Jose María Álvarez-Rodríguez, Giner Alor-Hernández, Jezreel Mejía-Miranda, "Survey of Scientific Programming Techniques for the Management of Data-Intensive Engineering Environments", *Scientific Programming*, vol. 2018, Article ID 8467413, 21 pages, 2018.

DOI: 10.1155/2018/8467413

Hindawi
Scientific Programming
Volume 2018, Article ID 8467413, 21 pages
https://doi.org/10.1155/2018/8467413

Review Article

Survey of Scientific Programming Techniques for the Management of Data-Intensive Engineering Environments

Jose María Álvarez-Rodríguez,[1] **Giner Alor-Hernández**,[2] **and Jezreel Mejía-Miranda**[3]

[1]*Department of Computer Science and Engineering, Universidad Carlos III de Madrid, Madrid, Spain*
[2]*Division of Research and Postgraduate Studies, Tecnológico Nacional de México/I.T., Orizaba, Mexico*
[3]*Centro de Investigación en Matemáticas (CIMAT), Guanajuato, Mexico*

Correspondence should be addressed to Jose María Álvarez-Rodríguez; josemaria.alvarez@uc3m.es

Received 22 February 2018; Accepted 3 October 2018; Published 30 October 2018

Academic Editor: Giuseppe Scanniello

The present paper introduces and reviews existing technology and research works in the field of scientific programming methods and techniques in data-intensive engineering environments. More specifically, this survey aims to collect those relevant approaches that have faced the challenge of delivering more advanced and intelligent methods taking advantage of the existing large datasets. Although existing tools and techniques have demonstrated their ability to manage complex engineering processes for the development and operation of safety-critical systems, there is an emerging need to know how existing computational science methods will behave to manage large amounts of data. That is why, authors review both existing open issues in the context of engineering with special focus on scientific programming techniques and hybrid approaches. 1193 journal papers have been found as the representative in these areas screening 935 to finally make a full review of 122. Afterwards, a comprehensive mapping between techniques and engineering and nonengineering domains has been conducted to classify and perform a meta-analysis of the current state of the art. As the main result of this work, a set of 10 challenges for future data-intensive engineering environments have been outlined.

1. Introduction

Digital technology fueled by software is currently embedded in any task, activity, and process that are done in any organization or even in our daily life activities. The Digital Age has come to stay meaning that any industry or business need to reshape strategies (operational, social, and economic) to become part of what is known as "the 4th Industrial Revolution" or "Industry 4.0" [1]. The first step towards the smart automation and data exchange in industrial environments relies on the application of new techniques to create new technology-driven business opportunities. However, technology is not completely the focus but people: consumers, workers, and partners. The change in the corporate culture through technology is considered a cornerstone to empower people and to drive the change and disruption in a domain turning a traditional organization into a leading digital entity. In this context, it is possible to find several consultancy reports, such as the "2016 Accenture Technology vision report" that outlines five trends to shape this new environment. Organizations shall focus on the creation of data-driven solutions powered by artificial intelligence ("Intelligent Automation") equipping people ("Liquid Workforce") with the required skills to build new execution platforms ("Platform Economy"), boosting disruption ("Predictable Disruption"), and trustworthy digital ecosystems ("Digital Trust").

In the frame of engineering processes and methods, the initiatives of "Industrial Internet" or "Industrial Data Spaces" among others are trying to define the new methodologies and good practices to bring the digital age to the industry and engineering. In this light, a set of technologies such as Security, Big Data, Mobility, Natural Language Processing, Deep Learning, Internet of X (things, people, tools, everything, etc.), User Interfaces, 3D Printing, Virtual Reality, or Cloud Computing are aimed at changing both the

development and production environments of complex products and services. The notion of the cyberphysical system [2] (CPS) is currently gaining momentum to name those engineering systems-combining technologies coming from different disciplines such as mechanical, electrical, and software engineering. CPSs represent the next generation of interconnected complex critical systems in sectors such as automotive, aerospace, railway, or medical devices in which the combination of different engineering areas are governed by software. The increasing complexity in the development of such systems also implies unprecedented levels of interaction between models and formalisms from the inception of the system to the production, distribution, support, and decommissioning stages.

In order to tackle the needs of new engineering environments, collaborative engineering [3], "*concept of optimizing engineering processes with objectives for better product quality, shorter lead time, more competitive cost, and higher customer satisfaction*" [4], represents a shifting paradigm from islands of domain knowledge to an interconnected knowledge graph of services [5], engineering processes, and people. Methods such as requirements engineering, modelling, and simulation or cosimulation to support processes such as analysis, design, traceability, verification, validation, or certification demand now integration and interoperability in the development toolchain to automatically exchange data, information, and knowledge. That is why, last times have seen the emergence of model-based systems engineering (MBSE) [6] as a complete methodology to address the challenge of unifying the techniques, methods, and tools to support the whole specification process of a system. In the context of the well-known Vee life cycle model, it means that there is "formalized application of modelling" (http://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf?sfvrsn=0) to support the left-hand side of this system life cycle implying that any process, task, or activity will generate different system artifacts but all of them represented as a model. In this manner, the current practice in the Systems Engineering discipline is improved through the creation of a continuous and collaborative engineering environment easing the interaction and communication between both tools and people.

However, there is much more at stake than the connection of tools and people, the increasing complexity of products and services also requires the improvement and extension of existing techniques. In this context, scientific programming techniques such as computational modelling and simulation, numerical and nonnumerical algorithms, or linear programming to name a few have been largely used and studied [7, 8] and applied to solve complex problems in disciplines such as natural sciences, engineering, or social sciences. Furthermore, the exponential increase of data has created a new complete environment in which computational scientists and practitioners must concentrate [8] not only in the formulation of hypotheses, creation of models, and execution of experiments but also in the complexity of techniques and combination of tools [9] to deal with large amounts of data.

Considering the current digital transformation in the industry, the need of producing complex CPS through the collaboration between different engineering domains and the increasing amounts of data, this survey looks for summarizing the last existing studies applying scientific programming techniques in the context of data-intensive engineering environments.

2. Background

The development of a complex system such as CPS requires the involvement of hundreds of engineers working with different tools and generating thousands of system artifacts such as requirements specifications, test cases, or models (logical, physical, simulation, etc.). These large amounts of data must then be integrated together to give support to those engineering processes that require a holistic view of a system such as traceability management or verification. In this frame, data management techniques are becoming a cornerstone to the proper exploitation of the underlying data and for the successful development of the system.

Last years have also seen a large body of work in the field of Big Data covering from the creation of tools and architectural frameworks to its application to different domains [10–12] such as social network analysis [13], bioinformatics [14], earth science [15], e-government [16], e-health [17, 18], or e-tourism [19]. More specifically, in the context of engineering and industry, some works have been reported [20] by large companies such as Teradata in particular domains such as maintenance of aircraft engines. All the work around tries to provide a response for data-centric environments in which it is necessary to deal with the well-known V's of a data ecosystem: variety, volume, velocity, and veracity. Big Data technology has been successfully applied to deal with the large amounts of data that are usually created during simulation processes of complex critical systems such as CPS. Moreover, last times have seen the emergence of a new notion "digital twin." A digital twin is defined as a "*digital replica of physical assets, processes and systems*" that looks for replying, in a digital environment, the same working conditions than a physical environment. This approach is being used to improve the verification and validation stages of CPS such as smart cars. For instance, autonomous cars need to pass certification processes in which manufacturers must demonstrate the proper behavior of the car under certain circumstances and consider the new artificial intelligence capabilities. Since it is not completely possible to create a complete digital environment for these large simulations, there is currently a trend to design a kind of validation loop in which real data are used to feed synthetic data. From a data management perspective, this approach implies the need of providing a data life cycle management process to represent, store, access, and enrich data. After several simulations, data gathered from car sensors under real conditions can be over hundreds of petabytes. Similar approaches are used to validate aircraft engines or wind turbines.

Thus, data storage systems and processing frameworks have been developed [21] demonstrating the viability of a technology that can now be considered mature. NoSQL data storage systems [22] based on different representation mechanisms such as key-value stores, documents, distributed

files, wide column stores, graphs, object databases, and tuple stores such as Apache Hive, MongoDB, ArangoDB, Apache Cassandra, Neo4J, Redis, Virtuoso, or CouchDB can be found to handle large volumes of evolving data. In the case of processing frameworks, technology offering capabilities to process data under different models (stream, batch, or event) such as the Apache projects: Hadoop (and its ecosystem of technology), Storm, Samza, Spark, or Flink can also be found. Furthermore, most of them usually offer us not just a distributed computational model to tackle the well-known CAP theorem [23] but a set of libraries [24] for implementing applications on top of existing machine learning techniques [25] (e.g., Mlib in Spark or FlinkML in Flink) or large-scale graph processing [26] (e.g., GraphX in Spark).

This plethora of tools and technology has also generated the development of technology and tools for the management of Big Data infrastructures and resources such as Apache Messos or YARN and the emergence of companies and commercial tools such as Cloudera, MapR, Teradata, or HortonWorks (apart from the toolsets offered by the big software players such as Amazon, Google, IBM, Oracle, or SAP). Moreover, Big Data technology has found in the cloud computing area a good travelling companion [27, 28] since the cloud provides a powerful and massive scale technology for complex computation decreasing the cost of hardware and software maintenance. In this sense, the current challenges rely on the automatic deployment of Big Data infrastructures under different topologies and technologies keeping nonfunctional aspects such as scalability, availability, security, regulatory and legal issues, or data quality as main drivers for research and innovation.

Once Big Data and cloud computing technology have been briefly summarized, it is important to highlight what is being considered one of the next big things in this area: the combination of high-performance computing (HPC) and Big Data technology [29]. The growing interest in this area [30] looks for joining the efforts to shift the paradigm of large and complex computations to a kind of Big Data analysis problem. In this way, new computational and programming models taking advantage of new data storage systems and extensions in programming languages such as R, Fortran, or Python are becoming critical to address the objective of performing time-consuming tasks (computational complexity) over large amounts of data.

That is why, the main aim of this review is to examine the existing works in the field of Big Data and scientific programming in the engineering discipline. A better understanding of this new environment may potentially allow us to address existing challenges in the new computational models to be designed.

3. Review Protocol

As it has been previously introduced, a good number of systematic reviews can be found in the field of Big Data [18, 27] and specific domains such as the health sector [17, 18, 31] the same manner, scientific programming techniques have been widely studied and reviewed [7, 8]. However, the application of Big Data technology and scientific programming techniques to the engineering domain has not been fully reviewed, and it is not easy to draw a comprehensive picture of the current state of the art apart from works in specific domains such as climate [32] or astronomy [20]. That is why, this work aims at providing an objective, contemporary, and high-quality review of the existing works following the formal procedures for conducting systematic reviews [31, 33]. The first step relies then on the definition of the systematic review protocol (SRL) as follows:

(1) *Research Question*. The main objective of this work is to provide a response to the following research question:

Which are the last techniques, methods, algorithms, architectures, and infrastructures in scientific programming/computing for the management, exploitation, inference, and reasoning of data in engineering environments?

To formulate a researchable and significant question, the PICO model is used to specify the different elements of the query to be formulated; see Table 1 outlining the description of each PICO element.

(2) *Search String*. According to the PICO model and the application of natural language processing techniques to add new terms to the query, the next search string has been formulated; see Table 2.

(3) *Bibliographic Database Selection*. In this case, common and large bibliographic databases in the field of computer science have been selected as in other previous survey works [34]:

(a) ACM Digital Library (ACM): this library comprises the most comprehensive collection of full-text articles in the fields of computing and information technology. The ACM Digital Library consists of 54 journals, 8 niche technology magazines, 37 special interest group newsletters, +275 conference proceedings volumes (per year), and over 2,237,215 records in the Guide to Computing Literature Index.

(b) IEEE Xplore Digital (IEEE): the IEEE Xplore® digital library provides "*access to the cutting-edge journals, conference proceedings, standards, and online educational courses that define technology today.*" "*The content in IEEE Xplore comprises 195 + journals, 800+ conferences, 6,200+ technical standards, approximately 2,400 books, and 425+ educational courses. Approximately 20,000 new documents are added to IEEE Xplore each month.*"

(c) ScienceDirect (Elsevier): ScienceDirect "*is a leading full-text scientific database offering science, medical, and technical (STM) journal articles and book chapters from more than 3,800 peer-reviewed journals and over 37,000 books. There are currently more than 11 million articles/chapters, a content base that is growing at a rate of almost 0.5 million additions per year.*"

Table 1: The PICO model applied to formulate the research question.

PICO element	Description
Population	Engineering environments and methods
Intervention	Scientific programming/computing techniques, algorithms, methods, architectures, and infrastructures to deal with data-intensive engineering
Comparison	Performance measures, benchmarks, and datasets
Outcomes	New and collaborative techniques, algorithms, architectures, infrastructures, domains, and case studies

Table 2: Search string including operators.

(scientific AND (programming OR computing) AND (model OR technique OR method OR algorithms OR architecture OR infrastructure) AND (data OR "big data") AND engineering)

(d) SpringerLink (Springer): ScienceDirect is a leading full-text scientific database offering science, medical, and technical (STM) journal articles and book chapters. It comprises 12,258,260 resources consisting of a 52% of articles, 34% of chapters, 8% of conference papers, and 4% reference work entries.

Moreover, some aggregation services such as Google Scholar and DBLP have also been checked with the aim of getting those results that can be found on other sources such as webpages or nonscientific articles. However, these aggregation services can also include works that have been published under different formats such as technical reports, conference, and journal papers, so it is important to carefully select the complete and most up-to-date version of the works to avoid duplicates. That is why, during the selection procedure, works were selected removing potential duplicates to provide a whole and unique picture of the research landscape.

(4) *Inclusion Criteria*. This review collects studies that have been published in English as a journal paper. English has been selected as the target language since relevant journals in these topics mainly include works to get the attraction of researchers and practitioners at a worldwide scale. Although many conferences and workshops in the field of data science and engineering have emerged during the last times generating a quite good number of papers, this review looks for works with strong foundations as those available as journal papers. In the case of the timeframe, a range of 5 years (2012–2017) has been defined to include relevant and up-to-date papers. Furthermore, a special attention has been paid to remove duplicate studies or extensions that can appear in several publications keeping in that case the most complete and up-to-date version of the work.

In terms of the contents, studies to be included in the review shall contain information about the scientific programming methods, Big Data technologies, and datasets (if any) that have been used. Since a huge amount of works (thousands) can be found in the field of Big Data technologies, only those papers related to scientific programming techniques shall be selected. However, in some cases, such as the use of hybrid approaches between scientific programming and artificial intelligence implies the need of extending the scope to explore other possibilities that may have impact in the current state of the art of scientific programming for data-intensive engineering environments.

In the same manner, novel techniques applied to other domains such as earth sciences, biology, meteorology, or social sciences may be included to check whether such works represent progress regarding the current techniques.

As a result of the application of the inclusion criteria, Figure 1 shows the number of papers published per year and database (Figure 2). According to the trendline, there is increasing interest in the creation of know-how around these techniques, methods and technologies having in the last 5 years an increment of a 56% of the number of published papers just in journals what indicates a rising demand of new techniques to take advantage of all the technology and approaches that are currently available. In the case of the databases, Springer is becoming a key player promoting the research and innovation in the data science and engineering areas, Figures 3 and 4 also shows the distribution of papers per year and database. However, the type of search engine in each database may have impact in the number of articles that are returned as the result explaining the big difference between Springer and the rest of publishers.

(5) *Selection Procedure*. Those papers fulfilling the inclusion criteria are summarized in a spreadsheet including the main facts of each work. The proposed approach seeks for ensuring that papers suitable for review must first pass a quality check. More specifically, the identifier, title, abstract, keywords, conclusions, methods, technology, measures, scope, and domain are extracted to create a table of metainformation that serve us to present the information to two different experts to decide whether the work is finally selected for review or not. To do so, a quantitative value is assigned to each entry being that: 1: applicable; 0.5: unknown; 0: not applicable. This approach allows us to accomplish with the two-fold objective of having a qualitative and quantitative selection procedure and follow the guidelines established in the PRISMA model [35].

As a result, Figure 5 depicts the number of works that have been identified after searching in the database (1193), the number of works that have been screened successfully (935) and excluded (258), the number of works selected for quality assessment (322) and, finally, the number of works that have been included in the review (114).

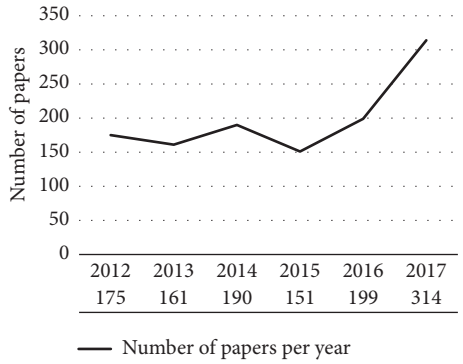

Figure 1: Distribution of the number of papers per year.

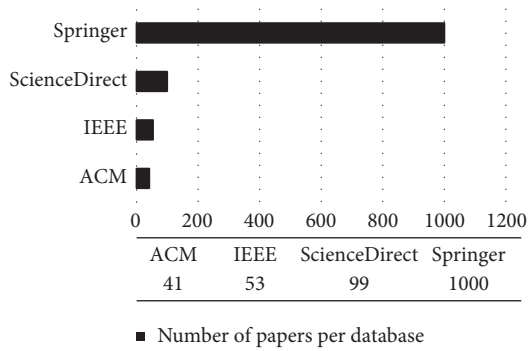

Figure 2: Distribution of the number of papers per database.

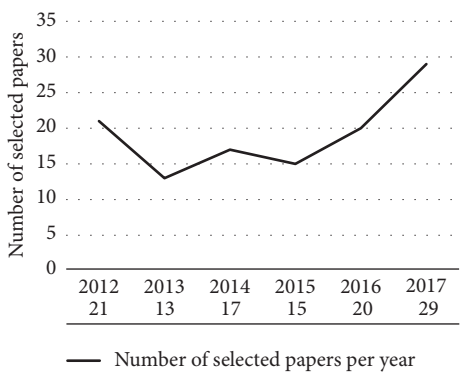

Figure 3: Distribution of the number of selected papers per year.

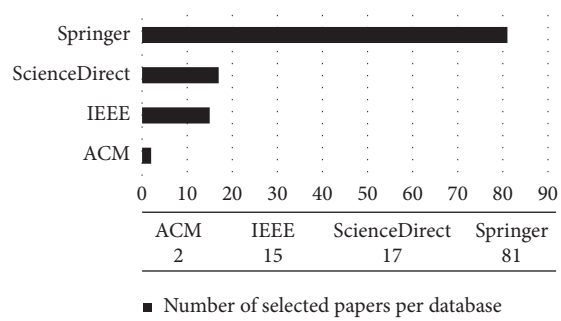

Figure 4: Distribution of the number of selected papers per database.

(6) *Evaluation Procedure and Data Extraction Strategy*. In this step, the selected papers are evaluated according to a Likert-scale 1–6 being 1 applicable but not fully representative for the scientific programming community and 6 applicable and fully representative. In this way, it is possible to create a sort of the selected papers to finally select those which are beyond a value of 4. In this manner, only papers beyond the average are then evaluated. To provide a proper classification and mapping of the works, the topics covered by the works are organized in broader themes. This classification looks for organizing the works providing a comprehensive view of the existing state of the art. In case of a work that can be applied to different domains, it is classified as a general domain work.

(7) *Synthesis of Data and Analysis of Results*. The synthesis of findings in different studies is hard to draw since many factors can affect a data-intensive environment. In this work, after passing the qualitative and quantitative evaluation, we have realized that the best approach is to group the different works according to themes relevant to a specific field. In this manner, we have selected, following a typical Big Data architecture [36], the next dimensions: infrastructure, software technique/method, and application domain. All data generated during this review are publicly available in the following repository: https://github.com/catedra-rtve-uc3m/smart-public/tree/master/papers/data-scientific-programming-review-2018.

4. Analysis of Results

In the field of data-intensive environments, it is necessary to separate the different responsibilities and aspects of both hardware and software components. To do so, as it has been previously introduced, the NIST (National Institute of Standards and Technology) has published the "Big Data Interoperability Framework [36]" allowing us to perfectly define and sort responsibilities and aspects of functional blocks in a Big Data environment.

In this work, we take as a reference this architectural framework simplifying and grouping works in three big themes: hardware infrastructures, software components being that techniques, methods, algorithms, and libraries and applications.

In the first case, hardware resources are becoming critical to offer high-performance computing (HPC) capabilities to data-intensive environments [29]. Last times have seen the growing interest to take advantage of new hardware infrastructures and techniques to optimize the execution of time-consuming applications. In this sense, the use of GPUs (graphical processing units) is a cornerstone to accelerate data-intensive applications being a critical component for performing complex tasks keeping a balance between cost and performance. More specifically, GPUs and CUDA (GPU and a programming language) are being currently used to train machine learning algorithms decreasing the training time from weeks to days or even hours. The potential use of

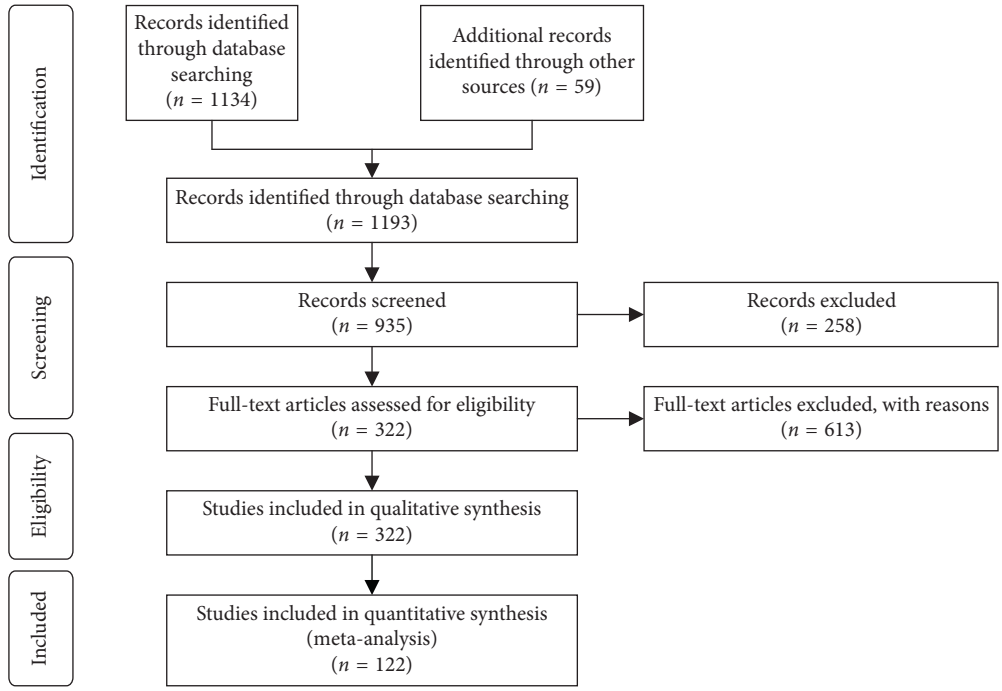

Figure 5: PRISMA flow diagram of the systematic review.

many processing cores enables algorithms to parallelize data and processing in a real multithread environment taking advantage of a high computing power, a large memory bandwidth and, in general, a very low power consumption. However, the main drawback of this type of architecture relies on the difficulties to design and code algorithms due to the expressivity of the language and the need of handling interdependencies in a highly parallelized environment.

FPGAs (field-programmable gate arrays) are other type of integrated circuit and programming language (hardware description language, a kind of block-based language) to run jobs repeatedly avoiding unnecessary processing cycles. It perfectly fits to problems in which repetitive tasks must be done several times such as pattern matching. However, the on-the-fly configuration of a new algorithm is not flexible as in GPUs or CPUs, and more advanced programmer toolkits are still missing.

Multiprocessor architectures are hardware infrastructures based on the use of several CPUs in the same computer system (commonly a super computer). According to the well-known Flyinn's taxonomy, multiprocessors-based systems lay on different configurations depending on how data are input and processed. Shared memory and message passing are also typical synchronization mechanisms of these computer systems that usually present symmetry in the type of CPUs and a set of primitives to easily process data, e.g., vectors.

Finally, grid computing represents a set of geographically distributed computer resources that are combined to reach a common goal. Several configurations and processing schemes can be found in each node for processing different jobs and workloads. To manage a grid computing system, a general-purpose middleware is used to synchronize both data sharing and job processing what is considered critical in a data-intensive environment where a data management strategy is required to organize, store, and share data items across the grid computing system.

As a final remark, parallelization and distribution are again general approaches [12] for improving performance while executing time-consuming tasks. This situation is also applicable to data-intensive environments where the combination of large amounts of data and complex calculations require new strategies and configurations to avoid the bottlenecks produced by the need of synchronizing data.

Secondly, techniques, methods, and algorithms for scientific computation are generally complex implying the need of optimization techniques to avoid recomputation. When large amounts of data meet this type of techniques, this situation is becoming critical since complexity is increased due to the processing of large inputs and intermediate results (e.g., cosimulation in engineering). That is why, it is possible to find combination of different techniques trying to tackle the problem of implementing a technique while keeping a reasonable balance between time and cost.

In this sense, after screening existing works, a plethora of techniques and methods was found trying to provide innovative ways of solving existing problems. With the aim of grouping and sorting them adequately, a first classification was made according to the 2012 ACM Computing Classification System at different levels of granularity. To do so, techniques and methods were grouped under general themes trying to make a comprehensive and clear distinction between the type of problem and the technique and/or method. In some cases, it was not completely possible to make such clear distinction since the same

technique can be applied to solve a huge set of problems (e.g., deep learning).

(i) Artificial intelligence (AI): nowadays, artificial intelligence techniques are spread to solve a huge variety of problems from space observation to autonomous driving. Robotics, medicine, and many other domains are being affected by AI in which more computational power and a change of the people mindset is still required. This category technique tries to equip machines with intelligence to be able to perceive its environment and take actions reaching specific goals. Some techniques such as deep learning, fuzzy logic, gene programming, neural networks, or planning methods fall in this category. Although AI techniques can directly be applied to solve particular problems, in the context of this review, they are usually combined with other techniques to optimize the resolution of a more complex problem.

(ii) Computation architecture: as it has been previously introduced, a hardware setting may have a critical impact in terms of time and costs. Instead of lowering the configuration to a hardware level, software-defined architectures and methods are used to provide a virtual infrastructure that practitioners can easily use and configure. Infrastructure customization, scheduling techniques, and high-level design of workflows can be found as abstract methods to manage complex hardware settings.

(iii) Computation model: a computational model can be defined as the formal model that is applied to solve a complex system. In this case, the computational models that have been selected refer to the different strategies to manage, synchronize, and process data and tasks. To do so, event calculus, message passing interface (MPI), parallel programming, query distribution, and stream processing are the main models that have been found in the review.

(iv) Computational science: complex problems may require the collaboration of different disciplines to provide innovative solutions. Computational science, and more specifically its application to the engineering domain, is the core of this review. The use of models, simulations, and many other techniques such as Euler models or statistical methods are widely spread to understand and model the behavior of complex systems such as those that can be found in engineering. However, it seems that the combination of these techniques under different hardware settings and software models is being a trend that must be systematically observed and evaluated.

(v) Graph theory: last times have seen an increasing use of graph theory to model problems in domains such as social network analysis, telecommunications, or biology. Any problem that can be modeled as a network is a good candidate to apply graph theory techniques understanding how the network is built and to infer new relationships through the exploitation of the underlying knowledge coded in the different nodes, relationships, and layers. Multilayered and multimode networks are a common logical representation for relational data that can be used to perform tasks of structural analysis [37] or relational learning [38]. It is also well known that techniques and methods in this domain [39] are usually complex and time-consuming. That is why, new data-intensive environments may require taking advantage of techniques such as complex network analysis or automata, but they will also require a good number of computational resources.

(vi) Engineering: building on the notions presented in computational science, the different disciplines in engineering make intensive use of formal models, simulations, and numerical and nonnumerical methods for building complex systems. In this case, methods such as finite elements and simulation have been identified as critical techniques already available in the literature as candidates to make use of data-intensive environments.

(vii) Machine learning: it is considered a brand of artificial intelligence and defined as a set of techniques to equip machines with intelligence in changing environments. Basically, a model is built through a training method that it is then evaluated in the testing phase. Large amounts of data directly impact the performance of both phases, so it is possible to find libraries that can work stand-alone (e.g., Weka, Python, and Scikit) or in a distributed environment (e.g., Spark Mlib). Machine learning techniques based on unsupervised and supervised learning methods are commonly applied to solve multiple problems in classification, computer vision, data mining, natural language processing and text mining, sentiment analysis, opinion mining, speech systems, pattern recognition, or question answering to name just a few. Once a model is created using a concrete technique, it can easily be used to perform predictive and prescriptive analysis processes. Potential applications of the techniques falling in this category range from social network, medicine, or biology to logistics or manufacturing. In this review, Bayesian machine learning, data mining, information fusion, pattern recognition, predictive models, support vector machines, and regression models have been found as representatives and popular types of problems and techniques in machine learning for engineering in combination with other scientific programming mechanisms.

(viii) Mathematics and applied mathematics: the use of strong theoretical foundations is critical to build complex systems such as those in engineering.

Mathematics (in this review, mainly Algebra) and, more specifically, applied mathematics offer use of a set of formal methods in science, engineering, computer science, and industry to solve practical problems. Gradient descent optimization, integer linear programming, linear algebra/solvers, linear and nonlinear programming, matrix calculation, numerical methods, and symbolic execution represent a set of common types of problems and techniques widely studied and applied to the scientific programming area.

(ix) Programming techniques: programming as a practice represents the main driver of implementation of a system after the analysis and design processes. Different programming techniques can be found to optimize the development of algorithms and take advantage of the different programming language primitives. In this sense, constraint programming, cube computation, dynamic programming, domain specific languages (DSL), and stochastic programming have been found as representative programming techniques in a scientific environment to take the most data and perform tasks such as analysis or data quality checking.

Once the hardware settings and the main methods, techniques, and algorithms at a software level have been presented, a set of tentative application domains can be outlined. In this light, the review has found two main directions: (1) engineering domains such as Aerospace, Automotive, Civil engineering, Cyberphysical systems, Feature engineering, Industrial applications, Iron mining, Manufacturing or Nuclear domain, and (2) other domains such as Earth science, Geometry, Image analysis, Internet of things, Medicine, Scientific research, Social Sciences, or Spatial modelling.

All these domains share some common characteristics and needs strong mathematical foundations to govern the different systems, processing of large amounts of data, application of methods, techniques, and models such as simulation and, in general, criticality (they are considered safety-critical or life-critical systems; a failure can imply death or serious injury to people and severe damage to equipment or environment). That is why, the emerging use of innovative hardware architectures and exploitation of data must be controlled by strict requirements that make scientific programming techniques even more important than in any other application domain.

4.1. Hardware Infrastructure for Data-Intensive Engineering Environments. The previous section has introduced the main hardware settings for data-intensive environments. In Table 3, a mapping between the hardware architectures and the different software-based techniques is presented.

The use of GPUs (and CUDA) [41–50, 54–56, 58, 60] is a widely accepted hardware setting for providing a hardware infrastructure to process large amounts of data regardless of the type of software technique. In this manner, the review has found 18 out of 24 (75%) works in this field. GPUs clearly represent a major step to optimize the execution of complex tasks iterating over data. Here, the possibility of reducing the number of repetitions in terms of data processing becomes relevant and can be critical for some time-restricted domains in which it is necessary to make decisions in near real-time. More specifically, GPUs are mainly used to solve linear algebra problems [54–56] or to perform large-scale matrix calculations [54–56, 58]. These are common problems in the field of engineering.

Other alternatives for large-scale computational infrastructures seem to be the traditional environments for parallel (FPGA and multiprocessor systems) and distributed computing. In the first case, the works that the review has found in the field of FPGAs [53] are not very representative (just 1 out of 24) while multiprocessor architecture works [40, 51, 57, 59] are still relevant (4 out of 24). Although FPGAs represent a powerful alternative to GPUs, the lack of (1) abstraction to code programs and (2) of a kind of hot-deployment method makes this alternative not very attractive for data science and engineering.

In the case of multiprocessor architectures, they represent the traditional powerful and expensive data processing centers for high-performance computation that are available in large research institutions. As it has been introduced in the first section, the research efforts [29] to merge Big Data and HPC are becoming popular to take advantage of existing large-scale computational infrastructures. Moreover, multiprocessor architectures also provide high-level APIs (application programming interfaces) that allow researchers and practitioners to easily configure, deploy, and run computing-intensive tasks. Finally, grid computing approaches [52] that have been widely spread for the deployment of cost-effective Big Data frameworks are not yet a popular option for scientific computation of large amounts of data.

4.2. Software Methods and Techniques for Data-Intensive Engineering Environments. To present the meta-analysis of the works regarding the typology of techniques and domains, Table 4 groups the different works making a mapping between the type of technique and its application domain. According to this table, AI techniques are the most prominent methods, representing the 21.43% of the reviewed works, to tackle problems in different sectors what is completely aligned to new perspectives open by the Industry 4.0 and digitalization processes.

Mathematics and applied mathematics methods still represent a 16.07% of the existing works bringing formal foundations for the computation of large amounts of data applying techniques such as integer linear programming, linear algebra, or symbolic execution. This situation makes sense since engineering domains are based on modelling physical systems using mathematics; so the existence and generation of more data only reinforces the need of improving the performance of existing methods to deal with more and greater amounts of data. This situation also affects computation architectures, trying to improve infrastructure

Table 3: Mapping between methods/models/techniques and hardware domains.

		GPU	FPGA	Multiprocessor	Grid computing
Artificial intelligence	*Deep learning* *Fuzzy logic* *Gene programming* *General techniques* *Neural networks* *Planning*				
Computational architecture	*Infrastructure* *Scheduling techniques* *Workflow*			[40]	
Computation model	*Event calculus* *MPI* *Parallel programming* *Query distribution* *Stream processing*	[41, 42] [43]			
Computational science	*Euler models* *Scientific computation* *Statistical methods*	([44], p. 2)			
Graph theory	*Automata* *Complex network analysis*				
Engineering	*Finite elements* *Simulation*	[45] [46, 47]			
Machine learning	*Bayesian machine learning* *Data mining* *Information fusion* *Pattern recognition* *Predictive models* *Support vector machines* *Regression model*	[48] [49, 50]	[53]	[51]	[52]
Mathematics and applied mathematics	*Gradient descent search* *Integer linear programming* *Linear algebra/solvers* *Linear programming* *Matrix calculation* *Nonlinear programming* *Numerical methods* *Symbolic execution*	[54–56] [54–56, 58]		[57] [59]	
Programming techniques	*Constraint programming* *Cube computation* *Dynamic programming* *DSL* *Stochastic programming*	[60]			

Table 4: Mapping between methods/models/techniques and domains including aggregated percentages.

	General application to engineering	Engineering	Nonengineering	Aggregated (%)
Artificial intelligence	[62, 63] (1.79%)	[64–73] (8.93%)	[74–85] (10.71%)	21.43%
Computational architecture	[86–88] (2.68%)	[89–95] (6.25%)	[90–95] (5.36%)	14.29%
Computation model	[96–98] (3.57%)	[99–101] (2.68%)	[102, 103] (1.79%)	8.04%
Computational science	[104] (0.89%)	[85, 105, 106] (2.68%)	[107] (0.89%)	4.46%
Graph theory	[37, 108–117] (9.82%)		[118, 119] (1.79%)	11.61%
Engineering methods	[96, 120–124] (5.36%)			5.36%
Machine learning	[125, 126] (1.79%)	[127] (0.89%)	[128–130] (2.68%)	5.36%
Mathematics and applied mathematics	[62, 88, 131–136] (7.14%)	[70, 137–141] (5.36%)	[141–144] (3.57%)	16.07%
Programming techniques	[74, 114, 122, 145–147] (5.36%)	[69, 148] (1.79%)	[149] (0.89%)	8.04%
General software application	[89, 115, 150–153] (5.36%)			5.36%
Aggregated (%)	43.75%	28.57%	27.68%	100/100%

settings via software-defined systems. Works in this area represent around 14.29% of the selected articles and, in general, they look for easing the creation of large software-defined infrastructures and APIs that can be used by the previous areas of AI and Mathematics and Applied Mathematics.

Graph techniques (11.61%), computational models (8.04%), and programming techniques (8.04%) represent the midclass methods in this classification. It is especially relevant to highlight the number of works that have emerged in the field of complex network analysis. Historically, large graph processing was a very time- and resource-consuming task that was preventing the broad use of these techniques. Currently and since new hardware infrastructures and APIs are available for scientific computation of large graphs, these techniques have gained momentum and they are being applied to different domains such as social network analysis, telecommunications, or biology.

Finally, the last section of the classification includes very specific works in the areas of Machine learning (5.36%), Engineering methods (5.36%), and Computational science (4.46%) that could be classified in other broader areas, but they represent concrete and representative works for data-intensive engineering environments.

On the other hand, selected works can be aligned to their scope in the different domains. In this sense, it has been found that techniques that can be applied to engineering (but not directly designed for engineering) represent a 43.75% what means that existing papers are reporting works to offer general-purpose solutions instead of solving specific problems. This is also a trend in the digitalization [61] in which one of the main cornerstones is the delivery of platforms ("Platform Economy") that can be extended and enable people to build new things on top of a baseline technology. In the case of Engineering, the selected works represent a 28.57% focusing on specific engineering problems that are now facing some new data landscape where existing techniques must be reshaped to offer new possibilities and more integrated and smart engineering methods. Finally, the review has also included a 27.68% of works in other domains to demonstrate that scientific computation and data are also an open issue in the other science-oriented domains. The two major conclusions of this meta-analysis are as follows:

(1) Engineering is not being indifferent to data-driven innovations. Assuming that more data will imply more knowledge, engineering methods are trying to find new opportunities in a data world to become more optimized and accurate.

(2) Hybrid approaches mixing existing techniques such as computational science and mathematics (deterministic) and AI techniques (probabilistic) are an emerging research area to optimize the use and exploitation of data.

In both cases, hardware- and software-defined infrastructures will play a key role to provide a physical or virtual environment to run complex tasks consuming and generating large amounts of data (e.g., simulations).

Once a whole picture of the techniques and methods in use for engineering has been depicted, it is necessary to identify and align existing research to specific engineering domains. To do so, Table 5 shows a mapping between the different methods and techniques and its application to the engineering domains.

Here, the development and operation of safety-critical systems implies an implicit need of managing data, information and knowledge that is generated by the different engineering teams. In this light, it is necessary to emphasize the research advances in Aerospace [105, 137, 138, 150] and Automotive [64, 65, 72, 89, 139] engineering. In both cases, the "one-size fits all" is again not true. Although formal methods are completely necessary to model this kind of complex systems, it is also necessary to combine different techniques that can shift the current practice in systems engineering. Existing engineering processes such as requirements engineering, modelling, simulation, co-simulation, traceability, or verification and validation are now completely impacted by an interconnected data-driven environment in which interoperability, collaboration, and continuous integration of system components are completely necessary.

Civil engineering [66, 67, 151], Industrial applications [148], Iron Mining [70], Manufacturing [73] Nuclear domain [143] works are also relevant in terms of using a huge up-to-date variety of techniques to exploit data that are continuously being generated by tools, applications, sensors, and people.

Finally, other rising sectors completely fueled by data such as CPS [71] or Feature Engineering [127] represent the update of classical engineering disciplines in aerospace, automotive, or robotics. As a final remark, general purpose engineering techniques have been also reported [68, 69, 99–101, 106, 140, 152, 154], making use of data, scientific programming techniques, and hybrid approaches.

On the other hand, as it has been previously introduced, the present work also looks for reviewing the impact of new techniques and methods in other nonengineering domains (Table 6). Earth science [78, 79, 84, 102, 149] is a common domain in which large amounts of data are generated by satellites and other data sources that must be processed to provide services such as environmental management and control, urban planning, and so on.

Scientific programming techniques are becoming even more relevant to Geometry [77, 80, 118, 130] paving the way to solve complex problems in areas such as civil engineering, physics, or architecture. In the case of image analysis [75, 76, 81, 144], it is possible to find techniques that are now gaining popularity for equipping smart cars with autonomous driving capabilities, identifying people, assisting physicians in medical diagnosis and surgery, or drone field monitoring with object recognition capabilities. More specifically, Medicine [74, 115, 128, 129] is taking advantage of scientific programming techniques and data for improving its processes of health monitoring and prevention. Social sciences [83] or Spatial modelling [94, 103] are also offering new capabilities such as social network analysis or simulation through the application of existing scientific

Table 5: Mapping between methods/models/techniques and engineering domains.

		General application to engineering	Aerospace	Automotive	Civil engineering	Cyberphysical-systems	Feature engineering	Engineering methods	Industrial applications	Iron mining
General software application			[150]	[89]	[151]			[152]		
Artificial intelligence	*Deep learning*									
	Fuzzy logic									
	General techniques			[64, 65]	[66, 67]					
	Gene Programming							[68]		
	Neural networks	[62, 63]						[69]		[70]
	Reasoning techniques					[71]				
	Planning			[72]						
Computational architecture	*Infrastructure*	[86–88]								
	Scheduling techniques			[89]						
	Workflow									
Computation model	*Event calculus*	[96]						[99–101]		
	MPI	[97]								
	Parallel programming									
	Query distribution	[87, 98]								
	Stream processing									
Computational science	*Euler models*									
	Scientific computation	[104]								
	Statistical methods		[105]					[106, 154]		
Graph theory	*Automata*	[124]								
	Graph/complex network analysis	[37, 108–117]								
Engineering methods	*Finite elements*	[155]								
	Simulation	[96, 120–124]								
Machine learning	*Bayesian machine learning*	[125, 126]								
	Data mining									
	Information fusion									
	Pattern recognition									
	Predictive models									
	Support vector machines									
	Regression model						[127]			
Mathematics and applied mathematics	*Gradient descent search*									
	Integer linear programming	[88]								
	Linear algebra/solvers	[131, 132]	[137]	[139]						
	Linear programming	[136]								
	Matrix calculation	[133]								
	Nonlinear programming	[62]	[138]							[70]
	Numerical methods							[140]		
	Symbolic execution	[134, 135]								
Programming techniques	*Constraint programming*	[74, 145–147]								
	Cube computation									
	Dynamic programming	[114]								
	Domain specific languages	[153]								
	Stochastic programming	[122]						[69]	[148]	

Table 6: Mapping between methods/models/techniques and other data-intensive domains.

		Earth science	Geometry	Image analysis	Internet of things	Manufacturing	Medicine	Nuclear domain	Scientific research	Social sciences	Spatial modelling
General software application							[115]				
Artificial intelligence	*Deep learning*			[75, 76]							
	Fuzzy logic										
	General techniques		[77]								
	Gene Programming	[78]				[73]					
	Neural networks	[79]	[80]	[81]					[82]	[83]	
	Reasoning techniques										
	Planning	[84]	[85]				[74]				
Computational architecture	*Infrastructure*								[90–93]		[94]
	Scheduling techniques										
	Workflow								[95]		
Computation model	*Event calculus*										
	MPI										
	Parallel programming	[102]									
	Query distribution										
	Stream processing										[103]
Computational science	*Euler models*										
	Scientific computation								[107]		
	Statistical methods										
Graph theory	*Automata*										
	Graph/complex network analysis		[118]							[119]	
Engineering methods	*Finite elements*										
	Simulation										
Machine learning	*Bayesian machine learning*						[128]				
	Data mining										
	Information fusion						[129]				
	Pattern recognition										
	Predictive models		[130]								
	Support vector machines										
	Regression model										
Mathematics and applied mathematics	*Gradient descent search*			[144]							
	Integer linear programming										
	Linear algebra/solvers								[141]		
	Linear programming				[142]						
	Matrix calculation							[143]			
	Nonlinear programming										
	Numerical methods										
	Symbolic execution										
Programming techniques	*Constraint programming*										
	Cube computation										
	Dynamic programming										
	Domain specific languages	[149]									
	Stochastic programming										

programming techniques over large datasets. Finally, this review has also found a good number of works in the field of scientific research [82, 90–93, 95, 107, 141] providing foundations for analysis and exploitation of data for other domains.

4.3. Additional Remarks about Benchmarking and Datasets. As it has been reviewed in the previous sections, a good number of hardware configurations and software frameworks to deal with large amounts of data in different domains can be found. However, it is also relevant to briefly introduce the notion of benchmarking for large-scale data systems as a method to evaluate this huge variety of tools and methods. In general, benchmarking comprises three main stages [156]: workload generation, input data or ingestion, and calculation of performance metrics. Benchmarking has been also widely studied [156, 157] in the field of Big Data systems and the main players in the industry [158] such as TPC ("Transaction Processing Performance Council"), SPEC ("the Standard Performance Evaluation Corporation") or CLDS ("Center for Large-scale Data System Research") have published different types of benchmarks that usually fall into three categories: microbenchmarks, functional benchmarks, or genre-specific benchmarks. The Yahoo! Cloud Serving Benchmark (YCSB) Framework, the AMP Lab Big Data Benchmark, BigBench [158], BigDataBench [159], BigFrame [160], CloudRank-D [161], or GridMix are some of the benchmarks that have been evaluated in [157] apart from others [157, 162] directly designed for Hadoop-based infrastructures such as HiBench, MRBench, MapReduceBenchmarkSuite (MRBS), Pavlo's Benchmark, or PigMix. Benchmarking is therefore a key process to ensure the capabilities of the different hardware settings and software frameworks in comparison with others.

Finally, it is also convenient to point out the possibility of accessing large datasets (open, free, and commercial) for computational and research purposes. These large datasets are usually managed by public institutions such as the European Data Portal (https://www.europeandataportal.eu/) (European Union), the Data.gov initiative (https://catalog.data.gov/dataset) (U.S. General Services Administration), research centers such as "Institute for Data Intensive Engineering and Science" (http://idies.jhu.edu/research/) (John Hopkins University), or large research projects such as Big Data Europe (https://www.big-data-europe.eu/). However, one of the best options to search and find high-quality datasets (license, provider, size, domain, etc.) is the use of some aggregating service such as the Amazon AWS Public Dataset Program, Google Public Data Explorer, or the most recent Google Dataset Search (https://toolbox.google.com/datasetsearch) (September 2018) that indexes any dataset published under a schema.org description. Kaggle (https://www.kaggle.com/datasets) and services organizing data-based competitions are also a good alternative to find complete and high-quality datasets. As a final type of data source, public APIs such as those from large social networks (e.g., Twitter, Linkedin, or Facebook) or specific sites such as GeoNames can also be used to access a good amount of data under some restrictions (depending on the API terms of service).

5. Answer to the Research Question and Future Challenges

The main objective of this systematic review was to provide a comprehensive and clear answer to the following question:

Which are the last techniques, methods, algorithms, architectures, infrastructures in scientific programming/computing for the management, exploitation, inference, and reasoning of data in engineering environments?

According to the results provided in the previous section and the meta-analysis, it is possible to define the different techniques, methods, algorithms, architectures, and infrastructures that are currently in use for scientific programming in data-intensive engineering environments. More specifically, in terms of hardware infrastructures, systems based on GPUs are now the main type of hardware setting to run complex and time-consuming tasks. Other alternatives such as FPGAs, multiprocessor architectures, or grid computing are being used for solving specific problems. However, the main drawback of FPGAs lies on the need of higher levels of abstraction to ease the development and deployment of complex problems. In the case of multiprocessor architectures, they are becoming popular since there is an effort to merge Big Data and HPC, but there is not yet so much works reporting relevant works in this area. Finally, grid computing represents a good option for Big Data problems where distribution is a key factor. However, when complex computational tasks must be executed, it does not seem to be a good option due to the need of synchronizing processes and data. In terms of architectures, there are also a good number of works looking for creating software-defined infrastructures easing the configuration and management of computational resources. In this sense, research works in the field of workflow management represent a trend to manage both the data life cycle and the execution of complex tasks.

On the other hand, scientific programming techniques and computational science methods such as integer linear programming, linear algebra/solvers, linear programming, matrix calculation, nonlinear programming, numerical methods or symbolic execution are being challenged by a complete new data environment in which large datasets of information are available. However, all these techniques have strong theoretical foundations that will be necessary to tackle problems in engineering. Moreover, programming techniques such as constraint and *stochastic* programming seem to be a good option to implement existing formal models in engineering.

This review has also identified that graph-based techniques, mainly complex network analysis, are becoming popular since a good number of problems can be represented as a set of nodes and relationships that can then be analyzed using graph theory. Considering that graph analysis techniques are based on matrix calculations, the possibility of having high-performance scientific methods

and infrastructures to execute such operations is being a key enabler for this type of analysis.

A relevant outcome of this review comes with the identification of AI and machine learning techniques as a counterpart of traditional scientific programming techniques. In the era of the 4th industrial revolution, the possibility of equipping not just machines but existing software techniques with intelligence is becoming a reality. These techniques are used to prepare the input of existing methods, to optimize the creation of computational models, and to learn and provide feedback on existing scientific programming techniques depending on the output. Although not every process, task, or engineering method is expected to include AI techniques, the review outlines the current trend of merging deterministic and probabilistic approaches. However, the use of AI is still under discussion since the impact of AI in engineering processes such as certification or assurance of safety-critical systems is completely open, e.g., autonomous driving.

Finally, the main engineering domains affected by huge amounts of data are Aerospace and Automotive. In these safety-critical sectors, engineering claims for innovative methods to enable collaboration between processes, people, and tools. The development and operation of a complex system cannot be anymore a set of orchestrated tasks via documents but a data-driven choreography in which each party can easily exchange, understand, and exploit data and information generated by others.

5.1. Future Challenges. Data-driven systems are already a reality ready to use. Big Data technologies and tools are enough mature and continuously improved and extended to cover all the stages of the data life cycle management. Capabilities for data ingestion, cleaning, representation, storage, processing models, and visualization are also available as stand-alone applications or as a part of a Big Data suite. Many use cases have successfully applied Big Data technology to solve problems in different sectors such as marketing, social network analysis, medicine, finances, and so on. However, a paradigm shift requires some efforts [163, 164] in the following topics:

(1) A reference and standardized architecture is necessary to have a separation of concerns between the different functional blocks. The standardization work in [36] represents a first step towards the harmonization of Big Data platforms.

(2) A clear definition of interfaces to exchange data between data acquisition processes, data storage techniques, data analysis methods, and data visualization models is completely necessary.

(3) A set of storage technologies to represent information depending on its nature. A data platform must support different types of logical representations such as documents, key/values, graphs, or tables.

(4) A set of mechanisms to transport (and transform) data between different functional blocks, for instance, between the data storage and analysis layers.

(5) An interface to provide data analysis as a service. Since problems to be solved and data may have different nature and objectives, it is necessary to avoid a kind of vendor lock-in problem and support a huge variety of technologies to be able to run diverse data analysis processes. Thus, it is possible to take the most of the existing libraries and technologies, e.g., libraries in R, Python, or Matlab. Here, it is also important to remark again that "no one size fits all" and hybrid approaches for analytical processes may be considered for the next generation of Big Data platforms.

(6) A set of processing mechanisms that can minimize the consumption of resources (in-memory, parallel, and distributed) maximizing the possibilities of processing data under different paradigms (event, stream, and batch) and analyzing data with different methods and techniques (AI, machine learning, scientific programming techniques, or complex network analysis) are also required.

(7) A management console to monitor the status of the platform and the possibility of creating data-oriented workflows (like the traditional methodologies in business process management but focusing on data). Data-intensive environments must take advantage of last technologies in cloud computing and enable users and practitioners to easily develop and deploy more complex techniques for the different phases and activities of the data life cycle.

(8) A catalogue of available hardware and software resources. A Big Data platform must offer capabilities to manage computational resources, tools, techniques, methods, or services. Thus, the operator can configure and build their own data-driven system combining existing resources.

(9) A set of nonfunctional requirements. Assuming that scalability can be easily reached through the use of cloud computation environments, interoperability, flexibility, and extensibility represents major nonfunctional characteristics that future platforms must include.

(10) A visualization layer or, at least, a method to ingest data in existing visualization platforms must be provided to easily summarize and extract meaning of huge amounts of data.

Apart from these general-purpose directions for the future of data-intensive environments, the engineering domain must also reshape the current methods to develop complex systems. Engineering is not anymore an isolated activity of experts in some discipline (software, mechanics, electronics, telecommunications, etc.) that produces a set of work products that serve us to specify and build a system.

Products and services are becoming complex and that complexity also implies the need of designing new ways of doing engineering. Tools, applications, and people

(engineers in this case) must collaborate and improve the current practice in systems engineering processes through the reuse of existing data, information, and knowledge. To do so, data-intensive engineering environments must be equipped with the proper tools and methods to ease processes such as analysis, design, verification and validation, certification, or assurance. In this light, scientific programming techniques must also be enriched with new and existing algorithms coming from the AI area. Hybrid approaches of techniques to solve complex problems represent the natural evolution of scientific programming techniques to take advantage of AI models and existing infrastructure.

5.2. Data-Intensive Engineering Environments and Scientific Programming. The development of complex engineering systems is challenging the current engineering discipline. From the development life cycle to the operation and retirement of an engineering product, an engineering product is now a combination of hardware and software that must be designed considering different engineering disciplines such as software engineering, mechanics, telecommunications, electronics, and so on. Furthermore, all technical processes, perfectly describe in standards such as the ISO/IEC/IEEE 15288:2015 "Systems and software engineering—System life cycle processes," are now more interrelated than never. A technical process implemented through an engineering method requires data and information that has been generated in previous development stages enriching the current engineering practice.

From a technical perspective, interoperability is becoming a cornerstone to enable collaborative engineering environments and to provide a holistic view of the system under development. Once data and information can be integrated together, new analytical services can be implemented to check the impact of a change, to discover traces, or to ensure the consistency of the system. However, it is necessary to provide common and standardized data models and access protocols to ensure that it is possible to build an integrated view of the system such as a repository. In this sense, the Open Services for Lifecycle Collaboration (OSLC) initiative applying the principles of Linked Data and REST and the Model-Based Systems Engineering approach are two of the main approaches to create a uniform engineering environment relying on existing standards. Approaches such as the OSLC Knowledge Management (KM) specification and repository [165] or the SysML 2.0 standard are defined under the assumption of providing standardized APIs (application programming interfaces) to access any kind of system artifact (in the case of OSLC KM) or model (in the case of SysML 2.0) meaning that the exchange of system artifacts is not anymore a single and isolated file but a kind of service. In general, engineering environments are already fueled by interconnected data shifting the paradigm of data silos to a kind of industrial knowledge graph. The impact of having an interconnected engineering environment relies on the possibility of increasing the time to release an engineering product or service meeting the evolving needs of customers.

In the operational environment, complex engineering systems are an example of Internet of Things (IoT) products in which thousands of sensors are continuously producing data that must be processed with different goals: prediction of failures, making decisions, etc. This environment represents a perfect match for designing and developing analytical services. Examples of data challenges for specific disciplines can be found in the railway sector [166], aerospace [167], or civil engineering [168] where Big Data technologies are mainly used to exploit data and information generated during the operation of the system.

However, in both cases, it is possible to find some common barriers to implement a data management strategy: data privacy, security, and ownership. For instance, in the automotive industry, it would be nice that the data used to validate the behavior of an autonomous car were shared among car manufacturers to ensure that all cars accomplish with a minimum level of compatibility and to ease the activity of certification bodies. Nevertheless, it seems also clear that this will not happened since it can represent a competitive advantage in a market-oriented economy. That is why, it is completely necessary to design policies at a political level that can ensure a proper development of new engineering products and services where manufacturers must focus on providing better user experiences under a common baseline of data.

Finally, it is necessary to remark again that scientific programming techniques have been widely used to solve complex problems in different domains under several hardware settings. The rising of Big Data technologies has created a new data-intensive environment in which scientific programming techniques are still relevant but facing a major challenge: How to adapt existing techniques to deal with large amounts of data?

In this context, scientific programming techniques can be adapted at hardware or software (platform, technique, or application) levels. For instance, it is possible to find again examples of GPUs architectures [169] to improve the practice in parallel programming for scientific programming. Cloud-based infrastructures [170, 171] and platforms for analytics [172] are another field of study. In the case of foundations of scientific programming, the work in [173] reviewing the main foundations of scientific programming techniques and the use of pattern matching techniques in large graphs [174] are examples of improvements and works in the scope of software techniques. Finally, applications in coal mining [175], recommendation engines for car sharing services [176], health risk prediction [177], text classification [178], or information security [179] are domains in which data are continuously being generated representing good candidates to apply scientific programming techniques.

6. Conclusions

Data are currently fueling any task, activity, or process in most industries. A Big Data ecosystem of infrastructures, technologies, and tools is already available and ready to tackle complex problems. Scientific programming techniques are being disrupted for this *mare magnum* of

technology. Complexity is not just the main driver to compute large and complex tasks. Large amounts of data are now used as input of complex algorithms, techniques, and methods to generate again huge amounts of more data items (e.g., simulation processes). These data-intensive environments strongly affect the current engineering discipline and methods that must be reshaped to take advantage of more and smarter data and techniques to improve both the development and operation of complex and safety-critical systems. That is why, the provision of new engineering methods through the exploitation of existing resources (infrastructure) and combination of well-known techniques will enable the industry to build complex systems faster and safer. However, there is also an implicit need to properly manage and harmonize all the aspects concerning a data-driven engineering environment. New integration platforms (and standardized architectural frameworks) must be designed to cover all stages of the data life cycle encouraging people to run large-scale experiments and improve the current practice in systems engineering.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The current work has been partially supported by the Research Agreement between the RTVE (the Spanish Radio and Television Corporation) and the UC3M to boost research in the field of Big Data, Linked Data, Complex Network Analysis, and Natural Language. It has also received the support of the Tecnologico Nacional de Mexico (TECNM), National Council of Science and Technology (CONACYT), and the Public Education Secretary (SEP) through PRODEP.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] N. Jazdi, "Cyber physical systems in the context of Industry 4.0," in *Proceedings of 2014 IEEE International Conference on Automation Quality and Testing, Robotics*, pp. 1–4, IEEE, Cluj-Napoca, Romania, May 2014.

[3] M. Borsato and M. Peruzzini, "Collaborative engineering," in *Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges*, J. Stjepandić, N. Wognum, and W. J. C. Verhagen, Eds., pp. 165–196, Springer International Publishing, Cham, Switzerland, 2015.

[4] W. Shen, Q. Hao, and W. Li, "Computer supported collaborative design: retrospective and perspective," *Computers in Industry*, vol. 59, no. 9, pp. 855–862, 2008.

[5] J. Lee, H.-A. Kao, and S. Yang, "Service innovation and smart analytics for industry 4.0 and big data environment," *Procedia CIRP*, vol. 16, pp. 3–8, 2014.

[6] INCOSE, *Systems Engineering Vision 2020, INCOSE*, Technical INCOSE-TP-2004-004-02, 2004.

[7] G. Wilson, D. A. Aruliah, C. Titus Brown et al., "Best practices for scientific computing," *PLoS Biology*, vol. 12, no. 1, article e1001745, 2014.

[8] P. Prabhu, T. B. Jablin, A. Raman et al., "A survey of the practice of computational science," in *State of the Practice Reports on-SC 11*, p. 19, ACM, New York, NY, USA, 2011.

[9] J. E. Hannay, C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson, "How do scientists develop and use scientific software?," in *Proceedings of 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, pp. 1–8, IEEE Computer Society, Montréal, Canada, 2009.

[10] N. Khan, I. Yaqoob, I. A. Targio Hashem et al., "Big Data: survey, technologies, opportunities, and challenges," *Scientific World Journal*, vol. 2014, pp. 1–18, 2014.

[11] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big Data computing and clouds: trends and future directions," *Journal of Parallel and Distributed Computing*, vol. 79, pp. 3–15, 2015.

[12] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: a survey on Big Data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[13] S. Gole and B. Tidke, "A survey of big data in social media using data mining techniques," in *Proceedings of 2015 International Conference on Advanced Computing and Communication Systems*, pp. 1–6, Coimbatore, India, 2015.

[14] C. S. Greene, J. Tan, M. Ung, J. H. Moore, and C. Cheng, "Big data bioinformatics," *Journal of Cellular Physiology*, vol. 229, no. 12, pp. 1896–1900, 2014.

[15] G. Boulton, "The challenges of a big data earth," *Big Earth Data*, vol. 2, no. 1, pp. 1–7, 2018.

[16] G.-H. Kim, S. Trimi, and J.-H. Chung, "Big-data applications in the government sector," *Communications of the ACM*, vol. 57, no. 3, pp. 78–85, 2014.

[17] C. S. Kruse, R. Goswamy, Y. Raval, and S. Marawi, "Adoption factors of the electronic health record: a systematic review," *JMIR Medical Informatics*, vol. 4, no. 4, p. e38, 2016.

[18] S. Hamrioui, I. de la Torre Díez, B. Garcia-Zapirain, K. Saleem, and J. J. P. C. Rodrigues, "A systematic review of security mechanisms for big data in health and new alternatives for hospitals," *Wireless Communications and Mobile Computing*, vol. 2017, pp. 1–6, 2017.

[19] R. Kitchin, "The real-time city? Big data and smart urbanism," *GeoJournal*, vol. 79, no. 1, pp. 1–14, 2014.

[20] S. Yin and O. Kaynak, "Big data for modern industry: challenges and trends [point of view]," *Proceedings of the IEEE*, vol. 103, no. 2, pp. 143–146, 2015.

[21] W. Fan and A. Bifet, "Mining big data: current status, and forecast to the future," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 1–5, 2013.

[22] F. Gessert, W. Wingerath, S. Friedrich, and N. Ritter, "NoSQL database systems: a survey and decision guidance," *Computer Science-Research and Development*, vol. 32, no. 3–4, pp. 353–365, 2017.

[23] S. Gilbert and N. Lynch, "Perspectives on the CAP theorem," *Computer*, vol. 45, no. 2, pp. 30–36, 2012.

[24] C.-H. Chen, C.-L. Hsu, and K.-Y. Tsai, "Survey on open source frameworks for big data analytics," in *Proceedings of Third International Conference on Electronics and Software Science*, , Takamatsu, Japan, 2017.

[25] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[26] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One trillion edges: graph processing at Facebook-scale," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1804–1815, 2015.

[27] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of 'big data' on cloud computing: review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[28] D. Agrawal, S. Das, and A. El Abbadi, "Big data and cloud computing: current state and future opportunities," in *Proceedings of the 14th International Conference on Extending Database Technology*, pp. 530–533, 2011.

[29] V. Holmes and M. Newall, "HPC and the BIG Data challenge," *Safety and Reliability*, vol. 36, no. 3, pp. 213–224, 2016.

[30] I. Corporation, *Big Data Meets High Performance Computing*, Intel HPC, 2014, https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/big-data-meets-high-performance-computing-white-paper.pdf.

[31] B. Kitchenham, *Procedures for Performing Systematic Reviews*, Vol. 33, Keele University, Keele, UK, 2004.

[32] S. Bhattacharyya and D. Ivanova, "Scientific Computing and Big Data Analytics: Application in Climate Science," in *Distributed Computing in Big Data Analytics*, S. Mazumder, R. Singh Bhadoria, and G. C. Deka, Eds., pp. 95–106, Springer International Publishing, Cham, Switzerland, 2017.

[33] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, pp. 1051-1052, Shanghai, China, 2006.

[34] J. M. Álvarez-Rodríguez, J. E. Labra-Gayo, and P. O. de Pablos, "New trends on e-Procurement applying semantic technologies: Current status and future challenges," *Computers in Industry*, vol. 65, no. 5, pp. 800–820, 2014.

[35] L. Shamseer et al., "Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015: elaboration and explanation," *BMJ*, vol. 349, p. g7647, 2015.

[36] NIST Big Data Public Working Group Technology Roadmap Subgroup, *NIST Big Data Interoperability Framework: Volume 7*, Standards Roadmap, National Institute of Standards and Technology, NIST SP 1500-7, 2015.

[37] Z. Halim, M. Waqas, A. R. Baig, and A. Rashid, "Efficient clustering of large uncertain graphs using neighborhood information," *International Journal of Approximate Reasoning*, vol. 90, pp. 274–291, 2017.

[38] T. Kajdanowicz, R. Michalski, K. Musial, and P. Kazienko, "Learning in unlabeled networks – An active learning and inference approach," *AI Communications*, vol. 29, no. 1, pp. 123–148, 2015.

[39] P. Kazienko, R. Alhajj, and J. Srivastava, "Computational aspects of social network analysis," *Scientific Programming*, vol. 2015, pp. 1-2, 2015.

[40] E. I. M. Zayid and M. F. Akay, "Predicting the performance measures of a message-passing multiprocessor architecture using artificial neural networks," *Neural Computing and Applications*, vol. 23, no. 7–8, pp. 2481–2491, 2013.

[41] Y. Ma, L. Chen, P. Liu, and K. Lu, "Parallel programing templates for remote sensing image processing on GPU architectures: design and implementation," *Computing*, vol. 98, no. 1-2, pp. 7–33, 2016.

[42] T. S. Sliwinski and S.-L. Kang, "Applying parallel computing techniques to analyze terabyte atmospheric boundary layer model outputs," *Big Data Research*, vol. 7, pp. 31–41, 2017.

[43] M. M. Rathore, H. Son, A. Ahmad, A. Paul, and G. Jeon, "Real-time big data stream processing using GPU with spark over Hadoop ecosystem," *International Journal of Parallel Programming*, vol. 46, no. 3, pp. 630–646, 2017.

[44] W. Xue, C. Yang, H. Fu et al., "Ultra-scalable CPU-MIC acceleration of mesoscale atmospheric modeling on Tianhe-2," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2382–2393, 2015.

[45] Y. Aksari and H. Artuner, "Forward and back substitution algorithms on GPU: a case study on modified incomplete Cholesky Preconditioner for three-dimensional finite difference method," *Journal of Supercomputing*, vol. 62, no. 1, pp. 550–572, 2012.

[46] A. Neic, M. Liebmann, E. Hoetzl et al., "Accelerating Cardiac Bidomain Simulations Using Graphics Processing Units," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 8, pp. 2281–2290, 2012.

[47] E. Thompson, N. Clem, D. A. Peter, J. Bryan, B. I. Peterson, and D. Holbrook, "Parallel cuda implementation of conflict detection for application to airspace deconfliction," *Journal of Supercomputing*, vol. 71, no. 10, pp. 3787–3810, 2015.

[48] L. Jian, C. Wang, Y. Liu, S. Liang, W. Yi, and Y. Shi, "Parallel data mining techniques on Graphics Processing Unit with Compute Unified Device Architecture (CUDA)," *Journal of Supercomputing*, vol. 64, no. 3, pp. 942–967, 2013.

[49] W. Zhou, Z. Cai, B. Lian, J. Wang, and J. Ma, "Protein database search of hybrid alignment algorithm based on GPU parallel acceleration," *Journal of Supercomputing*, vol. 73, no. 10, pp. 4517–4534, 2017.

[50] P. Leite, J. M. Teixeira, T. Farias, B. Reis, V. Teichrieb, and J. Kelner, "Nearest neighbor searches on the GPU: a massively parallel approach for dynamic point clouds," *International Journal of Parallel Programming*, vol. 40, no. 3, pp. 313–330, 2012.

[51] P. Liu, A. Hemani, K. Paul, C. Weis, M. Jung, and N. Wehn, "3D-stacked many-core architecture for biological sequence analysis problems," *International Journal of Parallel Programming*, vol. 45, no. 6, pp. 1420–1460, 2017.

[52] T. A. Wassenaar, M. van Dijk, N. Loureiro-Ferreira et al., "WeNMR: structural biology on the grid," *Journal of Grid Computing*, vol. 10, no. 4, pp. 743–767, 2012.

[53] S. Venkateshan, A. Patel, and K. Varghese, "Hybrid working set algorithm for SVM learning with a kernel coprocessor on FPGA," *IIEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2221–2232, 2015.

[54] J. K. Debnath, A. M. Gole, and W.-K. Fung, "Graphics-processing-unit-based acceleration of electromagnetic transients simulation," *IEEE Transactions on Power Delivery*, vol. 31, no. 5, pp. 2036–2044, 2016.

[55] A. Munir, F. Koushanfar, A. Gordon-Ross, and S. Ranka, "High-performance optimizations on tiled many-core embedded systems: a matrix multiplication case study," *Journal of Supercomputing*, vol. 66, no. 1, pp. 431–487, 2013.

[56] L. Ziane Khodja, R. Couturier, A. Giersch, and J. M. Bahi, "Parallel sparse linear solver with GMRES method using minimization techniques of communications for GPU clusters," *Journal of Supercomputing*, vol. 69, no. 1, pp. 200–224, 2014.

[57] H. Ltaief, P. Luszczek, and J. Dongarra, "Profiling high performance dense linear algebra algorithms on multicore architectures for power and energy efficiency," *Computer Science-Research and Development*, vol. 27, no. 4, pp. 277–287, 2012.

[58] M. A. Al-Mouhamed and A. H. Khan, "SpMV and BiCG-Stab optimization for a class of hepta-diagonal-sparse matrices on GPU," *Journal of Supercomputing*, vol. 73, no. 9, pp. 3761–3795, 2017.

[59] S. Li, C. Hu, J. Zhang, and Y. Zhang, "Automatic tuning of sparse matrix-vector multiplication on multicore clusters," *Science China Information Sciences*, vol. 58, no. 9, pp. 1–14, 2015.

[60] G. Zhou and H. Chen, "Parallel cube computation on modern CPUs and GPUs," *Journal of Supercomputing*, vol. 61, no. 3, pp. 394–417, 2012.

[61] A. Digital, "People first: the primacy of people in a digital age," *Accenture Technology Vision*, vol. 2016, 2016.

[62] A. Mansoori, S. Effati, and M. Eshaghnezhad, "An efficient recurrent neural network model for solving fuzzy non-linear programming problems," *Applied Intelligence*, vol. 46, no. 2, pp. 308–327, 2017.

[63] T. Muhammad and Z. Halim, "Employing artificial neural networks for constructing metadata-based model to automatically select an appropriate data visualization technique," *Applied Soft Computing*, vol. 49, pp. 365–384, 2016.

[64] Z. Halim, R. Kalsoom, S. Bashir, and G. Abbas, "Artificial intelligence techniques for driving safety and vehicle crash prediction," *Artificial Intelligence Review*, vol. 46, no. 3, pp. 351–387, 2016.

[65] J. Quadflieg, M. Preuss, and G. Rudolph, "Driving as a human: a track learning based adaptable architecture for a car racing controller," *Genetic Programming and Evolvable Machines*, vol. 15, no. 4, pp. 433–476, 2014.

[66] S. Terzi and S. Serin, "Planning maintenance works on pavements through ant colony optimization," *Neural Computing and Applications*, vol. 25, no. 1, pp. 143–153, 2014.

[67] A. Anton and A. Aldea, "Pumping stations: solving algorithm with inverse functions," *Procedia Engineering*, vol. 70, pp. 67–74, 2014.

[68] D. G. Savakar and A. Kannur, "A practical aspect of identification and classifying of Guns based on gunshot wound patterns using gene expression programming," *Pattern Recognition and Image Analysis*, vol. 26, no. 2, pp. 442–449, 2016.

[69] S. Aras and I. D. Kocakoç, "A new model selection strategy in time series forecasting with artificial neural networks: IHTS," *Neurocomputing*, vol. 174, pp. 974–987, 2016.

[70] Y. He, S. Gao, N. Liao, and H. Liu, "A nonlinear goal-programming-based DE and ANN approach to grade optimization in iron mining," *Neural Computing and Applications*, vol. 27, no. 7, pp. 2065–2081, 2016.

[71] L. Petnga and M. Austin, "Ontologies of time and time-based reasoning for MBSE of cyber-physical systems," *Procedia Computer Science*, vol. 16, pp. 403–412, 2013.

[72] G. Hongbo, X. Guotao, Z. Xinyu, and C. Bo, "Autonomous parking control for intelligent vehicles based on a novel algorithm," *Journal of China Universities of Posts and Telecommunications*, vol. 24, no. 4, pp. 51–56, 2017.

[73] S. Bingöl and H. Y. Kılıçgedik, "Application of gene expression programming in hot metal forming for intelligent manufacturing," *Neural Computing and Applications*, vol. 30, no. 3, pp. 937–945, 2016.

[74] L. Dioşan and A. Andreica, "Multi-objective breast cancer classification by using multi-expression programming," *Applied Intelligence*, vol. 43, no. 3, pp. 499–511, 2015.

[75] K. Charalampous and A. Gasteratos, "On-line deep learning method for action recognition," *Pattern Analysis and Applications*, vol. 19, no. 2, pp. 337–354, 2016.

[76] G. Ososkov and P. Goncharov, "Shallow and deep learning for image classification," *Optical Memory and Neural Networks*, vol. 26, no. 4, pp. 221–248, 2017.

[77] H. Karami, S. Karimi, H. Bonakdari, and S. Shamshirband, "Predicting discharge coefficient of triangular labyrinth weir using extreme learning machine, artificial neural network and genetic programming," *Neural Computing and Applications*, vol. 29, no. 11, pp. 983–989, 2016.

[78] V. Havlíček, M. Hanel, P. Máca, M. Kuráž, and P. Pech, "Incorporating basic hydrological concepts into genetic programming for rainfall-runoff forecasting," *Computing*, vol. 95, no. S1, pp. 363–380, 2013.

[79] A. Alexandridis, E. Chondrodima, E. Efthimiou, G. Papadakis, F. Vallianatos, and D. Triantis, "Large earthquake occurrence estimation based on radial basis function neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 9, pp. 5443–5453, 2014.

[80] J. Li, C. Li, Z. Wu, and J. Huang, "A feedback neural network for solving convex quadratic bi-level programming problems," *Neural Computing and Applications*, vol. 25, no. 3–4, pp. 603–611, 2014.

[81] M. Mulas and P. Massobrio, "NeuVision: A novel simulation environment to model spontaneous and stimulus-evoked activity of large-scale neuronal networks," *Neurocomputing*, vol. 122, pp. 441–457, 2013.

[82] K. J. Turner and P. S. Lambert, "Workflows for quantitative data analysis in the social sciences," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 3, pp. 321–338, 2015.

[83] S. Chakraborty, A. Cortesi, and N. Chaki, "A uniform representation of multi-variant data in intensive-query databases," *Innovations in Systems and Software Engineering*, vol. 12, no. 3, pp. 163–176, 2016.

[84] S. Multsch, D. Grabowski, J. Lüdering et al., "A practical planning software program for desalination in agriculture -SPARE:WATERopt," *Desalination*, vol. 404, pp. 121–131, 2017.

[85] K. Zafar, R. Baig, N. Bukhari, and Z. Halim, "Route planning and optimization of route using simulated ant agent system," *Journal of Circuits, Systems and Computers*, vol. 20, no. 03, pp. 457–478, 2011.

[86] A. Alexandrov, R. Bergmann, S. Ewen et al., "The Stratosphere platform for big data analytics," *VLDB Journal*, vol. 23, no. 6, pp. 939–964, 2014.

[87] F. Nadeem, D. Alghazzawi, A. Mashat, K. Fakeeh, A. Almalaise, and H. Hagras, "Modeling and predicting execution time of scientific workflows in the Grid using radial basis function neural network," *Cluster Computing*, vol. 20, no. 3, pp. 2805–2819, 2017.

[88] S. J. van Zelst, B. F. van Dongen, W. M. P. van der Aalst, and H. M. W. Verbeek, "Discovering workflow nets using integer linear programming," *Computing*, vol. 100, no. 5, pp. 529–556, 2017.

[89] G. Li, D. Wu, J. Hu, Y. Li, M. S. Hossain, and A. Ghoneim, "HELOS: heterogeneous load scheduling for electric vehicle-integrated microgrids," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 5785–5796, 2017.

[90] D. de Oliveira, K. A. C. S. Ocaña, F. Baião, and M. Mattoso, "A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds," *Journal of Grid Computing*, vol. 10, no. 3, pp. 521–552, 2012.

[91] I. K. Musa, S. D. Walker, A. M. Owen, and A. P. Harrison, "Self-service infrastructure container for data intensive application," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 3, no. 1, p. 5, 2014.

[92] M. Atkinson, C. S. Liew, M. Galea et al., "Data-intensive architecture for scientific knowledge discovery," *Distributed and Parallel Databases*, vol. 30, no. 5–6, pp. 307–324, 2012.

[93] M. S. Mayernik, J. C. Wallis, and C. L. Borgman, "Unearthing the infrastructure: humans and sensors in field-based scientific research," *Computer Supported Cooperative Work (CSCW)*, vol. 22, no. 1, pp. 65–101, 2013.

[94] J. Zhang, J. Yan, Y. Ma, D. Xu, P. Li, and W. Jie, "Infrastructures and services for remote sensing data production management across multiple satellite data centers," *Cluster Computing*, vol. 19, no. 3, pp. 1243–1260, 2016.

[95] Q. Wu, M. Zhu, Y. Gu et al., "A distributed workflow management system with case study of real-life scientific applications on grids," *Journal of Grid Computing*, vol. 10, no. 3, pp. 367–393, 2012.

[96] A. Pellegrini, S. Peluso, F. Quaglia, and R. Vitali, "Transparent speculative parallelization of discrete event simulation applications using global variables," *International Journal of Parallel Programming*, vol. 44, no. 6, pp. 1200–1247, 2016.

[97] C. Lively, X. Wu, V. Taylor et al., "Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems," *Computer Science-Research and Development*, vol. 27, no. 4, pp. 245–253, 2012.

[98] Y. Eom, J. Kim, and B. Nam, "Multi-dimensional multiple query scheduling with distributed semantic caching framework," *Cluster Computing*, vol. 18, no. 3, pp. 1141–1156, 2015.

[99] M. Z. A. Bhuiyan, J. Wu, G. Wang, T. Wang, and M. M. Hassan, "e-Sampling: event-sensitive autonomous adaptive sensing and low-cost monitoring in networked sensing systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 12, no. 1, pp. 1–29, 2017.

[100] Z. Milosevic, W. Chen, A. Berry, and F. A. Rabhi, "An open architecture for event-based analytics," *International Journal of Data Science and Analytics*, vol. 2, no. 1–2, pp. 13–27, 2016.

[101] S.-W. Lee, "Evidence-driven decision support in critical infrastructure management through enhanced domain knowledge modeling," *Multimedia Tools and Applications*, vol. 71, no. 1, pp. 309–330, 2014.

[102] R. Jacob, J. Krishna, X. Xu et al., "ParNCL and ParGAL: data-parallel tools for postprocessing of large-scale earth science data," *Procedia Computer Science*, vol. 18, pp. 1245–1254, 2013.

[103] D. Amagata and T. Hara, "A general framework for MaxRS and MaxCRS monitoring in spatial data streams," *ACM Transactions on Spatial Algorithms and Systems*, vol. 3, no. 1, pp. 1–34, 2017.

[104] J. Weinbub, K. Rupp, and S. Selberherr, "ViennaX: a parallel plugin execution framework for scientific computing," *Engineering with Computers*, vol. 30, no. 4, pp. 651–668, 2014.

[105] G. Zuo, G. Guan, and R. Wang, "Numerical modeling and optimization of vacuum membrane distillation module for low-cost water production," *Desalination*, vol. 339, pp. 1–9, 2014.

[106] M. D. G. Garcia-Hernandez, J. Ruiz-Pinales, E. Onaindia et al., "New prioritized value iteration for Markov decision processes," *Artificial Intelligence Review*, vol. 37, no. 2, pp. 157–167, 2012.

[107] G. Simonin, C. Artigues, E. Hebrard, and P. Lopez, "Scheduling scientific experiments for comet exploration," *Constraints*, vol. 20, no. 1, pp. 77–99, 2015.

[108] W. Horn, M. Kumar, J. Jann et al., "Graph programming interface (GPI): a linear algebra programming model for large scale graph computations," *International Journal of Parallel Programming*, vol. 46, no. 2, pp. 412–440, 2017.

[109] S. Lai, G. Lai, F. Lu, G. Shen, J. Jin, and X. Lin, "A BSP model graph processing system on many cores," *Cluster Computing*, vol. 20, no. 2, pp. 1359–1377, 2017.

[110] V. Boulos, S. Huet, V. Fristot, L. Salvo, and D. Houzet, "Efficient implementation of data flow graphs on multi-gpu clusters," *Journal of Real-Time Image Processing*, vol. 9, no. 1, pp. 217–232, 2014.

[111] J. Dümmler, R. Kunis, and G. Rünger, "SEParAT: scheduling support environment for parallel application task graphs," *Cluster Computing*, vol. 15, no. 3, pp. 223–238, 2012.

[112] Q. D. Pham, Y. Deville, and P. Van Hentenryck, "LS(Graph): a constraint-based local search for constraint optimization on trees and paths," *Constraints*, vol. 17, no. 4, pp. 357–408, 2012.

[113] W. Ju, J. Li, W. Yu, and R. Zhang, "iGraph: an incremental data processing system for dynamic graph," *Frontiers of Computer Science*, vol. 10, no. 3, pp. 462–476, 2016.

[114] O. Çeliktutan, C. Wolf, B. Sankur, and E. Lombardi, "Fast exact hyper-graph matching with dynamic programming for spatio-temporal data," *Journal of Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 1–21, 2015.

[115] J.-H. Jung, J. Lee, K. R. Hoffmann, T. Dorazio, and E. B. Pitman, "A rapid interpolation method of finding vascular CFD solutions with spectral collocation methods," *Journal of Computational Science*, vol. 4, no. 1–2, pp. 101–110, 2013.

[116] J. Xu, J. Han, K. Xiong, and F. Nie, "Robust and sparse fuzzy K-means clustering," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 2224–2230, New York, NY, USA, 2016.

[117] F. Nie, C. Ding, D. Luo, and H. Huang, "Improved MinMax cut graph clustering with nonnegative relaxation," in *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, pp. 451–466, Berlin, Heidelberg, 2010.

[118] H. Motallebi and S. Parsa, "Data locality optimization of interference graphs based on polyhedral computations," *Journal of Supercomputing*, vol. 61, no. 3, pp. 935–965, 2012.

[119] Z. Halim, M. Atif, A. Rashid, and C. A. Edwin, "Profiling players using real-world datasets: clustering the data and correlating the results with the big-five personality traits," *IEEE Transactions on Affective Computing*, p. 1, 2017.

[120] A. Weiß and D. Karastoyanova, "Enabling coupled multi-scale, multi-field experiments through choreographies of data-driven scientific simulations," *Computing*, vol. 98, no. 4, pp. 439–467, 2016.

[121] S. P. Muszala, G. Alaghband, J. Hack, and D. Connors, "Natural Load Indices (NLI) for scientific simulation," *Journal of Supercomputing*, vol. 59, no. 1, pp. 392–413, 2012.

[122] R. R. Upadhyay and O. A. Ezekoye, "libMoM : a library for stochastic simulations in engineering using statistical moments," *Engineering with Computers*, vol. 28, no. 1, pp. 83–94, 2012.

[123] B. Ben Youssef, "A parallel cellular automata algorithm for the deterministic simulation of 3-D multicellular tissue growth," *Cluster Computing*, vol. 18, no. 4, pp. 1561–1579, 2015.

[124] H. Mohamed and S. Marchand-Maillet, "Distributed media indexing based on MPI and MapReduce," *Multimedia Tools and Applications*, vol. 69, no. 2, pp. 513–537, 2014.

[125] W. Wang and G. Zeng, "Bayesian Cognitive Model in scheduling algorithm for data intensive computing," *Journal of Grid Computing*, vol. 10, no. 1, pp. 173–184, 2012.

[126] C. Ling, T. Hamada, J. Gao, G. Zhao, D. Sun, and W. Shi, "MrBayes tgMC ^{3} ++: a high performance and resource-efficient GPU-oriented phylogenetic analysis method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 5, pp. 845–854, 2016.

[127] P. Arroba, J. L. Risco-Martín, M. Zapater, J. M. Moya, and J. L. Ayala, "Enhancing regression models for complex systems using evolutionary techniques for feature engineering," *Journal of Grid Computing*, vol. 13, no. 3, pp. 409–423, 2015.

[128] G. Zhang, H. Pu, W. He, F. Liu, J. Luo, and J. Bai, "Bayesian framework based direct reconstruction of fluorescence parametric images," *IEEE Transactions on Medical Imaging*, vol. 34, no. 6, pp. 1378–1391, 2015.

[129] L. Peng, B. Liao, W. Zhu, Z. Li, and K. Li, "Predicting drug–target interactions with multi-information fusion," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 2, pp. 561–572, 2017.

[130] E. Park, J. Cavazos, L.-N. Pouchet, C. Bastoul, A. Cohen, and P. Sadayappan, "Predictive modeling in a polyhedral optimization space," *International Journal of Parallel Programming*, vol. 41, no. 5, pp. 704–750, 2013.

[131] M. Kreutzer, J. Thies, M. Röhrig-Zöllner et al., "GHOST: building blocks for high performance sparse linear algebra on heterogeneous systems," *International Journal of Parallel Programming*, vol. 45, no. 5, pp. 1046–1072, 2017.

[132] A. B. Manic, A. P. Smull, F.-H. Rouet, X. S. Li, and B. M. Notaros, "Efficient scalable parallel higher order direct MoM-SIE method with hierarchically semiseparable structures for 3-D scattering," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 5, pp. 2467–2478, 2017.

[133] K. V. Ryabinin and S. I. Chuprina, "A unified approach to adapt scientific visualization systems to third-party solvers," *Programming and Computer Software*, vol. 42, no. 6, pp. 347–355, 2016.

[134] X. Deng, J. Lee, and Robby, "Efficient and formal generalized symbolic execution," *Automated Software Engineering*, vol. 19, no. 3, pp. 233–301, 2012.

[135] Y. Chen, D. Chen, S. U. Khan, J. Huang, and C. Xie, "Solving symbolic regression problems with uniform design-aided gene expression programming," *Journal of Supercomputing*, vol. 66, no. 3, pp. 1553–1575, 2013.

[136] A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck, "Automation and combination of linear-programming based stabilization techniques in column generation," *INFORMS Journal on Computing*, vol. 30, no. 2, pp. 339–360, 2018.

[137] P. Chobeau, G. Guillaume, J. Picaut, D. Ecotière, and G. Dutilleux, "A Transmission Line Matrix model for sound propagation in arrays of cylinders normal to an impedance plane," *Journal of Sound and Vibration*, vol. 389, pp. 454–467, 2017.

[138] S. Deng, T. Meng, and Z. Jin, "Nonlinear programming control using differential aerodynamic drag for CubeSat formation flying," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 7, pp. 867–881, 2017.

[139] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles," *Automatica*, vol. 86, pp. 18–28, 2017.

[140] S. Dey, T. Mukhopadhyay, A. Spickenheuer, S. Adhikari, and G. Heinrich, "Bottom up surrogate based approach for stochastic frequency response analysis of laminated composite plates," *Composite Structures*, vol. 140, pp. 712–727, 2016.

[141] H. Astsatryan, V. Sahakyan, Y. Shoukouryan et al., "On the easy use of scientific computing services for large scale linear algebra and parallel decision making with the P-grade portal," *Journal of Grid Computing*, vol. 11, no. 2, pp. 239–248, 2013.

[142] A. Umbarkar, V. Subramanian, and A. Doboli, "Linear programming-based optimization for robust data modeling in a distributed sensing platform," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1531–1544, 2014.

[143] H. M. Aktulga, M. Afibuzzaman, S. Williams et al., "A high performance block eigensolver for nuclear configuration interaction calculations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1550–1563, 2017.

[144] F. Kizel, M. Shoshany, N. S. Netanyahu, G. Even-Tzur, and J. A. Benediktsson, "A stepwise analytical projected gradient descent search for hyperspectral unmixing and its code vectorization," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 9, pp. 4925–4943, 2017.

[145] A. Demiriz, N. Bagherzadeh, and A. Alhussein, "Using constraint programming for the design of network-on-chip architectures," *Computing*, vol. 97, no. 6, pp. 579–592, 2015.

[146] Á. Horváth and D. Varró, "Dynamic constraint satisfaction problems over models," *Software & Systems Modeling*, vol. 11, no. 3, pp. 385–408, 2012.

[147] W.-M. Ching and D. Zheng, "Automatic parallelization of array-oriented programs for a multi-core machine," *International Journal of Parallel Programming*, vol. 40, no. 5, pp. 514–531, 2012.

[148] D. Carvalho, L. R. de Souza, R. G. Barbastefano, and F. M. G. França, "Stochastic product-mix: a grid computing industrial application," *Journal of Grid Computing*, vol. 13, no. 2, pp. 293–304, 2015.

[149] A. N. Johanson and W. Hasselbring, "Effectiveness and efficiency of a domain-specific language for high-performance marine ecosystem simulation: a controlled experiment," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2206–2236, 2017.

[150] A. Aguilera, R. Grunzke, D. Habich et al., "Advancing a gateway infrastructure for wind turbine data analysis," *Journal of Grid Computing*, vol. 14, no. 4, pp. 499–514, 2016.

[151] L. Linzer, L. Mhamdi, and T. Schumacher, "Application of a moment tensor inversion code developed for mining-induced seismicity to fracture monitoring of civil engineering materials," *Journal of Applied Geophysics*, vol. 112, pp. 256–267, 2015.

[152] S. Zhu, R. Gao, and Z. Li, "Stereo matching algorithm with guided filter and modified dynamic programming," *Multimedia Tools and Applications*, vol. 76, no. 1, pp. 199–216, 2017.

[153] T. Muhammad, Z. Halim, and M. A. Khan, "Visualizing trace of Java collection APIs by dynamic bytecode instrumentation," *Journal of Visual Languages & Computing*, vol. 43, pp. 14–29, 2017.

[154] A. Kern, C. Schelthoff, and M. Mathieu, "Probability of lightning strikes to air-terminations of structures using the electro-geometrical model theory and the statistics of lightning current parameters," *Atmospheric Research*, vol. 117, pp. 2–11, 2012.

[155] H. Hakula and T. Tuominen, "Mathematica implementation of the high order finite element method applied to eigenproblems," *Computing*, vol. 95, no. S1, pp. 277–301, 2013.

[156] R. Han, L. K. John, and J. Zhan, "Benchmarking big data systems: a review," *IEEE Transactions on Services Computing*, vol. 11, no. 3, pp. 580–597, 2018.

[157] T. Ivanov, "Big data benchmark compendium," in *Performance Evaluation and Benchmarking: Traditional to Big Data to Internet of Things*, pp. 135–155, Cham, Switzerland, 2016.

[158] C. Baru, M. Bhandarkar, R. Nambiar, M. Poess, and T. Rabl, "Setting the direction for big data benchmark standards," in *Selected Topics in Performance Evaluation and Benchmarking*, pp. 197–208, Berlin, Heidelberg, 2013.

[159] L. Wang, "BigDataBench: a big data benchmark suite from Internet services," in *Proceedings of 2014 IEEE 20th International Symposium on High Performance Computer Architecture*, pp. 488–499, HPCA, Orlando, FL, USA, 2014.

[160] M. Kunjir, P. Kalmegh, and S. Babu, "Thoth: towards managing a multi-system cluster," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1689–1692, 2014.

[161] C. Luo, J. Zhan, Z. Jia et al., "Cloudrank-D: benchmarking and ranking cloud computing systems for data processing applications," *Frontiers of Computer Science*, vol. 6, no. 4, pp. 347–362, 2012.

[162] A. Pavlo, E. Paulson, A. Rasin et al., "A comparison of approaches to large-scale data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pp. 165–178, 2009.

[163] L. Cao, "Data science: nature and pitfalls," *IEEE Intelligent Systems*, vol. 31, no. 5, pp. 66–75, 2016.

[164] L. Cao, "Data science: challenges and directions," *Communications of the ACM*, vol. 60, no. 8, pp. 59–68, 2017.

[165] J. M. Alvarez-Rodríguez, J. Llorens, M. Alejandres, and J. M. Fuentes, "OSLC-KM: a knowledge management specification for OSLC-based resources," *INCOSE International Symposium*, vol. 25, no. 1, pp. 16–34, 2015.

[166] N. Attoh-Okine, "Big data challenges in railway engineering," in *Proceedings of IEEE International Conference on Big Data*, pp. 7–9, 2014.

[167] A. M. Chandramohan, D. Mylaraswamy, B. Xu, and P. Dietrich, "Big data infrastructure for aviation data analytics," in *Proceedings of IEEE International Conference on Cloud Computing in Emerging Markets*, pp. 1–6, 2014.

[168] O. Kapliński, N. Košeleva, and G. Ropaitė, "Big data in civil engineering: a state-of-the-art survey," *Engineering Structures and Technologies*, vol. 8, no. 4, pp. 165–175, 2016.

[169] L. Xu, A. Song, and W. Zhang, "Scalable parallel algorithm of multiple-relaxation-time lattice boltzmann method with large eddy simulation on multi-GPUs," *Scientific Programming*, vol. 2018, pp. 1–12, 2018.

[170] S. Ullah, M. D. Awan, and M. Sikander Hayat Khiyal, "Big data in cloud computing: a resource management perspective," *Scientific Programming*, vol. 2018, pp. 1–17, 2018.

[171] I. Kholod, I. Petukhov, and A. Shorov, "Cloud for distributed data analysis based on the actor model," *Scientific Programming*, vol. 2016, pp. 1–11, 2016.

[172] B. R. Chang, Y.-D. Lee, and P.-H. Liao, "Development of multiple big data analytics platforms with rapid response," *Scientific Programming*, vol. 2017, pp. 1–13, 2017.

[173] W. Zhao, L. Gao, and A. Liu, "Programming foundations for scientific big data analytics," *Scientific Programming*, vol. 2018, pp. 1-2, 2018.

[174] L. Zhang and J. Gao, "Incremental graph pattern matching algorithm for big graph data," *Scientific Programming*, vol. 2018, pp. 1–8, 2018.

[175] X. Xia, Z. Chen, and W. Wei, "Research on monitoring and prewarning system of accident in the coal mine based on big data," *Scientific Programming*, vol. 2018, pp. 1–10, 2018.

[176] Z. Liu, Y. Jia, and X. Zhu, "Deployment strategy for car-sharing depots by clustering urban traffic big data based on affinity propagation," *Scientific Programming*, vol. 2018, pp. 1–9, 2018.

[177] H. Zhong and J. Xiao, "Enhancing health risk prediction with deep learning on big data and revised fusion node paradigm," *Scientific Programming*, vol. 2017, pp. 1–18, 2017.

[178] W. Aziguli, Y. Zhang, Y. Xie et al., "A robust text classifier based on denoising deep neural network in the analysis of big data," *Scientific Programming*, vol. 2017, pp. 1–10, 2017.

[179] A. AlShawi, "Applying data mining techniques to improve information security in the cloud: a single cache system approach," *Scientific Programming*, vol. 2016, pp. 1–5, 2016.