problems. Fundamental constraints such as the speed of light and atomic granularity limit how fast a single processor core can execute an instruction stream.

HPC architecture therefore depends heavily on facilities that allow many actions to be performed simultaneously-the ability to do many things at once. This characteristic is called *parallelism*, and different classes of parallel computer architectures are defined and distinguished by the diversity of structures employed to achieve parallelism in different ways, but the HPC architecture is also determined by the way in which such parallelism is controlled. Thus, both the data path and the control path are factors that determine how parallelism is exploited by an HPC architecture.

3. **Efficiency**. The third factor that determines the performance provided for a user workload is efficiency. Efficiency is primarily system utilization, that is, the percentage of time that critical components are used. For HPC, a common measure of efficiency is the ratio of the sustained floating-point performance to the theoretical peak floating-point performance, both measure in flops, floating-point operations per second.

$$e_{flops} = \frac{P_{sustained}}{P_{peak}} \tag{2.2}$$

where e_{flops} is floating-point efficiency such that $0 \leq e_{flops} \leq 1$, P_{peak} is the theoretical peak performance of the HPC architecture measured in flops, and $P_{sustained}$ is the achieved average floating-point performance. However, this typical efficiency measure reflects an earlier era when a floating-point operation was expensive and time-consuming to perform or expensive and complicated floating-point hardware. Today, data movement and memory access costs are much more significant in terms of space, time and energy than a register-to-register floating-point operation. Nevertheless, this is the metric most often encountered.

4. **Power**. Every computer, large or small, uses electrical energy for its operation. The speed of processor cores is partly proportional to their clock frequency, which in turn is related to the power applied. For example, a laptop computer requires a sustained power of 80 W or less. A desktop workstation may require 200-400 W, depending on characteristics such as the number of screens and disk capacity. These values are within the capacity of the electrical service infrastructure of a light industry building, even for many workstations. Electricity is needed not only to provide the power to drive the many integrated circuits, interconnect channels, and input/output (I/O) devices, but also to remove the resulting heat from the system. Thermal control is essential to prevent the system from overheating and consequently failing. A high-end processor can consume from about 80 W to over 200 W. Small and medium-sized computers are air-cooled. Cool air is pushed through system modules and over processor, memory, and control sockets to remove heat generated during operation. For higher wattage parts, metal radiators are mounted directly on the sockets to ensure thermal conductivity, providing more surface area and cooling capacity. This reduces the packing density and final computing capacity per module unit. Additional electrical power is required to cool the air and pass it through the systems. This can easily reach 20% of the total power budget. Liquid cooling takes advantage of the higher specific heat of fluids, in most cases water, to increase the packing density and allow the use of higher power components for higher clock speeds or larger logic arrays. There is a wide range of liquid

cooling systems and mechanisms and hybrids that combine air and liquid cooling. Active thermal control is becoming increasingly important and common in the new generation of high-performance computers. Measuring temperatures throughout the system, including those of key chips, allows thermal gradients to be monitored and can support thermal control. Modern multicore processors allow variable clock speeds, voltage regulation, and variable number of active cores, which makes it easier to achieve a balance between power and performance. This requires some level of software management, both to establish settings at the beginning of execution of an application program and to adjust them continuously during execution as the needs of the application change.

5. **Reliability**. No system operation is perfect, and the reliability of HPC systems is further compounded by their size. Errors can occur periodically due to hardware or software failures. Severe failures occur when a part of the hardware breaks down permanently, causing failure of a component within the chip, inoperability of a processor socket core, or inoperability of the entire socket. *Hard failures* can affect cores, memory, communications, secondary memory, and control. A *soft failure* occurs when a component fails intermittently but otherwise functions properly. These transient failures are due to a number of possible causes, including occasional cosmic rays or noise from low voltage margins. Software errors are due to incorrect coding of the user's application program or system support software, such as the operating system. Programmer errors are routine, and the process of debugging the program is part of the application development task. An interruption due to an operating system error is more difficult to handle because it is usually the responsibility of the system vendor. Different application problems may require different answers to arrive at a final and correct solution. For large problem computations on high-scale machines, a common methodology is *checkpoint/restart*. Periodically, the system stops a computation in progress and stores all the state of the program at that moment, i.e., *checkpoint*, usually on a secondary memory. If an error occurs after such time of execution, it is not necessary to restart the problem from the beginning, but rather from the last checkpoint. If the error is caused by a hard fault, the system must be reconfigured to remove the broken hardware from the part of the system being used before restarting the application. If it is a soft fault, the program can be restarted from the last checkpoint without reconfiguration. In the case of a software bug, the code will have to be corrected by the user or the application vendor before it can proceed. In all three cases, diagnosis is required to determine the cause and possible source of the error before proceeding with execution.

6. **Programmability**. The difficulty of writing or developing complex application code reflects the programmability of the system. While other parameters, such as performance or power, can be easily defined and quantified, programmability largely eludes specification, although there have been many attempts; for example, standard lines of code (SLOC). Although less simple to define, it is still extremely important to the overall utility of HPC. While the cost of implementing a large HPC platform can reach into the hundreds of millions of dollars, the cost of the software running on it can reach billions of dollars in total. Many factors contribute to programmability (or lack thereof), including processor core and system architectures, programming models and language facility, the effectiveness of system software such as compilers, runtime systems, and the operating system, and the skills of the programmers themselves. The

level of effort required to write the application is strongly related to the performance achieved. Within a range of behaviors, the greater the performance required, the more difficult it is to optimize the user program. This interrelationship between achieved performance and programming effort is sometimes referred to as *productivity*. Improving the ease of use of HPC systems for domain applications benefits from a number of techniques that constitute a code development discipline. Many common code libraries have been developed by experts and optimized for different types and scales of HPC systems. Code reuse is critical to managing the complexity and difficulty of application development. An application program can become as simple as building a high-level framework that invokes a sequence of existing library routines and successfully passes data between them. When programs become very complicated and are borrowed by many different users, managing the code base itself can become challenging. The discipline of software engineering provides principles and practices that guide the overall control of workflow management, including testing. These and other methods contribute to programmability.

2.2.3.4 Application Programming

The user's main point of view on an HPC system is through one or more programming interfaces, which take the form of programming languages, libraries or other services that assist in creating, optimizing and debugging application codes. There are a variety of computer programming languages, from very low-level ones such as assemblers to the declarative regime, but for HPC the number of conventionally adopted programming interfaces is relatively small, in the tens, although there are many more experimental or research models. At the risk of oversimplification, a programming language defines a set of named objects that can be manipulated, the basic operations that can be performed on these objects, flow control mechanisms to establish the conditions and order of execution of operations, and the means of encapsulation for modularity and I/O, including mass storage.

Performance is the main requirement that differentiates HPC programming from other areas. It is second only to correctness and repeatability, which are of great interest. Performance is significantly represented by the need to represent and exploit computational parallelism, that is, the ability to perform multiple tasks simultaneously. Parallel processing involves defining parallel tasks, defining criteria that determine when a task is executed, synchronization between tasks in part to coordinate sharing and allocation to computational resources. A second aspect of programming for HPC is controlling the relationship between data and task allocation and the physical resources of parallel and distributed systems. The nature of parallelism can vary significantly depending on the architecture of the computer system at which the application program is aimed. Also important are issues of determinism, correctness, performance debugging, and performance portability.

The increasing prevalence of parallelism in all application domains has warranted significant research into how the scheduling and partitioning of parallel codes can be automated. In the following, we summarize some frequently used languages and libraries for data-intensive HPC applications.

- **Message Passing Interface (MPI)** is a standard that defines the syntax and semantics of the function provided by a given library implementing it. This interface defines a way to effectively program concurrent applications on multiprocessing environments such as "clusters". Several implementations of this

standard can be found on the state-of-the-art such as MPICH [37] or OpenMPI [38]. These frameworks provide synchronization and communication between processes that may run in different processors, e.g., cores in the same CPU or different CPUs on different machines. This way, using these mechanisms, multiple processes can run concurrently on the same application. However, the communication and synchronization should be explicitly incorporated in the application by means of sending or receiving messages to/from other processes. Therefore, it becomes clear that efficient use of these mechanisms requires a profound knowledge of both the framework and the target application in order to determine when and what should be passed between processes.

- **Legion** [39] is a data-centric programming language for developing HPC applications running on distributed heterogeneous architectures. By understanding the dependencies among program data and variables, Legion provides implicit parallelism and automatic data movement operations among computational nodes. Moreover, Legion supports programmers to explicitly manage several data aspects, including partitioning, privileges, and coherence. The programming model of Legion is based on three abstractions: *local regions*, *tasks* and *mapping interface*. Local regions represent relational data in the application, described by an index space of rows and a field space of columns that can be arbitrarily partitioned into sub-region or sliced on their field space. Data partitioning can be used to express different levels of locality, while field slicing can be used for removing dependencies in data. A Legion application executes as a tree of tasks that, starting from a root task, can recursively spawn further tasks. All tasks are assigned to some logical regions they will access with given privileges. Through a user-controlled *mapping interface*, Legion allows programmers to directly control over all the details of how tasks and logical regions are assigned to processors and memories respectively.

- **Charm++** [40] is a parallel programming system based on an asynchronous message driven execution model. Both data and tasks are represented as C++ objects, called chares, which are mapped (by the runtime system) to specific computational cores for exploiting data locality. Thus, an application in Charm++ is composed of several chares, which interact each other through asynchronous method invocations with messaging. Extensive work have be done to improve interoperability with MPI and OpenMP runtimes. For example, an extension of the Charm++ runtime has been proposed for enabling chares to create tasks using OpenMP parallel construct [41]. Even though Charm++ is the result of 20 years of research, many development efforts are needed to deal with the complexities of the Exascale systems, including load balancers, scalable collective communication, support for GPU computing, language abstractions and techniques for efficient data exchange.

- **High Performance ParallelX (HPX)** is a general purpose C++ runtime for parallel and distributed application [42]. This library provides a set of parallel algorithms that extend the C++ standard library algorithms to be computed in parallel (Parallel STL) along with other utilities such as new container types or hardware counters support. The interface for the high-level algorithms is similar to that designed for this thesis and provides support for multiple execution policies. However, this interface lacks high-level patterns targeted to stream processing applications such as the farm or pipeline patterns.

- **SYCL** is a high-level programming framework that introduces an abstraction layer between the users and the OpenCL framework using an interface similar to the STL [43]. This library provides an implementation of the parallel STL by defining a new execution policy to support OpenCL. This way, the code developed using this framework becomes portable and cross-platform thanks to the OpenCL environment.

- **Celerity** [44] seeks to enable developers to scale C++ applications to accelerator clusters with relative ease, while leveraging and extending the SYCL domain-specific embedded language. This is achieved by (i) a concise API extension focusing on flexible, reusable range mapper functors, (ii) a multi-pass runtime execution model which builds an implicit, shared understanding of the data and work primitives – buffers and kernel invocations – involved in the computation at runtime, and (iii) a fully asynchronous execution environment implementation for this model.

- **OmpSs** [45] is a task-based programming model that aims to provide portability and flexibility for sequential codes while the performance is achieved by the dynamic exploitation of the parallelism at task level. It has been used to promote the StarSs [46] ideas (tasking, dependences, support to heterogeneity) into the OpenMP standard. In particular, the main objective is to extend OpenMP with new directives to support asynchronous parallelism and heterogeneity (devices like GPUs, FPGAs). The novelty of OmpSs is the concept of data dependences between tasks. Tasks in OmpSs are annotated with data directionality clauses that specify the data used by it, and how it will be used (read, write or read&write). This information is used during the execution by the underlying OmpSs runtime to control the synchronization of the different instances of tasks by creating a dependence graph that guarantees the proper order of execution. This mechanism provides a simple way to express the order in which tasks must be executed, without the need of adding explicit synchronization.

- **Kokkos** [47] is a library consisting of a performance portable abstraction layer in C++, with OpenMP and Pthreads[3] backend for multi-core CPUs and a CUDA[4] backend for NVIDIA GPUs. The compiler support is very wide, spanning the *gcc*, *clang-llvm*, and *pgi* open-access compilers. The Kokkos syntax is based on Kokkos data arrays, that map the data between CPU and device memory. Its unique quality is the polymorphic layout of the 2D arrays, that can automatically switch between row- and column-major layout, to achieve coalesced memory access on different architectures. Up to three levels of hierarchical parallelism can be specified and data arrays can be explicitly placed on different cache levels or texture memories.

- **Chapel** [48] is another programming language for advanced parallel computing developed as part of a DARPA program on HPCS. Chapel is portable and available as an open-source project. Chapel supports the execution of multithreaded applications exploiting different forms of parallelism (e.g., data parallelism, task parallelism and its variant called nested parallelism). Chapel is based on the PGAS memory model so that a task running on a given node can access shared variables allocated in the memory of other nodes. In order to exploit data locality and data affinity, Chapel enables developers with a fine-grained control over

[3] http://pubs.opengroup.org/onlinepubs/007908799/xsh/threads.html
[4] http://docs.nvidia.com/cuda/pdf/CUDA_Toolkit_Release_Notes.pdf

many aspects, such as the placement of distributed data and tasks on computing nodes. To improve interoperability, Chapel programs can include functions written in several other languages, including C, Fortran, Java, and Python [49].

- **StarPU** [50] is a programming API for shared-memory and distributed-memory parallel programming in C and C++ languages that provides data management facilities and sophisticated task scheduling algorithms for heterogeneous platforms, with the ability to easily implement custom schedulers. Its API is however still relatively low-level, and does not provide multi-node distributed memory parallelism out of the box. While it does feature facilities to integrate with MPI, even automatically transferring data between nodes based on specified task requirements [51], the splitting and distribution of work still remains the responsibility of the user.

2.3 Future Trends and Research Directions: The Opportunities and Challenges of Exascale Computing

Computer system performance and storage capacity have increased very significantly in the past decades. This prodigious growth has powered new advances in biology and medicine, physics and engineering, energy, goods design and manufacturing, transportation, environmental modeling, Internet services, financial analysis, and social media mainly depend on unceasing rises in computer performance and data storage. Current Cloud Computing platforms and parallel computing systems are capable of storing large amounts of data, but they do not provide the high performance expected for the growing number of problems where experiments are impossible, dangerous or excessively expensive. In this scenario, the design of Exascale computers is a very significant research challenge with the goal of building computers composed of a large number of multicore processors (with more than 100 cores per chip) expected to deliver a performance of 10^{18} operations per second. Extreme-scale computing will enable the solution of much more accurate predictive models and the analysis of huge amounts of data, producing needed advances in areas of science and technology. For example, Exascale computing will push the frontiers of (i) adaptation to regional climate changes, such as sea level rise, droughts and floods, and more severe weather patterns; (ii) design, control and production of advanced materials; (iii) reverse engineering of the human brain; (iv) efficiency and safety of the nuclear energy sector; and (v) drug discovery to find the best tailored therapeutic options. This section outlines the main challenges for implementing massively parallel and/or distributed applications in the area of Big Data analysis on Exascale systems.

Section 2.3.1 discusses the issues raised by "going to the exascale", which will involve a radical change in computing architecture-essentially, a huge increase in the levels of parallelism to the point where millions of processors are working in tandem-that will impose radical changes in hardware design, problem solving (e.g., application codes), and the way application codes must communicate to the underlying hardware (e.g., compilers, I/O, middleware, and related software tools). Section 2.3.2 discusses issues and challenges for implementing massively parallel and/or distributed applications in the area of Big Data analysis on exascale systems. Finally, Section 2.3.3 discusses key research areas for the implementation of scalable data analytics dealing with huge, distributed data sources.

2.3.1 On the Road to Exascale

"Why move to exascale?" This question is not intended in the trivial sense that one would ask of any expenditure on cutting-edge computing technologies, but rather is motivated by the fact that the transition from current petascale computing to exascale will involve cross-cutting investments-from hardware to fundamental algorithms, programming models, compilers, and application codes-that will exceed previous levels of investment made as computing architectures evolved in the past.

As a first step, it is useful to be explicit about the types of issues that will justify large investments in next-generation computing. In general, one can divide forefront computing problems into three general categories:

- *Incrementally advanced computing.* In traditional scientific computing, many computational problems are characterized by a benign connection between advances in computational capabilities and the benefits gained from those advances. In other words, these problems have the characteristic that computing a little better yields a little better results. At times, these advances may be important-for example, they allow for greater predictability in calculations-but they rarely, if ever, transform a discipline.

- *Voracious computing.* In some problem areas of computational science, there is no a priori reason to believe that increased computational capabilities will lead to increased knowledge, even incrementally. A classic example is isotropic and incompressible homogeneous hydrodynamic turbulence: after solving the viscous subfield, there is currently no reason to believe that larger computational domains (i.e., with higher spatial and temporal resolution) will enable us to better understand the fundamental physics of turbulence. Therefore, these problems are voracious in the sense that no a priori reason can be given for limiting the computational resources required to a fixed level.

- *Transformational computing.* In some cases, computational problems have the property that they can be completely solved by sufficiently large computations, where "sufficiently large" means that one can specify quantitatively in advance what "sufficiently large" means. Turbulence in a finite domain (such as in a tea cup or the combustion chamber of a jet engine) is such a problem: a "sufficiently large" calculation will be able to completely solve the problem. This type of calculation is transformative, meaning that a smaller calculation will not solve the underlying problem, but a sufficiently large calculation can solve it completely.

As in any case involving the potential for large expenditures, it is incumbent upon us to consider the question: 'why now?' That is, does there exist urgency for proceeding to the exascale? What are the risks involved if we do not proceed? A leadership role of countries that invest in exascale computing will be accompanied by the collateral benefits of leadership and innovation in a variety of sectors, including aerospace, automotive, energy, healthcare, computing, and manufacturing, as well as the job creation that accompanies such leadership. Because the aerospace industry has long been an early adopter of HPC for simulating airframes and jet turbines, the exascale capability will allow aerospace companies to gain a competitive advantage over those that will not adopt such strategies. The automotive industry, on the other hand, has adopted HPC more recently, but given the competitive challenges for fuel efficiency, access to exascale simulations of internal combustion engines could significantly shorten the design cycle by using advanced computational fluid dynamics to meet the federal obligation to reduce greenhouse gas emissions by 80% by 2050

and reduce oil use by 25% by 2020. In the health care sector, exascale computing can contribute not only to advancing the scientific research needed to prevent and treat cancer and other challenging diseases, but also to reducing costs and increasing the effectiveness of health care services through, for example, the use of techniques such as compressive sensing to achieve lower radiation dosages. Exascale computing can enable the development of new materials, along with the infrastructure for high-value manufacturing processes involving these materials, as has been done in the past for materials such as semiconductors and polymers. For the information technology industry, leadership in hardware and software innovation for exascale computing will be accompanied by leadership in computing, from embedded computers and portable mobile devices to laptop and desktop servers to departmental servers, which include the use of high levels of multithreading parallelism within the chip. There is a critical need to find hardware and software solutions (e.g., the power challenges for mobile devices are very similar to those for nodes in an exascale system), and whichever country is a leader in exascale computing will also enjoy technology leadership in IT in general. Finally, leadership in exascale computing is also critical for national security, whether because of the need for simulation to replace nuclear weapons testing, nuclear test ban treaties, or the needs of the intelligence communities to mine large datasets.

2.3.2 Data-Intensive Exascale Computing

Extreme Data is an incarnation of Big Data concept distinguished by the massive amounts of data that must be queried, communicated and analyzed in (near) real-time by using a very large number of memory/storage elements and Exascale computing systems. Immediate examples are the scientific data produced at a rate of hundreds of gigabits-per-second that must be stored, filtered and analyzed, the millions of images per day that must be mined (analyzed) in parallel, the one billion of social data posts queried in real-time on an in-memory components database. The data intensity of scientific and engineering applications dictates the expansion of Exascale systems. This places emphasis on architectures, programming models, and runtime systems to improve data-intensive processing. A major challenge is to use available technologies and large-scale computing resources effectively to address scientific and societal challenges. This section describes programming models and runtimes required to facilitate the task of scaling and extracting performance of data-intensive applications [52].

2.3.2.1 Programming Issues for Exascale Data Analysis

Data analysis gained importance because of the large and pervasive availability of data sources and the continuous enhancements of techniques and algorithms to find insights in them. When the volume of data to be analyzed is of the order of terabytes or petabytes (billions of Tweets or posts), scalable storage and computing solutions must be used. Processing very large data volumes requires operations and new algorithms, that are able to scale in loading, storing, and processing massive amounts of data that generally must be partitioned in very small data grains, on which analysis is done by thousands to millions of simple parallel operations.

Reaching exascale in terms of computing nodes requires the transition from current control of thousands of threads to billions of threads as well as the adaptation of the performance models to cope with an increased level of failures. In order to explore this situation more precisely, the limitations of current programming models along-with evaluations of promises of hybrid programming model to solve these scaling-related difficulties are investigating in the following.

- **Limitations of message passing programming model**. The current vision for exascale systems is to exploit distributed-memory parallelism, and so it is likely that the message passing model will be used at least in part. Moreover, the most popular implementation of this model, MPI, has been shown to work with millions of cores for particular problems. However, it has been shown that the all-to-all communication algorithms used in message passing models are not scalable (the most commonly used implementations often assume a fully connected network and have dense communication patterns), whereas all-to-all, one-sided, or sparse communication patterns are more reliable. In addition, parallel I/O is a limiting factor in MPI systems, showing that the current MPI-IO model should be reconsidered. In particular, the limitations are related to collective access to I/O request and data partitioning.

- **Limitations of shared-memory programming models**. The exascale system is expected to handle hundreds of cores in a single CPU or GPU. The use of shared memory systems is a feasible alternative to message passing in the case of medium-sized parallel systems in order to reduce the programming overhead and shift the burden of parallelization from the programmer to the compiler. The most popular shared-memory system, OpenMP, follows a parallelism control model that does not allow control of data distribution and uses a non-scalable synchronization mechanism. In addition, the global view of data easily leads to inefficient scheduling, as it encourages synchronization of remote accesses to data from all threads, similar to local accesses. The emerging Partitioned Global Address Space (PGAS) model seeks to overcome the scalability problems of the shared-memory global model. The PGAS model is likely to be advantageous when non-global communication models can be implemented with minimal synchronization and overlap of computation and communication. Moreover, the scalability of I/O mechanisms in PGAS depends only on the scalability of the underlying I/O infrastructure and is not limited by the model. However, scalability is limited to thousands of cores. Load balancing is still an open problem for systems implementing the model.

- **Limitations and innovations of heterogeneous programming**. Heterogeneous node clusters composed of multicore CPUs and GPUs are increasingly used for High Performance Computing due to their advantages in peak performance and energy efficiency. To fully exploit the computational capabilities of these architectures, application developers often use a combination of different parallel programming paradigms (e.g., OpenCL, MPI, and OpenMP). Heterogeneous computing poses the new challenge of how to manage the diversity of execution environments and programming models. In recent years, heterogeneous systems have received a great amount of attention from the research community. Although several projects have been recently proposed to facilitate the programming of clusters with heterogeneous nodes [53, 54], none of them combines support for high performance inter-node data transfer, support for a wide number of different devices and a simplified programming model. An innovative approach to program clusters of nodes composed of multi-core CPUs and GPUs has been introduced through *libWater* [55], a library-based extension of the OpenCL programming paradigm that simplifies the development of applications for distributed heterogeneous architectures. Besides OpenCL-based approaches, also CUDA solutions have been proposed to simplify distributed

systems programming, such as CUDASA [56], an extension of the CUDA programming language which extends parallelism to multi-GPU systems and GPU-cluster environments, or rCUDA [57], a distributed implementation of the CUDA API that enables shared remote GPGPU in HPC clusters. However, all of these approaches are limited to devices that support CUDA, i.e. NVIDIA GPU accelerators, and therefore they cannot be used to address heterogeneous systems which combines CPUs and accelerators from different vendors.

2.3.2.2 Runtime requirements

The development of data-intensive applications on Exascale platforms requires the availability of effective runtime systems. Future runtime systems must support functional and nonfunctional requirements to support users in designing and running complex data-intensive applications on parallel and large-scale distributed platforms in an effective and scalable manner. In particular, the functional requirements can be grouped into four areas: data management, tool management, design management, and execution management.

- **Data management**. The data to be processed can be stored in different formats, such as relational databases, NoSQL databases, binary files, simple files, or semi-structured documents. The runtime system should provide mechanisms to store and access such data regardless of their specific format. In addition, metadata formalisms should be provided to describe the relevant information associated with the data (e.g., location, format, availability, available views) in order to enable efficient access, manipulation, and processing.

- **Tool management**. Data processing tools include programs and libraries for data selection, transformation, visualization, extraction, and evaluation. The runtime system must provide mechanisms for accessing and using these tools, regardless of their specific implementation. Again, metadata must be provided to describe the most important features of these tools (e.g., their function, location, usage).

- **Design management**. From a design perspective, three main classes of data-intensive applications can be identified: single-task applications, in which a single sequential or parallel process task is executed on a given dataset; parameter sweep applications, in which data are analyzed using multiple instances of a data processing tool with different parameters; and workflow-based applications, in which data-intensive applications are specified as possibly complex workflows.

- **Execution management**. The system must provide a parallel/distributed runtime environment to support the efficient execution of user-designed data-intensive applications. Since applications range from single tasks to complex workflows, the runtime system must be able to handle such a variety of applications. Specifically, the runtime environment must provide the following capabilities, related to the different phases of application execution: access to the data to be processed; allocation of the necessary computational resources; execution of the application according to user specifications, which can be expressed as a workflow; and the ability for users to monitor the execution of an application.

Non-functional requirements can be defined at three levels: user, architecture and infrastructure. From the user's perspective, nonfunctional requirements to be met include:

- **Data protection**. The system must protect data from both unauthorized access and intentional/incidental loss.
- **Usability**. A system must be easy for users to use, without the need for special training.

From the point of view of architecture, the following principles should guide the design of the system:

- **Openness and extensibility**. The architecture must be open to the integration of new data processing tools and libraries; in addition, existing tools and libraries must be open to extension and modification.
- **Independence from infrastructure**. The architecture must be designed to be as independent of the underlying infrastructure as possible; in other words, the system services must be able to take advantage of the basic functionality provided by different infrastructures.

Finally, from an infrastructure perspective, non-functional requirements include:

- **Heterogeneous/Distributed data support**. The infrastructure must be capable of handling very large, high-dimensional datasets stored in different formats at a single site or geographically distributed across many sites. geographically distributed across many sites.
- **Availability**. The infrastructure must function even in the presence of failures that affect a subset of the hardware/software resources. Therefore, effective mechanisms (e.g., redundancy) must be implemented to ensure reliable access to sensitive resources such as user data.
- **Scalability**. The infrastructure must be able to handle an increasing workload (resulting from bigger data to process or heavier algorithms to execute) efficiently and effectively by dynamically allocating the necessary resources (processors, storage, network). In addition, as soon as the workload decreases, the infrastructure must free up unneeded resources.
- **Efficiency**. The infrastructure must minimize resource consumption for the execution of a given task. In the case of parallel/distributed tasks, efficient allocation of processing nodes must be ensured. In addition, the infrastructure must be highly utilized to provide efficient services.

2.3.2.3 Exascale Scalability in Data Analysis

Implementing scalable data analysis applications in Exascale computing systems is a complex task, requiring fine-grained high-level parallel constructs and expertise in parallel and distributed programming. In particular, mechanisms and skills are needed to express task dependencies and parallelism among tasks, to use synchronization and load balancing mechanisms, and to properly manage memory and communication among a very large number of tasks. Given the variety of data analysis applications and types of users (from end users to skilled programmers) that can be envisioned in future Exascale systems, there will be a need for scalable programming models with different levels of abstractions. Using high-level scalable models, the programmer defines only the high-level logic of an application, while low-level details that

are not essential to the design of the application are hidden, including infrastructure-dependent execution details. The programmer is helped in defining the application, and application performance depends on the compiler analyzing the application code and optimizing its execution on the underlying infrastructure. Instead, low-level scalable models allow the programmers to interact directly with computing and storage elements of the underlying infrastructure and thus to define the applications parallelism directly. In this case, programming an application requires more skills, and the application performance strongly depends on the quality of the code written by the programmer.

Data-intensive applications can be designed through a visual programming formalism, which is a convenient design approach for high-level users, e.g., domain expert analysts with a limited understanding of programming. In addition, the graphical representation of workflows inherently captures parallelism at the task level, without the need to make parallelism explicit through control structures [58]. *Code-based formalism* allows users to program complex applications faster, more concisely, and with greater flexibility [59]. *Code-based applications* can be defined in several ways: i) with a language or language extension that allows parallel applications to be expressed; ii) with some annotations in the application code that allow the compiler to understand which instructions will be executed in parallel; and iii) using a library in the application code that adds parallelism to the application.

An accurate modeling of basic operations and the programming languages/APIs will require to support the efficient implementation of Exascale data analysis applications composed of up to millions of computing units, which process small data elements and exchange them with a very limited set of processing elements. At the same time, a significant programming effort of developers will be needed to implement complex algorithms and data-driven applications. At the Exascale scale, the cost of accessing, moving, and processing data across a parallel system is enormous. This requires mechanisms, techniques and operations for efficient data access, placement and querying. In addition, scalable operations must be designed in such a way so as to avoid global synchronizations, centralized control, and global communications.

2.3.3 Key Research Area

Cloud-based solutions for Big Data analytics tools and systems are well advanced on both the research and commercial fronts. On the other hand, new Exascale hardware/software solutions need to be researched and designed to enable the mining of large-scale datasets on these new platforms. Exascale systems place new requirements on application developers and programming systems that must focus on architectures composed of a very large number of homogeneous and heterogeneous cores. General issues such as power consumption, multitasking, scheduling, reproducibility, and resilience need to be addressed, along with other data-oriented issues such as data distribution and mapping, data access, communication, and data synchronization. Scheduling constructs and runtime systems will play a crucial role in enabling future data analytics programming models, runtime models, and hardware platforms to address these challenges and support scalable implementation of real-world Big Data analytics applications.

In particular, here we summarize a number of open design challenges that are fundamental to the design of Exascale programming systems and their scalable implementation.

2.3.3.1 The Hardware Challenges

1. **Power consumption**. Computer power consumption in Exascale systems is the biggest challenge in hardware research. Historically, HPC has focused on building the fastest system for maximum throughput. As a result, trends in computer architecture show higher clock frequencies and increased hardware parallelism in the form of transistors, cores, and nodes (Moore's Law). Although this trend continues to dominate the chip industry, it has raised some critical issues in recent times. Higher clock frequencies and increased hardware parallelism increase the amount of heat generated by transistor switching. This heat must be dissipated to keep the components at a reasonable temperature and avoid hardware failure. This results in high power consumption, since power is directly related to the amount of heat generated. The higher the power consumption, the more expensive the annual operating cost for the supercomputing center. With high power consumption, the operating cost becomes expensive and unsustainable for a supercomputing center. As summarized in Table 2.2, comparing a petascale machine in 2010 to an exascale machine in 2023, power consumption must only increase by one order of magnitude, while performance will increase by three. Thus, solutions will need to be 100X more power-efficient at exascale than petascale.

Parameter	2010	2023	Ratio
System Peak	2 PF	1 EF	500
Power Consumption	6 MW	20 MW	3
System Memory	0.3 PB	10 PB	33
Node Performance	0.125 GF	10 TF	80
Node Memory Bandwidth	25 GB/s	400 GB/s	16
Node Concurrency	12 CPUs	1K CPUs	83
Interconnect Bandwidth	1.5 GB/s	50 GB/s	33
System Size (Nodes)	20K	1M	50
Total Concurrency	225K	1B	4,444
Storage	15 PB	300 PB	20
I/O Bandwidth	0.2 TB/s	20 TB/s	100

TABLE 2.2: Predicted Exascale design targeted for 2023, and the resulting change from a petascale system in 2010. The large imbalance between performance and power at exascale is highlighted in red [60].

2. **Data movement**. From a scientist's point of view, the ratio of memory to processor is critical in determining the size of the problem that can be solved. The processor determines the amount of computation that can be performed, while the memory determines the size of the problem that can be handled. The system memory challenge is only one important aspect of the larger data movement challenge for Exascale systems. Figure 2.11 shows the conceptual organization of the hierarchy of components that store and move data in HPC systems. This hierarchy is designed to provide and store the data needed by the processing units. At the top of the pyramid are the registers and cache. These are the fastest, smallest, and most expensive memory modules in the system. Main memory is much slower (processors can perform 100 operations in the time it takes to get a word of data from memory) and much larger than the cache. External memory is much larger, but it too runs at extremely low

speeds compared to the speed of the processor. Since all these levels of the pyramid are needed to run scientific applications, the challenge is to provide the maximum capacity at each level (consistent with a plausible overall cost of the machine) and to provide the most efficient methods of moving data between levels, depending on the needs of the various applications.

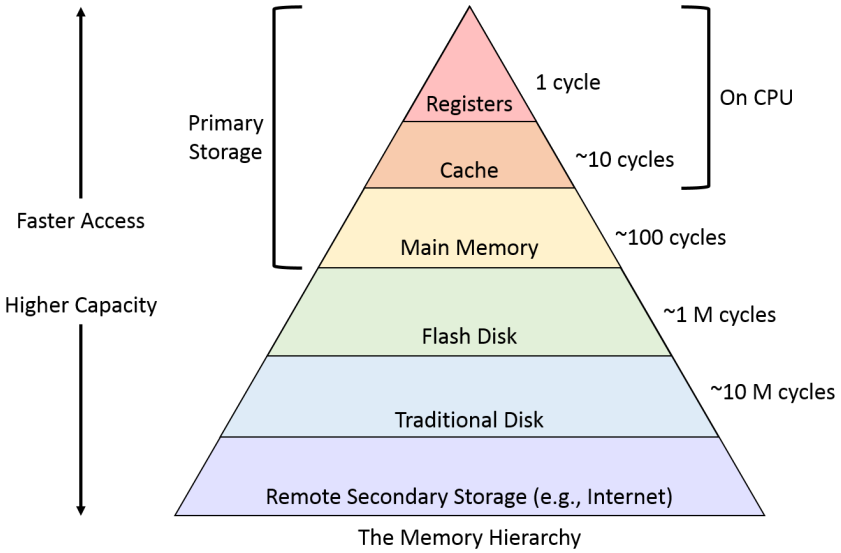


FIGURE 2.11: The data access/storage hierarchy. Source image: https://www.cs.swarthmore.edu/~kwebb/cs31/f18/memhierarchy/mem_hierarchy.html

The two main obstacles to providing adequate capacity at each level are: i) achieve adequate data transfer rate, or bandwidth, and reduce time delays, or latency, between layers, and ii) performance per disk, failure rates, and energy efficiency no longer improve. In today's petaflop systems, memory access at all levels is the limiting factor in almost all applications, but work will also need to be done to improve the efficiency of data when and how it needs to be moved. Research options include better data analysis to anticipate what data is needed before it is requested, determining when data can be efficiently recomputed instead of stored, and improving data layout to maximize data usage when moved between layers. Further hardware research, including non-volatile memory fills and advanced packaging will help close the current latency gap and provide sufficient overall storage capacity to run exascale applications.

3. **System resiliency**. With the transition from petascale systems to exascale systems, the number of system components will increase faster than the reliability of the components, with projections on the order of minutes or seconds for exascale systems. Based on current knowledge and observations of existing large systems, exascale systems are expected to experience failures of various types many times a day. Increasing evidence points to an increase in silent errors (failures that are never detected, or are detected long after generating erroneous results), which cause havoc and will become increasingly problematic as the number of components in exascale systems increases. Systems running 100 million cores will continually see core failures, and the tools to handle them will need to be rethought [61]. The current approach for resilience, which relies on automatic or application-level checkpoint/restart, will not work because the time to checkpoint and restart will exceed the mean time to failure (MTTF) of

a complete system. This set of projections presents a difficult challenge: finding new approaches to run applications to their normal termination, despite the expected unstable nature of exascale systems. The ability for a scientist to make progress will be difficult unless alternative methods for fault recovery that do not involve checkpoints/restarts are provided [62].

2.3.3.2 The Algorithmic Challenges

The availability of new hardware and exascale computer supercomputer architectures will require the radical redesign, reimplementation, and even reinvention of algorithms to take advantage of massively parallel and heterogeneous processing capabilities. While application porting often dramatically increases the performance and reduces the power consumption of HPC applications, the breakthrough performance gains that enable solving computational problems currently considered intractable require the conversion of innovative concepts into new algorithms and their efficient and reliable implementation. The move to exascale will place greater demands on algorithms in at least two areas: the need for an increasing amount of data locality to perform computations efficiently, and the need to achieve much higher factors of fine-grained parallelism as high-end systems support an increasing number of computational threads.

1. **New multicore-friendly and multicore-aware algorithms**. Scalable multicore systems bring new relationships between computation and communication. Within a node, data transfer between cores is relatively inexpensive, but time affinity is still important for effective cache use. Between nodes, the relative cost of data transfer is increasing greatly. Developing new algorithms that take these issues into account can often yield excellent results, as can algorithms that avoid communication and increase the computation/communication ratio or algorithms that support simultaneous computation/communication or algorithms that vectorize well and have a large volume of functional parallelism.

2. **Fault tolerance and robustness for large-scale systems**. Modern PCs can run for weeks without rebooting, and data servers should run for years. However, because of their scale and complexity, today's supercomputers run for only a few days before rebooting. Exascale systems will be even more complex with millions of processors. The main challenge in fault tolerance is that failures in extreme-scale systems will be continuous and not an exceptional event. This requires a radical change from today's software infrastructure. Every part of the exascale software ecosystem must be able to cope with frequent failures, otherwise applications will not be able to run all the way through. System software must be designed to detect and adapt to frequent failures of hardware and software components. Research into the reliability and robustness of exascale systems for running large simulations is critical to the effective use of these systems. New paradigms for fault management need to be developed in both system software and user applications. Equally important are new approaches to integrate detection algorithms into both hardware and software and new techniques to help simulations adapt to faults.

3. **Adaptive Response to Load Imbalance**. Adaptive multiscale algorithms are critical because they apply computational power precisely where it is needed. However, they introduce challenging computational requirements because they involve dynamic variation in computation that results in load imbalances due to the static distribution of tasks. As we move toward systems with billions of processors, even algorithms with natural load balancing on homogeneous hardware

will present many of the same adaptive load balancing problems seen in today's adaptive codes. For example, software-based recovery mechanisms for fault tolerance or power management functions will create substantial load imbalances as tasks are delayed by rolling back to a previous state or correcting detected errors. Directed acyclic graph (DAG)-based scheduling also requires new approaches to optimize resource utilization without compromising spatial locality. These challenges require the development and implementation of sophisticated software approaches to rebalance computation dynamically in response to changing workloads and operating environment conditions.

2.3.3.3 The Computer Science Challenges

Exascale systems place new requirements on application developers and programming systems that must focus on architectures composed of a very large number of homogeneous and heterogeneous cores. General issues such as power consumption, multitasking, scheduling, reproducibility, and resilience need to be addressed, along with other data-oriented issues such as data distribution and mapping, data access, communication, and data synchronization. Scheduling constructs and runtime systems will play a crucial role in enabling future data analytics programming models, runtime models, and hardware platforms to address these challenges and support scalable implementation of real-world Big Data analytics applications.

In particular, here we summarize a number of open design challenges [63] that are fundamental to the design of Exascale programming systems and their scalable implementation. The following design choices, among others, must be considered:

- *Programming models for Big Data analytics.* Programming tools for Big Data analytics require new complex abstract structures. The MapReduce model is often used on clusters and clouds, but further research is needed to develop high-level scalable models and tools. State-of-the-art solutions have generated great successes, but they are not mature and suffer from several problems, from data transfer bottlenecks to unpredictable performance. bottlenecks to data transfer and unpredictable performance.

- *Data and tool interoperability and openness.* Interoperability is one of the main problems of large-scale applications using resources such as data and compute nodes. Standard formats and templates are needed to support interoperability and facilitate cooperation among teams using different data formats and tools.

- *Integration of Big Data analytics frameworks.* The service-oriented paradigm enables large-scale distributed workflows on heterogeneous platforms together with software components developed with different programming languages or tools.

- *Scalable software architectures for fine grain in-memory data access and analysis.* Exascale processors and storage devices must be exploited with fine-grained runtime models. Software solutions to manage many cores on processors and scalable communications between processors must be designed to take advantage of exascale hardware.

- *Tools for data exploration and model visualization.* New approaches to data exploration and model visualization are needed, taking into account the size of the data and the complexity of the extracted knowledge. As the size of the data

increases, visualization tools will become increasingly useful for summarizing and displaying it in a compact and easy-to-see way.

- *Local mining and distributed model combination.* Because Big Data applications often involve multiple local sources and distributed coordination, collecting distributed data sources in a centralized server for analysis is neither practical nor possible. Scalable data analysis systems must allow local data source extraction, pattern exchange and fusion mechanisms to compose the results produced in distributed nodes. According to this approach, global analysis can be performed by distributing local extraction and supporting the global combination of each local knowledge to generate the complete model.

- *In-memory analysis.* Most data analysis tools query data sources on disks, whereas, unlike these, in-memory analytics query data in main memory (RAM). This approach brings many advantages in terms of query speed and faster decisions. For example, in-memory databases are very effective in real-time data analysis, but they require high-performance hardware support and fine-grained parallel algorithms. New 64-bit operating systems allow memory addressing up to 1 TB, making it realistic to cache a very large amount of data in RAM. Therefore, this area of research is very promising.

2.4 Summary and Outcomes of Chapter

In recent years, the field of Big Data analytics is a very active research discipline that has a significant impact on industrial and scientific processes and applications. This chapter has explored the evolution of Big Data technologies aimed at providing a performance- oriented and systems-oriented introduction to HPC. Since the introduction of MapReduce, the world of data intensive processing has evolved rapidly. The original inflexible programming model of MapReduce has been expanded to incorporate the full spectrum of data intensive processing paradigms. Now that access to data stored in a distributed file system has been separated from the MapReduce programming model, new systems such as Apache Spark have been able to push application performance further by using memory locality. MapReduce was only the first step toward democratizing parallel processing; future frameworks will continue this trend and make massive computing power available to nonspecialists. Sections are organized to deliver three kinds of information: concepts, knowledge, and skills. Concept discussions aim to teach those ideas that have established theoretical foundations, enjoy longevity, and largely will not change. Knowledge sections aim to impart information about supercomputing to the reader that will evolve with time and will need to be added to in the future. Finally, skills needed for entry-level work in supercomputing are presented in tutorial style for ease of learning. These skills may change over time, but represent current conventional practice in the field.

Another aspect discussed in this chapter concerns a number of open design challenges that are critical for future Exascale platforms will be exploited for implementing scalable Big Data analysis in all the areas of science and engineering. In many cases, Big Data is being stored and analyzed on cloud platforms. In particular, the development of exascale computers is a very significant research challenge that is being studied with the goal of building computers composed of hundreds of thousands of cores to provide performance of 10^{18} operations per second. We focused particularly on answering a central question, "Are there scientific disciplines in which exascale computing can presumably lead to transformative changes in those disciplines?" From

biology to nuclear engineering, exascale computing promises remarkable advances in our ability to model and simulate complex phenomena, at levels of fidelity that have the potential to radically change both our understanding and our ability to understand. Therefore, it is almost certain that the transition to the exascale will bring great benefits. Next, we addressed questions directly related to the transition to the exascale: "What will it entail and what will be the challenges? And how difficult will the transition from the current era of tera- and peta-scale computing to the new era of exascale computing be?" The challenges to be overcome are numerous and difficult; and the resulting need for substantial investment in research and development, on all fronts, cannot be downplayed: from hardware details, to algorithms and programming models, to error and fault detection and handling at both the hardware and software levels, and finally to the tools that make efficient computation possible (i.e., compilers (optimizers), load balancers, performance diagnostic tools, and the programming languages). In addition, efficient methods for storing, accessing, communicating, and extracting data are needed. When these methods are designed and implemented, Exascale computing systems will be leveraged to implement scalable Big Data analysis in all areas of science and engineering.

Chapter 3

Real Data Analysis Applications for Society

Extracting useful knowledge from large digital datasets requires intelligent and scalable analysis algorithms, services, programming tools and applications. For example, social data analytics is emerging as a rapidly growing area of research to analyze collective sentiments and understand the behavior of groups of people or the dynamics of public opinion. One of the most interesting features of social networks is the ability to associate a spatial context with social posts. For example, Twitter, Facebook, Flickr and Instagram take advantage of GPS readings from cell phones to tag tweets, posts and photos with geographic coordinates. Therefore, social network users traveling through groups of places produce a huge amount of geo-location data that incorporate extensive knowledge of human dynamics and mobility behaviors. Leveraging the rich information provided by geo-referenced social data can impact many areas, including urban planning, smart traffic management, route recommendations, safety, and health monitoring.

From social data to healthcare, the collection and use of data from settings that reflect the realities of everyday health care is likely to become increasingly important in the drug approval process. This is because clinical trial data alone often do not accurately reflect the effect a drug would have on a larger population of patients in the real world, due to strict inclusion and exclusion criteria. For example, elderly patients, patients with a particularly severe or rare case of a disease, or patients using concomitant medications are often excluded from studies. Big Data analysis can then be used to understand how these excluded individuals react to a drug, which will provide a much broader view of the effect a product will have on the heterogeneous patient population outside the clinical trial setting. This broader insight into the patient population can be beneficial as it leads to: (i) greater understanding of the varied patient journeys, treatment pathways, and levels of drug effectiveness; (ii) more effective drug development programs; (iii) minimization of the number of patients exposed to less efficacious therapy.

In this chapter, we describe methods, techniques, and prototypes designed and used to implement Big Data solutions on massive data sources for: (i) understanding urban public transport mobility patterns much better and increasing the efficiency of public transport systems (Section 3.1); (ii) extracting relevant information about users behavior, interests and activities (Section 3.2); (iii) collecting high-dimensional epidemiological data and transforming those data into a consistent schema to enable open data sharing and rapid science during health emergencies. (Section 3.3).

3.1 Urban Computing

3.1.1 ASPIDE project prototype: Trajectory mining application

Urban computing is the process of acquiring, integrating, and analyzing large and heterogeneous urban data to address the major problems facing cities today, including air pollution, energy consumption, traffic flows, human mobility, environmental conservation, business activities, and public spending savings. The widespread use of social media platforms allows scientists to collect huge amounts of data posted by people interested in a given topic or event. This data can be analyzed to infer patterns and trends about people's behaviors related to a topic or an event on a very large scale. Social media posts are often tagged with geographical coordinates and/or other information that allows applications to identify user positions, thus enabling mobility pattern analysis using trajectory mining techniques. The goal of this application is to discover frequent trajectories from people movements, so as to find the common routes followed by social media users. The input of our application is a large set of geotagged items gathered from Flickr.

The goal of the urban computing application we developed for the ASPIDE project (more details can be found in Appendix A.1) is to discover sequential patterns from people movements, so as to find the common routes followed by social media users. The analysis will be carried out by analyzing 16 millions of Flickr posts. The whole dataset is stored on a set of JSON files with a total size of 50 GB. The JSON files contains textual information, in particular each text line represents a Flickr post, i.e., the information associated to a photo uploaded in Flickr. Considering that, each photo referred in a Flickr post has an average dimension of 500 KB (Flickr large format, 1024px width), totally additional 8 TB (16 M*500 KB) of photos can be further downloaded. An example of Flickr post is shown in Figure 3.1.

```
{       "ID":"43012793876",
        "DATETIME":"2016-11-21T22:12:36.000",
        "LOCATION":{"LNG":12.456661,"LAT":41.90245},
        "USER":{ "USERID":"111222333@N00", "USERNAME":"fm84"}
        "TITLE":"Basilica di San Pietro",
        "DESCRIPTION":"St Peter's church in Rome"
        "HASHTAGS":["holiday","vatican"],
        "ACCURACY": 16}
}
```

FIGURE 3.1: Metadata of a Flickr post serialized in JSON format.

In the following is described the trajectory data analysis workflow proposed by our research group. The workflow, shown in Figure 3.2, is composed by five steps (some of them are optional)[1]:

1. *Crawling*: during this step, it is possible to run multiple crawlers in parallel for collecting large amounts of data from social media. If data have already been downloaded and stored in files, a specific crawler is used to load the data.

2. *Filtering*: this step uses a set of filtering functions to verify whether the social media items meet certain conditions or not.

[1] https://www.aspide-project.eu/wp-content/uploads/sites/62/2021/05/D2-6.pdf

3. *Automatic keywords extraction and data grouping*: thisstepextractskeywords identifying the places of interests (PoIs); then, it uses these keywords for grouping social media items according to the places they refer to. If the keywords are provided as a workflow input, the extraction step can be skipped.

4. *RoIs extraction using a parallel clustering approach*: starting from grouped social media data, a parallel clustering approach (ParCA) is exploited to identify Regions-of-Interest (RoIs) efficiently. The RoI extraction step can be skipped if the RoIs are provided as input of the workflow.

5. *Frequent pattern mining*: this step discovers behavior rules, correlations and mobility patterns of people, by analysing geotagged social media items. In particular, both associative and sequential analysis can be performed, using FPGrowth and Prefix-Span algorithms, respectively.

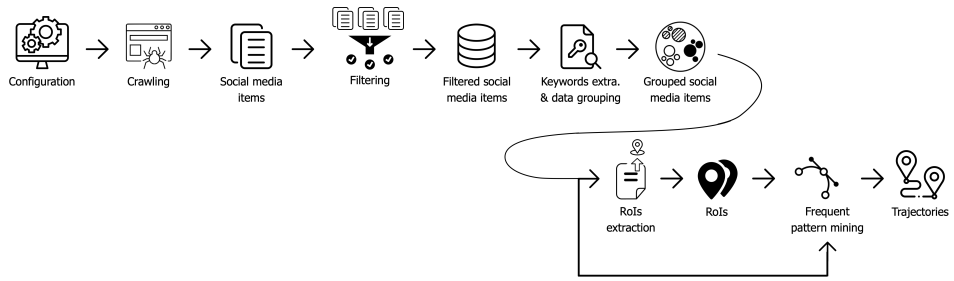


FIGURE 3.2: Workflow of the urban computing use-case.

Overall, the analysis process produces the following intermediate/final outputs: (i) the sets of keywords that are used to refer the PoIs in the area under analysis; (ii) the RoIs calculated for the different PoIs, which can also be easily displayed on a map; (iii) the most frequent trajectories followed by users.

Figure 3.3 shows an example of outputs produced by our algorithm in an area of Rome. In particular, starting from the geotagged social media posts generated by four users (U_1,...,U_4) , the method identifies five Places-of-Interest and the related keywords (e.g., ⟨Colosseum, Colosseo, Coliseo, Colaseum, Coloseu, ...⟩ for the Colosseum). From such places, the associated RoIs are generated to understand whether a user has visited or not a certain PoI. For each user, we can extract his/her movements across places, grouping social media posts by user id and sorting them by date and time. As an example, Figure 3.3 shows the trajectory of user U1: Colosseum → Roman Forum → Circus Maximus → Tiber Island. Through the analysis of the trajectories of different users we can discover the most frequent behaviors and trajectories.

Some of the most interesting results that have been obtained are summarized in Table 3.1, which shows the top 5 places visited in Rome, with the corresponding support in the data. The Colosseum is the most visited place, followed by the St. Peter's Basilica.

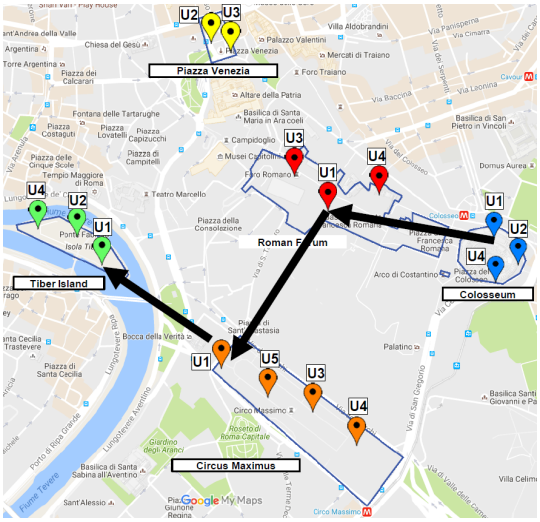


FIGURE 3.3: An example of the outputs produced.

Place	Support
Colosseum	21.7%
St Peter's Basilica	13.9%
Trastevere	8.7%
Pantheon	6.5%
Trevi Fountain	5.3%

TABLE 3.1: Top 5 places visited in Rome.

Table 3.2 shows the most frequent itemsets of length 3 that have been discovered in Rome. As reported, ⟨Pantheon, St. Peter's Basilica, Colosseum⟩ is the most frequent set of places visited by social users in Rome, with a support of 5.3%. Combining the information contained in Tables 3.1 and 3.2, an interesting result is that Trastevere, a popular district of Rome, is the third most visited place, but it is not present in any frequent itemset. This could happen because Trastevere is visited by people during the evening, for having a dinner in one of its many restaurants or pubs, but it is not part of common tourist routes during the daylight.

Set of places	Support
Pantheon, St. Peter's Basilica, Colosseum	5.3%
Trevi Fountain, St. Peter's Basilica, Colosseum	4.5%
Roman Forum, St. Peter's Basilica, Colosseum	4.4%
Vatican Museums, St. Peter's Basilica, Colosseum	4.4%
Trevi Fountain, Pantheon, Colosseum	4.0%

TABLE 3.2: Top 5 frequent sets of places visited in Rome.

3.1.2 Ticket Sales Prediction and Dynamic Pricing Strategies in Public Transport

The debate over the concept of smart cities seems to be growing by the day, with new investments and governments supporting the concept. Smart cities leverage data from connected devices and powerful analytics tools to keep traffic flowing, protect

public safety, reduce pollution, maintain public assets, and improve the delivery of city services. Use cases for smart city technologies range from monitoring the condition of roads and bridges and detecting water levels in flood-prone areas, to optimizing the use of streetlights and enabling citizens to access city services around the clock via mobile devices. However, since the smart city concept depends heavily on data and its processing and analysis, High Performance Computing (HPC) is already playing a key role in helping cities pursue goals of social safety, efficient use of resources, and a better quality of life in general.

This study [2] presents a methodology, namely DA4PT (Data Analytics for Public Transport), for discovering the factors that influence users' behavior in ticket purchasing. In particular, DA4PT allows analyzing large amounts of user-generated event logs from bus ticketing platforms for finding the correlation rules between booking factors and purchase of a ticket. Then, such rules are used: (i) for training machine learning models able to predict whether a user will buy or not a ticket, and (ii) for defining different dynamic pricing strategies with the purpose of increasing ticket sales on the platform and the total revenue. The proposed methodology was applied to a real case study composed by more than 3 million of event logs of an Italian bus ticketing platform, collected from 1 August 2018 to 20 October 2019. The results obtained show that factors such as number of days passed from booking to departure, occupancy rate and fare of tickets, significantly influence travelers in their purchases. We experimentally evaluated the accuracy of our methodology comparing some of the most relevant machine learning algorithms used in the literature. XGBoost and Random Forest proved to be the best classification models in the demand forecasting domain compared to other algorithms, with an accuracy of 95% and 93%, respectively. To evaluate the pricing strategies, we defined an algorithm for generating synthetic event logs similar to the real ones, and an algorithm for evaluating the impact of a pricing strategy in terms of number of purchased tickets and total revenue. We compared the different pricing strategies, showing that a dynamic pricing strategy based on occupancy rate and number of days passed from booking to departure are capable of increasing the number of purchased tickets by 6% and the total revenue by 9%. For the sake of clarity, this paper extends the work presented in [64] including several new original contributions with respect to the previous one. Specifically, the methodology is explained in more detail by analyzing new purchase factors such as the number of attempts made by users before buying a ticket or leaving the platform. A new section on dynamic pricing was added to explain how the synthetic event generator was defined and how the different pricing strategies were evaluated. The experiments section was also extended with new results that show how the number of ticket sales and total revenue increase by using different pricing strategies.

As shown in Figure 3.4, the proposed methodology consists of five main steps:

- **Steps 1-2. Data Collection and Process Mining**. The first two steps aim at extracting relevant information about users behavior, interests and activities by the event logs (*EL* of a bus ticketing platform). Specifically, during step 1, all the interactions of users with the bus ticketing platform (e.g., whether a user buys or not a ticket, or in which step of the buying decision process a user leaves the platform) are collected (Figure 3.5). Step 2 leverages a process mining algorithm to learn and support the design of purchasing processes by automatically discovering patterns that explain the events recorded in the input log traces [65] (Figure 3.6). The set of event logs EL is pre-processed to make it ready for analysis. In particular, we first process events for fixing wrong and missing data. Then, we proceed by selecting only the events that end successfully

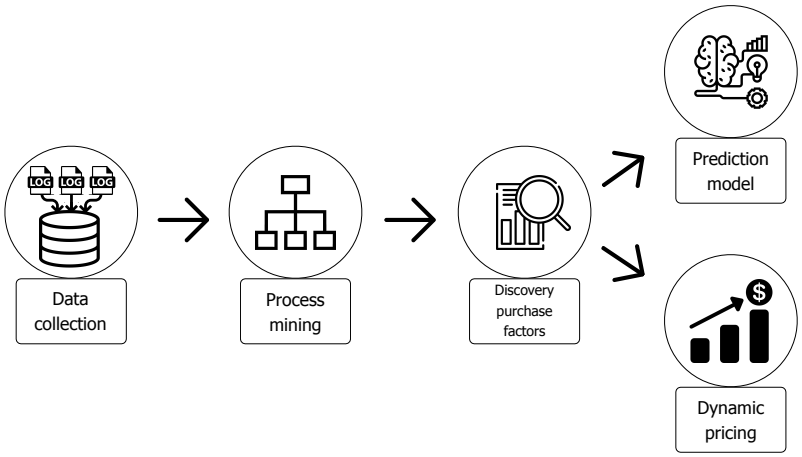


FIGURE 3.4: The steps of the DA4PT methodology.

and unsuccessfully (e.g., the purchase of a ticket and the abandonment of the platform by users, respectively).

COOKIE	ACTION	TIMESTAMP	TRIP ID	DEPARTURE DATE	BOOKING DATE	ORIGIN CITY	DESTINATION CITY	No. SEAT	BUS SEAT	FARE	BOUGHT
1JYASX	list_trips	2018-10-16 11:31:19		2018-10-22		Soverato	Rome				
1JYASX	estimate_ticket	2018-10-16 11:31:37	141772	2018-10-22		Soverato	Rome	1	45	35 €	
1JYASX	choice_seat	2018-10-16 11:36:28	141772	2018-10-22		Soverato	Rome	1	45	35 €	
1JYASX	purchased_ticket	2018-10-16 11:42:20	141772	2018-10-22	2018-10-16	Soverato	Rome	1	45	35 €	YES
28UAKS	list_trips	2019-02-24 18:15:07		2019-02-26		Milan	Lamezia Terme				
28UAKS	estimate_ticket	2019-02-24 18:15:40	408003	2019-02-26		Milan	Lamezia Terme	2	52	64 €	
28UAKS	choice_seat	2019-02-24 18:20:05	408003	2019-02-26		Milan	Lamezia Terme	2	52	64 €	NO

FIGURE 3.5: Example of the log events.

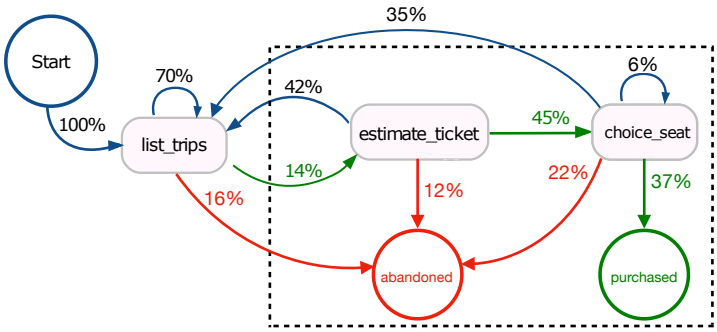


FIGURE 3.6: Example of process mining algorithms applied to user event logs.

- **Step 3. Discovery of Purchase Factors**. The goal of step 3 is to identify the key factors that drive a user to purchase a ticket. Firstly, we perform an exploratory analysis to reduce the set of attributes for keeping only those necessary to model travelers' purchasing behaviors. Then, we apply a correlation analysis for defining conditions that tend to occur simultaneously and/or patterns that recur in certain conditions. In particular, the goal of our analysis is to generate correlation rules such as $f \rightarrow e_{purchased}$ (if factor $f \in F$ occurs, then it is likely that also event $e_{purchased}$ occurs). The correlations between an attribute and the class attribute (*purchased* or *abandoned*) is evaluated using the Pearson's

correlation coefficient [66]. The values of the Pearson's correlation can be in the range [−1,1], where the value of 1 represents a strong linear relationship, 0 no linear correlation, while −1 corresponds to a negative linear correlation. To rule out collinearity issue, we calculate the variance inflation factor (VIF) [67] for each attribute. A VIF value of 1 indicates that there is no correlation between this attribute and any others. VIF values between 1 and 5 suggest that there is a moderate correlation, but it is not severe enough to request corrective measures. VIF values greater than 5 represent critical levels of collinearity which indicate that the correlation coefficients calculated among attributes may be unreliable. From an implementation point of view, the correlation is calculated with an R script, while the collinearity through the *statsmodels*[2] library. Before performing exploratory factor analysis, we added some attributes in addition to those given in the input dataset: (i) days before departure (*DBD*), by calculating the difference between booking and departure date; (ii) booking day of the week (*BDOW*), by extracting the day from a booking date; (iii) occupancy rate for a bus (*OCCR*), by evaluating the number of required bus seats per passenger; (iv) number of attempts (*NA*), by evaluating the number of attempts made by a user before buying a ticket or leaving the platform; (v) fare of a ticket (*HMLF*), by dividing the price of each trip itinerary into three bands (high, medium, and low).

- **Step 4. Prediction Model**. After defining the purchase factors, we train predictive algorithms to automatically learn whether or not a user will finalize the purchase of a ticket. To perform this task and evaluate the accuracy of the relative predictive models, a 10-fold cross-validation process is exploited. However, it should be noted that the training and test sets that are generated in such a process are unbalanced because the two classes, *purchased* and *abandoned*, are not equally represented in the original log. In particular, there is a high percentage of users who visit the bus website without buying any tickets, and a low percentage who instead purchase tickets. To get accurate prediction models and correctly evaluate them, we used balanced training sets and test sets in which half of the log events end with the purchase of a ticket and half with the abandonment of the platform. For this reason, we used the random under-sampling algorithm [68], which balances class distribution through random discarding of major class tuples as described in [1]. Moreover, to make our evaluation more accurate and complete, we evaluated the performance of the proposed approach comparing five classification algorithms (i.e., Naïve Bayes [69], Logistic Regression [70], Decision Tree [71], Random Forest [72], and XGBoost [73]). Specifically, the performance of the algorithms was evaluated through a confusion matrix. Tickets that are correctly predicted as *purchased* are counted as True Positive (TP), whereas tickets that are predicted as purchased but are actually abandoned are counted as False Positive (FP). Similarly, tickets that are correctly predicted as abandoned are counted as True Negative (TN), whereas tickets that are predicted as *abandoned* but are actually *purchased* are counted as False Negative (FN). Then, starting from the confusion matrix, we computed four metrics (accuracy, precision, recall and F1-score) which are commonly used in regressive analysis literature to quantify forecast performance. The machine learning algorithms were implemented in Python using the library *sklearn*[3] for

[2] https://www.statsmodels.org/
[3] https://scikit-learn.org/

producing the confusion matrix and the resulting measures, and its library *imblearn* to deal with the class-imbalance problem. Table3.3 summarizes the results obtained by the five machine learning algorithms we used. Specifically, XGBoost and Random Forest proved to be the best classification models with $R = 0.95$ ($R_p = 0.97$ and $R_a = 0.90$) and $R = 0.93$ ($R_p = 0.95$ and $R_a = 0.85$), respectively. A high value of accuracy is also obtained by Decision Tree with $R = 0.86$ ($R_p = 0.87$ and $R_a = 0.84$), showing good robustness and stability. The same value of accuracy is observed for Naïve Bayes and Logistic Regression ($R = 0.61$), but Naïve Bayes is less accurate on the *purchased* class than *abandoned* class ($R_p = 0.40$ and $R_a = 0.82$).

Algorithms	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Naïve Bayes	0.61	0.64	0.61	0.59
Logistic Regression	0.61	0.61	0.61	0.61
Decision Tree	0.86	0.86	0.86	0.86
Random Forest	0.93	0.93	0.93	0.93
XGBoost	0.95	0.95	0.95	0.95

TABLE 3.3: Performance evaluation.

- **Step 5. Dynamic Pricing**. To evaluate the different pricing strategies (standard and dynamic), we defined an algorithm for generating synthetic event logs similar to the real ones, and an algorithm for evaluating the impact of a pricing strategy in terms of number of purchased tickets and total revenue. Figure 3.7 shows the two algorithms, *event log generator* and *event log processing* and their inputs and outputs. The *event logs generator* algorithm generates synthetic events similar to those registered in original logs file. Specifically, synthetic data is generated by using a multivariate empirical distribution based on input data $\{\{x_1, x_2, ...\}, \{y_1, y_2, ...\}, ...\}$, where x_i are the observed values for the first attribute, y_i are the observed values for second attribute, and so on. The different synthetic log files that we can produce (one for each seed we use) allow us to perform extensive testing on pricing strategies. The *event logs processing* algorithm exploits a pricing strategy and the purchase behaviors of users, to estimate the number of purchased tickets and the total revenue. The pricing strategy sets a price for a ticket based on the user's choices, for example what type of trip a user chose, how many days before he/she booked the ticket, how much is the percentage of occupancy of the bus, on which day of week he/she booked the ticket.

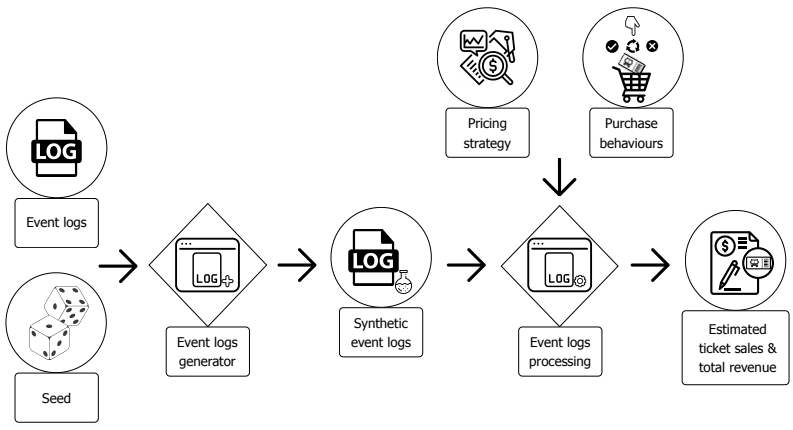


FIGURE 3.7: Evaluation steps of a dynamic pricing strategy.

Figure 3.8 shows the results achieved using the different pricing strategies. Since the number of purchased tickets depends on the number of purchased events and they are directly proportional, we show in the chart only the value of the purchased tickets. With the standard strategy, the number of tickets is practically the same as found in the input log file with a slightly changes in the total revenue (on average -2%). Using the fixed strategies, when the ticket price is always *low*, there is an increase of the number of ticket sales ($+5\%$) but a decrease of the total revenue (-9%). When the price is *medium*, the average number of purchased tickets goes down (-3%) but the revenue rises slightly ($+1\%$), whereas when the price is always *high* the number of ticket sales goes down (-5%) but the total revenue increases ($+3\%$). With dynamic strategies, we are able to increase both the number of purchased tickets and the total revenue. With the strategy that considers only the days before departure (*Dyn-Day*), we can increase the tickets sold by $+4\%$ and the total revenue by $+6\%$. Even considering only the bus occupancy rate (*Dyn-Occ*), we obtain similar results with 4% increase in the number of tickets and 5% of total revenue. With the mixed strategy (*Dyn-Mix*), we are able to have the maximum increase in the number of purchased tickets ($+6\%$) and total revenue ($+9\%$). This is due to a good sales strategy that sell the largest number of tickets at variable prices in order to try to maximize revenues. These experimental results show that dynamic pricing strategies allow bus companies to sell a greater number of tickets (thus occupying more seats inside the bus) and also to increase revenues by intercepting user demand.

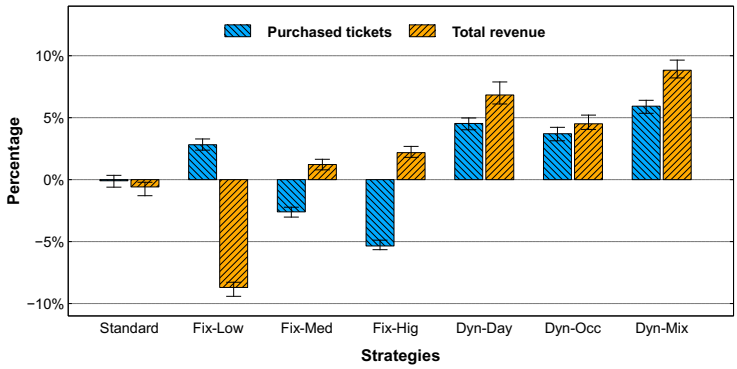


FIGURE 3.8: Comparative analysis among pricing strategies, evaluating the number of purchased tickets and the relative revenue.

Our methodology can be used for analyzing the buying behavior of large communities of people and providing valuable information and high-quality knowledge that are fundamental for the growth of business and organization systems. Moreover, the use of dynamic pricing strategies allows companies to sell a greater number of goods and things and also increase revenues by intercepting user demand.

In future work, additional research issues will be investigated. For example, our prediction model can be extended considering also external factors that certainly influence users in purchasing tickets, such as the strategies and prices offered by competitors (train, airline and other bus companies). In addition, we can use our methodology in other sales areas by demonstrating how the use of advanced forecasting techniques together with dynamic pricing strategies can increase tickets purchased and company revenues.

3.2 Social Network Analysis

3.2.1 Using social media for sub-event detection during disasters

A huge amount of user-generated data in social networks can be exploited to extract valuable information about human dynamics and behaviors. Social data analytics is emerging as a rapidly growing area of research that aims to extract useful information from this mass of data. It can be used for the analysis of collective feelings, to understand the behavior of groups of people or the dynamics of public opinion.

One of the most widely used data sources comes from microblogging services such as Twitter, Facebook and Instagram. Data elements contained in social media posts are often unstructured and require advanced analysis in order to extract useful knowledge. In the context of natural disasters, the very large use of social media platforms has enabled eyewitnesses and other disaster-affected people to share information about their damages, risks and emergencies in real time. Research studies show the importance and usefulness of the information shared during disasters, both through traditional infrastructures [74] and social media [75, 76].

Despite these advantages, the use of social media posts to help rescue and intervention activities remains an open challenge as users often publish posts containing inaccurate information, slang or abbreviated words, or without using geolocalization. While extensive research work has been done on the classification of posts to understand their high-level informational categories [75], little focus has been given to understand and extract small-scale events that affect small communities. In fact, every disaster creates a series of small-scale emergencies (*sub-events*), such as family members stranded, power outage, damage to buildings, school closure, or damage to bridges. Normally, these sub-events affect only a small portion of the population in the disaster area and thus receive less attention and delayed response. Among other causes, the lack of information about these events causes a slow response from the authorities, especially during an ongoing disaster.

In [77], we aim at identifying small-scale events that occurred after a natural disaster or catastrophic event. For this purpose, we present a new method, namely *SEDOM-DD* (*Sub-Events Detection on sOcial Media During Disasters*), for detecting sub-events during disasters from social media data. Specifically, the proposed method addresses two important issues: understanding whether a post is relevant about a disaster and discovering the sub-events that occurred in the disaster area. SEDOM-DD performs these tasks in four main steps: (*i*) collection of posts that are potentially related to the disaster; (*ii*) filtering of posts to keep only the relevant ones; (*iii*) data enrichment by using information contained in posts to increase the number of posts for which it is possible to estimate their geolocalization; and (*iv*) use of clustering techniques on geotagged relevant posts for detecting sub-events.

To identify sub-events during a disaster, the proposed method mainly relies on four important steps. Figure 3.9 shows these steps together with their inputs and outputs:

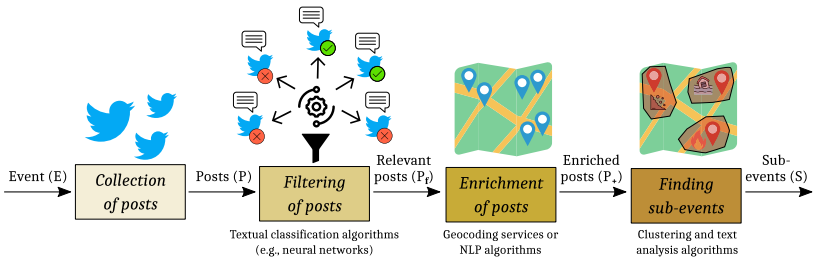


FIGURE 3.9: Execution flow of SEDOM-DD.

1. **Data collection**. Given a disaster event and its impact areas, all the posts generated in the event's area are collected. These posts can be collected from social media platforms (e.g., Twitter) through queries based on keywords or locations.

2. **Filtering of posts**. During this step, posts collected from social mediam are processed and filtered for keeping only the ones that are *relevant* for the analysis. A post is *relevant* when it contains text concerning a catastrophic event (e.g., earthquake, flooding) that happened in the area under analysis. *Relevant* posts can be further divided into two categories: (i) *generic*, which generically refer to a catastrophic event, without mentioning any specific sub-event (e.g., *yesterday there was an earthquake, we were very scared!*); (ii) *not-generic*, which explicitly refer to problems/sub-events that occurred as secondary effects of a catastrophic event (e.g., *we have been without electricity since yesterday*). We are mainly interested in relevant posts and, in particular, non-generic ones that mention some sub-events that have occurred. It is evident that the classification of posts is a crucial step for obtaining accurate sub-event results.

3. **Enrichment of posts**. The proposed method uses geotagged posts to identify the areas where the sub-events occurred. The main problem with posts from social media is that they are not always geotagged, which makes them not always useful for the analysis. The data enrichment step aims at estimating the coordinates of relevant but not geotagged posts through the analysis of the text. In this way, it is possible to increase the volume of geotagged data to be analyzed, which should lead to better accuracy in the identification of sub-events. Posts that are not geotagged can include textual information that allows to estimate their geographical coordinates. For instance, users often report in the text the name of the street or the district where the event occurred (e.g., *Washington Street in Cork closed to traffic following the partial collapse of a building*). Several studies have proposed techniques for geotagging posts exploiting the textual information they contain [78, 79]. In addition, different geocoding services, such as Google Map[4] or Nominatim[5], can be exploited for converting an address, even partial, into coordinates. In some cases, natural language processing techniques, e.g., based on CoreNLP [6], can also be used for identifying the locations mentioned in the text of a post. Our method uses the following approach for estimating the coordinates of a post. Given a geographical area to be analyzed, we exploit geocoding web services for retrieving Points-of-Interest (PoIs) in the area and the most common names used to refer to them. Then, we extract street and district names from a text through textual

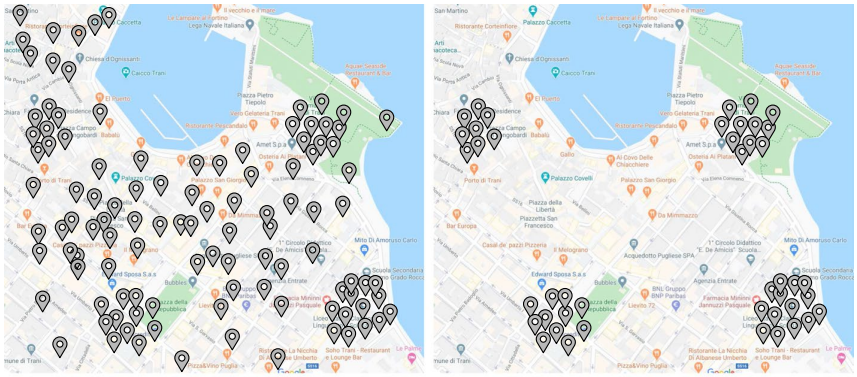
[4] https://developers.google.com/maps
[5] https://nominatim.openstreetmap.org
[6] https://stanfordnlp.github.io/CoreNLP/

patterns. Once we have identified this information in the text, we translate it into coordinates with four levels of accuracy: PoI, street, district, or city. For example, a post that refers to a PoI is geotagged with the coordinates of such PoI. While a post that refers to a street (without a house number) or district is assigned to a point randomly chosen inside the street/district where the sub-event occurred. A post about a catastrophic event that cannot be associated with a specific point or area is placed at the city level.

4. **Finding sub-events**. Following the same approach proposed in [80], we used a clustering algorithm to aggregate the posts that refer to the same sub-event and discover the area where it occurred. In particular, DBSCAN [81] has been chosen for its ability to detect clusters with different sizes and shapes, tolerate noise, and be applicable on small or large data sets. Moreover, in the context of the extraction of areas or regions-of-interest, it is one of the most used algorithms in the literature [82]. For each cluster identified by DBSCAN, a procedure is carried out for identifying the sub-event that occurred in the cluster's area. In particular, we extract the keywords (and their frequency) contained in the posts from such a cluster. The keywords are then sorted by frequency. A high frequency does not necessarily denote high representative keywords, but it is a useful starting point. As an example, the keyword *earthquake* may have a higher frequency than *building collapse*, although *building collapse* is evidently more representative as a sub-event that occurred in an area. The most representative keywords are then compared with a manually trained dictionary, which contains a list of terms that are commonly used to report specific sub-events that occurred. The dictionary associates a term, representing a type of sub-event, with some synonyms. As an example, for the sub-event *collapsed house* we also consider a list of similar terms, such as *destroyed house*, *house collapse*, and *unsafe house*. As stated in [83], the terms used to report a sub-event in the text are usually composed of a pair $\langle$ subject entity, action happened$\rangle$, such as *bridge collapsed* and *power outage*.

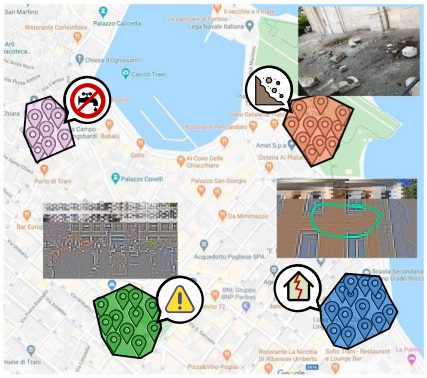
Figure 3.10 shows an example of how our method works. Specifically, it was built starting from an earthquake that really happened. On May 21, 2019, the province of Barletta-Andria-Trani in Apulia (Italy) was affected by an earthquake of magnitude 3.9 having an epicenter at 34 km of depth 4 km from the city of Barletta. After the shock warning in several municipalities across the province, many public institutions had to be evacuated, including schools, judicial offices and other facilities. The discomforts have also spread on public transport, in fact, many railway lines have been interrupted for a few hours, in order to guarantee passengers safety. The old town of the city of Trani turned out to be one of the most affected areas.

During these panic hours, we collected posts from social media focusing on the catastrophic event that occurred in the area (see Figure 3.10a). Starting from such posts, those that do not explicitly refer to any sub-event have been filtered out (Figure 3.10b). Posts regarding sub-events have been clustered and their text analyzed to understand the type of sub-event that occurred in each cluster's area (Figure 3.10c). In particular, Figure 3.10c highlights four significant sub-events that happened in the old town of Trani: the fall of material from the church of San Domenico, a structural problem with the Liceo De Santis, a water outage in the St. Mary district, and people evacuated from the judicial offices. More details on the algorithms used in the different steps of our method are provided below.



(a) Relevant posts.

(b) Filtering of posts related to sub-events.



(c) Sub-events detection.

FIGURE 3.10: An example of using SEDOM-DD on posts collected during the earthquake in the old town of Trani (May 21, 2019).

Figure 3.11, on the other hand, shows some significant sub-events that were discovered. In particular, two large areas with a high density of sub-events (red areas) have been discovered in the cities of Houston and Rockport. Other areas, smaller and with a lower density of sub-events (blue-green areas), have been identified elsewhere. Table 3.4 reports the sub-events that have been identified in the main cities involved in the disaster. Notably, Houston was found to be the city with the highest number of sub-events that occurred after the passing of Harvey, including flooded houses and damages to toxic waste sites. Also Rockport reported a high number of sub-events, such as collapsed houses, power lines downs, and damaged boats. The obtained results confirm that SEDOM-DD is able to discover a high number of sub-events that occurred after a large-scale natural disaster.

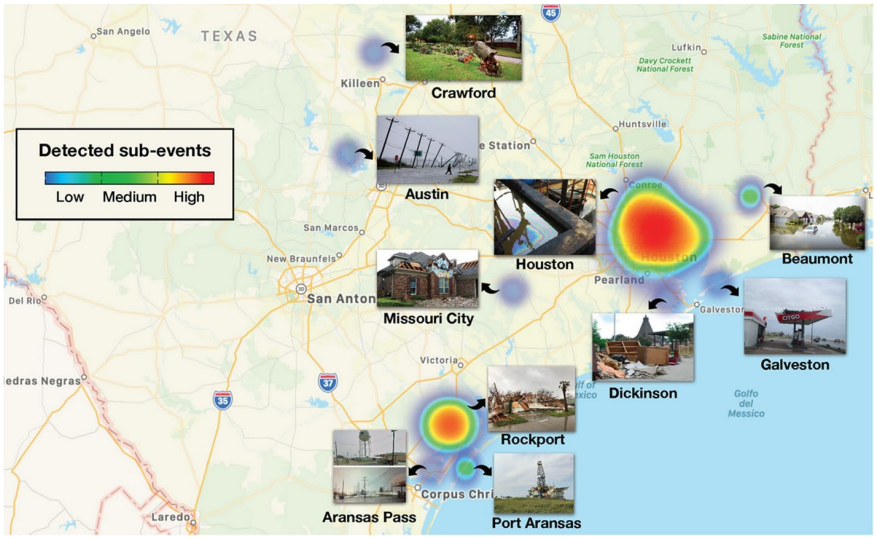


FIGURE 3.11: Sub-events detected by SEDOM-DD using tweets collected after Hurricane Harvey.

City	Types of sub-events
Houston	Flooded houses, airports runways and highways, damaged toxic waste sites and electrical station, destroyed cars.
Rockport	Damaged boat storage, collapsed houses, power line down.
Beaumont	Flooded houses, damaged oil refineries.
Port Aransas	Collapsed houses, damaged ferries and vehicles, power line down.
Austin	Power outage.
Crawford	Downed trees, collapsed houses.
Dickinson	Flooded houses and roads, destroyed churches.
Missouri City	Roofless houses, big trees down.
Aransas Pass	Water service down.
Galveston	Damaged gas station.

TABLE 3.4: Main sub-events detected in tweets about Harvey.

The widespread use of social media allows people who are victims of disasters (e.g., earthquakes, fires) to share real time information about damages, problems, and sub-events that can take place at different locations after a disaster (e.g, collapsed buildings, broken gas pipes). This valuable information is known only to people located where the events occurred and can be shared with rescue teams and authorities that are far away from the area. In this paper we presented SEDOM-DD, a new method that combines text mining and clustering analysis for discovering critical sub-events from social media data during natural disasters.

Several experiments have been carried out on both real and synthetic datasets for evaluating the performance of SEDOM-DD. In particular, an analysis of a large dataset containing real tweets about Hurricane Harvey showed that SEDOM-DD was able to discover a large number of sub-events that occurred after the disaster. Moreover, other experiments on synthetic datasets demonstrated that SEDOM-DD is able to identify sub-events with a very good F1-score (greater than 85%), which confirms the high accuracy and effectiveness of the proposed approach.

For this reason, SEDOM-DD can be integrated with existing systems for coordinating and enhancing emergency response. The detected sub-events, together with the posts and photos that made it possible to detect such events, can be analyzed and validated by a group of experts to establish the type and the priority of interventions to be carried out.

3.2.2 Analyzing voter behavior on social media during the 2020 US presidential election campaign

In recent years, the growing use of social media during political events can be exploited to extract valuable information about human dynamics and behaviors. For example, the textual content of a post may contain information about the discussion topic [84], the sentiment of the user who wrote the posts [85], the place where the post was written [86] and user opinion on a certain argument [87]. To obtain this information, advanced machine learning techniques, such as Natural Language Processing (NLP), neural networks and deep learning techniques, must be exploited [88, 89].

This work [90] presents an in-depth analysis of the posts published on Twitter during the 2020 US election campaign, aiming at outlining an accurate view of the political event from different points of view. Specifically, several techniques of topic discovery, opinion mining, and emotion analysis were combined in a unified data analysis workflow for investigating: (i) trending topics and their evolution over time, (ii) users' political alignment and publishing behavior, and (iii) users' sentiment and emotional aspects.

Firstly, we extracted the main discussion topics characterizing the 2020 US election campaign by leveraging the unsupervised approach proposed in [91], which relies on the density-based clustering of the latent representation of trending hashtags. Afterward, in order to achieve a more accurate representation of social media conversation, we studied the weekly evolution of the detected topics, which is useful to understand how online discussion evolves over time.

Secondly, we modeled the political alignment of social media users, in order to understand which candidate or party public opinion is most in favor of in the weeks preceding the Election Day. For this purpose, we exploited IOM-NN (Iterative Opinion Mining using Neural Networks), a neural-based opinion mining methodology we previously proposed in [87]. Specifically, a real-time analysis was carried out during the 2020 US presidential election campaign using data gathered from Twitter, correctly determining Joe Biden's lead over Donald Trump before the Election Day.

Thirdly, we analyzed the relationship between the emotional sphere of Twitter users and their political alignment. In particular, we jointly exploited sentiment analysis and text mining techniques for extracting the sentiment of social media users. Then, we combined this information with the polarization achieved by IOM-NN for investigating how a user refers to the candidates while supporting his/her preferred faction, with respect to a broad spectrum of emotions. This step is useful for understanding how the supporters of a particular candidate express their preference on social media. Specifically, they can praise, as in the case of pro-Trump users, their favorite candidate with positive content that shows emotions like joy and confidence. Alternatively, they may be more likely to discredit the opposing candidate, as in the case of pro-Biden users, by producing negative online content characterized by emotions like anger, disgust and sadness.

Several techniques were combined in a unified analysis workflow, represented in Figure 3.12, composed of the following steps:

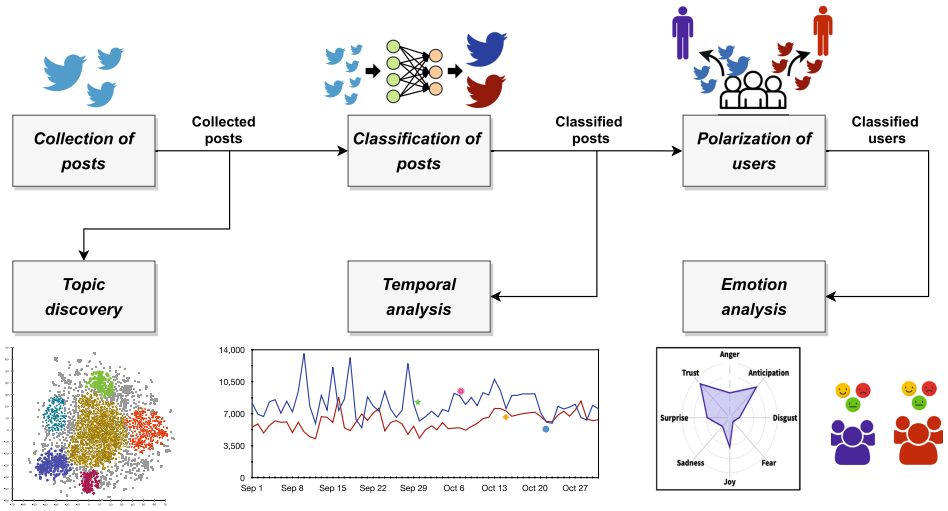


FIGURE 3.12: A graphic representation of our analysis workflow.

1. **Collection of posts**. The goal of this step is to collect a set $\mathcal{P}$ of social media posts from different sources (e.g., Twitter), related to the political event $\mathcal{E}$ under analysis. As a first step, the different factions, parties or candidates involved political event are identified, defined as the set $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$. In particular, in the case of 2020 US presidential election, we focused on the two main candidates Joe Biden and Donald Trump. Afterwards, geotagged posts are gathered by using a set of keywords $\mathcal{K}$ that is partitioned as follows:

   - *neutral keywords* ($\mathcal{K}_{context}$) that contains generic keywords that can be associated to $\mathcal{P}$ without referring to any specific faction in $\mathcal{F}$(e.g., #*vote*, #*election*2020);
   - *faction keywords* ($\mathcal{K}^{\oplus}_{\mathcal{F}} = \mathcal{K}^{\oplus}_{f_1}, \ldots, \mathcal{K}^{\oplus}_{f_n}$) that contains the keywords used for supporting each faction (e.g., #*votebiden*, #*maga*).

   The keywords selection process requires a small amount of domain knowledge, as these keywords can be manually selected among the trending hashtags that people commonly use to refer to $\mathcal{E}$ on social media. Moreover, the keyword selection process can be automatized by searching for specific patterns, like #vote + *candidate*, often used for labeling politically polarized post. In order to improve the representativeness of the collected posts, users' account are analyzed, filtering out content posted by users that show an anomalous publishing activity, such as social bots or news sites, or inconsistent profile information. Collected posts undergo the following preprocessing operations: (i) the text of each post is converted to lowercase and accented characters are normalized; (ii) words are lemmatized and stemmed (e.g., vote or votes or voted → vot); (iii) stopwords are removed; and (iv) bigrams are identified (e.g., San Francisco → San_Francisco). Figure 3.13 shows an example of how posts are collected using keywords about the 2020 US presidential election. Some of these keywords are generic (e.g., #*vpdebate*2020), and others are used to support a specific candidate (e.g., #*voteblue* for Biden and #*trump*2020 for Trump).

2. **Classification of posts and temporal analysis**. During this step, the posts $\mathcal{P}$ collected in the previous step are classified in favor of a faction using an incremental procedure implemented through neural networks, as described in [87].

FIGURE 3.13: Example of how the *collection of posts* step works.

Specifically, IOM-NN performs a preliminary iteration for classifying input posts according to the keywords in $\mathcal{K}^{\oplus}_{\mathcal{F}}$. Posts containing keywords related to exactly one faction are polarized towards that faction, while remaining posts are labeled as neutral. Then, neutral posts undergo an iterative classification process, during which the model exploits the knowledge acquired at the previous iterations. It is worth noticing that, due to the incremental nature of the annotation process, IOM-NN is not tied to a specific set of initial faction keywords and does not require an in-depth knowledge of the political event under consideration. In fact, even starting from a small but representative set of faction keywords, IOM-NN is able to infer new classification rules iteratively, which implies a good robustness and generalizability of the methodology. Figure 3.14 shows a classification example of a small set of tweets about the 2020 US presidential election, which exploits the following faction keywords.

- $\mathcal{K}^{\oplus}_{Biden} = \{\#voteblue, \#backtheblue, \#votebiden, \dots\}$;
- $\mathcal{K}^{\oplus}_{Trump} = \{\#votered, \#trump2020, \#maga, \dots\}$.

At iteration 0, IOM-NN uses the keywords in $\mathcal{K}^{\oplus}_{\mathcal{F}}$ for classifying 5 tweets. In the subsequent iterations, the neural model iteratively exploits the tweets classified in the previous steps for generating new classification rules based on co-hashtag relationships. As an example, at iteration 1 the model is trained with the tweet classified at iteration 0, discovering new political-oriented topics of discussion and generating the following classification rules:

- tweets with keywords *#bountygate* are classified in favor of Biden since Donald Trump was accused of paying Moscow's secret agents for the killing of US servicemen in Afghanistan;
- tweets with keywords *#crookedbiden* are classified in favor of Trump since Hunter Biden (i.e., second son of US President Joe Biden), was accused by Donald Trump of wrongdoing in regards to China and Ukraine.

This learning process iterates until the algorithm is no longer able to generate new classification rules and therefore to identify the polarization of new tweets.

3. **Polarization of users**. This step is aimed at analyzing the set of previously classified posts in order to determine the *polarization of users* towards a faction. Specifically, the list of classified posts for each user u is computed, filtering out

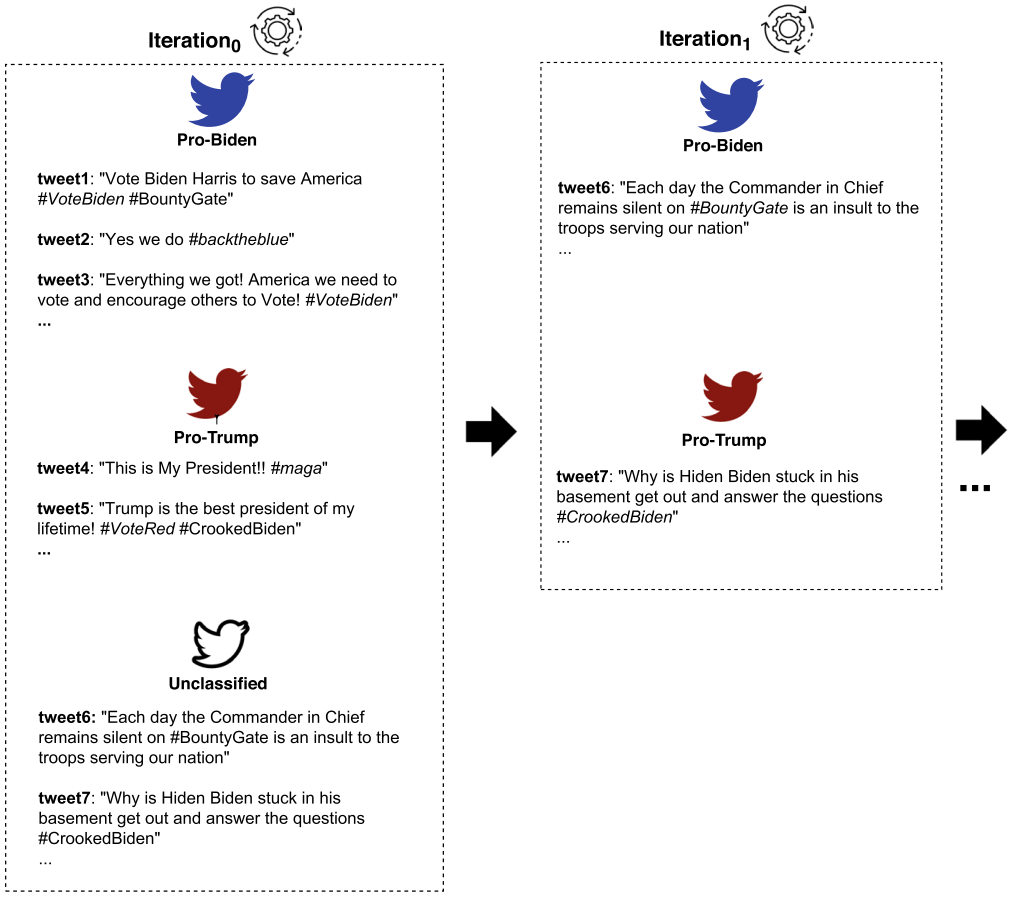


FIGURE 3.14: Example of how the *classification of posts* step works.

those users that published a number of posts below a given threshold. Afterwards, a score vector v_s^u for each user u is computed, which contains his/her score for each faction. Finally, IOM-NN calculates the overall faction score $\mathcal{S}$ as the normalized sum of the score vectors. Figure 3.15 shows how the *polarization of users* step works on the classified posts reported in Figure 3.14.

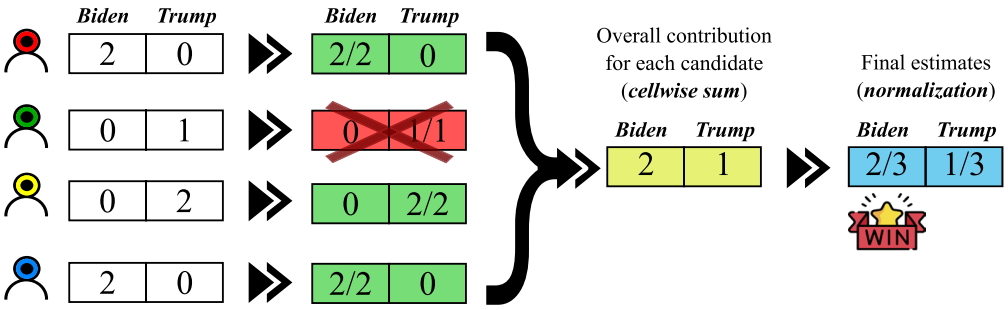


FIGURE 3.15: Example of how the *polarization of users* step works.

For each user, the posts in favor of Biden and Trump are counted, discarding those users who have published less than two tweets. Then, the final vector $\mathcal{S}$ is computed, that contains the overall polarization percentages for the two candidates.

4. **Emotion analysis**. This step analyzes the polarized posts for identifying the users' sentiment underlying the online discussion about the presidential candidates. Firstly, we exploited SentiStrength [92] to annotate social media data. In particular, for each polarized post we computed a positive $\mathcal{S}c^+(p)$ and negative $\mathcal{S}c^-(p)$ sentiment score, both ranging between 1 (neutral) and 5 (strongly positive/negative). Then, the overall sentiment score $\mathcal{S}c(p)$ of a polarized post is obtained as follows: $\mathcal{S}c(p) = \mathcal{S}c^+(p) - \mathcal{S}c^-(p)$. Secondly, we modeled the political orientation of social media users from an emotional point of view by exploiting NRC-EmoLex [93], a publicly available emotion lexicon which has proven its performance in several sentiment and emotion classification tasks, as described in [94], [95], and [96]. Specifically, NRC contains more than 14 thousand English terms labeled by the expressed polarity (i.e., positive or negative) and eight basic emotion categories of Plutchik [97] (i.e., joy, trust, anticipation, sadness, surprise, disgust, fear or anger). As an example, Figures 3.16 and 3.17 describe the sentiment and the emotional state of the tweets with the relative intensity of the tweets produced by Trump and Biden supporters, respectively. What appears evident is that, on average, the tweets produced by Trump's supporters are significantly more positive than those produced by Biden's supporters, which devote a significant number of negative tweets to their opponent. For what concerns the detected emotions, Trump's supporters express joy and confidence about Trump, while fear about Biden's election. Biden's supporters, instead, show trust and anticipation in having Biden as future president of the USA, with a more marked presence of negative emotions about Trump, like anger, disgust and sadness.

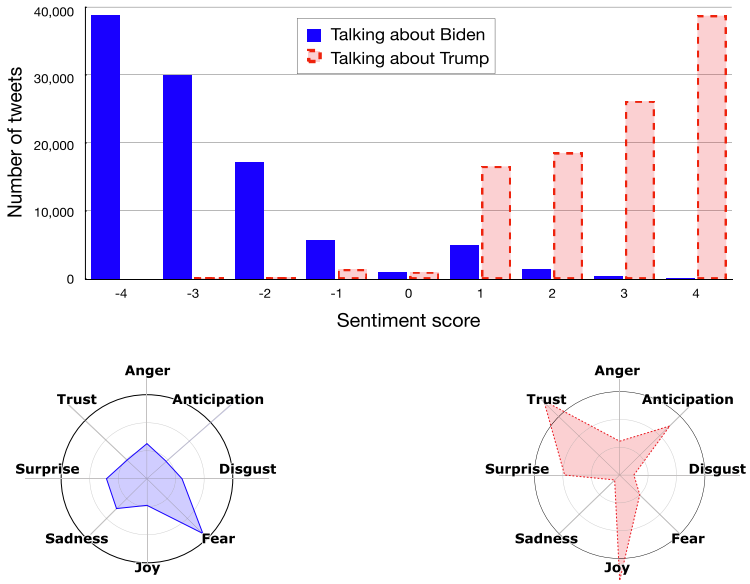


FIGURE 3.16: Distribution of sentiments and emotions of pro-Trump tweets

The results of the experimental evaluation are summarized in Table 3.5, which shows that IOM-NN was able to correctly identify the winning candidate in 10 out of 11 cases with an average accuracy of 91%, outperforming the opinion polls that correctly classifies 9 out 11 states with an average accuracy of 82%. Using this metric we penalize the inversions of predicted polarity which can be a crucial issue while

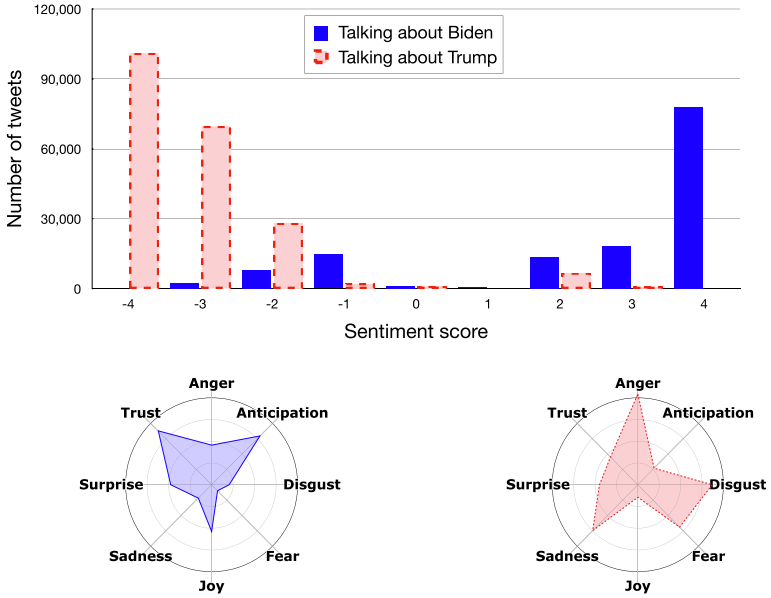


FIGURE 3.17: Distribution of sentiments and emotions of pro-Biden tweets

analyzing these kinds of states, since they are characterized by a high degree of uncertainty. Notice that the result of North Carolina that was not correctly predicted by our analysis was not correctly predicted even by opinion polls. Moreover, a noteworthy advantage of IOM-NN is the ability to capture the opinion of a larger number of people more quickly and at a lower cost, which makes our approach a valid alternative to traditional opinion polls.

State	Real percentages		Opinion polls		IOM-NN	
	B	***T***	***B***	***T***	***B***	***T***
Arizona	49.4	49.1	**48.0**	45.8	**50.2**	48.3
Florida	47.9	51.2	48.7	46.0	48.0	**51.1**
Georgia	49.5	49.2	**47.6**	47.4	**52.7**	46.0
Michigan	50.6	47.8	**49.9**	44.4	**55.4**	43.0
Minnesota	52.4	45.3	**51.6**	41.8	**55.1**	42.6
Nevada	50.1	47.7	**49.4**	44.4	**49.8**	48.0
New Hampshire	52.7	45.4	**53.4**	42.4	**50.9**	47.3
North Carolina	48.6	49.9	47.8	47.5	56.6	41.9
Pennsylvania	50.0	48.8	**49.4**	45.7	**55.7**	43.1
Texas	46.5	52.1	47.5	**48.8**	46.1	**52.5**
Wisconsin	49.4	48.8	**52.0**	42.8	**56.3**	41.9
Correctly classified	-		**9/11**		**10/11**	
Tweets	-		-		670,451	
Users	-		$\approx$ 11,000		57,116	
Avg. Acc	-		0.82		0.91	

TABLE 3.5: Voting percentages estimates of the 2020 US presidential election.

As future work, we will apply the presented analysis workflow to other scenarios, such as product adoption analysis and reputation evaluation of companies. In fact, it can be easily generalized to different use cases, as it is not tied to any specific application domain, and only relies on the representativeness of the analyzed posts. Moreover, we can integrate other techniques in our workflow, introducing new steps aimed at improving the quality of the achieved results. As an example, a hashtag

recommendation model can be used for enriching the information content of the analyzed data, since keyword-based approaches like IOM-NN are strongly dependent on the availability of consistent hashtags in social media posts.

3.3 Infectious Disease Surveillance

3.3.1 Challenges and Perspectives of Open Data in Modelling Infectious Diseases

The COVID-19 pandemic challenged the scientific community and governments around the world, who were looking for real-time answers but lacked the data or evidence to guide decision-making. This inevitably led to diverse and fragmented policies and responses. In this context, a positive note was the need, recognized by all, for an acceleration of several innovations that had already been introduced but had not yet taken hold: from decentralized clinical trials and digitization, to new ways of leveraging real-world data (RWD). However, RWDs is only the starting point: once the raw data is collected, it is necessary to structure all the information, thus obtaining a Real-World Evidence (RWE), this latter defined in 2016 by the Food and Drug Administration (FDA) as "the clinical evidence relating to the use and potential benefits, or risks of a medical product derived from the analysis of Real World Data" [98]. The challenges associated with data analysis affect the entire life cycle of an epidemic. In the early stages, there is a need to gain fundamental knowledge about the epidemiological characteristics of a new infection, from transmission potential to natural history. This requires rapid scaling up of testing and sequencing, rapid assessment of clinical impact, and open sharing of early results. As outbreaks grow, there is a need to predict disease dynamics, estimate potential burden, and evaluate interventions. Coordination and equitable data generation are critical to ensure that recommendations and responses are relevant to all demographic groups and affected settings. In the next steps, attention turns to estimating vaccine efficacy and monitoring outbreaks and evolutionary dynamics. This may include analysis of long-term data sets, including cohort studies, which can provide insights into epidemic processes over months or even years. Several lessons have been learned from the COVID-19 pandemic:

1. **Shareability and standardisation**. Effective response to outbreaks depends on rapid and reliable data sharing, especially given the global nature of infectious disease threats. From monitoring COVID-19 variants to analyzing measles outbreaks and selecting influenza vaccine strains, efficient and secure data sharing enables response based on the best possible evidence. It can also support rapid follow-up analysis that builds on previous work, reducing repetition. However, there can be challenges in ensuring rapid analysis and sharing while maintaining adequate quality control. Community-based platforms for data sharing and analysis, such as the COVID-19 prediction hubs coordinated by CDC and ECDC[7] offer a way to balance these considerations. In addition to sharing specific data, the ability to link different data streams, particularly combinations that provide orthogonal information about the epidemic, has significant advantages. For example, with each new variant of SARS-CoV-2 that has emerged, scientists and researchers from different specialties have been able to generate unprecedented data on its potential impact. However, obstacles remain to both collecting such data and sharing them ethically; detailed analysis can often involve potentially

[7] https://covid19forecasthub.org/

sensitive personal information, such as linking one or more pathogen genetic sequences, demographic metadata, human genomics, clinical outcomes, and data from wearable devices [99]. To protect the privacy of sensitive clinical data, platforms such as OpenSAFELY [100] have developed pipelines that allow researchers to ask questions of electronic health record datasets without viewing the underlying raw data. Meanwhile, projects such as Google Community Mobility Reports and Facebook Disease Prevention Maps have generated mobility and behavior data products that respect privacy and can be used more widely by disease researchers[8]. As the routine use of large-scale epidemic datasets increases, clear international ethical standards will be needed, particularly when information crosses borders or public-private partnerships. Such efforts could draw on the existing work of organizations such as the Ada Lovelace Institute and the Open Data Institute, with coordination from relevant regional and international bodies. In developing the COVID-19 digital certification of vaccination, for example, the European Commission has overseen standards within the continent, but integration with other countries remains a critical challenge. More generally, countries must proactively plan how to balance the legal aspects of privacy and public health during outbreaks. For example, General Data Protection Regulation (GDPR) concerns in Denmark prevented the sharing of SARS-CoV-2 sequences internationally for nearly two months in early 2021 until a new law was passed[9]. In contrast, South Korea changed some privacy laws after the 2015 MERS outbreak to accelerate data sharing in case of future infectious disease emergencies [101]. To build public trust, these initiatives will need to be supported by transparent communication from governments and health agencies. It would also be helpful to incorporate privacy and health issues into routine education programs. Although linked data can be shared and analyzed within a country, integration across geographic areas and populations may be limited. For example, mobility datasets may not cross national boundaries, serological studies and tests may not be standardized across laboratories and territories [102], and sample selection for sequencing may vary across countries [103]. These problems can be exacerbated by regional inequalities in data generation, from analysis capacity to smartphone coverage, leading to systematic biases in common data sets. As a first step, ensuring that existing national and international data portals provide ways to record why and how data are collected, as well as any known biases, would mean that subsequent analyses can seek to control for these effects [104]. In the long term, making sampling, standardization, and sharing strategies transparent will enable better integration of available information. In real time, data generation activities will often need to be adaptive and decentralized, but there is a need for health agencies to have scientific capabilities to ensure interoperability across scales. In addition, enabling more equitable access to and generation of data, ideally through long-term sustainable investments to develop local resources rather than external funding for specific projects, will reduce uneven coverage and improve the reliability and generalizability of results.

2. **Sustainability and scalability**. In addition to being shareable, data analysis must also be sustainable. The volume and diversity of data generated during the COVID-19 pandemic was unprecedented. To process and analyze these datasets,

[8] https://dataforgood.fb.com/tools/disease-prevention-maps/
[9] https://en.ssi.dk/news/news/2021/ssi-is-once-again-sharing-virus-sequences

scientists built new tools and found ways to improve and upgrade existing software to handle previously unimaginable large amounts of data. The urgency of the pandemic has spurred a proliferation of creations and improvements, but typically few of these tools and resources have long-term funding, making sustainability difficult. For example, several key resources such as the COVID Tracking Project in the United States[10] were created from scratch by committed volunteers, but often on a short-term basis. In contrast, successful open source data and software require ongoing maintenance to respond quickly to the changing needs of their user communities. This problem is far from new: the cycle of upgrading tools to replace those that are no longer maintained, often linked to precarious academic positions and lack of funding, has been documented long before 2020 [105]. However, the COVID-19 pandemic has highlighted the need for a broader view of resource development and maintenance. In the future, integrated programs linking partners such as universities, government departments, research funders, health organizations, and private sector groups could enable efficient coordination of analysis development and clear accountability for maintenance and implementation. These programs could build on existing structures, such as the WHO collaboration centers and the aforementioned foresight hubs; these structures provide academic groups with access to data and partnerships, and governments and health agencies with access to scientific knowledge. Such mechanisms would also ensure that policymakers have access to robust consensus estimates from multiple research groups with an established track record, rather than relying on ad hoc analyses from reactively convened groups. Defining clear accountability for upstream tasks, such as data generation and methodological development, as well as downstream implementation and communication, would help ensure that key tasks are not neglected during an outbreak. This would mean that countries are in a better position not only for the next pandemic, but also for the general growth in demand for real-time epidemic analysis that data advances are bringing. Countries will also need to consider the capacity of data beyond their borders, rather than waiting for a threat to arrive domestically to obtain vital information. The COVID-19 pandemic has reminded us how interconnected we are. What threatens some of us may soon threaten all of us, from the initial emergence and spread of SARS-CoV-2, to recent worrisome variants that have spread worldwide despite strict global travel restrictions. In addition to the spread of infections, we have witnessed the spread of information; rarely have societies benefited so clearly and immediately from globally shared information. From assessing the effectiveness of interventions to monitoring the evolution of pathogens, the accumulation of information from around the world is of tremendous value. It also means having the capacity to generate these data. While many countries have dramatically increased their public health and data activities in 2020/21, some have been able to draw into resources built for analysis of other infectious diseases; the Democratic Republic of Congo and South Africa have adapted infrastructures developed for Ebola and HIV to generate some of the first and largest sequences in Africa, respectively [106].

In the coming years [107], open data will be crucial for (i) conducting real-time situational analysis [108], implementing predictive models that can provide effective and timely responses for the effective containment of a disease; (ii) estimating key epidemiological parameters, such as incubation period, reproduction numbers, etc.;

[10] https://covidtracking.com/

(iii) ensuring community trust in the government through increased transparency and better communication; and (iv) countering misinformation.Importantly, open data are not only valuable for improving drug development and enabling the confirmation of efficacy and safety of post-marketing products, but are necessary for the development of effective policies and guidelines. For example, the use of masks is a good example to demonstrate this. Without clear, supportive, transparent and open data that can show the effectiveness of face masks in preventing airborne disease transmission, the result can be non-acceptance, as we witnessed at the beginning of the pandemic, and which still persists in some geographical areas and within some groups of people. The intent to create collaborative data spaces such as those defined in the context of the European data strategy[11] and the American GovLab Data Collaboratives[12], where any data, not only open, flow seamlessly between the public- and private-sector entities, makes clear the need for a focused effort to increase collaboration and invest in our collective capacity to identify and understand public health risks in order to be better prepared for future pandemics and epidemics. Promoting transparency and data sharing is more important than ever in a global context where openness, reliability, and trustworthiness of data will be critical to accelerating the global response to health emergencies. The value of data to society is at a critical juncture, and its utility will improve the ability of public health institutions to synthesize contextual information for risk assessment and decision making.

In the following sections, some contributions will be discussed where we try to explain the value of open data in an epidemiological context and how its usefulness will improve the ability of public health institutions to synthesize contextual information for risk assessment and decision making.

3.3.2 Building Surveillance Systems with a Special Focus on European and Italian Epidemic

Detailed epidemiological data are critical for making robust and reliable inferences about the spread of infectious diseases. Especially in the early stages of epidemics, detailed data are needed to estimate basic disease properties and evaluate disease surveillance systems (e.g., reporting delays), contributing to understanding the feasibility of controlling an epidemic in response to specific (initially nonpharmacological) interventions. To support global response efforts with real-time epidemiological data, we have created several open-access databases available at **dati.gov.it**[13], which is a platform that aims to aggregate the national list of Italian open data in different sectors-from environment to health, from education to transportation-in one portal:

- **"Emergenza COVID19 - Regione Calabria"**. From the infection report to the vaccines: all the data on the Covid emergency in Calabria, a region of Southern Italy[14].
- **"Vaiolo delle scimmie"**. Weekly European and Italian health monitoring summary describing the situation regarding the status of *monkeypox* (Mpox) infection evolution[15]. More details
- **"West Nile"**. The dataset describes the epidemiological trend of *West Nile* virus (Wnv) during the transmission season in Italy, expected between June and

[11] https://digital-strategy.ec.europa.eu/en/policies/strategy-data
[12] https://datacollaboratives.org/
[13] https://dati.gov.it/view-dataset?organization=universita-della-calabria
[14] https://dati.gov.it/view-dataset/dataset?id=3d374641-808c-4802-a6b1-dba96d968cb3
[15] https://dati.gov.it/view-dataset/dataset?id=7021f9f4-e468-4115-be41-7bdd452aa1a0

November. The following information is made available (i) veterinary surveillance (animal and entomological); (ii) human surveillance[16].

- **"Epatite di origine sconosciuta nei bambini"**. In [109] we make available data on cases of *hepatitis of unknown origin* in children aged 16 years and under reported to ECDC through the European Surveillance System (TESSy) and made available through a weekly report. The cases are stratified by age, gender, locations (aggregated to the country level), and clinical outcome (i.e., hospitalization, admission to an intensive care unit, and need for liver transplantation)[17].

- **"Influenza stagionale"**. The data describe the epidemiological (i.e., the onset, duration and intensity of the seasonal epidemic) and virological (i.e., monitoring the circulation of different types, as well as subtypes, of influenza viruses) trends of influenza[18]. Moreover, this data is being used by a major Italian newspaper (i.e., *Sole24Ore*[19]) to inform its readers about disease trends.

However, as epidemics expand and become pandemics, manual curation of data is no longer feasible and computational methodologies and infrastructure are needed to manage the collection of data from hundreds of disparate sources.

Considering the 2022 outbreak of *Mpox*, we developed a surveillance system, namely EpiMPX [110], that allows researchers and policymakers to monitor the impact of the virus in Europe, with a special focus on the epidemic trends in the Italian regions, based on an open-access database containing information on the laboratory confirmed Mpox cases reported by EU/EEA countries and updated once a week. EpiMPX monitors the space-time distribution of cases and their characteristics, such as age, gender, symptoms, clinical status, and sexual orientation, when available. To show how much the database can be useful, we estimated the effective reproduction number R_t from ECDC incidence (EpiCurves) by date of symptom onset and by country[20]. We used EpiEstim package [111] in R language with "Serial Interval Uncertainty" method to estimate the effective reproduction number R_t on a weekly sliding time window in European countries with more than 28 days of observed incidence, assuming that the Serial Interval (SI) early estimate in Italy is valid for other countries too. With the "Serial Interval Uncertainty" method, useful in the first outbreak phases when SI is still uncertain, Gamma distribution parameters are normally distributed, and a Markov Chain Monte Carlo simulation is used to explore SI distributions.

[16] https://dati.gov.it/view-dataset/dataset?id=32a9ef72-ec68-4f47-8df9-ca65c2a0a125
[17] https://dati.gov.it/view-dataset/dataset?id=25c4caf1-25e0-4d61-a5a4-8bf026685a0b
[18] https://dati.gov.it/view-dataset/dataset?id=54a7e912-dc0f-42a3-9759-5c6975dc2bad
[19] https://tinyurl.com/sole24ore-influenza-trend
[20] texthttps://github.com/fbranda/monkeypox-opendata

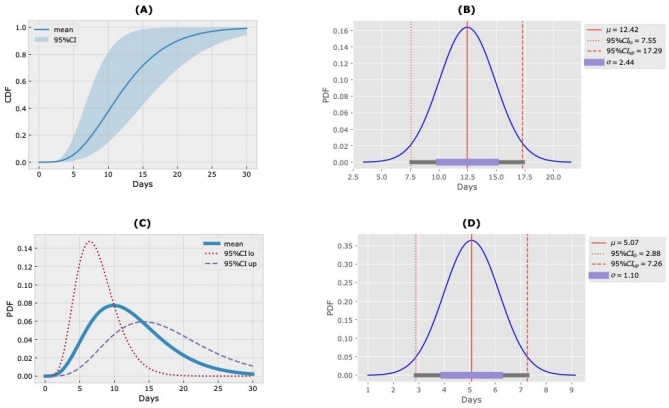


FIGURE 3.18: Early estimate of Serial Interval: (A) Cumulative Density Function with 95% CI; (B) Normal distribution of mean parameter μ; (C) Probability Density Function mean and 95% CI; (D) Normal distribution of standard deviation parameter σ.

Figure 3.18 shows the Serial Interval (SI) early estimate Cumulative Density Function (panel A) and Probability Density Function (panel C) Gamma distribution with 95% CI [112]; panels B and D show the normal distribution of SI Gamma distribution parameters (mean μ and standard deviation σ). SI early estimation has a mean that ranges from 7.55 up to 17.29 days.

The rapid outbreak sequencing of Ebola virus in 2022 demonstrated that the resurgence of Sudan virus disease (SVD) is a major public health concern in Uganda [113]. On 20 September 2022, Ugandan health authorities declared an outbreak of Ebola disease, caused by Sudan virus, following the confirmation of a fatal case in a young male resident of Ngabano village of Madudu sub-county in Mubende district[21]. On 11 January 2023, after 42 days with no new cases, the outbreak was declared over. A total of 164 cases (142 confirmed, 22 probable) and 77 deaths (55 among confirmed cases and 22 among probable cases) were reported from September 20 to 10 January 2023. Uganda has reported in its history four SVD outbreaks in 2000, 2011 and two in 2012, before the last one in 2022. It is therefore likely that filoviruses are present in the reservoir of wild animals in the region. Therefore, the risk of re-emergence of any filovirus through exposure to an animal host or from a persistent virus cannot be ruled out.

Figure 3.19 describes the epidemiological surveillance we implemented to collect, compare, and analyze information on all cases of Ebola infection reported by the World Health Organization-Regional Office for Africa (WHO AFRO)[22]. The system consists of the steps described below (see Figure 3.19A): (i) a data collection layer that collects shared data from verified sources, including reports from governments and public health organizations and statements from health officials reported in the media; (ii) a storage layer that facilitates the storage and organization of data in an easily identifiable structure; (iii) a processing layer that efficiently transforms, combines, and organizes data; iv) a publication layer that appropriately provides data and information to end users that they can use as a basis for epidemiological modeling to accelerate scientific discovery and response to the Ebola outbreak. Figure 3.19B summarizes the main tools used for each step. The main types of data we collected using an automated web scraping in R: a) key dates, which include the date of laboratory confirmed cases, including infections among healthcare workers; b) demographic

[21] https://www.who.int/emergencies/disease-outbreak-news/item/2022-DON410
[22] https://www.afro.who.int/countries/publications?country=879

information about the sex of patients/cases; c) geographic information, at the highest resolution available down to the district level; d) any additional information such as the status of hospitals, i.e., the bed capacity and occupancy rate of isolation units according to the severity status of the patient. Note that point b) and d) are not always shared in public official reports. For the rapid evolution of the epidemic and a data pattern not defined a priori given the dynamic context, we have chosen to adopt a No-SQL approach for data storage. Data processing was conducted using several programming languages, including R and Python. Specifically, data engineering activities, such as resolving inconsistencies in text formats through conversion, string matching and manipulation, merging files, reorganizing folders, and maintaining archives and folder locations that contained the latest version of official reports, were performed using R packages. These activities were programmed to operate semi-automatically and required human supervision to monitor and perform quality checks. All processed data were analyzed daily by a dedicated team of epidemiologists, data scientists, and statistical experts through Python scripts. Data analysis focused primarily on trends, geo- spatial distribution, and epidemiological characterization of cases by disease severity and sex. Other types of analysis performed included risk profiling of Ugandan districts by outbreak intensity. Finally, Ebola data were published through a GitHub repository[23].

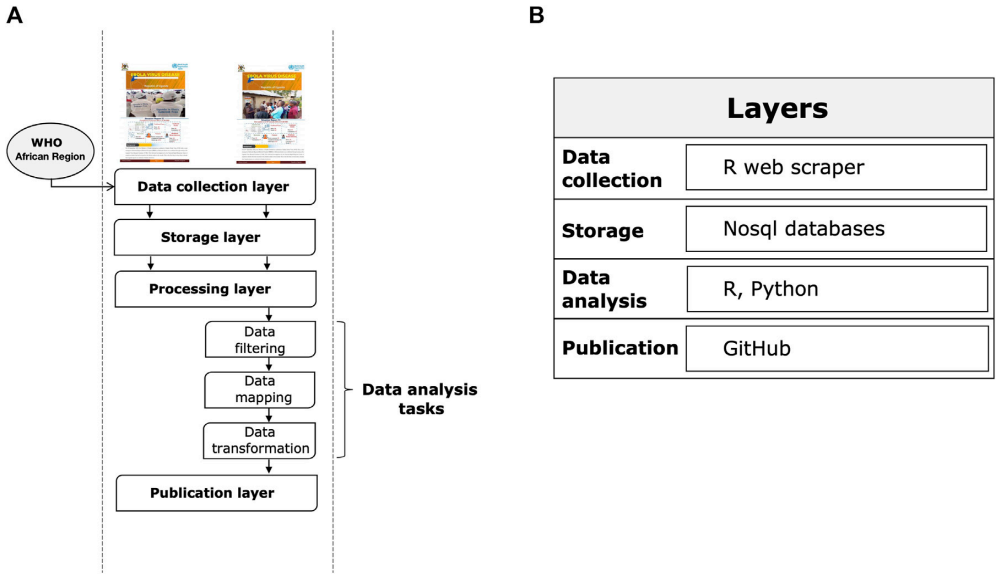


FIGURE 3.19: Layers of the Ebola information management system. (A) System execution flow. (B) Reference architecture.

3.3.2.1 Impact of SARS-CoV-2 infection in pediatric and school settings across COVID-19 waves in Italy: a pan-variant cross-sectional study

Since 2020, the COVID-19 pandemic has significantly affected the education sector around the world, with school closure causing important social costs in terms of learning, cognitive and socioemotional development [114, 115]. With the emergence of new severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) variants, against which existing vaccines are increasingly less effective [116, 117], monitoring the circulation of the virus in schools across different geographic contexts is necessary for

[23] https://github.com/fbranda/ebola

nearly real-time surveillance and to provide effective, timely, and inclusive responses to manage the pandemic [118, 119]. In this work, we report the epidemiological investigations of transmission of SARS-CoV-2 and their impact on the pediatric population and the Italian schools during different time periods of the pandemic, characterized by waves due to distinct virus lineages. Figure 3.20 describes an example of integrated epidemiologic surveillance that we implemented to continuously and systematically collect, compare, and analyze information on all cases of SARS-CoV-2 infection confirmed by molecular diagnostics at regional reference laboratories throughout Italy.

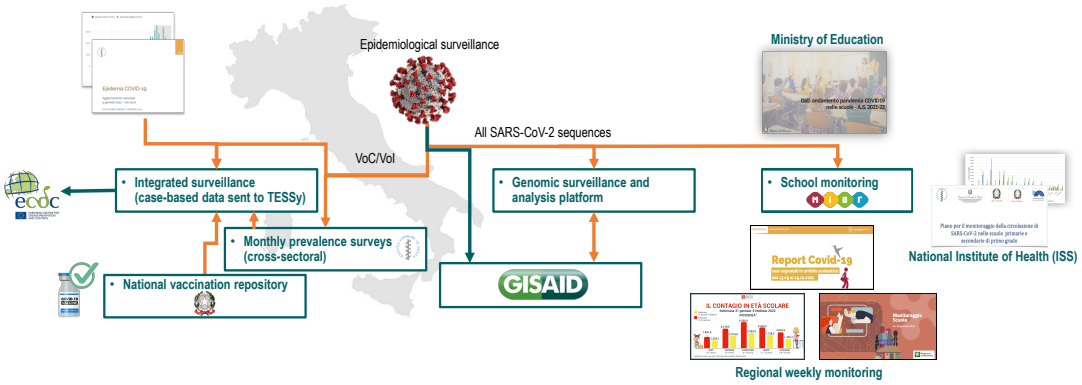


FIGURE 3.20: Example of SARS-CoV-2 integrated surveillance system.

To assess the impact of distinct SARS-CoV-2 variants on the pediatric population we integrated data of the CovidStat INFN[24] group, processing aggregate data from ISS and disseminating the results in machine-readable format, with the genomic data set from the GISAID repository [120] to gain insights on the epidemiological transmission dynamics of the disease per variant and age group. Furthermore, to evaluate the impact of the COVID-19 pandemic on Italian education, we analyzed data collected by the Ministry of Education Information System (SIDI)[25] through the "Rilevazione andamento emergenza COVID-19" initiative to gain insight on the spread of the disease at the national- and subnational levels. Specifically, the national monitoring plan for the 2021/22 school year stipulated that at the beginning of each week, schools would report through the Ministry of Education's Information System (SIDI) data on (i) teaching/non-teaching staff and students tested positive for COVID-19 and in quarantine; (ii) the number of classes in quarantine (known as "DAD", i.e., remote learning), and in presence and partially remotely (known as "DDI").

Since 2020, several notable variants of SARS-CoV-2 have emerged in Italy (see Figure 3.21), among which the most dominant ones were the B.1.1.7 (also known as "Alpha"), the B.1.617.2 (also known as "Delta") and the B.1.1.159 (also known as "Omicron". Figure 3.22 shows the daily incidence for each epidemic class and the information stratified by age group (0-5, 6-12, 3-19) and variant, obtained by matching the abundances obtained from genomic data and the time course of epidemic classes of interest, such as confirmed cases, symptomatic cases and hospitalizations. Since the Christmas holidays 2021, coinciding with the rapid spread of the Omicron variant, children have made up an increasing proportion of weekly case numbers with the most noticeable increase among those aged 6-12 years.

[24] https://covid19.infn.it/iss/
[25] https://www.istruzione.it/iotornoascuola/monitoraggi.html

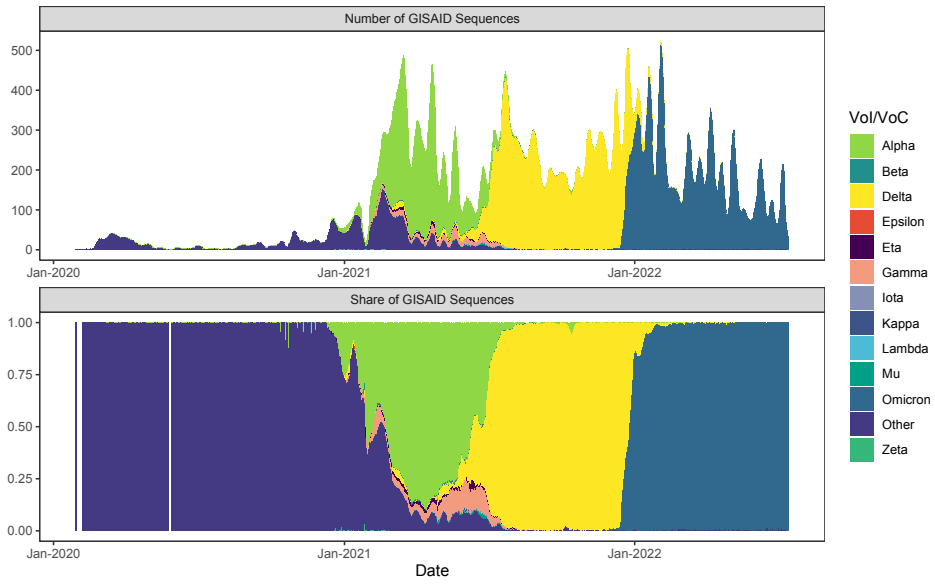


FIGURE 3.21: Evolution of SARS-CoV-2 variants across time, in terms of absolute number of sequences (top) and their relative share (bottom) in Italy, according to the GISAID repository.

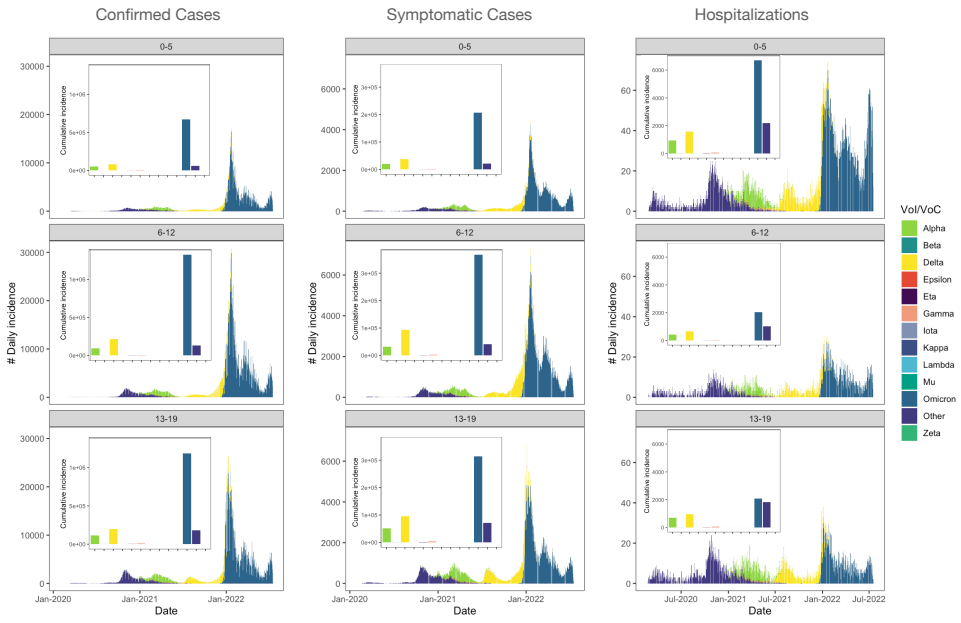


FIGURE 3.22: Epidemiological curves for SARS-CoV-2 infection in children. Observed daily incidence for confirmed (left), symptomatic (middle) and hospitalized (right) cases, stratified by age group: 0-5 (top), 6-12 (middle) and 13-19 (bottom). Color codes the variant inferred to be responsible for cases. Insets show the corresponding cumulative incidence due to each variant.

Our findings suggest that each new strain of the SARS-CoV-2 spreads faster than the previous one, with different effects on the school-age population, as summarized in Figure 3.23. The figure shows the incidence rate per 100,000 inhabitants in Italy for the three epidemic classes of interest, the three most dominant variants in the country and all age groups (upper panels), together with the relative difference between each

all rates and a reference baseline, here chosen to be the 0-5 years group during the Omicron wave. The laboratory-confirmed COVID-19–infections per 100,000 people were highest during the circulation of Omicron variant, with 34,959.3 cases among children aged 6-12 years (2,470.3 and 5,627.4 compared to Alpha and Delta, respectively) and adolescents aged 13–19 years (2,804.1 and 4,947.7 during Alpha and Delta waves, respectively), and slightly lower among children aged 0-5 years (24,734.7 vs 1,953.8 and 2,974.1 during Alpha and Delta waves, respectively). The Omicron wave caused an unprecedented surge in children's hospitalizations than the Alpha or Delta VOCs: 247.1 vs. 34.6 vs. 58 COVID-19–associated hospitalization rates per 100,000 were observed in children aged 0-5 years, 54 vs. 11.1 vs. 17.3 in children aged 6-12 years, and 52.1 vs. 17.8 vs. 23.9 in adolescents aged 13-19 years. To highlight how the spreading of COVID-19 impacts schools, we use a linear regression model to establish the relation between number of students in DAD and the same quantity one week ahead, from the week ending on the 16th of January to the week ending on the 12th of June, 2022. The data are stratified by age-group (prescholars, less than 5, primary school students, between 6 and 12, secondary school students, between 13 and 19) and region.

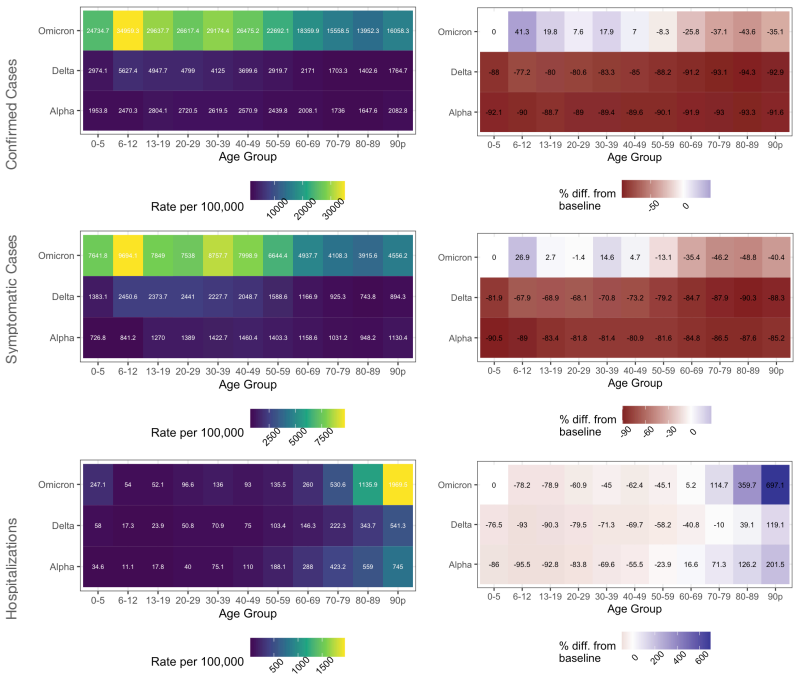


FIGURE 3.23: Pan-variant analysis of SARS-CoV-2 infection. Confirmed (top), symptomatic (middle) and hospitalized (bottom) cases, stratified by age group, are shown for Alpha, Delta and Omicron variants. Left panels report and color-code the observed rates per 100,000 inh. Right panels report and color-code the difference corresponding to each combination variant and age-group with respect to a baseline, here chosen to be the Omicron variant for children aged 0-5 years in the corresponding epidemiological compartment.

Our results show that the last wave, corresponding to the Omicron lineage, significantly hit the pediatric population in Italy both in absolute counts and in comparison, with previous waves, for what concerns confirmed and symptomatic cases. Hospitalization due to Omicron rates are lower in children 0-19 than older age groups: however, we measure a relevant increase in children aged 0-5 with respect to the Delta wave (247.1 vs 58 children per 100,000; +76.5%) and the Alpha wave (247 vs 34.6 children

per 100,000; +86%), with significant increase also in children aged 6-12 and 13-19. The largest impact on the aged 0-5 might be due to several factors, including the absence of vaccine coverage in this group. The impact of sustained community transmission on education, measured in terms of absences due to COVID-19 and by the weekly incidence, is significant: we have observed up to 25% of absences during the Omicron wave. The direct relationship between incidence and the number of pupils in remote learning can be used by policy makers as an early-warning indicator for short-term mitigation actions. In the future, it is desirable to integrate other relevant information sources, including clinical data, to develop an interdisciplinary collaboration between experts from various fields (e.g., epidemiologists, data scientists, policy makers, and response agencies) able to further strengthening data science capacity in public health institutions and leverage this knowledge to generate the most impact.

3.3.2.2 Comparing Worldwide, National, and Independent Notifications about Adverse Drug Reactions Due to COVID-19 Vaccines

Real-world outcomes have also been critically important for the rapidly developed COVID-19 vaccines, which reached the market through emergency use authorizations based on limited clinical data. As more people received COVID-19 vaccines, there was an increasing body of evidence suggesting that these vaccines were indeed safe and effective. In this paper [121], we define a general framework for a pharmaceutical data analysis application to automatically monitor increases in known adverse events and discover possible reporting clusters (e.g., suspected temporally localized or product-specific adverse event reporting). Results of large-scale data analysis show that: (1) most of the adverse events are labeled as non-serious and concern muscle/joint pain, chills and nausea, headache, and fatigue; (2) the notification rate is higher in the age group 20–39 years and decreases in older age groups and in very young people; and (3) the distributions of such reports concern women more than men.

Figure 3.24 depicts the entire workflow of the proposed method, which consists of three main steps. The first step consists of collecting adverse events to be processed. In the second step, the data are cleaned, selected, and transformed to make them suitable for analysis. Specifically, in this step, the following operations are performed: *data enrichment* combines data from multiple pharmacovigilance systems into a single, consistent database; *information extraction* involves the aggregation, filtering, cleaning, de-duplication, and validation of the data; and *data analysis* examines the transformed data to detect new, unusual, or rare vaccine adverse events, and to address possible reporting clusters (e.g., suspected or product-/batch-/lot-specific adverse event reporting).

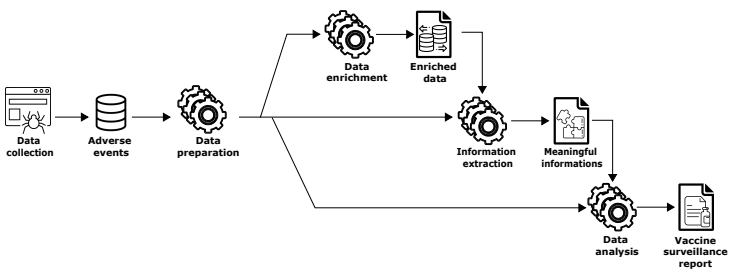


FIGURE 3.24: Workflow of the proposed framework.

Algorithm 3.25 shows the pseudocode for classifying a suspected adverse event.

Algorithm 1 Adverse events reports processing

```
   Input  :Reports of suspected adverse events ADRs
   Output:Dictionary of ⟨vaccine, reaction⟩ D_AE
1  D_AE ← ∅
2  for event ∈ SAE do
3  |  vaccine_type ← event.getVaccineType()
4  |  age_group ← event.getAge()
5  |  sex ← event.getSex()
6  |  reaction_type ← event.getReactionType()
7  |  D ← BuildPVData(vaccine_type, reaction_type, age_group, sex)
8  |  /* Generate signals of disproportionate reporting (SDR)        */
9  |  SDRs ← ApplyDisproportionalityAnalysis(D)
10 |  /* Determine if SDRs are relevant                              */
11 |  if (⌊SDRs⌋ ≥ 1) then
12 |  |  D_AE ← D_AE ∪ ⟨SDRs.vaccine_type, SDRs.reaction_type⟩
13 |  end
14 end
15 return D_AE
```

FIGURE 3.25: Adverse events reports processing algorithm.

The algorithm receives as input the reports of suspected adverse events, $ADRs$, and returns a collection D_{AE} of adverse reactions that have been classified as potential drug–event associations for further evaluation. The algorithm analyzes each *event* (lines 2–13) by performing the following operations:

- Retrieving the information about *vaccine_type* (i.e., typology of medical product such as Comirnaty, Spikevax, etc.), *age_group* (i.e., the patient's age), *sex* (i.e., the patient's sex), and *reaction_type* (i.e., the information about the suspected ADRs/*event*) (lines 3–6).
- Creating a pharmacovigilance dataset to make it suitable for analysis (line 7).
- Generating the statistical associations between medicinal products and adverse events, i.e., drug–event pairs, using disproportionality analysis methods (line 9).
- If the lower bound of the 95% confidence interval of signals of disproportionate reporting (SDRs) generated at the previous step is greater than or equal to one, the adverse reactions require further evaluation (lines 11–12).

Two examples of use of the proposed methodology is discussed below.

1. In the first case, we designed and developed an independent platform for collecting and analyzing spontaneous reports, where everyone (from health professionals to patients to citizens) can report a suspected adverse event through a secure online submission process[26], as shown in Figure 3.26. The form has required data fields for patient demographic information (sex and age), vaccine administered, and severity outcome of the adverse event on a scale of 1 to 5 (the severity of an event varies between 1 and 5, broken down as follows: *low* ("1", "2"), *medium* ("3"), *high* ("4", "5")), including the description of an adverse event if the "Other" option is clicked.

[26] http://www.covid19-italy.it/segnalazioni-covid19/index.html

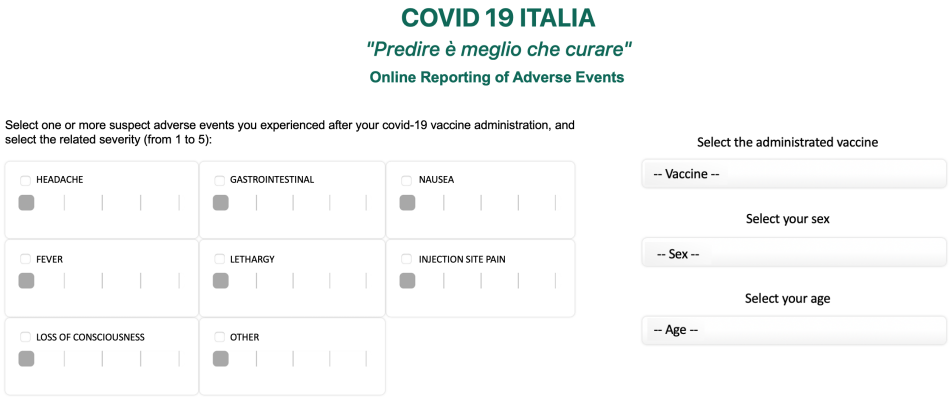
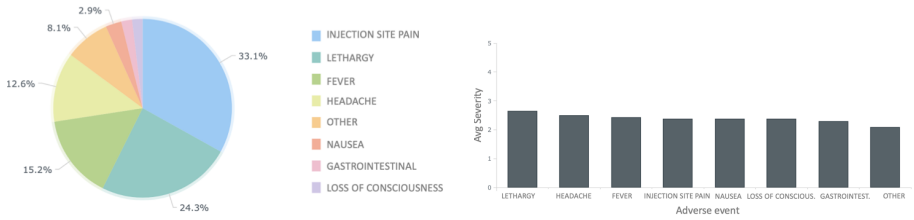


FIGURE 3.26: An example of online reporting of an adverse event on our platform.

Starting from the scientific divulgation page of one of the authors[27] with more than 10,000 followers, we collected about 2000 spontaneous and anonymous (GDPR-compliant) reports (third dose pre-booster) from 1 October 2021 to 20 November 2021, distributed as follows (see Figure 3.27a): 33.1% of reports referred to **injection site pain**, 24.3% to **lethargy**, 15.2% to **fever**, 12.6% to **headache**, 8.1% to **other** (i.e., more detailed reporting that does not fall into a predefined category, such as **menstrual cycle alteration**, **arrhythmia, tachycardia, chest pain**, etc.), 2.9% to **nausea**, and 2.0% to **gastrointestinal disorders** and **loss of consciousness**. However, the severity of these events on a scale of 1 (very low) to 5 (very high), is low with a value of 2.39 on average, a maximum value of 2.65 for lethargy, and a minimum value of 2.09 for the other adverse events collected, as shown in Figure 3.27b.



(a) (b)

FIGURE 3.27: Distribution of reported adverse events on our platform by type (a) and by severity (b).

Regardless of dose number, the distribution of suspected adverse events is in line with that reported in previous sections, as shown in Figure 3.28a,b. In fact, the charts reveal that the number of reports is highest in the 20-49 age groups, and about 61% of them involve women. Considering the *Other* category, Figure 3.28c shows the keywords most commonly used by platform users to describe the reactions that were observed after administration of the vaccine. Specifically, 104 reports fell within the category for **joint and muscle pain (back, neck)**, followed by 48 for **injection site warmth, erythema, and pruritus**, 44 for **lymphadenopathy**, 33 for **menstrual cycle alteration**, 28 for **arrhythmia, tachycardia, and chest pain**, 23 for **chills**, 18 for **drowsiness**, 12 for **transient hypertension** and **dizziness**, 8 for **eye disorders**

[27] https://www.facebook.com/prediremegliochecurare

(i.e., **blurred vision and hemorrhage**), 7 for **paresthesia**, and 6 for **tinnitus**. Regarding the severity of such events, 21.22% (83/391) were classified as *high*, 12.28% (48/391) as *medium*, and 66.5% (260/391) as *low*.

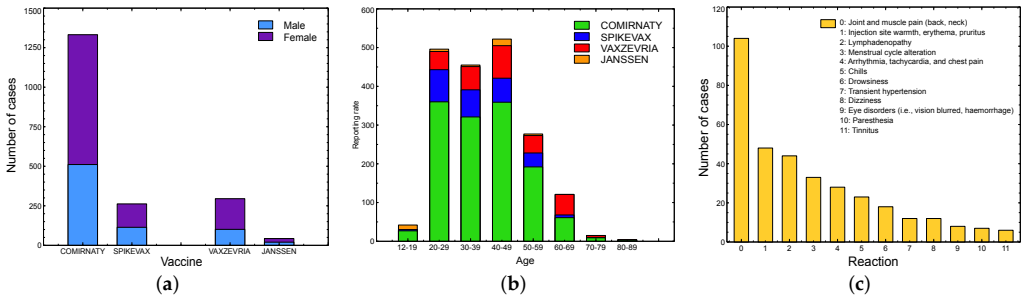


FIGURE 3.28: Main results of our pharmacovigilance activity: distribution of number of reported cases by sex (**a**), by age (**b**), and in "Other" category (**c**).

2. In the second case, we evaluated the impact of COVID-19 additional/booster vaccines against COVID-19-related symptoms, hospitalization, and death in Italy, between December 28, 2021, and May 18, 2022, when Omicron was the dominant variant in circulation [122]. The estimates of vaccine effectiveness (VE) reported in the weekly bulletin of Istituto Superiore di Sanità (ISS)[28] are analyzed to evaluate the impact the additional/booster doses against severe illness during the epidemic phase characterized by the predominance of the Omicron variant. Such estimates are calculated using generalized linear random-effects model with Poisson distribution, considering the number of events per day as the dependent variable, vaccination status as the independent variable, 10-year age groups and weekly regional incidence as adjustment variables, and including region of administration as a random effect. Using this model, it is possible to estimate the relative risk (RR), i.e., the ratio of the risks for an event for the exposure group to the risks for the non-exposure group. Moreover, the number needed to treat (NNT), i.e., the number of people who need to be treated to prevent one additional adverse outcome from a disease, is used to evaluate the impact of the VE. Since it is not possible to compare the absolute numbers of events in the different vaccination status within the same age group because they refer to different populations, it was necessary to calculate the incidence rate (IR), i.e., the number of new cases of a disease divided by the number of persons at risk for the disease. Figure 3.29 describes the IR per 100,000 persons stratified by vaccination status and by age group. During the period under analysis, rates of COVID-19 cases were lowest among fully vaccinated persons with a booster dose, compared with fully vaccinated persons without a booster dose, and much lower than rates among unvaccinated persons aged $\geq$ 60 years. Figure3.30A describes the vaccine effectiveness (VE) against SARS-CoV-2 infections VE of vaccinated persons with 2 doses within 90 days in preventing infection was 44.0% (95% CI: 43.8%-44.1%), i.e., there was a risk reduction of about 44% for those vaccinated within 90 days compared to the unvaccinated. Between 90 and 120 days after the administration of the second dose, the estimated VE in preventing diagnoses was 33.5% (95% IC: 33.3%-33.7%), rising to 45.9% (95% IC: 45.8%-46.0%) after 120 days, and 57.6% (95% IC: 57.5%-57.7%)

[28] https://www.iss.it/en/sorveglianze-covid-19

in individuals with additional/booster dose. In the case of severe disease (see Figure3.30B), VE for vaccinated persons with 2 doses within 90 days, between 91 and 120 days and over 120 days was 70.8% (95% IC: 69.9%-71.8%), 69.6% (95% IC: 68.3%-70.8%) and 71.5% (95% IC: 71.0%-72.0%) respectively, whereas it was 87.8% (95% IC: 87.6%-88.0%) in vaccinated with an additional/booster dose. The results of the study demonstrated the importance of receiving a third dose of COVID-19 mRNA vaccine to prevent both infection and severe COVID-19, especially when the effectiveness of 2 doses is significantly reduced against the Omicron variant.

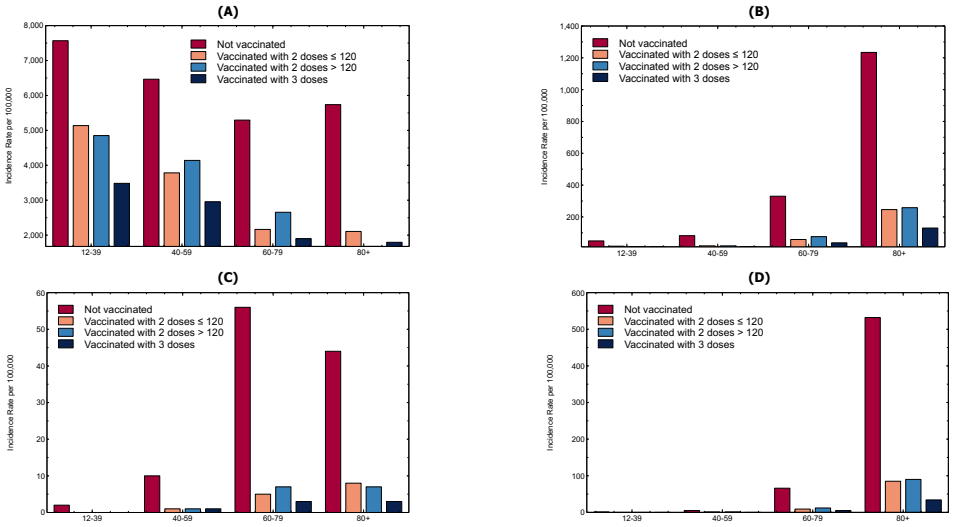


FIGURE 3.29: Incidence rate of COVID-19 diagnoses (A), and diagnoses with subsequent hospitalization (B), ICU admission (C), and death (D) by vaccination status and by age group.

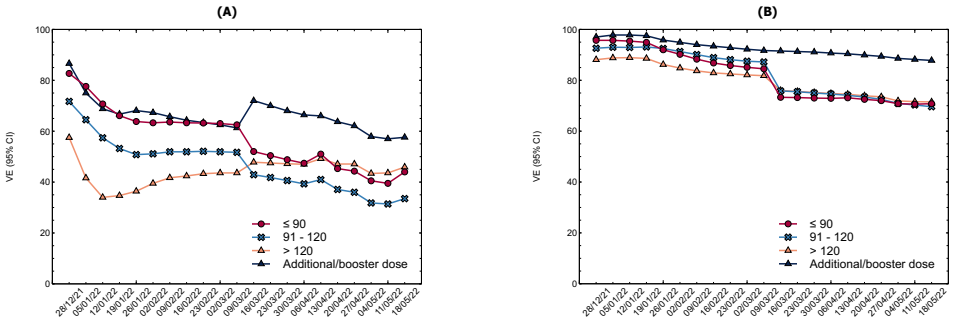


FIGURE 3.30: Vaccine effectiveness (VE) against SARS-CoV-2 infections (A), and severe disease (B), by vaccination status.

3.3.3 Applications of predictive modelling in the epidemiological context

During the onset of a new pandemic, predictive modeling is important for public health planning and response. Correlating models to data provides information on previously unseen variables, such as the occurrence of cryptic transmission and the prevalence of infection, and these models allow for the exploration of counterfactuals and hypothetical interventions. However, despite enormous advances in mathematical

epidemiology, predictions of epidemic outcomes are inherently error-prone. Predictive modeling is valuable when the assumptions are correlated, the variables to be estimated are clearly defined, and the researchers or policymakers using the model results have a clear understanding of what can and cannot be achieved by this method. Predictive modeling requires a diverse, often imperfect data set, especially during the crucial period of epidemic emergence when surveillance is imprecise and little is known about the epidemiology or clinical characteristics of the disease. For example, large clinical case counts and genomic data have been combined with large-scale records of human mobility and behavior using predictive modeling, thanks in part to the massive dissemination of digital information sources. In the following sections, we highlight some important findings from the application of predictive modeling to various data sources that can influence clinical and policy decisions.

3.3.3.1 Predicting the Spread of SARS-CoV-2 in Italian Regions: The Calabria Case Study, February 2020–March 2022

In December 2019, the new Coronavirus, SARS-CoV-2 [123], emerged and into a world population without proper specific immunization. Due to its high infectivity, the virus spread worldwide, beginning the new current pandemic [124]. During the epidemic and subsequent vaccine immunization, one of the main issues was the emergence of several Variants of Concern (VOC) [125] with consequent re-emergence of new infection cases. In particular, several notable variants of SARS-CoV-2 have emerged in recent months [126], such as B.A.2 [127], which expresses a very high infectivity rate and pathogenesis, and B.A.2.2 [128], responsible in Hong Kong for the significant increase in lethality among the unvaccinated and uninfected elderly and children.

This highlighted the importance of following the contagion evolution on a temporal scale inside the various populations, especially to define the restarting moment of new cases to prevent severe clinical cases, along with ICU admissions and deaths, to determine needed non-pharmacological intervention (NPI), to test the success or failure of containment measures in place, and to guide governments and decision makers. Moreover, it has become increasingly important to define and refine epidemiological forecasting methods in order to adequately monitor population infection cases and deaths. SARIMA(X) models are an example of methods that have already been used for prediction in epidemiology, such as malaria [129], influenza-like illness [130], dengue hemorrhagic fever [131], West Nile virus [132], scarlet fever [133], human brucellosis [134], and recently, COVID-19 [135, 136].

This paper [3] presents the design and implementation of an approach based on autoregressive models to reliably forecast the spread of COVID-19 in Italian regions. The method automatically collects daily data on the number of individuals infected with SARS-CoV-2 and performs the following operations: (i) *data pre-processing*, which consists of turning raw data into the required format for the purposes of analysis; (ii) *predictive modeling*, which regards the training of a model to forecast the number of infections that will happen in a specific area; (iii) *results visualization*, which presents the results in a graphical way, allowing users to visually explore the data.

Figure 3.31 presents the general idea of the approach through a graphic representation of the whole process as a sequence of three steps. The input data of the analysis is the set of collected epidemiological data to be processed. The second step consists of cleaning, selecting, and transforming raw data into the desired format so that useful information can be derived from it. In other words, (i) incorrect and incomplete data are removed; (ii) a subset of data is selected to make it suitable for analysis; (iii) third-party data from an external authoritative source are merged with the existing

database to enrich collected data. The third step is aimed at detecting the regions mostly affected by the pandemic and extracting a prediction model for providing dynamic monitoring of the spread of the disease, and supporting organizations in the evaluation of the effect of local containment measures. Finally, results visualization is performed by using an interactive dashboard to inform citizens in an intuitive way and make the collected data available.

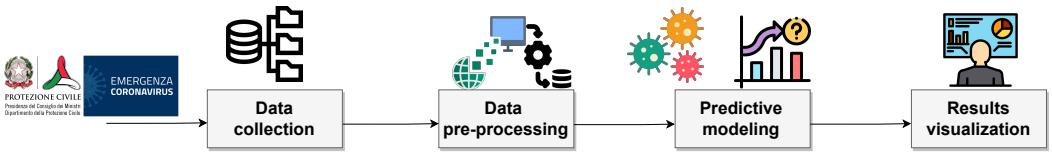


FIGURE 3.31: Proposed approach steps.

We carried out an experimental evaluation by analyzing the SARS-CoV-2 trend in the period between February 24, 2020, and March 27, 2022, in Calabria, one of the Italian regions which has been most affected during the fourth wave of the pandemic. The data that we used to evaluate the effectiveness and accuracy of the approach described above are gathered from the Italian Civil Protection Department (DPC)[29] dataset and enriched at the provincial level by using an automatic scraper that extracts and transforms data from the website of the Calabria region[30] into a machine-readable format in order to make its reuse easier on the COVIDA platform[31](more details can be found in Appendix A.2).

Figure 3.32 shows a preliminary view of the collected epidemiological data. Specifically, Figure 3.32A reports the time plot of the new confirmed cases, in which new positive cases are plotted versus the time of notification. From the plot, we see that the number of cases exhibits a stable trend until December 2021, followed by a sharp increase in infections in January 2022, due to the circulation of the SARS-CoV-2 Omicron variant. Omicron's higher infection rate have pushed health systems to the breaking point, causing a significant number of deaths and hospitalizations. A clearer view of the Omicron's wave can be seen in Figures 3.32B and 3.32C, which show the number of confirmed COVID-19 deaths and hospitalizations per day, respectively. In particular, the new record for deaths in one day was 18 on 28 January 2022 (previously it was 17 on 23 November 2020), whereas the hospitalizations have risen significantly, approaching the values of the prior wave (446 hospitalizations on 18 January 2022 versus 482 hospitalizations on 26 April 2021). However, for the innate features of Omicron and the protection offered by the vaccines, a smaller percentage of COVID patients were admitted to intensive care units (ICUs), as shown Figure 3.32D. The chart clearly show that during the first phase of Omicron surge, an initial increase of patients (with 38 persons admitted in ICU on 11 January 2022) was followed by a smooth decreasing trend.

[29] https://github.com/pcm-dpc/COVID-19
[30] https://regione.calabria.it/website/
[31] https://covida.ml/

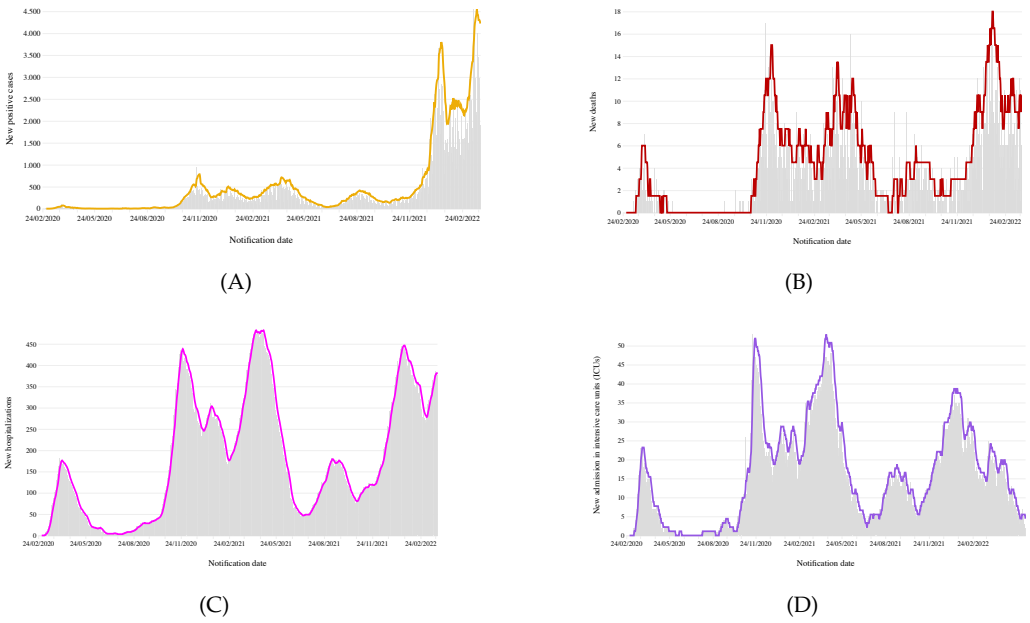


FIGURE 3.32: Calabria epidemiological data: (A) new positive cases; (B) total amount of deaths; (C) hospitalized patients with symptoms and (D) in intensive care.

To evaluate the impact of SARS-CoV-2 circulation, we analyzed the net reproduction number R_t (i.e., the transmission potential at a given time t of the epidemic once interventions are introduced or the susceptibility in the population decreases), calculated by the research group CovidStat INFN[32]. This method uses the growth rate determined over the last 14 days with an exponential fit to the number of infected persons per day [137] assuming the mean value of the generation time published by Cereda et al. [138]. As shown in Figure 3.33, with the national lockdown imposed in early March 2020, R_t estimates followed a constantly decreasing trend. Since late May 2020 with the gradual reopening of all activities, R_t started to fluctuate, reaching maximum values around 3 in the week from 3 to 10 August. From January 7, 2021 to March 27, 2022, R_t remained nearly constant at values around 1.5–1.8.

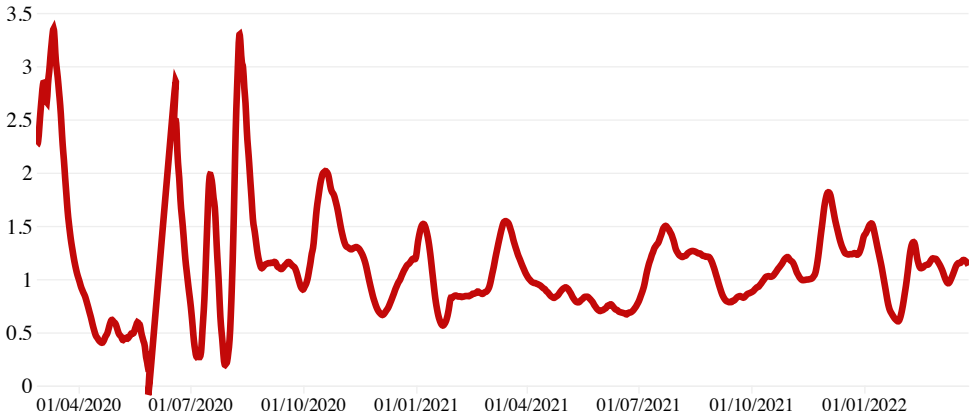


FIGURE 3.33: COVID-19 estimated R_t in Calabria over a 7-day moving average, February 2020, 24 – March 27, 2022.

[32] https://covid19.infn.it/progetto.html

To define the SARIMA model R-squared (R^2) score for out-of-sample 14-days prediction, for each day between January 1, 2021, and March 13, 2022, we have chosen the best SARIMA model via grid-search (based on previous observations) and forecasted the next 14 days. Pooled R^2 score of 14 days of observations and predictions is 0.90 at 1 day, greater than 0.80 within 6 days, and greater than 0.50 up to 14 days (Figure 3.34). These results confirm the appropriateness of the autoregressive model and its good performance in the epidemiology domain over rolling time horizons.

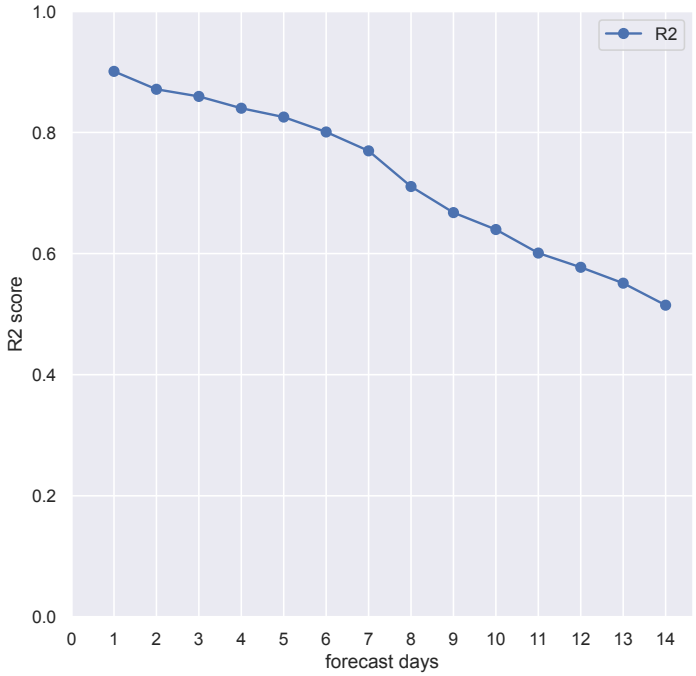


FIGURE 3.34: Pooled R^2 scores of the 14-days out-of-sample forecast of COVID-19 new daily cases in Calabria (Italy) with SARIMA model.

In future work, other research issues may be investigated. Firstly, we may explore a machine-learning-based approach, using feature importance for identifying potential patient risk factors for particular types of adverse events, as well as investigating further disproportionality analysis methods to generate hypotheses regarding possible causal relations between drugs and adverse effects. Secondly, we will improve the quality and completeness of reporting, as there is currently a lack of information about the total number of people vaccinated and the total number of people who experienced an adverse event, as well as about the incidence of adverse events in unvaccinated persons.

3.3.3.2 Statistical models to predict clinical outcomes with anakinra vs. tocilizumab treatments for severe pneumonia in COVID19 patients

Predictive modeling is neither a monolithic framework nor a single methodology, but rather encompasses a wide variety of statistical and mathematical models applied to different data to achieve different inference and prediction goals. How can we evaluate the performance of predictive modeling in driving the overall response to COVID-19? Regarding the most important application of these models, there has been considerable success in predicting a patient's clinical outcome.

The clinical course of coronavirus disease 2019 (COVID-19) varies from mild to severe, with a minority of patients evolving to critical condition. The most common complication of COVID-19-related pneumonia is a specific type of acute respiratory distress syndrome (ARDS), characterized by dramatic inflammatory features similar to cytokine release syndrome and macrophage activation syndrome (MAS) [139]. Initial understanding of the pathology of ARDS in COVID-19 suggested that interleukin blockade may control the observed cytokine storm that resembles MAS. A potential clinical option was found in the interleukin-6 antagonist tocilizumab and the recombinant interleukin-1 receptor antagonist anakinra [140]. Early clinical experience highlighted the possible role of these drugs in the treatment of COVID-19 and subsequent observational studies confirmed the positive impact of both tocilizumab and anakinra in the setting of hyper-inflammation or in cases with symptoms indicative of worsening respiratory function [141, 142]. Even though different results have been reported by randomized clinical trials (RCTs) a better clinical response has been shown in patients with severe or critical COVID-19 who were treated with anakinra or tocilizumab when hyper-inflammation was demonstrated [143, 144]. Moreover, little information is available about possible differential benefits of either drug. The purpose of our study [145] was to evaluate possible differences in biochemical markers and comorbidity conditions to predict outcome in order to define optimal therapy, the best timing of treatment initiation and the possible risks for severe adverse events.

We conducted a retrospective analysis including all patients admitted to the Infectious Diseases Unit, A. Manzoni Hospital, with a diagnosis of COVID-19 pneumonia during the first wave of the pandemic (from March 1 and May 15, 2020), who required different oxygen supports, and who received off-label treatment with anakinra or tocilizumab. Nonparametric tests (Mann–Whitney and Kruskal–Wallis) and Pearson χ test (or Fisher's exact test, when applicable) were used to compare the continuous and categorical variables of patients, respectively, with the different treatment strategies. Statistical analysis was performed using R (version 4.0.0) and R Studio (version 1.2.5042) software. Predictive models based on Artificial Intelligence (AI) were used to predict the outcome of a COVID-19 patient treated with anakinra or tocilizumab using CCI and biochemical/inflammatory parameters at baseline (day of start immunomodulant treatment). Specifically, our AI predictive models adopt Logistic Regression using the Scikit-learn package in Python. Logistic Regression identifies the relationship between a continuous dependent variable (30-day mortality and secondary infections) and one or more independent variables (CCI and blood inflammatory markers).

To provide a quantitative evaluation of the performance of the models and their effectiveness in making predictions on the corresponding test sets, we computed: (1) the R-squared (R^2), which explains to what extent the variance of one variable explains the variance of the second variable (e.g., if the R^2 of a model is 0.50, then approximately half of the observed variation can be explained by the model's inputs); and (2) the mean squared error (MSE), which measures the amount of error in statistical models. The K-fold cross-validation technique was also exploited to detect overfitting in a model (i.e., the model is not effectively generalizing patterns and similarities in the newly inputted data). K refers to the number of groups the data sample is split into. 71 and 39 patients were treated with anakinra and tocilizumab, respectively. 71.8% of patients were male, and the median age was 65 years (interquartile range [IQR] 59–71 years). No differences were observed in CCI and inflammatory biomarkers at baseline between the anakinra and tocilizumab groups. Overall, 30-day mortality was 23% (20/71) in patients receiving anakinra versus 35% (12/39) in those receiving tocilizumab, without a statistically significant difference between the groups. Same

results were found about major secondary infection events (19.7% vs 28.2%, anakinra vs tocilizumab; p=0.44). No significant difference in mortality was observed between patients experiencing secondary infections compared with those without this adverse event in the anakinra and tocilizumab groups (35.7% vs. 27.2%, p=0.40 and 27.3% vs. 32.1%, p=0.19, respectively).

A comparative analysis of anakinra and tocilizumab treatments to discover the relationship between CCI (independent variable) with secondary infection risk and probability of survival (dependent variables) was performed. The results show a positive correlation between CCI values and secondary infection (R^2=0.88) with anakinra, whereas inverse correlation (R^2 = 0.70) with tocilizumab treatment. The relationship between CCI and survival decreases as CCI values increase in both groups of patients (anakinra R^2=0.80 and tocilizumab R^2=0.64).

The advantages of using models like ours should be: (i) improved diagnostics; (ii) high cost-effectiveness; (iii) increased operational efficiency; (iv) reducing readmission rates; and finally (v) personalized medical care with predictive models that improve patient-centered care based on personal health records and contributes to the creation of the most effective and personalized treatment plans for each patient.

3.3.3.3 Monkeypox: Early estimation of basic reproduction number R0 in Europe

Starting from our EpiMPX open data [110] surveillance system, we defined a measure of early R_0, which is an epidemiological index expressing the potential transmissibility of an uncontrolled infectious disease and can be interpreted as the average number of people infected by an infected contact during the first exponential growth phase of an epidemic, before any countermeasure is taken. In particular, The aim of our study [146] was to develop and implement a novel Bayesian method to estimate the basic reproduction number R_0 in early epidemics' outbreaks when the Generation Time (GT, from infection to infection) is not yet known and the Serial Interval (SI, from symptoms onset to symptoms onset) is still highly uncertain [111]. We assume that an infected individual at time t generates new infections at time $t+s$ with a probability proportional to the SI distribution [111]. To account for SI uncertainty, we made use of a simulation, distributing the SI parameters and taking a proper number of random samples [111]. Since R_0 needs to be estimated in the first exponentially phase of an epidemic outbreak, we performed a sensitivity analysis, with a deviance-based R^2, to choose the best time window [147]. This method can be of great help in early R_0 estimation. We applied the method to the recent Monkeypox outbreak in Europe, using the data provided by the ECDC[33], made readily available in machine-readable format by the EpiMPX project [110]. We also assumed that the early SI estimate in Italy [112] is valid for other European countries too. A more detailed description of the method is available in the Supplemental Material and its implementation is public and open-source on the GitHub repository available at the link `https://github.com/maxdevblock/Monkeypox_R0_Europe`.

Results show high R_0 variability between the selected European countries (Figure 3.35B and Figure 3.36), from Belgium (1.54, 95%CI 1.28-1.9) to Germany (3.62, 95%CI 2.22-6.38). The causes of this variability should be investigated in detail and could be identified in i) different Serial Intervals (SI), ii) different social and/or sexual behaviors, iii) possible different Monkeypox virus variants, and iv) other hidden causes. Pooled European R_0 estimate (Figure 3.35A), obtained by pooling all R_0 samples and fitting the result to a Gamma distribution, shows a median of 2.44 (95%CI

[33] `https://monkeypoxreport.ecdc.europa.eu/`

1.35-4.9) which is compatible with previous estimations [112] and estimated epidemic potential [148]. This is, in general, a relatively high R_0 and needs to be confirmed by further studies and methods with i) less uncertain SI estimations, ii) country-specific SI estimations, iii) in-depth virological and epidemiological analysis of any factor that could impact the viral transmission and the basic reproduction number.

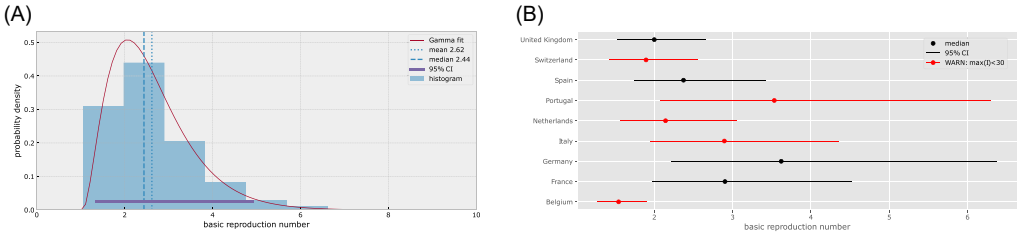


FIGURE 3.35: Monkeypox pooled early R estimate in (A) Europe and (B) selected countries.

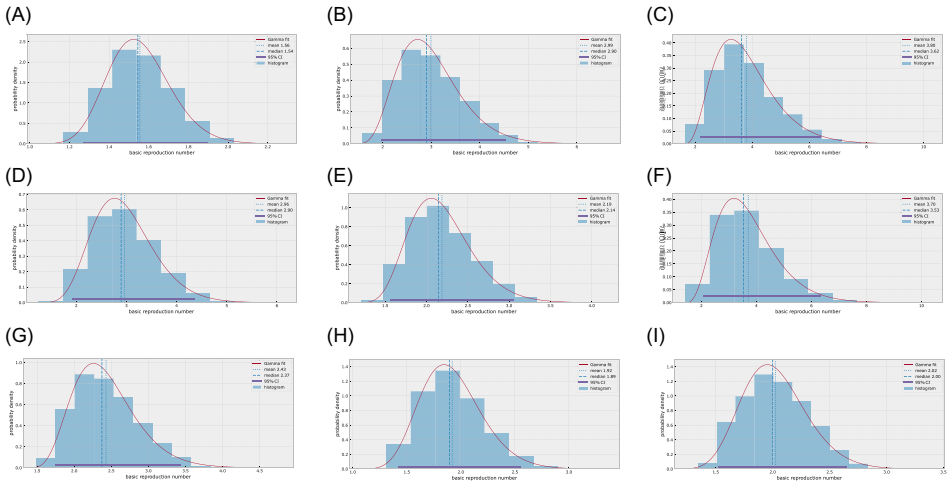


FIGURE 3.36: Monkeypox early R0 estimate in: (A) Belgium (window 67 days from outbreak); (B) France (window 126 days from outbreak); (C) Germany (window 36 days from outbreak); (D) Italy (window 73 days from outbreak); (E) Netherlands (window 53 days from outbreak).

WHO, ECDC, and governments after early reports defined and strongly supported a series of recommendations and guidelines, leaving to local health authorities the possibility of implementing measures in particular epidemiological and/or environmental contexts. The emergence of this "new but old pathology" outside Africa with the expression of a significant number of clinical cases, in the post-COVID-19 pandemic period, has seen a European and World younger populations with a waining or absent specific vaccine immunization and did increase the need of correct and special information. Moreover, the use of our novel early R_0 estimation method appears able to inform public health policy decisions, and contain MPX spreading in naive populations and core groups with risk factors.

3.4 Summary and Outcomes of Chapter

This chapter described the main research contributions during my Ph.D. and reported the main challenges carried out in implementing innovative data analytics applications.

- *Ticket sales prediction and dynamic pricing strategies in public transport* [2] proposes a bus data analysis methodology which shows the correlation rules between travel features and the purchase of a ticket. Such rules are then used to train a machine learning model for predicting whether a user will buy or not a ticket, and to define several dynamic pricing strategies which have the purpose of increasing the tickets sold by the platform and the related revenue of a bus company. Compared to the state of the art, the methodology includes the following innovative aspects: (i) it analyzes the user-generated event logs of a bus ticketing platform; (ii) it is the first methodology in this research field that uses the process mining algorithms to extract valuable information concerning user dynamics and behaviors from a bus ticketing portal; (iii) it discovers the main factors that influence user's buying decisions both to predict whether or not a user will buy a ticket, and to define a dynamic pricing strategy able to maximize the number of tickets sold and the total revenue of a bus company. The results obtained by this study reveal that factors such as occupancy rate and number of days passed from booking to departure, have significant influence on traveler's buying decisions. The dynamic pricing strategy based on these two factors can increase the number of tickets sold by 8% and the total revenue by 10%. Moreover, the methodology reaches an accuracy of 93% in forecasting the purchase of a ticket, showing the effectiveness of the proposed approach and the reliability of results. Using the methodology discussed in this work, the buying behaviour of large communities of people can be analyzed for providing valuable information and high-quality knowledge that are fundamental for the growth of business and organization systems. For the sake of clarity [2] extends the previous work Discovering Travelers' Purchasing Behavior from Public Transport Data [64]. The most significant extension concerns the experimental evaluation, which has been extended by developing a synthetic event generator to create events similar to those registered in log traces extracted from the bus ticketing platform, and by testing different pricing strategies.

- *Paradigms and Models at Run-time – Final Report*[34] presents and discusses the results of the research work done around the programming features and the runtime mechanisms of the Datacentric Programming Model for Exascale Systems (DCEx) framework prototype, a novel programming model for data-intensive applications supporting the scalable use of a massive number of processing elements. The basic idea of DCEx is structuring programs into data-parallel blocks, which are the units of shared- and distributed-memory parallel computation, communication, and migration in the memory/storage hierarchy. Computation threads execute close to data, using near-data synchronization based on the Partitioned Global Address Space (PGAS) model, which assumes the memory is partitioned into a global shared address space and a portion that is local to each process. To show through a real data-intensive application how the DCEx constructs can be used, it is also described a trajectory data analysis application coded in DCEx.

[34] https://www.aspide-project.eu/wp-content/uploads/sites/62/2021/05/D2-6.pdf

- *Using social media for sub-event detection during disasters* [77] presents SEDOM-DD (Sub-Events Detection on sOcial Media During Disasters), a new method that analyzes user posts to discover sub-events that occurred after a disaster (e.g., collapsed buildings, broken gas pipes, floods). SEDOM-DD has been evaluated with datasets of different sizes that contain real posts from social media related to different natural disasters (e.g., earthquakes, floods, and hurricanes). Experiments performed on both real and synthetic datasets showed that SEDOM-DD can identify sub-events with high accuracy. For example, with a percentage of relevant posts of 80% and geotagged posts of 15%, our method detects the sub-events and their areas with an accuracy of 85%, revealing the high accuracy and effectiveness of the proposed approach.

- *Analyzing voter behavior on social media during the 2020 US presidential election campaign* [90] aims to discover the political polarization of US citizens during the 2020 US presidential election, through the analysis of tweets shared on Twitter. In particular, the polarized posts are exploited for investigating the relationship between the political orientation of users and the sentiment they expressed in referring to the different candidates. To assess the statistical significance of the collected data, we studied the age, gender, and geographical distribution of Twitter users for understanding whether they can be considered voters in the political event under analysis. Furthermore, we investigated the temporal dynamics of social media conversation to analyze users' publishing behavior, also studying how it reflects the occurrence of external events (e.g., debates, rallies, etc.). In addition, we detected the main discussion topics, characterizing them from a political viewpoint and showing their weekly evolution within our observation period. Experimental results show the great effectiveness of the proposed methodology, which was able to correctly identify the winning candidate in 10 out of 11 swing states, compared to the average of latest opinion polls before the election, which identified the winner in 9 out of 11 states.

- *Impact of the additional/booster dose of COVID-19 vaccine against severe disease during the epidemic phase characterized by the predominance of the Omicron variant in Italy* [122]. The aim of this paper is to estimate the risk of COVID-19, and the risk of subsequent hospitalization, admission to an intensive care unit (ICU), and death, at different time intervals since vaccination, in individuals fully vaccinated with mRNA vaccines in Italy, both overall and separately by age group. The results of the work underscore the importance of receiving a third dose of COVID-19 mRNA vaccine to prevent both infection and severe COVID-19, especially when the effectiveness of 2 doses is significantly reduced against the Omicron variant. It is recommended that persons aged *ge* 60 years who have received the second dose of COVID-19 vaccine should receive a third dose when eligible. Moreover, all unvaccinated persons should get vaccinated as soon as possible.

- *Predicting the Spread of SARS-CoV-2 in Italian Regions: The Calabria Case Study, February 2020–March 2022* [3]. This paper presents the design and implementation of an approach based on autoregressive models to reliably forecast the spread of COVID-19 in Italian regions. Starting from the database of the Italian Civil Protection Department (DPC), the experimental evaluation was performed on real-world data collected from February 2020 to March 2022, focusing on Calabria. The experimental evaluation shows that the proposed approach achieves a good predictive power for out-of-sample predictions within

one week. This model can be used to provide researchers and policy makers with an open-access, real-time tool that can monitor the most recent COVID-19 pandemic trends in Italian regions with informative graphical outputs and be easily modified to adapt to other countries and future pandemics.

- *Comparing Worldwide, National, and Independent Notifications about Adverse Drug Reactions Due to COVID-19 Vaccines* [121]. This paper defines a general framework for a pharmaceutical data analysis application that detects unusual or unexpected patterns of adverse event reporting that might indicate a possible safety problem with a vaccine. The primary objectives of the work are to: i) detect new, unusual, or rare vaccine adverse events; ii) monitor increases in known adverse events; iii) identify potential patient risk factors for particular types of adverse events; iv) determine and address possible reporting clusters (e.g., suspected localized [temporally or geographically] or product-/batch-/lot-specific adverse event reporting); v) provide a national safety monitoring tool for countering misinformation. As a case study, we present here an analysis of the potential side effects observed following the administration of the COVID-19 vaccines. The experimental evaluation shows that: i) most adverse events can be classified as non-serious and concern muscle/joint pain, chills and nausea, headache, and fatigue; ii) the notification rate is higher in the age group 20–39 years and decreases in older age groups and in very young people.

- *Hepatitis of unknown origin in children: Why and how to create an open access database* [109]. The main goal of the paper is to provide effective, timely and comprehensive data to investigate this emerging virus, i.e., the acute hepatitis of unknown origin in children. These data can be used to: i) conduct real-time situation analysis, and early and timely diagnosis for effective containment; ii) facilitate coordination and collaboration between national and local governments; iii) inform citizens on the spread of the disease in the world; and iv) support governments in the future prevention decisions.

- *Monkeypox: EpiMPX surveillance system and open-data with a special focus on European and Italian epidemic* [110]. During the early stages of outbreaks, obtaining reliable, synthesised data on the characteristics of cases is a challenge, especially at a global scale. This work aims to build a surveillance system that allows researchers and policymakers to monitor the impact of Monkeypox in Europe, with a special focus on the epidemic trends in the Italian regions, based on an open-access database containing information on the laboratory confirmed Monkeypox cases reported by EU/EEA countries and updated once a week. In particular, EpiMPX monitors the space-time distribution of cases and their characteristics, such as age, gender, symptoms, clinical status, and sexual orientation, when available, providing additional data to support the epidemiological understanding of the origins and transmission dynamics of the outbreak. For the first time, Italian data on the regional distribution of cases and gender distribution were graphically evaluated. Data and research results are freely available and can be easily enriched to provide a prompt response to the scientific community and accelerate global efforts to contain the Monkeypox virus.

- *Impact of SARS-CoV-2 infection in pediatric and school settings across COVID-19 waves in Italy.* This work presents a systematic evaluation of the impact of COVID-19 waves on the Italian paediatric population, by providing an overview of the epidemiology and disease outcomes of COVID-19 in children (0-19 years),

and an assessment of the impact on childcare (preschools; ages 0-5 years) and educational (primary and secondary schools; ages 5-19 years) settings due to community transmission caused by SARS-CoV-2 variants. The work collects information from verified sources and transforms that information into consistent, high-granularity data to support epidemiological understanding of transmission dynamics in the Italian paediatric population. The study shows that, compared with earlier variants, the BA.* lineage resulted in notably higher infection and hospitalization rates, especially in the 0-5 cohort not yet vaccinated against COVID-19. With the appearance of more transmissible sublineages, such as BA.5, combinations of non-pharmaceutical interventions (NPIs), in the form of appropriate ventilation of indoor spaces as well as other measures to mitigate transmission risks and further impact on the pediatric population, are recommended.

- *Monkeypox: Early estimation of basic reproduction number R0 in Europe* [146]. This work defines a new epidemiological indice to estimate basic properties of the disease such as the reproduction number R0. Starting from the new surveillance system EpiMPX [110], the study observed the higher R0 in Portugal and Germany, followed by Italy, Spain, and France. The use of this novel R0 estimation method can be used to support the epidemiological understanding of transmission dynamics and contain MPX from spreading in naive populations and core groups with risk factors.

- *Statistical models to predict clinical outcomes with anakinra vs tocilizumab treatments for acute respiratory distress syndrome (ARDS) in COVID19 patients* [145]. The aim of this study is to evaluate the possible differences between two treatment options (anakinra and tocilizumab) in a real-world setting at the beginning of the pandemic, when no other immunomodulatory treatments were in use, in order to predict clinical outcome and define the optimal therapy, the best time to start treatment and the possible risks of serious adverse events. The analysis included 71 anakinra- and 39 tocilizumab-treated patients. Overall 30-day mortality (23% anakinra and 35% tocilizumab; p=0.82) and cumulative risk of death or ventilation-free survival did not differ between treatment groups. Tocilizumab had a more rapid effect on C-reactive protein and lymphocyte levels than anakinra. Baseline D-dimer levels predicted 30-day survival in anakinra-treated patients, baseline C-reactive protein levels predicted survival in tocilizumab-treated patients. Infectious events were more common in tocilizumab recipients (32% vs. 10.5%; p=0.44). These discrepancies may allow clinicians to better characterize possible scenarios in which to use each drug.

- *Challenges and Perspectives of Open Data in Modelling Infectious Diseases* [107]. The COVID-19 pandemic has demonstrated how important real-world data (RWD) are for informing public health policy decisions and improving clinical trials. This paper highlights the importance of open data in an epidemiological context for: (i) conducting real-time situational analysis, tracking contacts, and making early and timely diagnoses for effective containment; (ii) ensuring public trust in government through increased transparency and better communication; (iii) countering misinformation; (iv) identifying and addressing vulnerabilities and specific needs of vulnerable groups by collecting disaggregated data; and (v) supporting effective management of medical supplies and medical equipment requests.

- *The challenges of open data for future epidemic preparedness: The experience of the 2022 Ebolavirus outbreak in Uganda* [113]. Real-time information are needed to provide crucial information to understand transmissibility, risk of geographical spread, routes of transmission, risk factors of infection, and provide the basis for epidemiological modelling that can inform response and containment planning to reduce the burden of disease. We made an effort to build a centralized repository of the Ebola virus cases from verified sources, providing information on dates of symptom onset, locations (aggregated to the district level), and when available, the gender and status of hospitals, reporting bed capacity and isolation unit occupancy rate according to the severity status of the patient. The proposed data repository provides researchers and policymakers timely, complete, and easy-accessible data to monitor the most recent trends of the Ebola outbreak in Ugandan districts with informative graphical outputs. This favors a rapid global response to the disease, enabling governments to prioritize and adjust their decisions quickly and effectively in response to the rapidly evolving emergency, with a solid data basis.

- *Is a new COVID-19 wave coming from China due to an unknown variant of concern? Keep calm and look at the data* [108]. Supported by the GISAID data, we discuss on the more or less probable increase in the number of COVID-19 cases in the very next future. Currently, all the sequenced case in Europe, USA and Asia refer to Omicron-related variants and recombinations. Accordingly, the increasing in the attention to the Covid situation is currently not justified by looking at the available indicators and at the current variants of concern recorded worldwide.

Chapter 4

Conclusions

The scientific community's interest in large-scale data analysis has been continuously increasing over the last decades. The huge amount of data generated and its heterogeneity in terms of format (e.g., video, text, XML, email), represent a challenge to the current storage, process, and analysis capabilities. However, today only one-quarter of digital data available, commonly called Big Data, would be a candidate for analysis and about 5% of that is analyzed. The lack of and pressing need for a systematically developed, scientifically scrutinized and openly available base of analysis tools, is evident from the conducted survey in this thesis. The research problems addressed were expressed in the form of four main research questions, which provide guidelines for the analysis and discussion of the selected scientific use cases:

RQ 1. **"How can we exploit modern massively parallel HPC systems in advanced data analysis pipelines?"**

RQ 2. **"Is it possible to address urban transportation challenges with Big Data?"**

RQ 3. **"How can the implementation of artificial intelligence into Big Data analytics help analyze users on social media?"**

RQ 4. **"Is it possible to use advanced analytics solutions to improve epidemic control?"**

Publications described in Chapter 3 contribute to answering these questions. Paper [2] respond to the posed **RQ 2**, by proposing a data analysis workflow aimed at predicting the ticket sales. A proposed workflow aims at finding patterns in people mobility, which includes social media crawling, data filtering, keyword extraction, a clustering approach to identify regions of interest, and a final frequent pattern mining. The second part of this work proposes a methodology called DA4PT, which implements dynamic pricing through the discovery of purchase factors and a predictive model.

Social data analysis is emerging as a fast growing research area, which is aimed at extracting useful information from this mass of data. It can be used for the analysis of collective sentiments to understanding the behavior of groups of people or the dynamics of public opinion. Papers [77] and [90] include two major applications to address the challenges posed by **RQ 3**: (i) the use of social media for sub-event detection during disasters through the SEDOM-DD platform, and (ii) the analysis of voter behavior on social media during the 2020 US presidential election campaign for investigating the relationship between the political orientation of users and the sentiment they expressed in referring to the different candidates.

The collection and interpretation of data is central to the monitoring and decision-making during the global response to a pandemic. COVID-19 demonstrated how

important open data, i.e., freely accessible and available in a standardized machine-readable format and under a license that allows it to be re-used and re-shared, is for informing public health policy decisions and improving clinical trials. Papers [122], [3], [121], [109], [110], [146], [145], [107], [113], and [108] describe a series of analyses related to infectious disease monitoring that contribute to **RQ 4**. These include the impact of a booster dose of COVID-19 vaccine, predicting the spread of SARS-CoV-2 in Italy, an analysis of adverse drug reactions to COVID-19 vaccines, COVID-19 infection in schools, and other two scenarios regarding the resurgence of of *Mpox*(*monkeypox*) and the unknown epidemic of childhood *hepatitis*.

However, today data are being generated as never before in each of the application domains analyzed. Conventional computer systems are not so powerful to process Big Data and they are not able to scale with the size of problems to be solved. To face with limits of sequential machines when the volume of data to be analyzed is of the order of terabytes or petabytes (billions of tweets or posts), scalable storage and computing solutions must be used. Implementing scalable data analysis applications in high-performance computing systems is a very complex job and it requires high-level fine-grain parallel models, appropriate programming constructs and skills in parallel and distributed programming. The expertise gained from the ASPIDE project is used to guide the design of the DCEx prototype and described in *Paradigms and Models at Run-time – Final Report*[1], which contributes to respond **RQ 1**. In the next years, as parallel technologies advance, Exascale computing systems will be exploited for implementing scalable Big Data analysis in all the areas of science and engineering. Exascale systems refer to high performance computing systems capable of at least one exaFLOPS (i.e., 10^{18} operations per second), so their implementation represents a very significant research and technology challenge. The development of Exascale systems urges to address and solve issues and challenges both at hardware and software level. Indeed it requires to design and implement novel software tools and runtime systems able to manage a very high degree of parallelism, reliability and data locality in extreme scale computers. New programming constructs and runtime mechanisms able to adapt to the most appropriate parallelism degree and communication decomposition for making scalable and reliable data analysis tasks are needed. At the same time, reproducibility in scalable data analysis asks for rich information useful to assure similar results on environments that dynamically may change. Finally, reliable and effective methods for storing, accessing and communicating data, intelligent techniques for massive data analysis and software architectures enabling the scalable extraction of knowledge from data, are needed. To reach this goal, models and technologies enabling cloud computing systems and HPC architectures must be extended/adapted or completely changed to be reliable and scalable on the very large number of processors/cores that compose extreme scale platforms and for supporting the implementation of clever data analysis algorithms that ought to be scalable and dynamic in resource usage. Exascale computing infrastructures will play the role of an extraordinary platform for addressing both the computational and data storage needs of Big Data analysis applications. Pursuing this objective, new Exascale computing infrastructures will appear as the necessary platforms for Big Data analytics in the next decades, and data mining algorithms, tools and applications will be ported on such platforms for implementing extreme data discovery solutions.

[1] https://www.aspide-project.eu/wp-content/uploads/sites/62/2021/05/D2-6.pdf

Appendices

App. A

Projects

A.1 ASPIDE: exAScale ProgramIng models for extreme Data procEssing

Over the past decade, the scientific computing scenario is evolving significantly in two main areas. First, the focus for scientific computing is shifting away from CPU-intensive work, such as large-scale simulations or complex mathematical applications, toward a data-intensive approach. This new paradigm greatly affects the requirements of the underlying architecture, slowly disappearing the classic CPU bottleneck and exposing bottlenecks in I/O systems. Second, the evolution of computing technologies and restrictions on science funding are changing the computing resources available in the scientific community. Exascale computing faces many challenges as parallel codes will have to control millions of threads running on many cores. Such programs will have to avoid synchronization, use less communication and memory, and failures may be more frequent. Although these considerations are critically important for the scalability of future codes, programmers themselves generally want to focus on their own application and not have to deal with these complicated lower-level details. Today there are no programming models and languages that can solve these problems, especially when it comes to data-intensive applications.

The ASPIDE project aims to address these challenges by assisting developers in building data-intensive applications for Exascale systems while ensuring compliance with data management and performance requirements. The main objectives of the project are summarized below.

- **Objective 1.** Design and develop of a new Exascale programming models for extreme data applications.
- **Objective 2.** Build new tools for monitoring extreme data analytics algorithms and applications.
- **Objective 3.** Adapt the data management techniques to the extreme scale applications.
- **Objective 4.** Validate the concepts and tools usefulness through extreme data applications.

The indirect societal benefit of ASPIDE is the development of an energy-aware support system for extreme data processing, which will reduce the energy consumed by supercomputing centers (health benefits through reduced pollution, prevention of global warming through reduced CO2, reduced public costs). In addition, it provides user-friendly APIs and tools for the development of extreme data applications in the field of supercomputing. This allows an excellent opportunity to extend its use to

communities that are now limited by complexity, thus enabling new challenges to be solved and possibly reducing unemployment in application areas.

The evaluation and validation plan for the project activities included our research team's development of an urban informatics application that would be in line with efforts to evolve toward smart cities and provide a demonstration of how urban data can be harnessed for social good. In the following subsections, we report the main method and some of most important functions it recalls during the execution of the Urban Computing application.

A.1.1 Application main

The main method of the Urban Computing application starts collecting the command-line parameters, which are used to configure the execution environment and application workflow. Listing A.1 shows how the main method configures the DCEx [149] environment and creates the Execution Model to execute the different workflow stages.

```
...
using namespace dcex;
//initializes the dcex environment
std::map<int,string> machines = {<list-of-IP-addresses>};
dcex::initialize(machines);
//creates the dcex_grppi runtime
dcex_grppi runtime(node_id, server_id, server_port);
//defines a CArea
CArea ca(carea_size);
std::cout << ca.toString() << std::endl;
//gets the dcex execution model for grppi
auto exec = runtime.get_execution(ca);
...
```

Listing A.1: DCEx code snippet of the *main* function for setting up the environment.

After configuring the execution environment, the main method performs the data processing and analysis functions. As stated in the previous deliverables, the urban computing application works on geotagged social media items. In addition, the area under analysis, on which such social media items have been collected, has been divided into squared cells of equal size. Then, as shown in Listing A.2 the main method performs the following workflow steps:

```
...
// items filtering
filter_geoitems(exec, inputFile, "file:/./geoitems");

// keywords extraction
extract_keywordsInCell(exec, "file:/./geoitems", "file:/./
    keywordsInCell");

// top keywords extraction
extract_topKeywords(exec, "file:/./keywordsInCell", "file:/./
    topKeywords");

// data per keywords extraction
data_grouping(exec, "file:/./geoitems", "file:/./topKeywords", "file
    :/./dataPerKeywords");

// RoIs extraction
extract_rois(exec, "file:/./dataPerKeywords", "file:/./rois");

// Transactions extraction
extract_transactions(exec, "file:/./geoitems", "file:/./rois", "file
    :/./transactions");

```

```
// frequent pattern mining
frequentPatternsMining("file:/.transactions", "file:/./frequentPatterns
    ");
```

LISTING A.2: DCEx code snippet of the *main* method that executes the different workflow steps

A.1.1.1 Data filtering

Listing A.3 shows the GrPPi-DCEx code used for implementing the data filtering step, which parses a set of social media items (in JSON format) and produces a new set by including only the ones that are geolocalized and contain tags. In particular, the filtering method exploits the Pipeline pattern, which accepts: i) an input container from which the items are read, ii) two filtering transformers (i.e., *isGeotagged* and *hasTags*), and iii) an output container to which the filtered geotagged items are written. As shown in Figure A.1, the filtering function processes each social media item by removing those that do not meet the set criteria.

```
template<typename Execution>
void filter_geoitems(Execution & exec, string inputItemsFile, string
    geoitemsFile){
    aspide::text_in_container input_items(inputItemsFile, '\n');
    aspide::output_container geoitems_out(geoitemsFile);
  cout << "filtering geotagged items..." << endl;
  grppi::pipeline(
        exec,
        input_items,
        grppi::keep([&](string line){
            json obj = json::parse(line);
            return jsonDao->isGeotagged(obj);
        }),
        grppi::keep([&](string line){
            json obj = json::parse(line);
            return jsonDao->hasTags(obj);
        }),
        geoitems_out
    );

  cout << "geotagged items filtered." << endl;
}
```

LISTING A.3: DCEx code snippet for data filtering metho

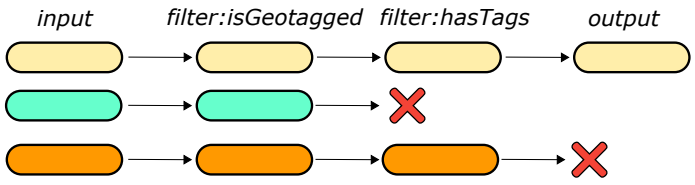


FIGURE A.1: Filtering data flow.

A.1.1.2 Keywords extraction

. Listing A.4 shows the GrPPI-DCEx code for the keywords extraction method. The algorithm exploits the Map-Reduce pattern. Specifically, for each geotagged item, the mapper determines the cell in which the item falls into and keywords it contains. In particular, the mapper processes the tags of each social media items by removing stop words and not useful dictionary terms. The reducer takes care of merging all the

keywords by cell. The output file contains the list of words contained in a cell and their frequency. In particular, such an output file is produced using a format writer object (i.e., *keywordsInCellFormatter*) that receives the output map from a reducer and encode its entries in JSON strings. The reducer exploits an utility function, namely (combineMaps), to merge partial data obtained from the different mappers.

```
template<typename Execution>
void extract_keywordsInCell(Execution & exec, string geoitemsFile,
    string keywordsInCellFile){
    cout << "computing keywordsInCell..." << endl;
    aspide::text_in_container geoitems_in(geoitemsFile, '\n');
    aspide::output_container keywordsInCell_out(keywordsInCellFile);

    aspide::format_writer keywordsInCellFormatter(
        keywordsInCell_out,
        [](unordered_map<Cell, unordered_map<string, int>> result){
            std::ostringstream s;
            for (auto & w : item)
            {
                Cell & cell = w.first;
                unordered_map<string, int> & tagFreq = w.second;
                json jsonPair;
                jsonPair["cell"] = {cell.getLatitude(), cell.
    getLongitude()};
                jsonPair["tagFreq"] = json(tagFreq);
                s << jsonPair.dump() << std::endl;
            }
            return s.str();
        }
    );
    grppi::map_reduce(
        exec,
        geoitems_in,
        keywordsInCellFormatter,
        // mapper
        [](std::string line)-> unordered_map<Cell, unordered_map<string
    , int>>
        {
            unordered_map<Cell, unordered_map<string, int>> m;
            json x = json::parse(line);
            auto key = getCell(x, cellSize);
            auto value = getTagFreq(x);
            m.insert({key,value});
            return m;
        },
        // reducer
        [](unordered_map<Cell, unordered_map<string, int>> &lhs,
            unordered_map<Cell, unordered_map<string, int>> &rhs)
            -> unordered_map<Cell, unordered_map<string, int>>
        {
            for (auto & w : rhs) {
            lhs[w.first] = combineMaps(
                    lhs[w.first],
                    w.second,
                    [](int x, int y){return x+y;}
                );
            }
            return lhs;
        }
    );
  cout << "keywordsInCell computed." << endl;
}

template<typename Map, typename ValueCombiner>
static Map combineMaps(Map & m1, Map & m2, ValueCombiner & combiner){
  Map result;
  result.insert(m1.begin(), m1.end());
```

```
59
60    for(auto kv : m2){
61      if(result.count(kv.first) > 0){
62        result[kv.first] = combiner(result[kv.first], kv.second);
63      } else {
64        result.insert(kv);
65      }
66    }
67    return result;
68  }
```

Listing A.4: DCEx code snippet for keywords extraction method

Figure A.2 illustrates the keywords extraction data flow. The intermediate data generated by mappers are aggregated by reducers so as to generate the final output. As shown, such a flow contains some stages for sorting and distributing intermediate data in order to generate the final output, which are typical of the Map-Reduce execution flow.

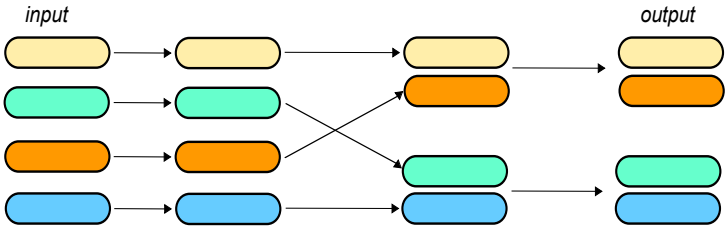


Figure A.2: Keywords extraction data flow.

A.1.1.3 Top Keywords extraction

Listing A.5 shows the code used to compute the most frequent keywords in each cell, starting from the output (i.e., *keywordsInCell*) of the previous step. Also in this case, the algorithm is based on the Map-Reduce parallel pattern. As shown in Figure A.3, for each cell, the mapper searches a number of keywords (i.e., *numTopKeywords*) that occur at least *minOcc* times. The helper function *getTopKeywords*, which is used by the mapper, computes the most frequent keywords in a cell. Finally, the reducer performs a merging of the results coming from the different mappers.

```
1  template<typename Execution>
2  void extract_topKeywords(Execution & exec, string keywordsInCellFile,
       string topKeywordsFile){
3      cout << "extracting top keywords..." << endl;
4      aspide::text_in_container keywordsInCell_in(keywordsInCellFile, '\n
       ');
5    aspide::output_container topKeywords_out(topKeywordsFile);
6
7      aspide::format_writer topKeywordsFormatter(
8          topKeywords_out,
9          [](set<string> out){
10             std::ostringstream s;
11             for (auto & w : out)
12                 s << w << std::endl;
13             return s.str();
14         }
15     );
16     grppi::map_reduce(
17         exec,
18         keywordsInCell_in,
19         topKeywordsFormatter,
20         // mapper
```

```
21          [](string line)->set<string>
22          {
23              json jsonPair = json::parse(line);
24              Cell cell(jsonPair["cell"].value("lat", 0.0), jsonPair["
    cell"].value("long", 0.0));
25              unordered_map<string, int> tagFreq = jsonPair.at("tagFreq")
    .get<unordered_map<string, int>>();
26              pair<Cell, unordered_map<string, int>> cellMap = make_pair(
    cell, tagFreq);
27              pair<Cell, vector<pair<string, int>>> topKeywords =
    topKeywords(cellMap, numTopKeywords, minOcc);
28              set<string> result;
29              if(topKeywords.second.size()>0){ //filter
30                  for(auto & p : topKeywords.second)
31                      result.push_back(p.first);
32              }
33              return result;
34          },
35          // reducer
36          [](set<string> & lhs, set<string> & rhs) -> set<string>
37          {
38              lhs.insert(rhs.begin(), rhs.end());
39              return lhs;
40          }
41      );
42      cout << "top keywords extracted..." << endl;
43  }
```

Listing A.5: DCEx code snippet of the top keywords extraction method

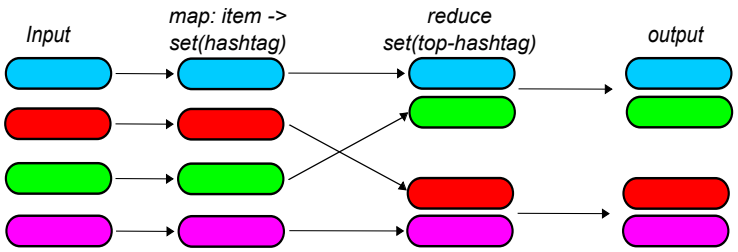


Figure A.3: Top Keywords extraction data flow.

A.1.1.4 Data grouping

Listing A.6 shows the Map-Reduce algorithm used to group the geotagged items that can be associated to each PoI. In particular, the algorithm, given a keyword k that identifies a PoI p, a social media item g is associated to p if it contains k. As shown in Figure A.4, each mapper associates social media items to PoIs and converts them in geographical points, so as to be ready for the clustering step. Then, the reducers aggregates all the points that are associated to the same PoI.

```
1  template<typename Execution>
2  void extract_dataPerKeywords(Execution & exec, string geoitemsFile,
       string topKeywordsFile, string dataPerKeywordsFile){
3      cout << "computing dataPerKeywords..." << endl;
4      aspide::text_in_container geoitems_in(geoitemsFile, '\n');
5      aspide::text_in_container topKeywords_in(topKeywordsFile, '\n');
6      aspide::output_container dataPerKeywords_out(dataPerKeywordsFile);
7
8      aspide::format_writer dataPerKeywordsFormatter(
9          dataPerKeywords_out,
10         [](unordered_map<string, list<ClusterPoint>> & map){
```

```
11              std::ostringstream s;
12              for(auto & kv : map){
13                  json jsonPair;
14                  jsonPair["first"] = kv.first;
15                  jsonPair["second"] = json::array(kv.second);
16                  s << jsonPair.dump() << endl;
17              }
18              return s.str();
19          }
20      );
21
22      grppi::map_reduce(
23          exec,
24          geoitems_in,
25          dataPerKeywordsFormatter,
26          //mapper
27          [&](string line)->unordered_map<string, list<ClusterPoint>> {
28              json x = json::parse(line);
29              vector<pair<string,ClusterPoint>> vect =
    splitDataPerKeywords(exec, x, topKeywords_in);
30
31              unordered_map<string, list<ClusterPoint>> dataPerKeywords;
32              for(pair<string,ClusterPoint> x : vect){
33                  if(dataPerKeywords.count(x.first)){
34                      list<ClusterPoint> current = dataPerKeywords[x.
    first];
35                      current.push_back(x.second);
36                      dataPerKeywords[x.first] = current;
37                  } else {
38                      dataPerKeywords[x.first] = {x.second};
39                  }
40              }
41              return dataPerKeywords;
42          },
43          //reducer
44          [](unordered_map<string, list<ClusterPoint>> & lhs,
    unordered_map<string, list<ClusterPoint>> & rhs)
45              -> unordered_map<string, list<ClusterPoint>>
46          {
47              for(pair<string,list<ClusterPoint>> & x : rhs){
48                  if(lhs.count(x.first)){
49                      list<ClusterPoint> & current = lhs[x.first];
50                      current.insert(x.second.begin(), x.second.end());
51                  } else {
52                      lhs[x.first] = x.second;
53                  }
54              }
55              return lhs;
56          }
57      );
58    cout << "dataPerKeywords computed." << endl;
59  }
```

Listing A.6: DCEx code snippet of the data grouping method

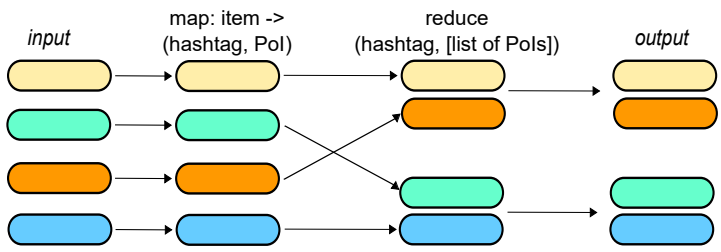


Figure A.4: PoIs extraction data flow.

A.1.1.5 RoI extraction

Listing A.7 shows the GrPPI-DCEx code for the RoIs extraction step. The goal of this step is defining a set of Regions-of-Interest (RoIs) from the points assigned to the different PoIs. The algorithm exploits the Map-Reduce pattern. Specifically, for each keyword (i.e., a PoI), all the associated geographical points are processed by using the DBSCAN algorithm, which aggregates them into clusters (see Figure A.5). Among these clusters, the one containing the greatest number of points is selected and used to represents the Region-of-Interest (RoI) associated to the PoI. Such cluster can be converted into a polygon, and eventually displayed on a map, by using a convex hull algorithm.

```
template<typename Execution>
void extract_rois(Execution & exec, string dataPerKeywordsFile, string roisFile){
    cout << "computing rois..." << endl;

    aspide::text_in_container dataPerKeywords_in(dataPerKeywordsFile, '\n');
    aspide::output_container rois_out(roisFile);
    map<string, Geometry> rois;

    aspide::format_writer roisFormatter(
        rois_out,
        [](map<string, Geometry> & rois_map){
            std::ostringstream s;
            rois = rois_map;
            for(auto & kv : rois_map){
                json jsonPair;
                jsonPair["first"] = kv.first;
                jsonPair["second"] = json(kv.second);
                s << jsonPair.dump() << endl;
            }
            return s.str();
        }
    );
    grppi::map_reduce(
        exec,
        dataPerKeywords_in,
        roisFormatter,
        //mapper
        [](string line)->map<string, Geometry>{
            json jsonPair = json.parse(line);
            string roi_name = jsonPair["first"].get<string>();
            list<ClusterPoint> pois = jsonPair["second"].get<list<ClusterPoint>>();
            Geometry* roi = getRoI(pois, eps, minPts);
            map<string, Geometry> roi_map;
            if(roi != NULL)
                roi_map[roi_name] = *roi;
            return roi_map;
        },
        //reducer
        [](map<string, Geometry> & lhs, map<string, Geometry> & rhs)
            -> map<string, Geometry>
        {
            for(auto & kv : rhs)
                lhs[kv.first] = kv.second;
            return lhs;
        }
    );
  cout << "rois computed." << endl;
}
```

Listing A.7: DCEx code snippet of the RoI extraction method

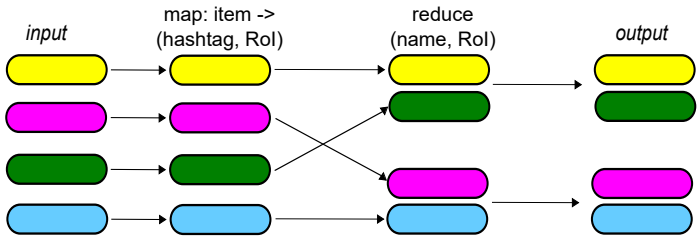


FIGURE A.5: RoI extraction data flow.

A.1.1.6 Transactions extraction

Starting from the RoIs calculated at the previous step, the Map-Reduce algorithm defined in Listing A.8 transforms the locations visited by users from pairs of coordinates (longitude/latitude) into of RoIs. Specifically, for each social media item, a mapper extracts the user and day of the post to create a $< user, day >$ pair. Exploiting the geographical coordinates of each social media item, such pairs are associated to a RoI. Then, the reduce step aggregates intermediate data coming from the mappers (i.e., « $user, day >$, $RoI >$) and produces a dataset containing the list of RoIs visited by each user in the different days (see Figure A.6).

```
template<typename Execution>
void extract_transactions(Execution & exec, string geoitemsFile, string
     roisFile, string transactionsFile){
    cout << "computing roiPaths..." << endl;
    aspide::text_in_container geoitems_in(geoitemsFile, '\n');
    aspide::text_in_container rois_in(roisFile, '\n');
    aspide::output_container transactions_out(transactionsFile);

    aspide::format_writer transactionsFormatter(
        transactions_out,
        [](map<string, list<string>> out){
            std::ostringstream s;
            for(auto & p : out){
                json jsonTransaction = json::array(p.second);
                s << jsonTransaction.dump() << std::endl;
            }
            return s.str();
        }
    );
    grppi::map_reduce(
        exec,
        geoitems_in,
        transactionsFormatter,
        //mapper
        [](string line)->map<string, list<string>>{
            json x = json::parse(line);
            string nameAndDay = getNameAndDay(x);
            string roi = getRoI(exec, item, rois_in);
            map<string, list<string>> out;
            if(roi != "")
                out[nameAndDay]={roi};
            return out;
        },
        //reducer
        [](map<string, list<string>> & lhs, map<string, list<string>> &
     rhs)
            -> map<string, list<string>>
        {
            for(auto & kv : rhs){
                if(!lhs.count(kv.first))
                    lhs[kv.first] = {};
```

```
40                  lhs[kv.first].insert(kv.second.begin(), kv.second.end()
    );
41              }
42              return lhs;
43          }
44      );
45    cout << "transactions computed." << endl;
46 }
```

Listing A.8: DCEx code snippet of the RoI path extraction method

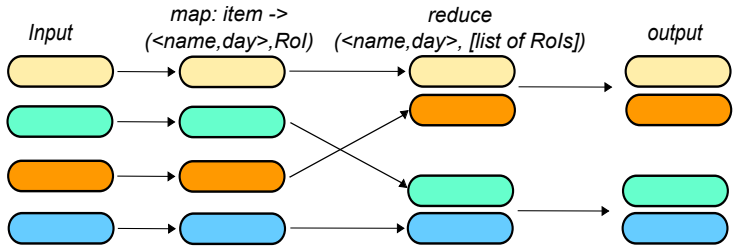


Figure A.6: RoI path extraction data flow.

A.1.1.7 Frequent pattern mining

During the last step of our use case, the FPGrowth algorithm is used for extracting the most frequent sets of RoIs visited by social media users in a single day. Listing A.9 shows three functions used in this step:

- *load_transactions*: it loads the transactions from the container created at the previous step in a local in-memory data structure;
- *write_frequentPatterns*: it writes the computed frequent patterns, received in a local data structure, to the final output container;
- *frequentPatternsMining*: it invokes a function that executes an instance of the FPGrowth algorithm.

Figure A.7 shows the data flow of the frequent pattern mining algorithm. In particular, the FPGrowth algorithm processes the input transactions and distributes results into a distributed storage system.

```
1  template<typename Execution>
2  void load_transactions(string transactionsFile, vector<vector<string>>
       transactions_output){
3      cout << "loading transactions..." << endl;
4      aspide::text_in_container file(transactionsFile,'\n');
5      auto iter = file.begin();
6      while(iter != file.end()){
7          auto line = *iter;
8          json tJson = json::parse(line);
9          vector<string> transaction = tJson.get<vector<string>>();
10         transactions_output.push_back(transaction);
11         ++iter;
12     }
13   cout << "transactions loaded." << endl;
14 }
15
16 template<typename Execution>
17 void write_frequentPatterns(vector<vector<string>> frequentPatterns,
       string patternsFile){
```

```
18      cout << "writing patterns..." << endl;
19      aspide::output_container frequentPatterns_out(patternsFile);
20      aspide::flusher flusher = frequentPatterns_out.get_flusher();
21      for(vector<string> pattern : frequentPatterns){
22          json jsonPattern = json::array(pattern);
23          flusher << jsonPattern.dump();
24      }
25    cout << "patterns written." << endl;
26  }
27
28  template<typename Execution>
29  void frequentPatternsMining(string transactionsFile, string
        patternsFile){
30      cout << "mining frequent patterns..." << std::endl;
31      vector<vector<string>> transactions;
32      vector<pair<vector<string>, uint64_t>> frequentPatterns;
33      // load
34      load_transactions(transactionsFile, transactions);
35      // compute
36      grppi::parallel_execution_native exec();
37      fpgrowth_grppi(exec, transactions, minSupport, frequentPatterns);
38      // write
39      write_frequentPatterns(frequentPatterns, patternsFile);
40      cout << "process successful." << endl;
41  }
```

Listing A.9: DCEx code snippet of the frequent pattern mining function

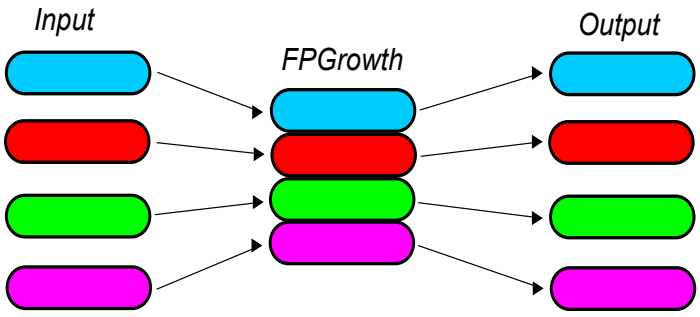


Figure A.7: Frequent pattern mining data flow.

A.2 COVIDA: COrona VIrus Data Analytics

The health emergency we have been experiencing for more than three years now has highlighted how crucial data has become, whether to monitor the emergency, make predictions or just correct information. The more accurate, complete, and timely the data, the more it becomes a tool for understanding where we are and then deciding what to do. COVIDA (CoronaVirus Data Analytics) was born from this consideration. The COVIDA project is dedicated to the collection and visualization of data related to the COVID-19 emergency in Calabria and makes available to the scientific community all the necessary information, such as the total number of infections recorded on the territory, with a series of detailed indications (hospitalized, cured, deceased, number of swabs performed) to monitor and classify the epidemic risk, and the number of subjects vaccinated with the first dose, those vaccinated with a full cycle, the progress of vaccinations by category and age group, to evaluate the progress of the vaccination campaign.

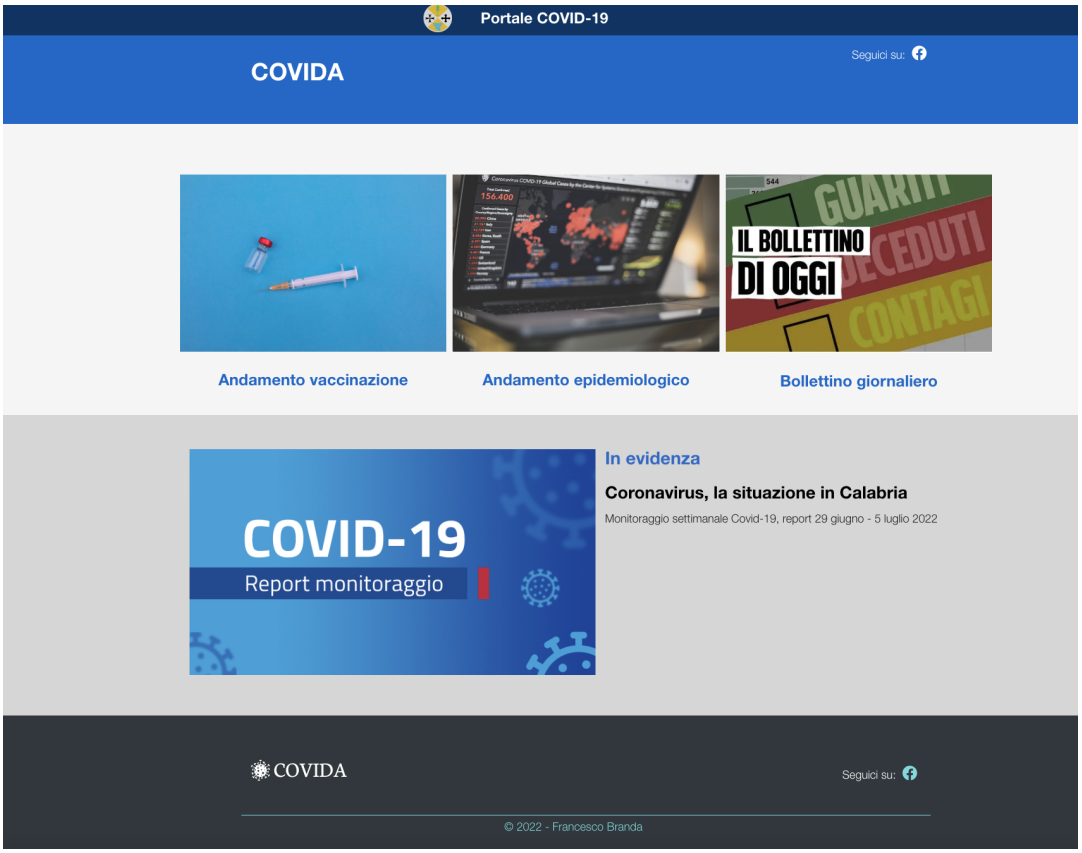


FIGURE A.8: COVIDA platform homepage.

The availability of the daily pandemic trend data in machine readable format was made possible by the implementation of an automatic scraper, i.e., a software program that establishes a connection with the Calabria Region website, extracts the data and transforms them into CSV format, so that their reuse is easier and accessible through the repository[1] on **dati.gov.it** platform, and can be consulted graphically on the COVIDA platform (Figure A.8) at the link https://covida.ml/.

In addition, the project since April 27 has been accepted on the COVID-19 Data Portal[2], a portal developed by ELIXIR-IT in collaboration with CNR, GARR and ISS, which provides information, guidelines, tools and services to support researchers in the process of creating and sharing research data on COVID-19. University of Calabria (UniCal) is currently one of the few Italian universities on the portal.

[1] https://dati.gov.it/view-dataset/dataset?id=3d374641-808c-4802-a6b1-dba96d968cb3
[2] https://www.covid19dataportal.it/data_types/health_data/data/

Bibliography

[1] Loris Belcastro, Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. "Using scalable data mining for predicting flight delays". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.1 (2016), pp. 1–20.

[2] Francesco Branda, Fabrizio Marozzo, and Domenico Talia. "Ticket sales prediction and dynamic pricing strategies in public transport". In: *Big Data and Cognitive Computing* 4.4 (2020), p. 36.

[3] Francesco Branda, Ludovico Abenavoli, Massimo Pierini, and Sandra Mazzoli. "Predicting the Spread of SARS-CoV-2 in Italian Regions: The Calabria Case Study, February 2020–March 2022". In: *Diseases* 10.3 (2022), p. 38.

[4] Vivien Marx. "The big challenges of big data". In: *Nature* 498.7453 (2013), pp. 255–260.

[5] Saint John Walker. "Big Data: A Revolution That Will Transform How We Live, Work, and Think". In: *International Journal of Advertising* 33.1 (2014), pp. 181–183. DOI: 10.2501/IJA-33-1-181-183. eprint: https://doi.org/10.2501/IJA-33-1-181-183. URL: https://doi.org/10.2501/IJA-33-1-181-183.

[6] Thomas Sterling, Maciej Brodowicz, and Matthew Anderson. *High performance computing: modern systems and practices*. Morgan Kaufmann, 2017.

[7] Rajkumar Buyya, Christian Vecchiola, S Thamarai Selvi, and RBVT Selvi. "Data-Intensive Computing: MapReduce Programming". In: *Mastering Cloud Computing*. Morgan Kaufmann, 2013, pp. 253–311.

[8] Saman Amarasinghe, Dan Campbell, William Carlson, Andrew Chien, William Dally, Elmootazbellah Elnohazy, Mary Hall, Robert Harrison, William Harrod, Kerry Hill, et al. "Exascale software study: Software challenges in extreme scale systems". In: *DARPA IPTO, Air Force Research Labs, Tech. Rep* (2009), pp. 1–153.

[9] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. "Apache spark: a unified engine for big data processing". In: *Communications of the ACM* 59.11 (2016), pp. 56–65.

[10] Michael Cox and David Ellsworth. "Application-controlled demand paging for out-of-core visualization". In: *Proceedings. Visualization'97 (Cat. No. 97CB36155)*. IEEE. 1997, pp. 235–244.

[11] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. "The Google file system". In: *Proceedings of the nineteenth ACM symposium on Operating systems principles*. 2003, pp. 29–43.

[12] GhemawatS DeanJ. "MapReduce: simplifieddataprocessing onlargeclusters". In: *CommunicationsoftheACM* 51.1 (2008), p. 107.

[13] Silvina Caíno-Lores, Jesús Carretero, Bogdan Nicolae, Orcun Yildiz, and Tom Peterka. "Spark-diy: A framework for interoperable spark operations with high performance block-based data models". In: *2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT)*. IEEE. 2018, pp. 1–10.

[14] Big Data Value Association et al. "European big data value strategic research and innovation agenda". In: *BDVA, Brussels, Belgium* (2017).

[15] Daniel A Reed and Jack Dongarra. "Exascale computing and big data". In: *Communications of the ACM* 58.7 (2015), pp. 56–68.

[16] Mark Asch, Terry Moore, R Badia, Micah Beck, P Beckman, T Bidot, François Bodin, Franck Cappello, A Choudhary, B De Supinski, et al. "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry". In: *The International Journal of High Performance Computing Applications* 32.4 (2018), pp. 435–479.

[17] Doug Laney et al. "3D data management: Controlling data volume, velocity and variety". In: *META group research note* 6.70 (2001), p. 1.

[18] N Marz and J Warren. "Big Data: Principles and Best Practices of Scalable Realtime Data Systems . 544 Manning Publications Co". In: *Greenwich, CT, USA* 545 (2015).

[19] Loris Belcastro, Fabrizio Marozzo, and Domenico Talia. "Programming models and systems for big data analysis". In: *International Journal of Parallel, Emergent and Distributed Systems* 34.6 (2019), pp. 632–652.

[20] David B Skillicorn and Domenico Talia. "Models and languages for parallel computation". In: *Acm Computing Surveys (Csur)* 30.2 (1998), pp. 123–169.

[21] Sameer Wadkar, Madhu Siddalingaiah, and Jason Venner. *Pro Apache Hadoop*. Springer, 2014.

[22] Michael J Flynn. "Some computer organizations and their effectiveness". In: *IEEE transactions on computers* 100.9 (1972), pp. 948–960.

[23] Marc Bux and Ulf Leser. "Parallelization in scientific workflow management systems". In: *arXiv preprint arXiv:1303.7195* (2013).

[24] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: Simplified data processing on large clusters". In: (2004).

[25] Can Uzunkaya, Tolga Ensari, and Yusuf Kavurucu. "Hadoop ecosystem and its analysis on tweets". In: *Procedia-Social and Behavioral Sciences* 195 (2015), pp. 1890–1897.

[26] Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica. "Discretized Streams: An Efficient and {Fault-Tolerant} Model for Stream Processing on Large Clusters". In: *4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 12)*. 2012.

[27] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster computing with working sets". In: *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*. 2010.

[28] Sherif Sakr, Anna Liu, and Ayman G Fayoumi. "The family of mapreduce and large-scale data processing systems". In: *ACM Computing Surveys (CSUR)* 46.1 (2013), pp. 1–44.

[29] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J Franklin, Scott Shenker, and Ion Stoica. "Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing". In: *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. 2012, pp. 15–28.

[30] Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, and Ion Stoica. "Graphx: A resilient distributed graph system on spark". In: *First international workshop on graph data management experiences and systems*. 2013, pp. 1–6.

[31] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy Katz, Scott Shenker, and Ion Stoica. "Mesos: A Platform for {Fine-Grained} Resource Sharing in the Data Center". In: *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*. 2011.

[32] Jack J Dongarra, Piotr Luszczek, and Antoine Petitet. "The LINPACK benchmark: past, present and future". In: *Concurrency and Computation: practice and experience* 15.9 (2003), pp. 803–820.

[33] David Schneider. "The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second". In: *IEEE Spectrum* 59.1 (2022), pp. 34–35.

[34] Toshiyuki Shimizu. "Supercomputer Fugaku: Co-designed with application developers/researchers". In: *2020 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE. 2020, pp. 1–4.

[35] David A Patterson and John L Hennessy. *Computer organization and design: the hardware/software interface, (Rev. ed. of: Computer organization and design/John L. Hennessy, David A. Patterson. 1998.)* 2012.

[36] *Dell Inc. "Design Principles for HPC"*. https://dl.dell.com/manuals/all-products/esuprt_software/esuprt_it_ops_datcentr_mgmt/high-computing-solution-resources_white-papers48_en-us.pdf (accessed on: 20 November 2022).

[37] William Gropp. "MPICH2: A new start for MPI implementations". In: *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 9th European PVM/MPI Users' Group Meeting Linz, Austria, September 29–Oktober 2, 2002 Proceedings 9*. Springer. 2002, pp. 7–7.

[38] Edgar Gabriel, Graham E Fagg, George Bosilca, Thara Angskun, Jack J Dongarra, Jeffrey M Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, et al. "Open MPI: Goals, concept, and design of a next generation MPI implementation". In: *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting Budapest, Hungary, September 19-22, 2004. Proceedings 11*. Springer. 2004, pp. 97–104.

[39] Michael Bauer, Sean Treichler, Elliott Slaughter, and Alex Aiken. "Legion: Expressing locality and independence with logical regions". In: *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE. 2012, pp. 1–11.

[40] Laxmikant V Kale and Sanjeev Krishnan. "Charm++ a portable concurrent object oriented system based on c++". In: *Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications*. 1993, pp. 91–108.

[41] Seonmyeong Bak, Harshitha Menon, Sam White, Matthias Diener, and Laxmikant Kale. "Integrating openmp into the charm++ programming model". In: *Proceedings of the Third International Workshop on Extreme Scale Programming Models and Middleware*. 2017, pp. 1–7.

[42] Hartmut Kaiser, Thomas Heller, Bryce Adelstein-Lelbach, Adrian Serio, and Dietmar Fey. "Hpx: A task based programming model in a global address space". In: *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models*. 2014, pp. 1–11.

[43] Michael Wong, Andrew Richards, Maria Rovatsou, and Ruyman Reyes. *Khronos's OpenCL SYCL to support heterogeneous devices for C++*. 2016.

[44] Peter Thoman, Philip Salzmann, Biagio Cosenza, and Thomas Fahringer. "Celerity: High-level c++ for accelerator clusters". In: *Euro-Par 2019: Parallel Processing: 25th International Conference on Parallel and Distributed Computing, Göttingen, Germany, August 26–30, 2019, Proceedings 25*. Springer. 2019, pp. 291–303.

[45] Alejandro Fernández, Vicenç Beltran, Xavier Martorell, Rosa M Badia, Eduard Ayguadé, and Jesus Labarta. "Task-based programming with ompss and its application". In: *Euro-Par 2014: Parallel Processing Workshops: Euro-Par 2014 International Workshops, Porto, Portugal, August 25-26, 2014, Revised Selected Papers, Part II 20*. Springer. 2014, pp. 601–612.

[46] Judit Planas, Rosa M Badia, Eduard Ayguadé, and Jesus Labarta. "Hierarchical task-based programming with StarSs". In: *The International Journal of High Performance Computing Applications* 23.3 (2009), pp. 284–299.

[47] H Carter Edwards and Christian R Trott. "Kokkos: Enabling performance portability across manycore architectures". In: *2013 Extreme Scaling Workshop (xsw 2013)*. IEEE. 2013, pp. 18–24.

[48] Bradford L Chamberlain, David Callahan, and Hans P Zima. "Parallel programmability and the chapel language". In: *The International Journal of High Performance Computing Applications* 21.3 (2007), pp. 291–312.

[49] Adrian Prantl, Thomas Epperly, Shams Imam, and Vivek Sarkar. *Interfacing Chapel with traditional HPC programming languages*. Tech. rep. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2011.

[50] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. "StarPU: a unified platform for task scheduling on heterogeneous multicore architectures". In: *Euro-Par 2009 Parallel Processing: 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings 15*. Springer. 2009, pp. 863–874.

[51] Emmanuel Agullo, Olivier Aumage, Mathieu Faverge, Nathalie Furmento, Florent Pruvost, Marc Sergent, and Samuel Paul Thibault. "Achieving high performance on supercomputers with a sequential task-based programming model". In: *IEEE Transactions on Parallel and Distributed Systems* (2017).

[52] Georges Da Costa, Thomas Fahringer, Juan Antonio Rico Gallego, Ivan Grasso, Atanas Hristov, Helen D Karatza, Alexey Lastovetsky, Fabrizio Marozzo, Dana Petcu, Georgios L Stavrinides, et al. "Exascale machines require new programming paradigms and runtimes". In: *Supercomputing frontiers and innovations* 2.2 (2015), pp. 6–27.

[53] Jungwon Kim, Sangmin Seo, Jun Lee, Jeongho Nah, Gangwon Jo, and Jaejin Lee. "SnuCL: an OpenCL framework for heterogeneous CPU/GPU clusters". In: *Proceedings of the 26th ACM international conference on Supercomputing*. 2012, pp. 341–352.

[54] Enqiang Sun, Dana Schaa, Richard Bagley, Norman Rubin, and David Kaeli. "Enabling task-level scheduling on heterogeneous platforms". In: *Proceedings of the 5th Annual Workshop on General Purpose Processing with Graphics Processing Units*. 2012, pp. 84–93.

[55] Ivan Grasso, Simone Pellegrini, Biagio Cosenza, and Thomas Fahringer. "LibWater: heterogeneous distributed computing made easy". In: *Proceedings of the 27th international ACM conference on International conference on supercomputing*. 2013, pp. 161–172.

[56] Magnus Strengert, Christoph Müller, Carsten Dachsbacher, and Thomas Ertl. "CUDASA: Compute Unified Device and Systems Architecture." In: *EGPGV@ Eurographics*. 2008, pp. 49–56.

[57] José Duato, Antonio J Pena, Federico Silla, Rafael Mayo, and Enrique S Quintana-Ortí. "rCUDA: Reducing the number of GPU-based accelerators in high performance clusters". In: *2010 International Conference on High Performance Computing & Simulation*. IEEE. 2010, pp. 224–231.

[58] Ketan Maheshwari and Johan Montagnat. "Scientific workflow development using both visual and script-based representation". In: *2010 6th World Congress on Services*. IEEE. 2010, pp. 328–335.

[59] Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. "JS4Cloud: script-based workflow programming for scalable data analysis on cloud platforms". In: *Concurrency and Computation: Practice and Experience* 27.17 (2015), pp. 5214–5237.

[60] Steve Ashby, Pete Beckman, Jackie Chen, Phil Colella, Bill Collins, Dona Crawford, Jack Dongarra, Doug Kothe, Rusty Lusk, Paul Messina, et al. "The opportunities and challenges of exascale computing". In: *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee* (2010), pp. 1–77.

[61] P Thibodeau. "Supercomputers with 100 million cores coming By 2018". In: *Computerworld* 11.16 (2009), p. 09.

[62] Franck Cappello, Al Geist, Bill Gropp, Laxmikant Kale, Bill Kramer, and Marc Snir. "Toward exascale resilience". In: *The International Journal of High Performance Computing Applications* 23.4 (2009), pp. 374–388.

[63] Domenico Talia. "A view of programming scalable data analysis: from clouds to exascale". In: *Journal of Cloud Computing* 8.1 (2019), pp. 1–16.

[64] Francesco Branda, Fabrizio Marozzo, and Domenico Talia. "Discovering Travelers' Purchasing Behavior from Public Transport Data". In: *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part I 6*. Springer. 2020, pp. 725–736.

[65] Claudia Diamantini, Laura Genga, Fabrizio Marozzo, Domenico Potena, and Paolo Trunfio. "Discovering mobility patterns of instagram users through process mining techniques". In: *2017 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE. 2017, pp. 485–492.

[66] Karl Pearson. "Determination of the coefficient of correlation". In: *Science* 30.757 (1909), pp. 23–25.

[67] Robert M O'brien. "A caution regarding rules of thumb for variance inflation factors". In: *Quality & quantity* 41 (2007), pp. 673–690.

[68] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. "Handling imbalanced datasets: A review". In: *GESTS international transactions on computer science and engineering* 30.1 (2006), pp. 25–36.

[69] Melvin Earl Maron. "Automatic indexing: an experimental inquiry". In: *Journal of the ACM (JACM)* 8.3 (1961), pp. 404–417.

[70] Strother H Walker and David B Duncan. "Estimation of the probability of an event as a function of several independent variables". In: *Biometrika* 54.1-2 (1967), pp. 167–179.

[71] S Rasoul Safavian and David Landgrebe. "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3 (1991), pp. 660–674.

[72] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[73] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[74] Mohsin Raza, Muhammad Awais, Kamran Ali, Nauman Aslam, Vishnu Vardhan Paranthaman, Muhammad Imran, and Farman Ali. "Establishing effective communications in disaster affected areas and artificial intelligence based detection using social media platform". In: *Future Generation Computer Systems* 112 (2020), pp. 1057–1069.

[75] Tahora H Nazer, Guoliang Xue, Yusheng Ji, and Huan Liu. "Intelligent disaster response via social media analysis a survey". In: *ACM SIGKDD Explorations Newsletter* 19.1 (2017), pp. 46–59.

[76] Tomer Simon, Avishay Goldberg, and Bruria Adini. "Socializing in emergencies—A review of the use of social media in emergency situations". In: *International journal of information management* 35.5 (2015), pp. 609–619.

[77] Loris Belcastro, Fabrizio Marozzo, Domenico Talia, Paolo Trunfio, Francesco Branda, Themis Palpanas, and Muhammad Imran. "Using social media for sub-event detection during disasters". In: *Journal of big data* 8.1 (2021), pp. 1–22.

[78] Ismini Lourentzou, Alex Morales, and ChengXiang Zhai. "Text-based geolocation prediction of social media users with neural networks". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 696–705.

[79] Wei Zhang and Judith Gelernter. "Geocoding location expressions in Twitter messages: A preference learning method". In: *Journal of Spatial Information Science* 9 (2014), pp. 37–70.

[80] Loris Belcastro, M Tahar Kechadi, Fabrizio Marozzo, Luca Pastore, Domenico Talia, and Paolo Trunfio. "Parallel extraction of Regions-of-Interest from social media data". In: *Concurrency and Computation: Practice and Experience* 33.8 (2021), e5638.

[81] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[82] Loris Belcastro, Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. "G-RoI: automatic region-of-interest detection driven by geotagged social media data". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12.3 (2018), pp. 1–22.

[83] Koustav Rudra, Pawan Goyal, Niloy Ganguly, Prasenjit Mitra, and Muhammad Imran. "Identifying sub-events and summarizing disaster-related information from microblogs". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 265–274.

[84] B Athira, Josette Jones, Sumam Mary Idicula, Anand Kulanthaivel, and Enming Zhang. "Annotating and detecting topics in social media forum and modelling the annotation to derive directions-a case study". In: *Journal of Big Data* 8.1 (2021), pp. 1–23.

[85] Aliza Sarlan, Chayanit Nadam, and Shuib Basri. "Twitter sentiment analysis". In: *Proceedings of the 6th International conference on Information Technology and Multimedia*. IEEE. 2014, pp. 212–216.

[86] Stuart E Middleton, Giorgos Kordopatis-Zilos, Symeon Papadopoulos, and Yiannis Kompatsiaris. "Location extraction from social media: Geoparsing, location disambiguation, and geotagging". In: *ACM Transactions on Information Systems (TOIS)* 36.4 (2018), pp. 1–27.

[87] Loris Belcastro, Riccardo Cantini, Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. "Learning political polarization on social media using neural networks". In: *IEEE Access* 8 (2020), pp. 47177–47187.

[88] Marco Roccetti, Giovanni Delnevo, Luca Casini, and Silvia Mirri. "An alternative approach to dimension reduction for pareto distributed data: a case study". In: *Journal of big Data* 8.1 (2021), pp. 1–23.

[89] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions". In: *Journal of big Data* 8 (2021), pp. 1–74.

[90] Loris Belcastro, Francesco Branda, Riccardo Cantini, Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. "Analyzing voter behavior on social media during the 2020 US presidential election campaign". In: *Social Network Analysis and Mining* 12.1 (2022), p. 83.

[91] Riccardo Cantini, Fabrizio Marozzo, Giovanni Bruno, and Paolo Trunfio. "Learning sentence-to-hashtags semantic mapping for hashtag recommendation on microblogs". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16.2 (2021), pp. 1–26.

[92] Mike Thelwall. "The Heart and soul of the web? Sentiment strength detection in the social web with SentiStrength". In: *Cyberemotions*. Springer, 2017, pp. 119–134.

[93] Saif M Mohammad and Peter D Turney. "Crowdsourcing a word–emotion association lexicon". In: *Computational intelligence* 29.3 (2013), pp. 436–465.

[94] Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. "Sentiment analysis of short informal texts". In: *Journal of Artificial Intelligence Research* 50 (2014), pp. 723–762.

[95] Saif Mohammad. "Portable features for classifying emotional text". In: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2012, pp. 587–591.

[96] Preslav Nakov, Sara Rosenthal, Svetlana Kiritchenko, Saif M Mohammad, Zornitsa Kozareva, Alan Ritter, Veselin Stoyanov, and Xiaodan Zhu. "Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts". In: *Language Resources and Evaluation* 50.1 (2016), pp. 35–65.

[97] Robert Plutchik. "The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice". In: *American scientist* 89.4 (2001), pp. 344–350.

[98] Nirosha Mahendraratnam, Kerra Mercon, Mira Gill, Laura Benzing, and Mark B McClellan. "Understanding Use of Real-World Data and Real-World Evidence to Support Regulatory Decisions on Medical Product Effectiveness". In: *Clinical Pharmacology & Therapeutics* 111.1 (2022), pp. 150–154.

[99] Jo Best. "Wearable technology: covid-19 and the rise of remote clinical monitoring". In: *bmj* 372 (2021).

[100] Elizabeth J Williamson, Alex J Walker, Krishnan Bhaskaran, Seb Bacon, Chris Bates, Caroline E Morton, Helen J Curtis, Amir Mehrkar, David Evans, Peter Inglesby, et al. "Factors associated with COVID-19-related death using OpenSAFELY". In: *Nature* 584.7821 (2020), pp. 430–436.

[101] Sangchul Park, Gina Jeehyun Choi, and Haksoo Ko. "Information technology–based tracing strategy in response to COVID-19 in South Korea—privacy controversies". In: *Jama* 323.21 (2020), pp. 2129–2130.

[102] Emma C Wall, Mary Wu, Ruth Harvey, Gavin Kelly, Scott Warchal, Chelsea Sawyer, Rodney Daniels, Philip Hobson, Emine Hatipoglu, Yenting Ngai, et al. "Neutralising antibody activity against SARS-CoV-2 VOCs B. 1.617. 2 and B. 1.351 by BNT162b2 vaccination". In: *The Lancet* 397.10292 (2021), pp. 2331–2333.

[103] Adam J Kucharski, Sebastian Funk, and Rosalind M Eggo. "The COVID-19 response illustrates that traditional academic reward structures and metrics do not reflect crucial contributions to modern science". In: *PLoS Biology* 18.10 (2020), e3000913.

[104] Emma J Griffiths, Ruth E Timme, Andrew J Page, Nabil-Fareed Alikhan, Dan Fornika, Finlay Maguire, Catarina Inês Mendes, Simon H Tausch, Allison Black, Thomas R Connor, et al. "The PHA4GE SARS-CoV-2 contextual data specification for open genomic epidemiology". In: (2020).

[105] Adam Siepel. "Challenges in funding and developing genomic software: roots and remedies". In: *Genome biology* 20.1 (2019), p. 147.

[106] Houriiyah Tegally, Eduan Wilkinson, Richard J Lessells, Jennifer Giandhari, Sureshnee Pillay, Nokukhanya Msomi, Koleka Mlisana, Jinal N Bhiman, Anne von Gottberg, Sibongile Walaza, et al. "Sixteen novel lineages of SARS-CoV-2 in South Africa". In: *Nature medicine* 27.3 (2021), pp. 440–446.

[107] Francesco Branda and Giorgia Lodi. *Challenges and Perspectives of Open Data in Modelling Infectious Diseases*. 2023.

[108] Francesco Branda, Fabio Scarpa, Massimo Ciccozzi, and Antonello Maruotti. "Is a new COVID-19 wave coming from China due to an unknown variant of concern? Keep calm and look at the data". In: *Journal of Medical Virology* (2023).

[109] Francesco Branda, Massimo Pierini, and Sandra Mazzoli. "Hepatitis of unknown origin in children: Why and how to create an open access database". In: *Journal of Clinical Virology Plus* 2.3 (2022), p. 100102.

[110] Francesco Branda, Massimo Pierini, and Sandra Mazzoli. "Monkeypox: EpiMPX Surveillance System and Open Data with a Special Focus on European and Italian Epidemic". In: *Journal of Clinical Virology Plus* 2.4 (2022), p. 100114.

[111] Anne Cori, Neil M Ferguson, Christophe Fraser, and Simon Cauchemez. "A new framework and software to estimate time-varying reproduction numbers during epidemics". In: *American journal of epidemiology* 178.9 (2013), pp. 1505–1512.

[112] Giorgio Guzzetta, Alessia Mammone, Federica Ferraro, Anna Caraglia, Alessia Rapiti, Valentina Marziano, Piero Poletti, Danilo Cereda, Francesco Vairo, Giovanna Mattei, et al. "Early estimates on monkeypox incubation period, generation time and reproduction number in Italy, May-June 2022". In: *arXiv preprint arXiv:2207.13483* (2022).

[113] Francesco Branda, Ahmed Mahal, Antonello Maruotti, Massimo Pierini, and Sandra Mazzoli. "The challenges of open data for future epidemic preparedness: The experience of the 2022 Ebolavirus outbreak in Uganda". In: *Frontiers in Pharmacology* 14 (2023). ISSN: 1663-9812. DOI: 10.3389/fphar.2023.1101894. URL: https://www.frontiersin.org/articles/10.3389/fphar.2023.1101894.

[114] Suzanne M Egan, Jennifer Pope, Mary Moloney, Clara Hoyne, and Chloé Beatty. "Missing early education and care during the pandemic: The socio-emotional impact of the COVID-19 crisis on young children". In: *Early Childhood Education Journal* 49.5 (2021), pp. 925–934.

[115] Annaleise R Howard-Jones, Asha C Bowen, Margie Danchin, Archana Koirala, Ketaki Sharma, Daniel K Yeoh, David P Burgner, Nigel W Crawford, Emma Goeman, Paul E Gray, et al. "COVID-19 in children: I. Epidemiology, prevention and indirect impacts". In: *Journal of Paediatrics and Child Health* 58.1 (2022), pp. 39–45.

[116] Baoqi Zeng, Le Gao, Qingxin Zhou, Kai Yu, and Feng Sun. "Effectiveness of COVID-19 vaccines against SARS-CoV-2 variants of concern: a systematic review and meta-analysis". In: *BMC medicine* 20.1 (2022), pp. 1–15.

[117] Sharon HX Tan, Alex R Cook, Derrick Heng, Benjamin Ong, David C Lye, and Kelvin B Tan. "Effectiveness of BNT162b2 vaccine against omicron in children 5 to 11 years of age". In: *New England Journal of Medicine* 387.6 (2022), pp. 525–532.

[118] Elisabetta Colosi, Giulia Bassignana, Alain Barrat, and Vittoria Colizza. "Modelling COVID-19 in school settings to evaluate prevention and control protocols". In: *Anaesthesia, Critical Care & Pain Medicine* 41.2 (2022), p. 101047.

[119] Luigi Schibuola and Chiara Tambani. "High energy efficiency ventilation to limit COVID-19 contagion in school environments". In: *Energy and Buildings* 240 (2021), p. 110882.

[120] S Khare, C Gurry, and L Freitas. "B Schultz". In: *M., Bach, G., Diallo, A., Akite, N., Ho, J., Tc Lee, R., Yeo, W., Core Curation Team, G., and Maurer-Stroh, S* (2021), pp. 1049–1051.

[121] Francesco Branda and Davide Tosi. "Comparing Worldwide, National, and Independent Notifications about Adverse Drug Reactions Due to COVID-19 Vaccines". In: *Information* 13.7 (2022), p. 329.

[122] Francesco Branda. "Impact of the additional/booster dose of COVID-19 vaccine against severe disease during the epidemic phase characterized by the predominance of the Omicron variant in Italy, December 2021-May 2022". In: *medRxiv* (2022), pp. 2022–04.

[123] *Pneumonia of unknown cause – China*. Available online: https://www.who.int/csr/don/05-january-2020-pneumonia-of-unkown-cause-china/en/ (accessed on 22 March 2022).

[124] *Listings of WHO's response to COVID-19*. Available online: https://www.who.int/news-room/detail/29-06-2020-covidtimeline (accessed on 20 March 2022).

[125] *Tracking SARS-CoV-2 variants*. Available online: https://www.who.int/en/activities/tracking-SARS-CoV-2-variants/ (accessed on 20 March 2022).

[126] *Update on Omicron*. Available online: https://www.who.int/news/item/28-11-2021-update-on-omicron (accessed on 20 March 2022).

[127] *Statement on Omicron sublineage BA.2*. Available online: https://www.who.int/news/item/22-02-2022-statement-on-omicron-sublineage-ba.2 (accessed on 20 March 2022).

[128] *Four possible cases of BA.2.2 Omicron sub-variant detected in Thailand no cause for alarm*. Available online: https://www.thaipbsworld.com/four-possible-cases-of-ba-2-2-sub-variant-detected-in-thailand-no-cause-for-alarm/ (accessed on 20 March 2022).

[129] Abiodun M Adeola, Joel O Botai, Hannes Rautenbach, Omolola M Adisa, Katlego P Ncongwane, Christina M Botai, and Temitope C Adebayo-Ojo. "Climatic variables and malaria morbidity in mutale local municipality, South Africa: a 19-year data analysis". In: *International journal of environmental research and public health* 14.11 (2017), p. 1360.

[130] Soo Beom Choi and Insung Ahn. "Forecasting seasonal influenza-like illness in South Korea after 2 and 30 weeks using Google Trends and influenza data from Argentina". In: *PloS one* 15.7 (2020), e0233855.

[131] Junyu He, Jimi He, Zhihai Han, Yue Teng, Wenyi Zhang, and Wenwu Yin. "Environmental Determinants of Hemorrhagic Fever with Renal Syndrome in High-Risk Counties in China: A Time Series Analysis (2002–2012)". In: *The American journal of tropical medicine and hygiene* 99.5 (2018), p. 1262.

[132] Abdulla Watad, Samaa Watad, Naim Mahroum, Kassem Sharif, Howard Amital, Nicola Luigi Bragazzi, and Mohammad Adawi. "Forecasting the West Nile virus in the United States: an extensive novel data streams–based time series analysis and structural equation modeling of related digital searching behavior". In: *JMIR public health and surveillance* 5.1 (2019), e9176.

[133] Yu Duan, Xiao-lei Huang, Yu-jie Wang, Jun-qing Zhang, Qi Zhang, Yue-wen Dang, and Jing Wang. "Impact of meteorological changes on the incidence of scarlet fever in Hefei City, China". In: *International journal of biometeorology* 60.10 (2016), pp. 1543–1550.

[134] Yongqing Zhao, Rendong Li, Juan Qiu, Xiangdong Sun, Lu Gao, and Mingquan Wu. "Prediction of human brucellosis in China Based on temperature and NDVI". In: *International Journal of Environmental Research and Public Health* 16.21 (2019), p. 4289.

[135] Vikas Chaurasia and Saurabh Pal. "COVID-19 pandemic: ARIMA and regression model-based worldwide death cases predictions". In: *SN Computer Science* 1.5 (2020), pp. 1–12.

[136] Cia Vei Tan, Sarbhan Singh, Chee Herng Lai, Ahmed Syahmi Syafiq Md Zamri, Sarat Chandra Dass, Tahir Bin Aris, Hishamshah Mohd Ibrahim, and Balvinder Singh Gill. "Forecasting COVID-19 Case Trends Using SARIMA Models during the Third Wave of COVID-19 in Malaysia". In: *International journal of environmental research and public health* 19.3 (2022), p. 1504.

[137] Gianluca Bonifazi, Luca Lista, Dario Menasce, Mauro Mezzetto, Daniele Pedrini, Roberto Spighi, and Antonio Zoccoli. "A simplified estimate of the effective reproduction number R_t using its relation with the doubling time and application to Italian COVID-19 data". In: *The European Physical Journal Plus* 136.4 (2021), pp. 1–14.

[138] Diletta Cereda, Marcello Tirani, Francesca Rovida, Vittorio Demicheli, Marco Ajelli, Piero Poletti, Frédéric Trentini, Giorgio Guzzetta, Valentina Marziano, Angelica Barone, et al. "The early phase of the COVID-19 outbreak in Lombardy, Italy". In: *arXiv preprint arXiv:2003.09320* (2020).

[139] Serena Colafrancesco, Cristiano Alessandri, Fabrizio Conti, and Roberta Priori. "COVID-19 gone bad: A new character in the spectrum of the hyperferritinemic syndrome?" In: *Autoimmunity reviews* 19.7 (2020), p. 102573.

[140] Luke Adam Monteagudo, Aaron Boothby, and Elie Gertner. "Continuous intravenous anakinra infusion to calm the cytokine storm in macrophage activation syndrome". In: *ACR open rheumatology* 2.5 (2020), pp. 276–282.

[141] Giovanni Guaraldi, Marianna Meschiari, Alessandro Cozzi-Lepri, Jovana Milic, Roberto Tonelli, Marianna Menozzi, Erica Franceschini, Gianluca Cuomo, Gabriella Orlando, Vanni Borghi, et al. "Tocilizumab in patients with severe COVID-19: a retrospective cohort study". In: *The Lancet Rheumatology* 2.8 (2020), e474–e484.

[142] Thomas Huet, Hélène Beaussier, Olivier Voisin, Stéphane Jouveshomme, Gaëlle Dauriat, Isabelle Lazareth, Emmanuelle Sacco, Jean-Marc Naccache, Yvonnick Bézie, Sophie Laplanche, et al. "Anakinra for severe forms of COVID-19: a cohort study". In: *The Lancet Rheumatology* 2.7 (2020), e393–e400.

[143] Olivier Hermine, Xavier Mariette, Pierre-Louis Tharaux, Matthieu Resche-Rigon, Raphaël Porcher, Philippe Ravaud, Serge Bureau, Maxime Dougados, Annick Tibi, Elie Azoulay, et al. "Effect of tocilizumab vs usual care in adults hospitalized with COVID-19 and moderate or severe pneumonia: a randomized clinical trial". In: *JAMA internal medicine* 181.1 (2021), pp. 32–40.

[144] Evdoxia Kyriazopoulou, Garyfallia Poulakou, Haralampos Milionis, Simeon Metallidis, Georgios Adamis, Konstantinos Tsiakos, Archontoula Fragkou, Aggeliki Rapti, Christina Damoulari, Massimo Fantoni, et al. "Early treatment of COVID-19 with anakinra guided by soluble urokinase plasminogen receptor plasma levels: a double-blind, randomized controlled phase 3 trial". In: *Nature medicine* 27.10 (2021), pp. 1752–1760.

[145] Stefania Piconi, Silvia Pontiggia, Marco Franzetti, Francesco Branda, and Davide Tosi. "Statistical models to predict clinical outcomes with anakinra vs. tocilizumab treatments for severe pneumonia in COVID19 patients." In: *European Journal of Internal Medicine* (2023).

[146] Francesco Branda, Massimo Pierini, and Sandra Mazzoli. "Monkeypox: Early estimation of basic reproduction number R0 in Europe". In: *Journal of Medical Virology* 95.1 (2023), e28270.

[147] Thomas Obadia, Romana Haneef, and Pierre-Yves Boëlle. "The R0 package: a toolbox to estimate reproduction numbers for epidemic outbreaks". In: *BMC medical informatics and decision making* 12.1 (2012), pp. 1–9.

[148] Rebecca Grant, Liem-Binh Luong Nguyen, and Romulus Breban. "Modelling human-to-human transmission of monkeypox". In: *Bulletin of the World Health Organization* 98.9 (2020), p. 638.

[149] Domenico Talia, Paolo Trunfio, Fabrizio Marozzo, Loris Belcastro, Javier Garcia-Blas, David del Rio, Philippe Couvée, Gael Goret, Lionel Vincent, Alberto Fernández-Pena, et al. "A novel data-centric programming model for large-scale parallel systems". In: *Euro-Par 2019: Parallel Processing Workshops: Euro-Par 2019 International Workshops, Göttingen, Germany, August 26–30, 2019, Revised Selected Papers 25*. Springer. 2020, pp. 452–463.