The Landscape of Exascale Research: A Data-Driven Literature Analysis

Heldens, S.; Hijma, P.; Van Werkhoven, B.; Maassen, J.; Belloum, A.S.Z.; Van Nieuwpoort, R.V.

DOI
10.1145/3372390

Publication date
2020

Document Version
Final published version

Published in
ACM Computing Surveys

License
CC BY-SA

Link to publication

Citation for published version (APA):
Heldens, S., Hijma, P., Van Werkhoven, B., Maassen, J., Belloum, A. S. Z., & Van Nieuwpoort, R. V. (2020). The Landscape of Exascale Research: A Data-Driven Literature Analysis. *ACM Computing Surveys*, *53*(2), [23]. https://doi.org/10.1145/3372390

The Landscape of Exascale Research: A Data-Driven Literature Analysis

STIJN HELDENS, Netherlands eScience Center and University of Amsterdam
PIETER HIJMA, Vrije Universiteit Amsterdam and University of Amsterdam
BEN VAN WERKHOVEN and JASON MAASSEN, Netherlands eScience Center
ADAM S. Z. BELLOUM, University of Amsterdam
ROB V. VAN NIEUWPOORT, Netherlands eScience Center and University of Amsterdam

The next generation of supercomputers will break the exascale barrier. Soon we will have systems capable of at least one quintillion (billion billion) floating-point operations per second (10^{18} FLOPS). Tremendous amounts of work have been invested into identifying and overcoming the challenges of the exascale era. In this work, we present an overview of these efforts and provide insight into the important trends, developments, and exciting research opportunities in exascale computing. We use a three-stage approach in which we (1) discuss various exascale landmark studies, (2) use data-driven techniques to analyze the large collection of related literature, and (3) discuss eight research areas in depth based on influential articles. Overall, we observe that great advancements have been made in tackling the two primary exascale challenges: energy efficiency and fault tolerance. However, as we look forward, we still foresee two major concerns: the lack of suitable programming tools and the growing gap between processor performance and data bandwidth (i.e., memory, storage, networks). Although we will certainly reach exascale soon, without additional research, these issues could potentially limit the applicability of exascale computing.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Computer systems organization**; • **Hardware**; • **Software and its engineering**;

Additional Key Words and Phrases: Exascale computing, extreme-scale computing, literature review, high-performance computing, data-driven analysis

ACM Reference format:
Stijn Heldens, Pieter Hijma, Ben Van Werkhoven, Jason Maassen, Adam S. Z. Belloum, and Rob V. Van Nieuwpoort. 2020. The Landscape of Exascale Research: A Data-Driven Literature Analysis. *ACM Comput. Surv.* 53, 2, Article 23 (March 2020), 43 pages.
https://doi.org/10.1145/3372390

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777533 (PROCESS) and No 823988 (ESiWACE2), and the Netherlands eScience Center under file number 027.016.G06.

Authors' addresses: S. Heldens, Netherlands eScience Center, Science Park 140, 1098 XG, Amsterdam, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam; email: s.heldens@esciencecenter.nl; P. Hijma, Vrije Universiteit Amsterdam, De Boelelaan 1111, 1081 HV, Amsterdam, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam; email: pieter@cs.vu.nl; B. van Werkhoven and J. Maassen, Netherlands eScience Center, Science Park 140, 1098 XG, Amsterdam; emails: {b.vanwerkhoven, j.maassen}@esciencecenter.nl; A. S. Z. Belloum, University of Amsterdam, Amsterdam, Science Park 904, 1098 XH; email: a.s.z.belloum@uva.nl; R. V. van Nieuwpoort, Netherlands eScience Center, Amsterdam, Science Park 140, 1098 XG, University of Amsterdam, Amsterdam, Science Park 904, 1098 XH; email: r.v.vannieuwpoort@uva.nl.

0360-0300/2020/03-ART23
https://doi.org/10.1145/3372390

1 INTRODUCTION

The massive computational power of supercomputers plays a major role in the advancement of many scientific disciplines. The performance of these systems, measured in *floating-point operations per second* (FLOPS), has increased substantially over the last decades. Since the early 1990s, the TOP500 [4] has kept track of the fastest supercomputers in the world based on the LINPACK benchmark [56]. The TOP500 reveals exponential growth from several gigaflops (10^9 FLOPS) in 1993 to hundreds of petaflops (10^{15} FLOPS) in 2018. The *high-performance computing* (HPC) community is working towards the next major milestone: a computer system capable of at least one exaflop (10^{18} FLOPS). The race towards these *exascale* systems is reaching its conclusion: The United States announced the exascale supercomputer *Aurora* to be operational by end of 2021 [9] and the 1.5 exaflops supercomputer *Frontier* a year later [2], the European Union aims to build an exascale system in 2022/2023 [7], and China is targeting 2020 for the *Tianhe-3* [1].

However, over the past decade, it has frequently been acknowledged that building and programming such systems would be highly challenging [54, 66, 75, 103, 115, 126, 151]. Tremendous amounts of research have been invested into overcoming the challenges of exascale computing.

With this work, we aim to acquire a better understanding of these research efforts by analyzing the major trends. The topic of exascale computing is broad and touches upon nearly all aspects of HPC, each worth a literature survey of their own. Therefore, we do not aim for an *exhaustive* review of all available literature, but instead, provide a *high-level overview* of the entire exascale computing landscape. We devised a three-stage methodology that incorporates data-driven techniques alongside manual analysis.

- In Section 2, we discuss various landmark studies on the challenges and opportunities of exascale computing. These studies originate from both academic (i.e., peer-reviewed journals) as well as non-academic (e.g., white papers and industry roadmaps) sources. The studies represent the view on exascale computing by different communities at different moments, and allow us to sketch a timeline.
- In Section 3, based on these landmark studies, we formulate a search query and gather exascale related literature. We use data-driven methods to analyze this large collection of publications by considering aspects such as influential articles, authors, and publication venues. Additionally, we use natural language processing techniques to identify important research topics in a semi-automated way.
- In Section 4, we define eight research themes based on the identified topics: fault tolerance, energy/power, data storage/analytics, network interconnects, computer architectures, parallel programming, system software, and scientific applications. For each theme, we discuss publications selected from the search results and perform and in-depth analysis by analyzing important trends and progress over the last decade.

Section 5 discusses limitations of our study and Section 6 concludes our work. With this survey, both novice and experienced researchers and engineers gain insight into the ongoing trends, important developments and exciting research opportunities in exascale research.

2 LANDMARK STUDIES

In this section, we discuss various studies that have been published over the last decade on the challenges and opportunities of exascale computing. Table 1 lists the studies that have been selected for this analysis. They were chosen based on their citation count, while also considering variety in year of publication, authors, and scope. These studies are representative for the prevalent ideas,

Table 1. Landmark Exascale Studies Discussed in This Work

Title	Year	Authors
Technology Challenges in Achieving Exascale Systems [103]	2008	US DARPA
Major Computer Science Challenges at Exascale [75]	2009	Geist & Lucas
IESP Roadmap [54]	2011	Dongarra et al.
Top Ten Exascale Research Challenges [115]	2014	ASCAC
EESI2 Vision & Recommendations [66]	2015	EESI consortium
Exascale Computing and Big Data [151]	2015	Reed & Dongarra
The Exascale Computing Project [126]	2016	Messina

opinions, and visions on exascale computing. We discuss these contributions in chronological order and end with a discussion of our findings.

2.1 Technology Challenges in Achieving Exascale Systems (2008)

In 2008, the first HPC system in the TOP500 [4] reached petascale performance: *Roadrunner* at Los Alamos National Laboratory. Shortly after, US DARPA (*Defense Advanced Research Projects Agency*) presented the results of the *Exascale Working Group* [103]. The objective of this study was to understand the trends in computing technology at the time and to determine whether it was possible to increase the capability of computing systems by 1000× before 2015. The study group recognizes four major technology challenges in reaching exascale performance for which the technology trends at the time were insufficient.

Energy and Power. The study group believes that power consumption is the most pervasive challenge of the four. They establish 20MW as a reasonable power limit for exascale systems, but trends at the time show that exascale systems in 2015 would be significantly off from this target.

Concurrency and Locality. The group points out that explicit parallelism might be the only solution to increase overall system performance, since single core performance will stagnate due to clock rates flattening out. Exascale systems might offer billion-way concurrency with thousands of cores per node. Software needs consideration to exploit this thousand-fold increase in concurrency.

Memory and Storage. The study foresees a lack of available storage technologies that offer sufficient capacity and access rate for exascale applications, while staying within the power budget. This concerns all levels of the hierarchy: main memory, scratch storage, file storage, and archives.

Resilience. The study predicts that exascale systems will experience faults and errors more frequently than petascale systems. Various reasons are given, including the huge number of components (millions of memory chips and disks), very high clock rates (to maximize bandwidth), and ultra-low voltages (to minimize power consumption). Fault tolerance will become crucial since critical system failures could occur several times per day.

2.2 Major Computer Science Challenges at Exascale (2009)

Besides technological challenges, there are also computer science issues on topics such as software engineering and data management. In 2009, Geist and Lucas [75] presented an article that summarizes several studies and workshops (in 2007/2008) on the computer science problems that the community faces when moving to exascale. These challenges are organized according to four aspects:

Challenges due to Scale and Complexity of Computer Architectures. Computer architectures are expected to become more complex: massive parallelism, inadequate memory performance (capacity, bandwidth, and latency), more specialized circuits, and increasing amounts of heterogeneity.

Developing software for these systems is challenging and the traditional programming method (MPI plus sequential programming language) will become less productive.

Challenges due to Complexity of Applications. As computational capabilities increase, so does the complexity and resolution of scientific simulations. There are many challenges in adapting existing programs or developing new applications for future architectures. Existing programs often consist of millions of lines of code that have been written over long periods of time by hundreds of scientists/engineers. Adapting such applications to different architectures is not straightforward.

Challenges due to Increased Data. Exascale applications are likely to consume and produce massive amounts of data and there are several issues related to managing these large datasets. First, there are technical problems in efficiently storing and retrieving data. Second, there are data management problems, such as how to transfer large datasets between different compute clusters or between software components. Third, turning raw data into scientific discoveries requires the ability to search, visualize, explore, and summarize these large datasets.

Software Sustainability. Reaching exascale computing requires building a software ecosystem, training scientists on how to use new tools, and raising existing software to production quality.

2.3 IESP Roadmap (2011)

At the beginning of 2009, a large international consortium of researchers initiated the *International Exascale Software Project* (IESP) [8, 54]. The IESP recognized that significant amounts of effort have been invested into software during the petascale era. However, they argue that “a great deal of productivity has also been lost because of lack of planning, coordination, and key integration of technologies” [54, p. 1]. The purpose of the IESP was two-fold: (1) “developing a plan for a common, high-quality computational environment for petascale/exascale systems” [54, p. 6] and (2) “catalyzing, coordinating and sustaining the effort of the international open-source software community to create that environment as quickly as possible” [54, p. 6]. In 2011, the IESP presented their technology roadmap [54] and they proposed a possible software stack for exascale systems. The project ended in 2012, but IESP’s mission was continued by the *Big Data and Extreme-Scale computing* (BDEC) project [5, 13] focusing on the convergence of exascale computing and Big Data.

2.4 Top Ten Exascale Research Challenges (2014)

In 2014, the US ASCAC (*Advanced Scientific Computing Advisory Committee*) published the report [115] of the subcommittee that was directed by the US DOE (*Department of Energy*) to compile “a list of no more than ten technical approaches (hardware and software) that will enable the development of a system that achieves the Department’s exascale goals” [115, p. 80].

The study group found that “the U.S. has the technical foundation to create exascale systems” [115, p. 66], but “an evolutionary approach to achieving exascale would not be adequate” [115, p. 66]. Additionally, they argue that the U.S. should increase investments in HPC, since they may otherwise fall behind the international competition. The top ten challenges are as follows:

- *Energy Efficiency*: Develop energy-efficient technology to reach the goal of 20MW.
- *Interconnect Technology*: Improve *vertical* (intra-node) and *horizontal* (inter-node) data movement in terms of energy-efficiency and performance.
- *Memory technology*: Integrate novel memory technologies (e.g., PCRAM, NOR Flash, ReRAM, memristor) to improve capacity, bandwidth, resilience, and energy-efficiency.
- *Scalable Systems Software*: Increase the scalability, energy-awareness, and resilience of system software (e.g., operating systems, runtime systems, monitor systems).

- *Programming Systems*: Develop new programming methods that enable expressing fine-grained concurrency, locality, and resilience.
- *Data Management*: Develop software that is capable of handling massive amounts of data. This concerns both *offensive* I/O (e.g., data analysis) and *defensive* I/O (e.g., fault tolerance).
- *Exascale Algorithms*: Redesign algorithms to improve their scalability (e.g., reduce communication, avoid synchronization).
- *Algorithms for Discovery, Design, and Decision*: Research should focus not only on "one-off heroic simulations" [115, p. 48], but also on ensembles of many small runs (e.g., common for uncertainty quantization or parameter optimization).
- *Resilience and Correctness*: Computations should be correct, reproducible, and verifiable even in the face of software and hardware errors.
- *Scientific Productivity*: Scientists should have tools to productively utilize exascale systems (e.g., develop programs, run applications, prepare input, collect output, analyze results).

These ten challenges are mostly a mixture of the previously recognized technological challenges (US DARPA) [103] and computer science challenges (Geist and Lucas) [75]. Overall, these challenges are broad and touch upon nearly all aspects of HPC, indicating that exascale is disruptive.

2.5 EESI2 Vision & Recommendations (2015)

In 2015, the experts of the EESI (*European Exascale Software Initiative*) presented their vision and recommendations on development of efficient exascale applications [66]. They consider exascale computing as "not only a 'bigger HPC'" [66, p. 3], but also as a new era that requires disruptive scalable solutions. Additionally, they believe exascale computing and Big Data to be closely associated since HPC requires processing large-scale data from scientific instruments and scientific simulations. They argue that Europe has strengths in certain areas (e.g., applications, scalable algorithms, and software couplers), but is weak in other domains (e.g., languages, programming tools). They consider it urgent for the European Commission to fund large holistic projects on topics from three pillars:

- *Tools and Programming Models*: Novel programming models, heterogeneity management, software engineering methods, resilience, validations, and uncertainty quantification.
- *Ultra-Scalable Algorithms*: Scalable algorithms that reduce communication, avoid synchronization, and exploit parallelism in the time domain (in addition to the spatial domain).
- *Data-Centric Approaches*: Flexible and efficient software couplers, in-situ data processing, and declarative processing frameworks for data analytics.

2.6 Exascale Computing and Big Data (2015)

In 2015, Reed and Dongarra [151] presented their thoughts on the unification of exascale ("high-end computing") and Big Data ("high-end data analytics"). They observe that the tools and cultures of these two communities differ significantly and bringing them closer together would be beneficial. They acknowledge the technical challenges recognized by others, including US DARPA [103] and ASCAC [115], but consider "research and development of next-generation algorithms, software, and applications [to be] as crucial as investment in semiconductor devices and hardware" [151, p. 68] since "historically the research community has underinvested in these areas" [151, p. 68].

2.7 The Exascale Computing Project (2016)

In 2016, the *Exascale Computing Project* (ECP) [126] was launched: a seven year project by the U.S. Department of Energy (DOE) to coordinate the effort to achieve exascale computing in the

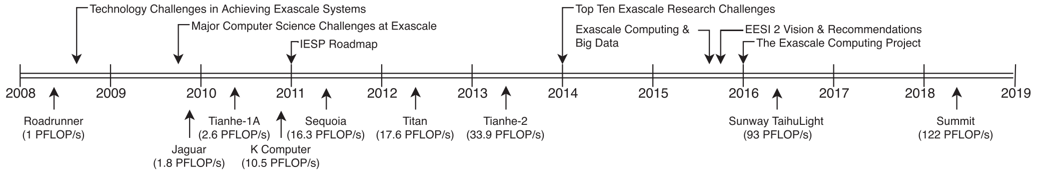

Technology Challenges in Achieving Exascale Systems
Major Computer Science Challenges at Exascale
IESP Roadmap
Top Ten Exascale Research Challenges
Exascale Computing & Big Data
EESI 2 Vision & Recommendations
The Exascale Computing Project
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
Roadrunner (1 PFLOP/s)
Jaguar (1.8 PFLOP/s)
Tianhe-1A (2.6 PFLOP/s)
K Computer (10.5 PFLOP/s)
Sequoia (16.3 PFLOP/s)
Titan (17.6 PFLOP/s)
Tianhe-2 (33.9 PFLOP/s)
Sunway TaihuLight (93 PFLOP/s)
Summit (122 PFLOP/s)

Fig. 1. Timeline showing landmark studies (top row) and #1 from TOP500 [4] (bottom row).

U.S. The project is a collaboration between six US-based computing laboratories and is funded by the U.S. National Nuclear Security Administration and the US DOE Office of Science. The project is organized into four focus areas: "application development", "software technology", "hardware technology", and "exascale systems". Additionally, the project includes training of scientists on software engineering and programming tools. Overall, the project emphasizes "co-design and integration of activities to ensure that the result is a robust exascale ecosystem" [126, p. 65].

2.8 Discussion

Figure 1 shows a timeline incorporating the discussed studies together with data from the TOP500 [4]. The figure clearly shows that growth has stagnated as we are reaching exascale: performance increased by ~33× between 2008 and 2013, but only by ~3.6× between 2013 and 2018.

Two primary challenges are acknowledged by nearly all of the above studies: *energy consumption* and *fault tolerance*. For the energy challenge, US DARPA set 20MW as a reasonable power budget [103] and most of the subsequent studies acknowledge this ambitious goal. For the fault tolerance challenge, all studies agree that handling hardware and software faults becomes increasingly more important as the scale of supercomputers continues to increase. These two challenges were foreseen in 2008 and many subsequent studies echo these concerns. Both topics will be discussed further in Section 4.

Various other challenges are mentioned, many of which can be attributed to one of two classes: challenges due to an *increase in complexity of software* or challenges due to an *increase in volume of data*. The first class addresses matters such as scientific applications, programming models, concurrency, heterogeneity, software sustainability, systems software, and software tools. Scalability needs to be incorporated across the entire software stack to create exascale-ready platforms. The second class concerns the fact that exascale systems deal with massive amounts of data. This has a major impact on hardware (e.g., memory, storage, networks) as well as on software (e.g., data management, visualization, scientific discovery). Some challenges are on the intersection between the two classes, such as algorithms that can increase scalability by exploiting parallelism in the time-domain (i.e., increase software complexity) or reducing communication (i.e., decrease data movement).

We observe two notable trends among these reports. The first trend is the rise of Big Data and the gap between large-scale data analytics (i.e., Big Data) and large-scale scientific computing (i.e., HPC). These two ecosystems appear to be disconnected and bridging this gap is a challenge acknowledged in recent studies such as "Exascale Computing and Big Data" [151], but also by the BDEC project and the EESI2 recommendations. The second trend is the acknowledgment that reaching exascale performance requires, besides solving technical roadblocks in software and hardware, to also work on the "social" problems in preparing the community for exascale computing. The IESP roadmap [54] emphasizes that much productivity was lost during the petascale-era due to lack of a cohesive group. The IESP [54] and the ECP [126] both emphasize collaboration among many different parties and focus on building a larger ecosystem surrounding exascale computing.

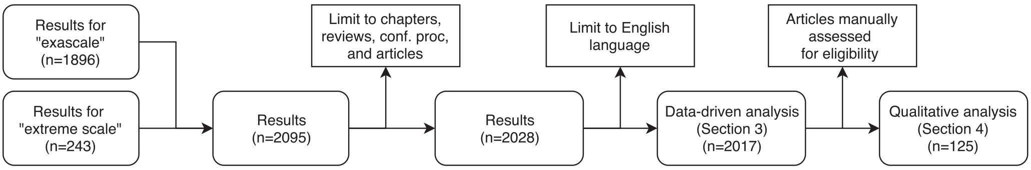

Results for "exascale" (n=1896)
Results for "extreme scale" (n=243)
Results (n=2095)
Limit to chapters, reviews, conf. proc, and articles
Results (n=2028)
Limit to English language
Data-driven analysis (Section 3) (n=2017)
Articles manually assessed for eligibility
Qualitative analysis (Section 4) (n=125)

Fig. 2. Flow diagram of article selection.

3 DATA-DRIVEN LITERATURE ANALYSIS

We perform a literature analysis to gain insight into the extensive peer-reviewed literature on exascale computing. In this section, we explain the method and results of our *data-driven* literature analysis. Section 4 presents the results of our *qualitative* literature analysis.

3.1 Methodology

Scopus [3] was used to get literature on exascale computing. Scopus is a citation database covering over 22,800 peer-review sources (e.g., journals, books, conference proceedings) from more than 500 publishers (including well-known publishers such as IEEE, ACM, Elsevier, Springer, and Wiley). The software tool that was used to perform the analysis is available online [91].

3.1.1 Search Query. The following search query was performed and yielded 2,017 search results.[1] Throughout this document, we use the terms *search result*, *article*, *publication*, and *document* interchangeably.

```
TITLE-ABS-KEY("exascale" OR "exa scale")
  OR TITLE-ABS-KEY("extreme scale comput*" OR "extreme scale system*")
```

The query requests publications that mention the term `exascale` or `extreme-scale` in the title, abstract, or author-specified keywords. The word `exascale` is an unambiguous term that is widely used and there appears to be no conflicting usages of the term. The term `extreme scale` is a strongly related term that gained popularity in recent years. Since solely the term `extreme scale` is heavily ambiguous (yielding 30,832 results), we refined this query by appending the term `comput*` (e.g., *computing*, *computer*, *computers*) or `system*` (e.g., *system*, *systems*). Note that Scopus ignores punctuation (e.g., `exa-scale` and `exa scale` are considered equivalent).

The search results are further refined using the following method (Figure 2). The Scopus document type must be *conference proceeding*, *article*, *review*, or *book*, thus excluding miscellaneous types defined by Scopus such as *erratum*, *editorial*, and *press releases*. Additionally, non-English documents are also excluded.

We are aware that an occurrence of the literal term `exascale` or `extreme-scale` does not necessarily imply that the entire document is dedicated to this topic, since the term could be used to provide context or background information. However, for this study, we deliberately make the assumption that any mention of these terms indicate that the authors are aware of the challenges in exascale computing and consider their contribution to play some role for future computing systems, thus making it an interesting candidate for this literature analysis.

Besides `exascale`, we experimented with three additional queries: `ultrascale`, `exaflop`, and `exabyte`. Results for `ultrascale` were limited and it appears to be a rare term. Results for `exabyte` were mostly out of scope and present a different line of research focusing on long-term storage systems. Few results for `exaflop` were related, but only 46 results were not covered by our query.

[1] The data was downloaded from Scopus API on 23 August 2019 via http://api.elsevier.com and http://scopus.com.

3.1.2 Metadata Analysis. Scopus provides rich metadata for the search results, including the document's title, abstract, DOI, authors, affiliations, publication date, and publication venue. This data is used to analyze the number of publications per year, per institute, per country, per continent, and per journal/conference.

To determine the institute, we use the author affiliation data reported by Scopus. These institutes were manually checked to consolidate different spellings (e.g., "*Jülich Supercomputing Centre*", "*Juelich Research Centre*", "*Forschungszentrum Jülich*" are mapped to "*Jülich Research Centre*").

To determine geographical regions, we use the country of the affiliations as reported by Scopus. For our analysis, we say a document belongs to a certain country if at least one author reports an affiliation in that country. One publication with multiple authors could belong to multiple countries.

To determine the publication venues, we use the *source title* attribute reported by Scopus. The source titles have been manually rewritten to canonical names (e.g., "*16th ACM Symposium on Principles and Practice of Parallel Programming*" is mapped to "*PPoPP*"). We omit workshops from the list of publication venues since Scopus often reports the name of the hosting conference (e.g., "*Workshops at International Conference on Parallel Processing (ICPP)*") instead of the official workshop title (e.g., "*Workshop on Runtime and Operating Systems for Supercomputers (ROSS)*"). Workshops are independently organized so including these results would be misleading. Overall, 19% of the search results originates from workshop proceedings.

3.1.3 Network Analysis. Network analysis helps to identify the contributions of influential publications and authors in the exascale landscape. To determine influential publications, we construct a *citation network* from the results: nodes represent documents and directed edges indicate citations between documents. To determine influential authors, we construct a *co-author network*: nodes correspond to institutes and edges indicate that two institutes have published at least one document together. Edges are weighted by the number of shared publications.

3.1.4 Text Analysis. To identify the most relevant research topics, we employ a method from *natural language processing* (NLP) known as *unsupervised topic modeling* [10] to extract abstract "topics" from texts. The model was trained on the article's title/abstract since they provide a concise summary of the article's contributions. Training topic models on academic abstracts is a proven method that has demonstrated its effectiveness in various domains, including information systems [165], software engineering [123], agriculture [101], bio-medicine [28], and fisheries science [169].

For this work, we selected the unsupervised topic modeling method based on *non-negative matrix factorization* (NMF) [161, 183]. We have chosen this method since it requires no critical parameters, has an intuitive interpretation, and has previously been used for various text-mining applications [19, 142, 180]. NMF detected topics automatically based on the correlations between word frequencies. For example, the words "*programming,*" "*code,*" "*compiler,*" "*parallel,*" "*performance,*" and "*software*" are likely to occur together and thus are strongly correlated.

NMF processes the titles/abstracts into one word-frequency vector per document (see Appendix A for details). It then takes the set of word count vectors and constructs two non-negative normalized matrices: **U** and **V** (Figure 3). Each row of **U** represents the topic distribution for one document with elements indicating the degrees of which topics are applicable to document. Each row of **V** represents the word distribution for one topic with elements indicating the degrees of which terms are applicable to the topic.

NMF requires the desired number of topics k as its input parameter. The appropriate value for k depends on the dataset at hand and the desired level of detail [161]. To determine the optimal number, we experimented with different values ($k = 5, 10, 25, 35, 40, 50$) and, for each k, evaluated

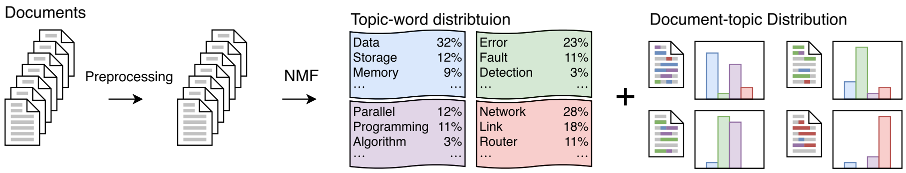


Fig. 3. Topic modeling detects topic-word distribution and document-topic distribution.

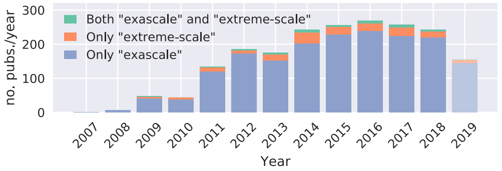


Fig. 4. Number of publications per year.

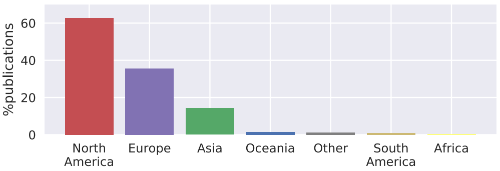


Fig. 5. Number of publications per continent.

the semantic quality of the detected topics by hand. We use $k = 25$ because we found fewer topics (e.g., $k = 10$) resulted in composite topics and more topics (e.g., $k = 50$) led to duplicates.

To gain insight into how these topics are "spread out" over the domain, we visualize the results using the method proposed by Choo et al. [38] based on t-SNE [174] (*t-Distributed Stochastic Neighbor Embedding*). This method embeds the documents into two-dimensional space such that the distance between documents inversely corresponds to their lexical similarity.

3.2 Results

In this section, we present the results of our data-driven literature analysis. The search was performed on 23 August 2019 and yielded 2,017 results. Of these results, 91% was discovered by the `exascale` query and 12% by the `extreme-scale` query. The term `exascale` is thus more popular than `extreme scale` and there is some overlap (only 3%) between the queries.

Figure 4 shows the number of documents per year for both queries. The earliest document originates from 2007, after which the number of articles increased, exceeding 100 articles in 2011 and 200 articles in 2014. Note that, at the time of writing, not all literature for 2019 is available yet.

Figure 5 shows the number of documents per continent based on author affiliation. The results indicate that the United States is the primary driver in exascale research, followed by Europe and Asia. Overall, affiliations from the U.S. are involved in 63% of the documents, Europe in 36%, and Asia in 14%. Many collaborations exist: nearly 10% of the publications is a collaboration between the U.S. and Europe (i.e., at least one affiliation from each continent), 3.7% for the U.S. and Asia, and 2.6% for Europe and Asia. The data per *country* is available in Appendix B (Figure B.3). For Europe, the top five countries are Germany (11%), United Kingdom (8%), France (8%), Spain (6%), and Switzerland (4%). For Asia, the top three consists of Japan (4%), China (4%), and India (2%).

To understand the role of the different institutes in exascale literature, we shift our focus to author affiliations. Figure 6 shows the number of documents per affiliation as reported by Scopus (top 25, extended results in Figure B.1 of Appendix B). The figure shows that the national laboratories in the United States play an important role in exascale research, with the top three consisting of *Oak Ridge National Lab.*, *Argonne National Lab.*, and *Sandia National Lab. in New Mexico*. For Europe, the top three is *INRIA*, *Jülich Research Center*, and *IBM Research Zurich*.

3.2.1 Metadata Analysis. Finally, we shift our attention to publication venues. Overall, 51% of the documents originates from conference proceedings, 26% from journals, 19% from workshops, and 5% from other sources (e.g., books, reports). See Figure 7 for the top publication venues (top 25,

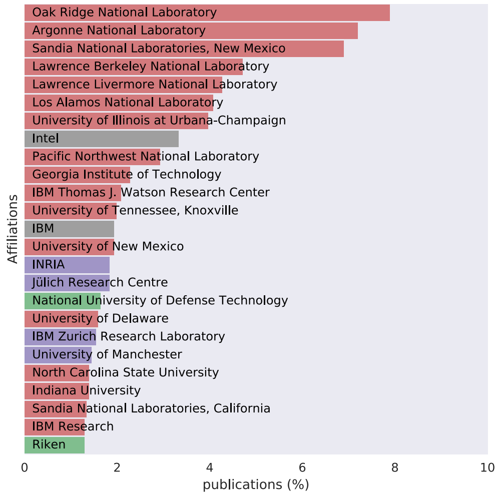


Fig. 6. Number of publications based on author affiliations (top 25). Color indicates continent (Figure 5).

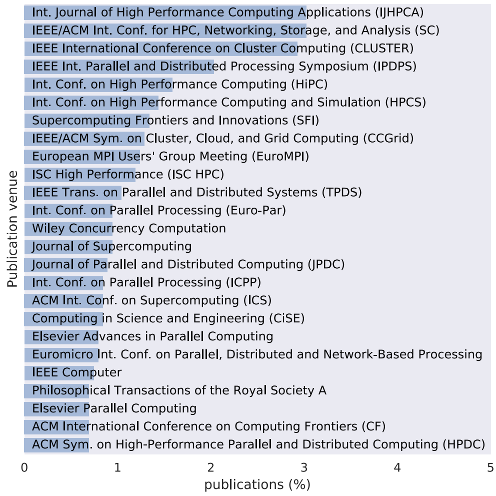


Fig. 7. Number of publications per publication venue (top 25). Workshops are omitted from figure, see text.

extended results in Figure B.2 of Appendix B). As mentioned in the methodology, we omit workshop names from this figure since Scopus inconsistently reports the name of the hosting conference instead of the workshop name. The figure shows that both conferences and journals are important. The top three conferences are *ACM/IEEE Supercomputing (SC)*, *IEEE Cluster*, and *IEEE IPDPS* (*International Parallel and Distributed Processing Symposium*). The top three journals are *IJHPCA* (*International Journal of High Performance Computing Applications*), *SFI* (*Supercomputing Frontiers and Innovations*), and *TPDPS* (*IEEE Transactions on Parallel and Distributed Systems*). Overall, exascale research is spread out over many venues.

3.2.2 *Network Analysis.* Figure 8 shows the citation network: nodes represent documents and edges are citations. An interactive version is available.[2] The visualization shows that the network contains a small number of "core" publications and a large number of "peripheral" publications. Appendix B (Table B.1) lists the complete top 10 publications according to the number of citations. Three of these publications were already discussed as landmark studies in Section 2. Surprisingly, six out of the ten publications are on the topic of fault tolerance. The remaining four papers are on diverse topics: software challenges, technology challenges, big data, and power constraints.

The collaboration network for the top 50 institutes is available in Appendix B (Figure B.3). The figure shows that the network is highly interconnected with many collaborations between different institutes. Additionally, national laboratories and supercomputing centers appear to play a central role in the network and they are often strongly connected to various universities and research institutes. Examples of strongly connected laboratory-university pairs are *Barcelona Supercomputing Center* ↔ *Polytechnical University of Catalonia*; *Riken* ↔ *University of Tokyo*; *Sandia Lab. New Mexico* ↔ *University of New Mexico*; and *Argonne Lab.* ↔ *University of Chicago*.

3.2.3 *Text Analysis.* Figure 9 visualizes the output of topic modeling. Each topic is assigned a letter (A-Y) and the words having the highest weights are shown. A table of the results is in Appendix B (Table B.2).

[2] Interactive network visualization available at https://exascale-survey.github.io/assets/citation.html.

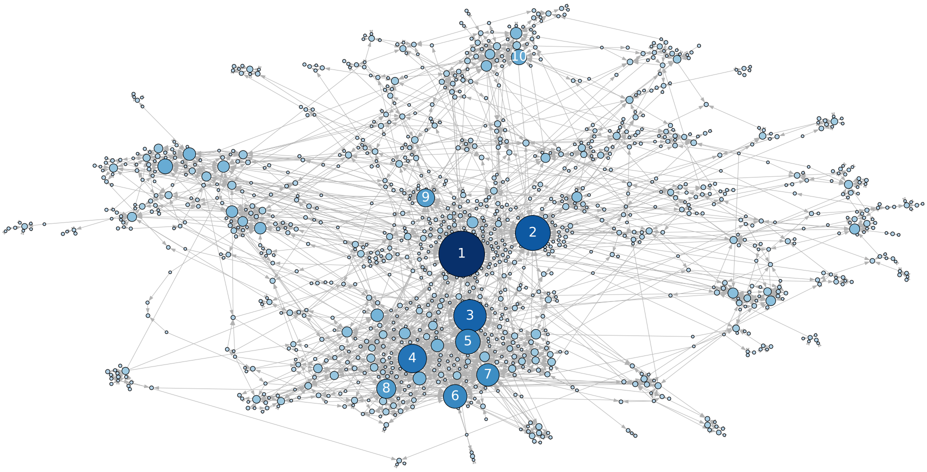


Fig. 8. Citation network. Nodes are publications and edges represent citations. Node size and color indicate number of citations. Numbers indicate top 10 publications (see Table B.1). Interactive view available in Footnote 2.

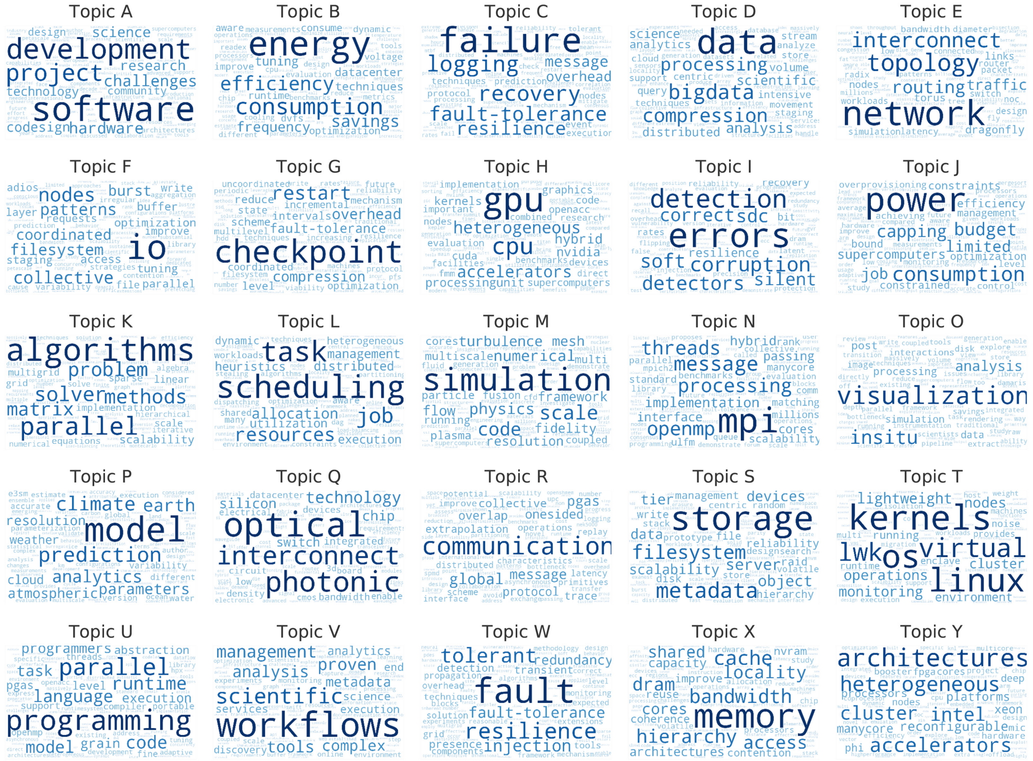
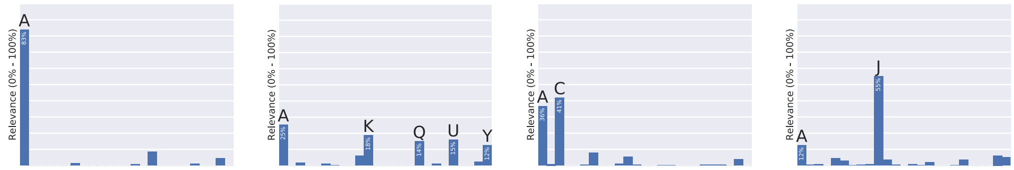


Fig. 9. Visualization of topic model using word clouds. Each word cloud represents one detected topic where the size of words indicates the relevance of each word to that particular topic (i.e., weights in matrix **U**).



(a) The International Exascale Software Roadmap [54].

(b) Exascale Computing Technology Challenges [162].

(c) Fault Tolerance in Petascale/Exascale Systems [31].

(d) Exploring Hardware Overprovisioning in Power-Constrained HPC [141].

Fig. 10. Examples of topic distribution for four articles. The height of the bars represents the elements of matrix U indicating the relative degree of which topics are applicable to articles.

Based on the dominant terms, it is clear that each topic represents some coherent area of research. For example, topic E (top three words are *network*, *topology*, *interconnect*) is on network topologies, topic X (*memory*, *bandwidth*, *hierarchy*) is on memories, topic L (*scheduling*, *job*, *resources*) is on job scheduling and, topic U (*programming*, *parallel*, *code*) is on parallel programming.

Figure 10 illustrates the topic distributions for four articles. In general, we observe that the weights are either large (i.e., the topic is strongly related) or near zero (i.e., the topic is unrelated). For most articles, only a small number of topics (i.e., between 1 and 5) is applicable. For example, the *International Exascale Software Roadmap* [54] (Figure 10(a)) strongly belongs to topic A (*development, software, hardware*). On the other hand, *Exascale Computing Technology Challenges* [162] (Figure 10(b)) belongs to five topics (topic H, K, Q, U, Y), indicating that this work touches upon multiple issues. More examples are available in Appendix B (Figure B.4).

Figure 11 shows the embedding of the documents into two-dimensional space. In this visualization, the documents form clusters according to their topics. Some of these clusters are more isolated than others. For example, documents for topic B (*energy, consumption, efficiency*) and for topic J (*power, consumption, budget*) are close to each other, indicating that these two topics are related. On the other hand, documents for topic O (*visualization, in-situ, analysis*) appear as an isolated cluster in the two-dimensional embedding.

3.2.4 Research Themes. Based on the detected terms for the topics (Figure 9) and the visualization (Figure 11) we group the topics into eight research themes based on our interpretation. Four research themes appear as clear clusters in the visualization: *fault tolerance* (Topic C, G, I, W), *energy/power* (Topic B, J), *networks* (Topic E, Q), and *data storage/analysis* (Topic D, S, F, V, O).

The remaining topics are present near one central "island" in the visualization, indicating that the corresponding publications are closely related to each other. These topics are mostly on various topics from HPC such as MPI, GPU programming, numerical simulations, parallel algorithms, scheduling, and operating systems. Based on our interpretation, we divide these topics into four research themes: topics at *architecture*-level such as memories, GPUs and FPGAs (topic H, X, Y), topics on *parallel programming* such as libraries and programming languages (topic U, R, N), topics on *system software* such as operating systems and job schedulers (topic L, T), and topics on *scientific applications* such as scientific simulations and parallel algorithms (topic A, K, M, P). Figure 12 shows where these research themes are located in the visualization.

4 QUALITATIVE LITERATURE ANALYSIS

In this section, we depart from the data-driven approach and we further explore each of the research themes from Section 3.2.4. We have manually selected relevant publications from the search results for each research theme and identified the promising research trends. This section mostly revolves around these publications, although additional sources are referenced when deemed

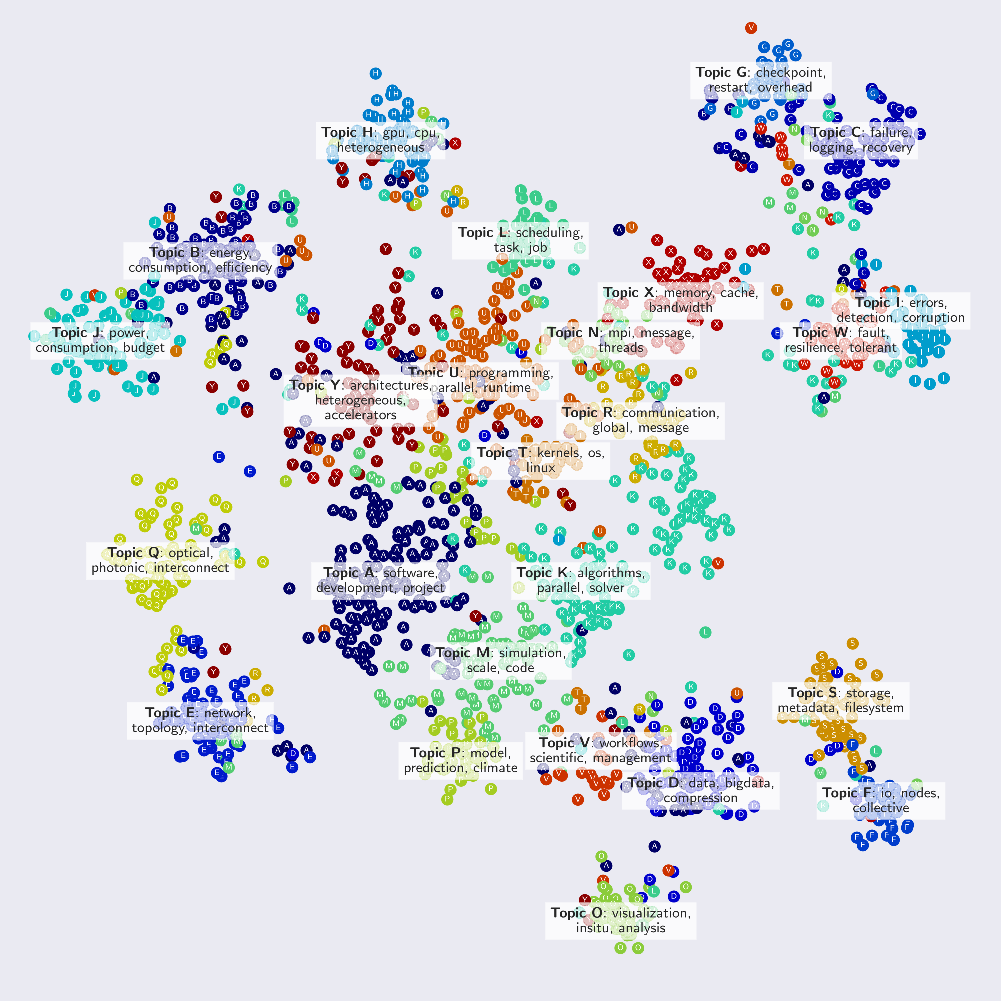


Fig. 11. Embedding of documents into two-dimensional space. Each point represents one document with letter/color indicating its dominant topic. Distance between documents inversely corresponds to lexical similarity. Labels have been placed manually and list the top three terms from Figure 9.

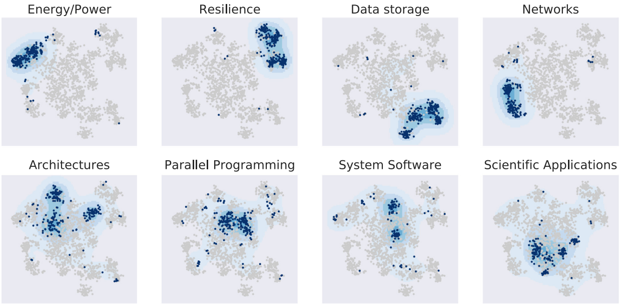


Fig. 12. Documents from Figure 11 highlighted for the eight research themes.

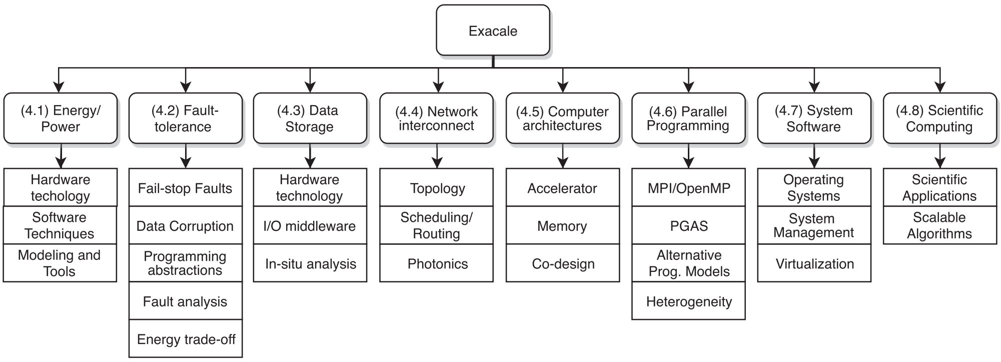


Fig. 13. Research themes and identified research trends within each theme from Section 4.

Table 2. Further in-depth Reading for Each Research Theme

Theme	Literature
4.1 Energy/Power	Jin et al. [98], Czarnul et al. [42], Zakarya et al. [186]
4.2 Fault Tolerance/Resilience	Herault and Robert [92], Snir et al. [166], Cappello et al. [33]
4.3 Data Storage	Lüttgau et al. [116], Kunkel et al. [105]
4.4 Network interconnect	Trobec et al. [172], Thraskias et al. [170]
4.5 Computer Architecture	Kogge and Shalf [104], Shalf et al. [163]
4.6 System Software	Naughton et al. [134], Gerofi et al. [76]
4.7 Parallel Programming	Gropp and Snir [84], Diaz et al. [51], Dongarra et al. [54]
4.8 Scientific Computing	Ashby et al. [14], Åström et al. [15]

necessary. For each research theme, we discuss relevant work and end with a conclusion of progress in this area.

Since each theme is broad, this section aims to provide a bird's-eye view of the different aspects of exascale computing. Figure 13 shows the eight research themes and the identified trends within each theme. Note that this is not a strict classification, as some articles touch upon multiple topics (e.g., research on energy-efficiency of storage systems). For expert technical readers, we compiled a list of relevant in-depth literature dedicated to each theme (Table 2).

4.1 Energy/Power

Energy consumption is an important challenge and there has been a tremendous amount of progress. Figure 14 shows the performance and energy efficiency of the TOP500 over time. In June 2009, *Roadrunner* was no. 1 in the TOP500 [4] and delivered 0.41 PFLOPS/MW (1.1 PFLOPS at 2.4MW). At that time, an exascale system would have required over 2,000MW. In June 2019, *Summit* was no. 1 in the TOP500 and delivered 14.7 PFLOPS/MW (149 PFLOPS at 10MW). US DARPA established 20MW as a reasonable power envelope for an exascale system (see Section 2.8), thus requiring an energy efficiency of 50 PFLOPS/MW for the whole system. The yearly energy-efficiency increase was ~56% and extrapolating this trend means this goal would be reached around 2022.

Much of this improvement came from *hardware technology*, but *software techniques* also play an important role. There has also been tremendous progress in *models* and *tools* on energy usage.

4.1.1 Hardware Technology. An HPC system consists of many components (i.e., processors, memory, interconnects, power delivery, cooling, persistent storage) and for each, there is active

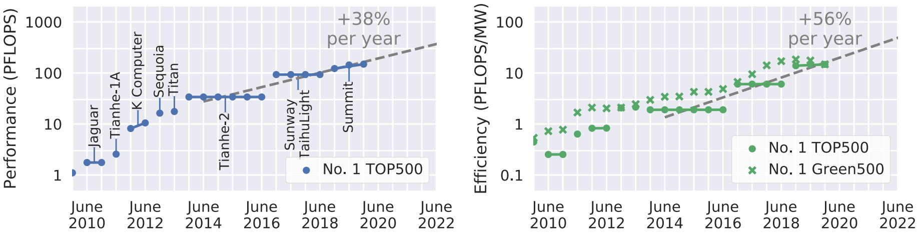


Fig. 14. Performance/energy efficiency of the no. 1 in the TOP500 [4]. Dashed lines extrapolate trends over the time period 2014-2019. The right figure also includes the no. 1 from the Green500 [4].

research into improve its energy efficiency. For example, optical interconnects [172] replace copper cables by optical links to deliver high bandwidth at low power consumption. Non-volatile memory (NVM) [178] is a novel memory technology that provides better energy efficiency than conventional DRAM. Villa et al. [179] explains how NVIDIA plans to reach exascale using low-voltage SRAM, low-energy signaling, and on-package memory. The Mont-Blanc project [149] experiments with building HPC clusters from low-power components from mobile computing.

4.1.2 Software Techniques. Solutions have been proposed at the node-level or the system-level. At the node-level, a common technique to improve energy efficiency is to dynamically scale the frequency, voltage, or level of concurrency for different components depending on the workload. For example, Porterfield et al. [147] show how energy consumption is affected by aspects such as the algorithm, compiler, optimizations, number of threads, and temperature of the chip. Sarood et al. [158] propose a thermal-aware load-balancing solution that limits processor temperatures and results show reduction of cooling demand of up to 63% while barely affecting run times. Haidar et al. [86] investigate the effect of power limiting for various scientific kernels and find reductions in energy consumption (up to 30%) without impacting performance.

At the system-level, research focuses on solutions to efficiently distribute available power over nodes while not exceeding the power cap. For instance, Bodas et al. [25] present a power-aware job scheduler that continuously monitors power consumption of jobs and adjusts clock frequencies. Patki et al. [141] compare traditional *worst-case* power provisioning (i.e., maximum power draw of hardware is within system power cap) to *overprovisioning* (i.e., maximum power draw exceeds system power cap) with dynamic power management and they find up to 50% improvement in performance. Gholkar et al. [79] propose a two-level power management solution: the system's power budget is divided over each job and each job further distributes the power across the nodes. Results show performance improvement of up to 29% compared to a static per-node power cap.

4.1.3 Modeling and Tools. Tools for measuring and modeling energy-efficiency of existing systems are important. Borghesi et al. [26] use machine-learning to predict the power consumption of typical HPC workloads and show promising results. Mair et al. [120] extensively analyze trends in the TOP500. They find that the greenest computers are often small-scale deployments and they propose a new energy-efficiency metric that also takes into account system scale. Cabrera et al. [30] present the *Energy Measurement Library* (EML): a framework that standardizes energy measurements across different devices (such as Intel CPUs, NVIDIA GPUs, and Power Distribution Units (PDU). Shoukourian et al. [164] propose the *PowerDAM* toolset for collecting sensor data in HPC clusters and correlating sensor data with resource management data.

4.1.4 Discussion. Overall, we see tremendous progress for energy-efficiency of hardware and energy-awareness of software. We also observe progress in tools for measuring/modeling energy consumption and in energy-efficient solutions for fault tolerance. Extrapolating the current trend shows that we can expect to reach the target of 20MW for one exaflop (see Section 2.8) around the year 2022. We conclude that, although energy efficiency is likely to remain a major concern in HPC, we are reaching the point where building an exascale system is feasible in terms of energy usage.

4.2 Fault Tolerance

Fault tolerance (also called *resilience*) refers to the ability of a system to continue operation, despite failures in the underlying software or hardware. In 2009 [32] and 2014 [33], Cappello et al. presented their thoughts on the state of resilience for exascale computing at that moment in time. They anticipated that exascale systems would experience much higher failure rates than petascale systems (possibly failing once every 30 minutes [166]) since the number of components is increasing while components' failure rate is not improving.

Much work in exascale fault tolerance is dedicated to handling *fail-stop faults* (i.e., faults where a node stops abruptly), but there is also increasing amounts of work on *fail-continue faults* (i.e., faults were a node continues), such as data corruption. Other trends are in *programming abstractions*, helping programmers integrate fault tolerance into their applications, and *fault analysis*, to better understand the cause and impact of faults for supercomputers.

4.2.1 Fail-Stop Faults. Traditionally, fault tolerance research in HPC has focused on handling *fail-stop* [33] faults (i.e., a process fails and stops abruptly) by performing globally synchronized checkpoint-restart: system execution is paused at regular intervals to save application state to remote storage, which can be used to restart the job in case of failure. However, if the number of nodes grows, the mean time between failures shrinks, thus checkpointing becomes less effective since execution time would be dominated by checkpoint/restart time.

Much work is thus dedicated towards reducing checkpointing/restart overhead. For example, Di et al. [49] consider *multi-level* checkpointing: writing checkpoints at different time intervals to different levels in the memory hierarchy (e.g., local memory, remote memory, local disk, and remote file system). They show that, with optimal selection of levels and parameters, their solution outperforms single-level methods by up to 50%. Zheng et al. [187] propose an in-memory checkpoint solution that does not use stable storage and their evaluation on 64,000 cores shows checkpoint/restart times in milliseconds. Sato et al. [159] consider non-blocking checkpointing (i.e., overlapping I/O with computation) and find high efficiency even when I/O bandwidth is limited. Dong et al. [52, 53] experiment with fast non-volatile memory (NVM) to reduce checkpoint overhead. Their predictions for exascale systems show overhead of at most 4%, a significant improvement over the current 25% for petascale systems. Ibtesham et al. [96] use compression to reduce checkpoint size and thus the amount of I/O, at the cost of additional compute time.

Alternatives to checkpointing are also subject of study. Ferreira et al. [68] evaluate the suitability of replication as the primary resilience method for exascale and show how it outperforms traditional checkpointing for some applications. Pickartz et al. [145] see opportunities for process migration by moving active processes away from faulty nodes. Results are promising, but require accurate fault prediction. Integrating fault tolerance into MPI is a common research topic [22, 23, 74].

We also observe that the number of compute nodes has not grown as expected. For example, *Roadrunner* (1.1 PFLOPS, 2008 [4]) consisted of 3,240 compute nodes [114]. *Titan* (27 PFLOPS, 2012 [4]) was the first GPU-accelerated supercomputing to be ranked 1st in the TOP500 and consisted of 18,866 compute nodes each having a NVIDIA K20x GPU [136]. *Summit* (149 PFLOPS, 2018 [4]) is currently ranked 1st in the TOP 500 and consists of just 4,608 compute nodes each having 6 NVIDIA V100 GPUs [106]. The move towards "leaner" clusters of "fatter" nodes may imply that traditional checkpointing methods are still adequate for the coming years.

4.2.2 Data Corruption. Besides fail-stop faults, an upcoming trend is research into *fail-continue* faults such *silent data corruptions* (SDC) [33]. These faults are caused by transient hardware errors, such as bit flips in memory or arithmetic units, and are difficult to combat since execution could continue normally. These events are exceedingly rare under normal operation, but several researchers [32, 33, 166] contest that these errors can no longer be ignored when further scaling systems. Di and Cappello [50] characterize the impact of bit-flip errors on execution results for 18 HPC applications. Liu and Agrawal [111] use machine learning with offline training to detect signatures of SDCs. Gomez and Cappello [83] present a solution that uses interpolation and prediction methods to detect and correct SDCs in stencil applications. Huber et al. [94] present a resilience solution for parallel multigrid solvers that recomputes corrupted values based on redundant storage of ghost values.

4.2.3 Programming Abstractions. Since much is unsure about future fault tolerance solutions, there is active research into programming abstractions for resilience. For example, Chung et al. [39] present *containment domains*: a programming construct that enables programmers to explicitly define the fault tolerance requirements for sections of code. Rolex [95] is a C/C++ language extension that incorporates resilience into application code. Chien et al. [37] propose *versioned distributed arrays* as a portable solution to extend existing codes with resilience. Results for four real-world applications (OpenMC, PCG, ddcMD, and Chombo) show that the required changes are small, localized, and machine-independent. Meneses et al. [125] demonstrate that migratable objects form a scalable programming model for exascale computing.

4.2.4 Fault Analysis. Much is unknown about the cause/impact of failures in supercomputing centers and exascale computing has sparked interest in this area. Martino et al. [122] analyze 5 million HPC jobs submitted between 2013 and 2014 to *Blue Waters* (13.1 PFLOP/s) and they present a large number of findings. For instance, while only 1.5% of the jobs fails due to system problems, they account for 9% of all compute time. Additionally, failure rates increase dramatically when increasing job size: going from 10,000 to 22,000 nodes increases failure probability from 0.8% to 16.2%. El-Sayed and Schroeder [64] analyze failures from a decade of historical data from Los Alamos National Laboratory. They present several conclusions, for example that some nodes experience significantly more failures than others (even if hardware is identical) and once a node fails, it is likely to experience follow-up failures. Gupta et al. [85] perform a large in-depth study using data from more than one billion compute hours across five different supercomputers over a period of 8 years. They present many findings, including that failures show temporal recurrence, failures show spatial locality, and reliability of HPC systems has barely changed over generations.

4.2.5 Energy Tradeoff. As discussed in Section 4.1, energy is one of the main exascale challenges. A notable research trend is the tradeoff between fault tolerance and energy-consumption: increasing system resilience often comes at a cost of decreased energy-efficiency and vice-versa. For example, Giridhar et al. [82] consider the tradeoff in memory design between the benefit of memory error correction versus its energy cost. Sarood et al. [157] study the tradeoff between component reliability and cooling (since high temperatures increase fault rates) and propose load-balancing techniques that move active processes to "cool" nodes. Chandrasekar et al. [36] propose a power-aware checkpoint framework and results show up to 48% reduction in energy consumption during checkpointing. Dauwe et al. [46] optimize multi-level checkpointing for performance and energy-efficiency and find that optimizing both metrics simultaneously is not possible.

4.2.6 Discussion. For fault tolerance, much of the work is related to improving existing checkpointing solutions, alternatives to checkpointing, and understanding faults in supercomputers.

There are also crossovers to the other research themes such as energy consumption (e.g., tradeoff between ECC and energy usage) and parallel programming (e.g., containment domains). Checkpointing seems to stay a feasible solution for exascale fault tolerance, taking into account recent developments such as multi-level checkpoints, non-volatile memory, and non-blocking checkpointing. Snir et al. [166] support this by arguing that "recent results in multilevel checkpointing and in fault tolerance protocol [...] make checkpoint/restart a viable approach for exascale resilience" [166, p.152]. However, silent data corruption (SDC) might require more research since their impact is not well understood. All in all, we argue that exascale systems are feasible in terms of fault tolerance.

4.3 Data Storage

Exascale storage systems must handle massive volumes of data, both for scientific discovery (i.e., inputs/outputs of jobs) as well as fault tolerance (i.e., checkpoints). Kunkel et al. [105] analyze historical data from the *German Climate Computing Center* (DKRZ) and predict processor performance growth by 20× each generation (~5 years), while storage throughput/capacity improves by just 6×. Hu et al. [93] quantify this problem and define the *storage wall* for exascale computing: some I/O-intensive applications may never scale to exaflops due to I/O bottlenecks.

As we are moving towards exascale, the gap between computing power and I/O bandwidth will widen and researchers are looking for solutions to tackle this problem. There are essentially three lines of research: at *hardware* level, at *middleware* level, and at *application* level. For application level, one prominent trend is *in-situ analysis.*

4.3.1 Hardware Technology. One important research trend in storage systems is the use of *non-volatile memory* (NVM). The capabilities of NVM (i.e., capacity, bandwidth, energy consumption) are somewhere in-between main memory and persistent storage, thus it is often used as a "caching" solution between these two layers. Bent et al. [18] propose *burst buffers*: extending I/O nodes (i.e., between compute and storage nodes) with *solid state drives* (SSDs) to aggregate many small I/O requests into few larger ones. HPC applications often show bursty I/O behavior (i.e., all processes read/write at the same time) and burst buffers help to absorb these workloads. Their simulations show performance increase of 30% and the authors consider inclusion of burst buffers to be necessary for exascale. da Silva et al. [69] measure the benefit of burst buffers at *Cori* (14 PFLOPS) [4] for two applications and find read and write performance increases by an order of magnitude. Congiu et al. [40] attach SSDs to compute nodes to speed-up MPI-IO operations and they find significant improvements in I/O bandwidth. The SAGE project [133] propose a four-layer storage hierarchy incorporating NVM, SSDs, and two types of hard disk drives. Overall, the exact storage solution for exascale is not clear, but it likely consists of multiple layers.

4.3.2 I/O Middleware. I/O performance can also be improved by solutions in I/O middleware (e.g., file systems, I/O interfaces). Dorier et al. [57] observe that I/O performance is unpredictable and processes often need to wait for the slowest one. They propose a software framework (*Damaris*) that overlaps computation and I/O operations by dedicating a single to core to I/O tasks. ADIOS [112] is an I/O abstraction framework for HPC applications that enables switching between different I/O transport methods with little modification to application code, enabling integration of new I/O solutions. Liu et al. [112] discuss ADIOS' history and lessons learned during its development. They foresee four major challenges in moving ADIOS towards exascale: data integrity, fault recovery, automated tuning, and data consistency. *DeltaFS* [188] is a file system that improves the scalability of file systems by letting compute nodes manage metadata instead of a centralized server (common in traditional distributed file systems). Kunkel et al. [105] review three methods to improve I/O performance at exascale: *re-computation* (found to be beneficial for infrequent accesses), *deduplication* (not promising), and *data compression* (promising, but application-dependent). Lofstead et al. [113] propose a solution that allows different storage technologies to be deployed over time without changes to application code. Their design exposes a high-level model-based programming interface and uses a low-level layer (*DAOS*) to translate the user-defined model to underlying storage.

4.3.3 In-Situ Analysis. Another promising trend is *in-situ* analysis [117]. Rather than applications writing their raw output to storage to later be read again for post-processing (e.g., visualization, filtering, statistics), in-situ processing removes this overhead by performing the analysis directly on the same machines as where the applications run. Ma et al. [117] provide an overview of the challenges and opportunities of in-situ processing for exascale computing. Yu et al. [185] show how large-scale combustion simulations benefit from in-situ visualization. ParaView [67], Dax [130], and Damaris/Viz [59] are tools for large-scale in-situ visualization.

4.3.4 Discussion. An important concern is the mismatch between the massive computational performance of processors and relatively limited I/O bandwidth of storage systems. There are essentially three methods to alleviate this problem: new hardware technology (e.g., non-volatile memory), new I/O middleware (e.g., ADIOS, DAOS), and application-specific solutions (e.g., in-situ analysis). Hardware technology shows promising solutions, but different systems might employ different solutions (i.e., SSDs, NVM, burst buffers), reducing the portability and increasing the complexity of software. Middleware can alleviate some of this complexity with solutions such as ADIOS. In-situ analysis is an example of how application-specific solutions can be used to improve I/O throughput and thus application performance. All in all, there appears no one-size-fits-all solution to the storage problem and programmers must take I/O into careful consideration when developing applications.

4.4 Network Interconnect

Exascale systems require ultra scalable, low latency, high bandwidth network interconnects. Trobec et al. [172] predict the important trends for future interconnects by reviewing the past networks from the fastest petascale systems. They observe that, while reasonable techniques for routing and flow-control algorithms are known, optimal network topology is still a topic of research. By extrapolating current trends, they find the requirements for exascale networks: novel topologies scalable to millions of nodes, adaptivity of scheduling/routing to specific applications, high-radix switches for high bandwidth, and optical links for low latency. In this section, we discuss three relevant research areas: *network topology*, *scheduling/routing*, and *optical interconnects*.

4.4.1 Network Topology. The topology of a network determines how nodes are connected and how they communicate. No topology is optimal since their design involves tradeoffs in terms of latency, bandwidth, wiring complexity, cost, power consumption, and resilience. Several designs have been proposed having exascale "ambitions". An important trend is the inclusion of *high-radix switches* [172] (i.e., network switches having hundreds of ports) to simplify network topology. Flajslik et al. [71] consider 3-hop topologies to be the most likely candidate for exascale system in which each path is at most three hops. They propose a design called *Megafly*: a hierarchical topology with high path diversity. *HyperX* [11] is an extension of the hypercube topology and also takes advantage of high-radix switches. *Tofu* [12] is a 6D torus topology developed by Fujitsu for the K computer (8 PFLOPS) [4]. *BXI* [48] (*Bull eXascale Interconnect*) is an interconnect developed by Atos that allows offloading many communication primitives to hardware.

Modeling and simulating is important to enable exploring different designs. Mubarak et al. [131] propose an accurate simulation framework for exascale networks. Pascual et al. [140] design and

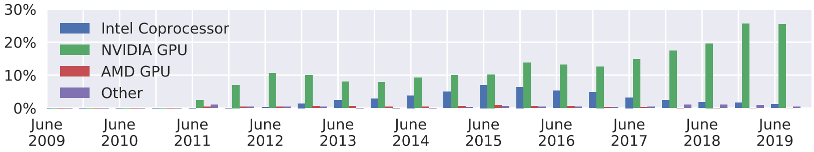


Fig. 15. Percentage of TOP500 systems [4] equipped with accelerator per vendor.

simulate interconnects by modeling them as a multi-objective optimization problem (i.e., fault tolerance, performance, cost).

4.4.2 Scheduling and Routing. Besides the (static) network topology, performance can also be improved by (dynamic) scheduling/routing algorithms that are workload- or topology-aware. For example, Prisacari et al. [148] study the effect of task placement in systems with dragonfly networks. Dorier et al. [58] evaluate different collective operations and study the effect of topology-awareness and sensitivity to network contention. Bhatele et al. [20] propose routing techniques to prevent hot-spots in dragonfly networks. Totoni et al. [171] study the effectiveness of disabling underutilized links and reduce power consumption by 16% without impacting performance.

4.4.3 Photonics. A promising technology is *optical* interconnects [154, 170] that replace traditional copper wires by optical links. Optical interconnects could potentially offer lower latency, lower power consumption, and higher throughput than current solutions. However, according to Rumley et al. [154], the technology is not yet cost-effective except for distances over several meters. There is also much ongoing work into on-chip optical interconnects [17, 182] that enable efficient small-scale links (e.g., core to core, processor to memory). Werner et al. [182] argue that the technology will be disruptive for HPC, but is not yet production-ready.

4.4.4 Discussion. For *network interconnects*, important trends are high-radix switches, modeling, scheduling/routing, and optical interconnects. Several topologies have been proposed for exascale, each having its own advantages and disadvantages. Photonics is a promising direction, both for inter- and intra-node connections, but the technology is not yet ready for all types of communication links. We foresee that the gap between computing power and network bandwidth will continue to increase and this could limit the scalability of various scientific applications.

4.5 Computer Architecture

It is likely that future platforms integrate new technology alongside conventional components. Two important developments are research into *hardware accelerators* and developments in *memory technology*. Another active topic of research is *co-design*: the process where hardware design is optimized for specific applications.

4.5.1 Accelerators. Figure 15 shows the percentage TOP500 systems equipped with accelerators. The figure shows that the popularity of GPUs in HPC systems has skyrocketed in just 8 years, thus is it likely that exascale system include GPU-like accelerators. Mittal and Vetter [129] performed an extensive survey on CPU-GPU heterogeneous computing and conclude that a collaboration between CPU and GPU is inevitable to achieve the exascale goals. To achieve exascale performance, AMD [160] envisions a heterogeneous processor (*accelerated processing unit*, APU) that integrates both CPUs and GPUs on the same chip. They consider such a design to be ideal for HPC since it combines the high throughput and energy-efficiency of GPUs with the fast serial processing of CPUs. Nikolskiy et al. [135] benchmark the energy-efficiency and performance of the Tegra X1 and K1 for algorithms from molecular dynamics and show promising results. Tegra is a system on a chip (SoC) that integrates an ARM-based CPU and NVIDIA-based GPU.

Some work focuses on exploring alternative accelerators besides GPUs. For instance, O'Brien et al. [137] evaluate the performance and energy efficiency of a dynamic programming application on three platforms: FPGA, GPU, and Intel Xeon Phi. Overall, they find that GPUs deliver the best performance, the FPGAs demonstrate the best energy efficiency, and the Xeon Phi is easiest to program but shows mediocre results (both run times and energy efficiency). Varghese et al. [176] discuss their experiences with the Adapteva Epiphany 64-core network-on-chip co-processor. They consider the device to be suitable for exascale computing due to its energy-efficiency, but programming is difficult due to the low-level primitives and the relatively slow external shared memory interface. Mitra et al. [128] experiment with the TI Keystone II architecture for exascale computing and conclude it is a low-power solution but programming is challenging.

There are also research projects exploring alternative system designs: DEEP [63] focuses on developing a *Booster* cluster composed of Intel Xeon Phi co-processors, EXTRA [167] proposes FPGAs as exascale technology, and Mont-Blanc [149] (discussed in Section 4.1.1) aims to build HPC systems out of low-power components.

4.5.2 Memory. Novel memory technology is an active area of research. Some contributions were already discussed in previous sections, such as Dong et al. [52, 53] using 3D stacked memory to reduce checkpointing overhead and Kannan et al. [102] using active *non-volatile memory* (NVM) to reduce I/O overhead. Meswani et al. [127] propose a two-level memory hierarchy for future systems: a first-level stacked memory to provide large bandwidth and second-level conventional memory to provide large capacity. They experiment with a programmer-managed memory system and use three exascale applications as case studies. Nair et al. [132] present an architecture called the *active memory cube* that performs computations within the memory modules, thus reducing data movement between memory and CPU. Li et al. [109] identify possible opportunities of NVM for exascale applications. Their simulations suggest that many applications are not affected by longer write latency and energy improvements are 27%. Vetter and Mittal [178] enumerate the many possible opportunities of NVM in exascale systems.

4.5.3 Co-design. An active topic of research in exascale computing is *co-design*: the process of optimizing hardware architectures and application software in unison, with the requirements of one influencing the other and vice-versa. Shalf et al. [163] argue that co-design is beneficial since it is application driven: scientists can decide what future systems need to offer to solve their scientific problems, instead of looking for specific applications that match existing systems. *CoDEx* [163] is a co-design environment that consists of three key elements: tools for extraction of memory/interconnect traces from applications, a cycle-accurate hardware simulator, and a coarse-grained interconnect simulator. ExaSAT [35, 173] is a framework for co-design that uses compiler analysis to automatically extract performance models from source code. Dosanjh et al. [60] propose a co-design methodology based on a hardware simulator and they study the accuracy for several algorithms and architectures.

4.5.4 Discussion. For *computer architectures*, important trends are accelerators, novel memory technology, and co-design. The popularity of GPUs in HPC has increased rapidly over the past decade and it is highly likely that future systems include GPUs (or similar accelerators). Novel memory technologies (e.g., non-volatile memory, processing-in-memory, multi-level memory) are promising, but they increase heterogeneity and introduce additional levels in the data hierarchy. Co-design helps to bring hardware capabilities closer to application requirements, but it is unlikely that the first exascale systems will be built using application-specific hardware. All in all, it is evident that computer architectures will continue to become more complex, heterogeneous, and hierarchical over time, increasing the complexity in understanding and programming these systems.

4.6 Parallel Programming

Programming exascale applications requires rethinking our current approaches. Sarkar et al. [156] argue that two aspects are crucial at exascale: *parallelism*, due to the massive concurrency, and *locality*, since moving data is expensive compared to computation. They consider the primary exascale challenges to "boil down to the ability to express and manage parallelism and locality in the applications" [156, p.10]. Da Costa et al. [41] discuss the limitations of existing programming models (e.g., MPI, OpenMP, OpenCL, CUDA) for exascale applications. They conclude that "new programming models are required that support data locality, minimize data exchange and synchronization, while providing resilience and fault tolerance mechanisms" [41, p. 21] since "no programming solution exists that satisfies all these requirements" [41, p. 7]. Parallel programming is clearly an important topic and this section discusses four important trends: *MPI/OpenMP*, *PGAS*, *alternative programming models*, and *heterogeneity*.

4.6.1 MPI and OpenMP. According to Gropp and Snir [84], MPI is ubiquitously used in HPC as communication library, often in combination with OpenMP for thread-level parallelism. They foresee issues in scaling MPI+OpenMP to exascale: MPI is too low-level and relies on the programmer to use it well and OpenMP encourages fine-grained synchronization that does not scale.

Several researchers investigated scaling the MPI+OpenMP approach to exascale. Bosilca et al. [27] investigate the scalability of MPI in practice and isolate two major considerations: parallel launching and distributed management. They estimate that it is possible to launch up to 20,000 MPI ranks within 1 minute. Engelmann [65] evaluates MPI runs for up to 2^{24} nodes using simulations. Iwainsky et al. [97] experiment with the scalability of OpenMP constructs. Results show that the OpenMP overhead is compiler-dependent and scales linearly with the number of threads, which is undesirable for massively parallel processors. Jin et al. [99] argue that OpenMP's flat memory model does not match current architectures which limits performance. OpenMP syntax extensions are proposed to control locality of tasks and data.

4.6.2 PGAS. PGAS (*partitioned global address space*) [84] is a possible parallel programming solution for exascale computing. PGAS aims to ease programming of distributed systems by offering a global memory address space partitioned across the nodes. Each node can access the entire address space (usually via *direct memory access* (DMA)), but locality of data is made explicit since local access is faster than remote access. DASH [72] is a data-structure oriented library build on the PGAS model. Chapel [89] is a PGAS-based programming language by Cray Inc. UPC++ [16] is a library based on *asynchronous* PGAS, adding the ability to spawn and coordinate tasks. The EPiGRAM project [121] aims to improve the interoperability of MPI and PGAS.

4.6.3 Alternative Programming Models. Several alternative programming platforms have been proposed. For example, HPX [100] is a parallel task-based runtime offering a global address space, fine-grained parallelism, lightweight synchronization mechanisms, message-driven computation, and explicit support for hardware accelerators. The *Open Community Runtime* (OCR) [124] is proposed as a runtime system for high-level exascale programming models, although recent development has been scarce. StarSs [168] is a programming model that enables programmers to parallelize existing sequential application using code annotations. Legion [90] is a scalable data-centric framework for programming distributed platforms. PaRSEC [45] is a data-flow programming framework in which dataflow dependencies can be expressed by means of a *Parameterized Task Graph* (PTG), an input-size independent representation of the dataflow graph that allows nodes to determine data dependencies in a distributed and independent way. Yang et al. [184] consider *hierarchically tiled arrays* (HTA) to be promising high-level abstraction for writing exascale applications. Van der Wijngaart et al. [175] compare the scalability and performance of eight different programming models having “exascale ambitions” on three different benchmarks, but found no clear winner.

4.6.4 Heterogeneity. As mentioned in Section 4.5.1, accelerators are likely to be included in future exascale systems. According to Da Costa et al. [41], these accelerators are programmed using either CUDA (NVIDIA GPUs) or OpenCL (OpenCL-compliant devices), but using these low-level languages remains challenging and high-level programming models are needed. Lee and Vetter [108] see directive-based programming models as a solution to simplify programming GPUs. They compare five different platforms (PGI accelerator, OpenACC, HMPP, OpenMPC, and R-Stream) and find that performance of these high-level models is competitive compared to hand-crafted GPU programs. The Kokkos library [34] is a portable many-core programming model that compiles to many different back-ends (e.g., OpenMP, CUDA, OpenCL). OpenMC [110] is a programming model which provides a unified abstraction of compute units, meaning code is portable across CPUs and accelerators. Hairi et al. [87] accelerate large-scale particle-in-cell (PIC) simulations using OpenACC. They experiment with different parallelization strategies and show how the optimum depends on the problem and architecture, meaning tuning of OpenACC applications is a necessity. Rasmussen et al. [150] explore how to extend Coarray Fortran with support for accelerators.

4.6.5 Discussion. For *parallel programming*, there are essentially two lines of research: improving the traditional programming models (i.e., MPI+OpenMP) and developing alternative models. MPI could potentially scale to thousands of nodes (shown in simulations and controlled experiments) and PGAS is an attractive alternative. However, it is questionable whether these tools are the right solution for exascale computing since they do not take into account many of the concerns of exascale such as fault tolerance, energy management, heterogeneity, and accelerators. Several programming frameworks have been proposed, such as HPX, StarSs, Kokkos, OCR, PaRSEC, but it is unclear which model, if any, will become the de-facto standard for future systems. The lack of a standardized programming model, combined with the increasing complexity of computer architectures (see Section 4.5), means that writing exascale applications becomes an extremely challenging task.

4.7 System Software

System software is the software that is situated in-between the hardware and application code. It is important for scheduling jobs and managing resources. We discuss three important directions relevant for exascale computing: *operating systems*, *system management*, and *virtualization*.

4.7.1 Operating Systems. One research direction is on LWKs (*light-weight kernel*) for exascale systems: operating systems that reduce overhead by providing minimal functionality and services. Giampapa et al. [81] describe their experience with the LWK on Blue Gene/P, carefully describe the tradeoffs between light-weight and full kernels, and discuss considerations for the future. Gerofi et al. [77] explore a hybrid solution that runs a LWK and FWK (*full-weight kernel*) side-by-side: applications run on the LWK and additional features are accessible through the FWK. Weinhold et al. [181] propose a full system architecture for exascale computers incorporating LWKs. Despite these efforts, the diversity in operating systems in the TOP500 is decreasing [4] and the list for June 2019 shows over 95% run a conventional Linux-based solution. For instance, *Argo NodeOS* [144] is a Linux-based system specifically designed for exascale computing.

4.7.2 System Management. PanDA [119] is a large-scale workload management system. Although originally developed for the ATLAS experiment at the Large Hadron Collider, the next generation of PanDA targets a wider audience. Argo [143] (project by Argonne National Laboratory) aims to develop a workload manager and OS for exascale systems. Argo divides the system into *enclaves*: sets of resources that can be controlled and monitored as a whole (e.g., power management, performance monitoring, fault tolerance). These enclaves form a hierarchy with the complete system at the root to enable resource management at different levels.

4.7.3 Virtualization. While *virtualization* completely dominates cloud computing, it is still a relatively rare sight in HPC clusters, although there is research into how to use these techniques for HPC. For instance, Pickartz et al. [145] argue that application migration has benefits for exascale computing (e.g., fault tolerance, maintenance, load balancing) and propose a method to migrate MPI processes using *virtual machines* (VM). Ouyang et al. [138] describe how to achieve *performance isolation* by running multiple VMs on the same physical machine, which is needed for concurrent jobs on one node (e.g., *in-situ* analysis).

Containerization is similar to virtualization in that it provides an isolated environment for running application software, with the difference being that containers run directly on host's operating system instead of a VM. This minimizes overhead and makes it an attractive technology for HPC. For instance, the Argo NodeOS [189] uses containers to isolate jobs and manage resources.

4.7.4 Discussion. For system software, important topics are operating systems, workload managers (e.g., Argo project, PanDA), and virtualization. Linux is likely to remain the dominant operating system. Workload managers and virtualization could benefit system administrators and ease deployment of complex applications. However, these topics do not appear to be the major roadblocks in reaching exascale computing.

4.8 Scientific Computing

Exascale computing will have a major impact on many computational sciences, although existing applications likely need to be adapted to exploit this immense computational power. For instance, Oak Ridge National Laboratory selected 13 applications to be prepared for exascale computing by adapting them for the Summit supercomputer [6]. These codes are from diverse fields including climate science, quantum chemistry, biophysics, seismology, and plasma physics. In this section, we discuss the impact of exascale computing for *scientific applications* and *algorithmic methods*.

4.8.1 Scientific Applications. Bhatele et al. [21] analyze the architectural constraints to reach one exaflop for three scientific applications: molecular dynamics, cosmological simulation, and finite element solvers. They conclude that network performance (low latency and high bandwidth) is crucial and emphasize research into communication-minimizing algorithms. Reed and Hadka [152] demonstrate that large-scale *multi-objective evolutionary algorithms* (MOEA) can be used in water management and predict performance at exascale. Páll et al. [139] discuss the challenges in moving *GROMACS*, a package for bio-molecular simulations, to exascale and they discuss issues such as parallelization schemes, algorithms, and profiling, but also lessons learned from open-source software development. *Alya* [177] is a multi-physics simulation code aimed at exascale computing and performance results for up to 100.000 processors are presented. *Atlas* [47] is a library for large-scale weather/climate modeling with exascale ambitions. *OpenMC* [62] is a software package for Monte Carlo particle transport simulations that should scale to exaflops. Lawrence et al. [107] argue that new libraries and tools are needed for weather/climate models to make effective use of future exascale systems. The Square Kilometer Array (SKA) [29] is a next-generation radio telescope consisting of thousands of antennas spread out over South Africa and Western

Australia. Broekema et al. [29] estimate that real-time processing of the sensor data requires several exaflops.

4.8.2 Scalable Algorithms. There is much research into how parallel algorithms would perform at exascale and how to improve their scalability. Czechowski et al. [43] discuss how 3D fast Fourier transforms (FFTs) would perform at exascale. They conclude that all-to-all *inter*-node communication is the major bottleneck, but *intra*-node communication also plays a critical role due to limited main memory bandwidth. Gahvari and Gropp [73] apply performance models to study two exascale use-cases (FFT and multigrids) and conclude that scalability of these applications is limited by interconnect bandwidth. Hasanov et al. [88] present a hierarchical approach to distributed matrix multiplication. Evaluation on existing systems (Grid500 and BlueGene/P) and prediction for future exascale systems show better performance than the state-of-the-art. Gholami et al. [78] compare the performance of different large-scale Poisson solvers (FFT, FMM, GMG, and AMG). Evaluation up to ~220,000 cores show that the methods scale well and differences are application-dependent. Ghysels et al. [80] analyze scalability of the *general minimal residual method* (GMRES) and find that global all-to-all communication is the main bottleneck. Their pipelined GMRES variant hides this communication cost and they predict speedups up to 3.5×, although empirical results are absent. Dongarra et al. [55] describe a hierarchical QR factorization algorithm that scales better than existing QR factorization implementations. Dun et al. [62] prepare OpenMC for exascale computing by incorporating global view arrays.

4.8.3 Discussion. There are many computational domains that could benefit from exascale systems. Studies on the scalability of current applications make clear that data movement (both intra- and inter-node), not computation, will be the limiting factor. Developments in network and memory technology would be beneficial.

5 LIMITATIONS

In this section, we discuss the threats to the validity of our work. Every literature study has limitations and this one is no exception.

We utilized Scopus for our literature search since it covers most publishers common in HPC. However, despite Scopus' extensive database, it does not index non-peer-reviewed contributions such as preprints (e.g., arXiv.org, personal homepages) or technical reports (e.g., osti.gov, science.gov, cds.cern.ch). Including these contributions could be an interesting extension to our work, but quality control would be essential since not all documents have been reviewed. Several landmark studies (Section 2) were deliberately chosen from non-peer-reviewed sources as compensation.

For our search query, we assume that work is related if it mentions the literal term `exascale` or `extreme-scale` in the title, abstract, or author-specified keywords. However, a small fraction of relevant publications might be excluded due to two reasons. The first reason is that authors might mention exascale computing in the full text, but refer to it in the title/abstract using alternative descriptions (e.g., "next-generation supercomputer", "future platforms", "post-petascale systems", "quintillion FLOPS"). We experimented with various search queries, but were unable to identify additional search terms that would extend the results while not also increasing our scope. The second reason is that some relevant publications might not mention exascale computing at all since the authors were (at the time) unaware of the relevance. However, we argue that, even when the original publication is not included in our analysis, later work that expands upon those ideas will be included. Overall, we believe our search query to be concise and precise.

For topic modeling, we selected the method based on *non-negative matrix factorization* (NMF) [161, 183]. Besides NMF, we also experimented with *Latent Direchlet Allocation* (LDA) [24] (poor semantic quality and sensitive to input parameters), *Latent Semantic Analysis* (LSA) [61] (decent quality but negative weights made interpretation difficult), and hierarchical clustering [153] (does not capture articles touching upon multiple topics).

For the qualitative literature analysis, we manually selected publications from the search results related to each research theme. This selection was made based on the title/abstract and topic distribution while also taking into account year of publication (favoring newer work) and number of citations (favoring highly cited work). This selection method is subjective and requires interpretation of the work. However, we argue that this combination of data-driven analysis and qualitative analysis provides a balance between objective and subjective views on exascale research.

6 CONCLUSION

In this work, we have explored and mapped the landscape of exascale research. Exascale computing is a broad topic that touches upon many different aspects of HPC and we captured the important trends using a three-stage approach. First, we identified the challenges and opportunities of exascale based on various landmark studies. Second, we used data-driven techniques to analyze the large collection of related academic literature. Finally, we identified eight research themes and discussed the trends and progress within each area of research. While the three-stage methodology itself was not the research focus of this manuscript, we believe our data-driven approach to be a promising tool for literature reviews.

Overall, great progress has been made in tackling two of the major exascale barriers: energy efficiency and fault tolerance. Energy efficiency has improved dramatically with an increase of ~35× over the past decade, meaning we are nearing the point where an exascale system would be feasible in terms of energy consumption (i.e., 20MW for one exaflop). Fault tolerance is another topic that has seen much attention with major developments in checkpoint-restart protocols, data corruption detection, and fault understanding.

As we move forward, we foresee that these two barriers will slowly be overshadowed by the other two challenges: *software complexity* and *data volume*. For the software complexity challenge, we see a lack of suitable programming models that simplify the development of scalable scientific applications. Programming is still often done using low-level solutions, such as Fortran and MPI. However, exascale applications must handle more and more complex issues such as power management, fine-grained fault tolerance, *in-situ* processing, topology-aware routing/scheduling, and heterogeneity. For the data volume challenge, we observe a growing gap between computing power of processors and data bandwidth (i.e., memory, storage, network). While we expect that computation (i.e., floating-point operations per second) will become cheaper over time, data movement (i.e., bytes per second) might not grow at the same pace and become relatively more expensive. Novel hardware technology (i.e., accelerators, memories, networks) are promising solutions to the data movement challenge, but they conflict with the software complexity challenge since they introduce additional complication when programming these systems.

Thanks to the substantial body of exascale research that has already been done, from where we are today, we estimate an evolutionary approach should be sufficient to eventually reach exaflop performance, but the major challenge for subsequent generations will be how we are actually going to use this massive computational power effectively and efficiently to advance scientific research.

APPENDICES

A PREPROCESSING OF DOCUMENTS

As is common in natural language processing [10], topic modeling requires preprocessing of the documents. We have used a standard preprocessing pipeline that we will demonstrate using Cappello et al. [32] as an example. First, the title and abstract are extracted and concatenated.

```
Toward Exascale Resilience Over the past few years resilience has became a
major issue for high-performance computing (HPC) systems, in particular in the
perspective of large petascale systems and future exascale systems.
```

Second, punctuation (i.e., dashes, commas, brackets) is replaced by whitespace and the remaining characters are converted to lowercase.

```
toward exascale resilience over the past few years resilience has became a major
issue for high performance computing hpc systems in particular in the perspective
of large petascale systems and future exascale systems
```

Third, neutral words that do not contribute to the content are removed (e.g., "*this*", "*an*", "*year*", "*author*", "*propose*"). This list of stopwords was manually compiled by the authors.

```
toward exascale resilience resilience major issue performance computing systems
particular perspective petascale systems future exascale systems
```

Words that are rare (occur in less than 5 documents) are removed. This removes names and other unique words.

```
exascale resilience resilience issue performance computing systems petascale
systems future exascale systems
```

Words that occur frequently (more than 75% of the documents) are removed. For our dataset, frequent words are the following: *exascale*, *computing*, *application*, *performance*, and *system*. They can be omitted since they contribute little to differentiating the documents.

```
resilience resilience major issue particular perspective petascale future
```

Words are stemmed using Porter's algorithm [146] (e.g., the algorithms maps "*communicating*", "*communicate*" and "*communication*" to their stem "*commun*").

```
resili resili major issu particular perspect petascal futur
```

The document is converted into a term frequency vector.

```
futur: 1, issu: 1, major: 1, particular: 1, perspect: 1, petascal: 1, resili: 2
```

As is common in natural language processing [10], the terms are weighed using TF-IDF [155] to reflect how important each word is to the entire collection. This final vector is the input to the topic modeling algorithm.

```
futur: 0.69, issu: 1.8, major: 2, particular: 1.1, perspect: 1, petascal: 3.2,
resili: 7.2
```

For results presented within this manuscript, words are manually "un-stemmed" for improved readability (e.g., "*resili*" is mapped back to "*resilience*").

B EXTENDED RESULTS OF LITERATURE ANALYSIS

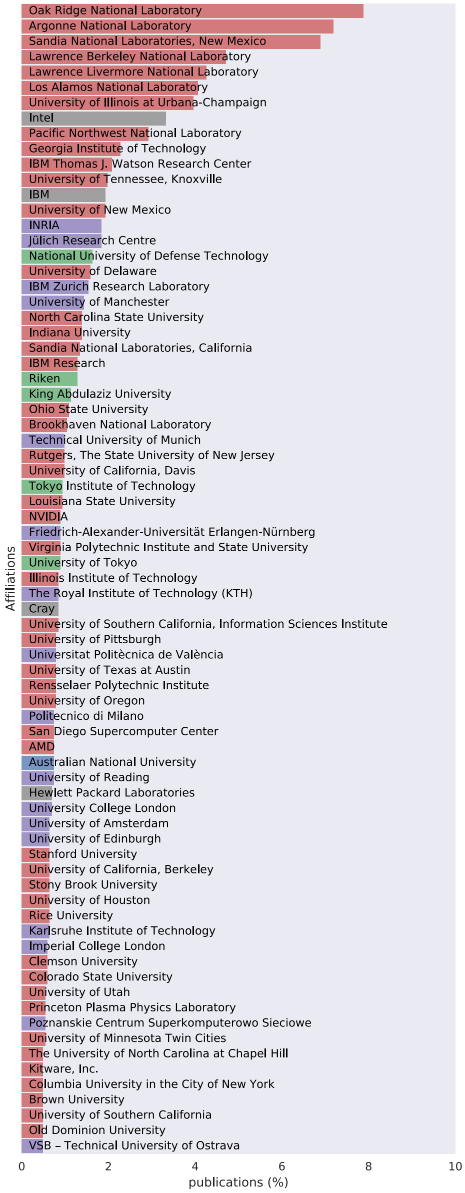


Fig. B.1. Number of publications based on author affiliations (top 75). Colors indicate continent as in Figure 5.

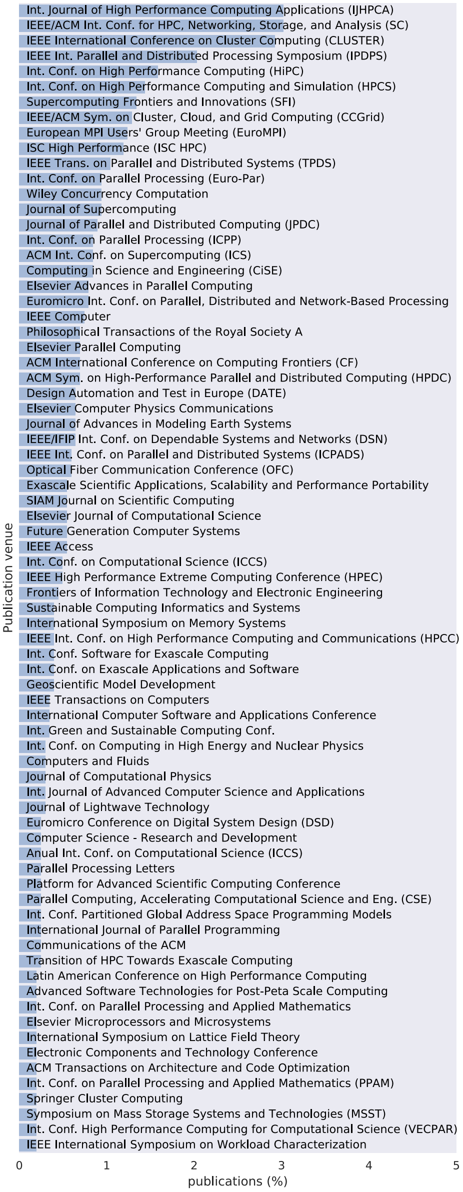


Fig. B.2. Number of publications per publication venue (top 75).

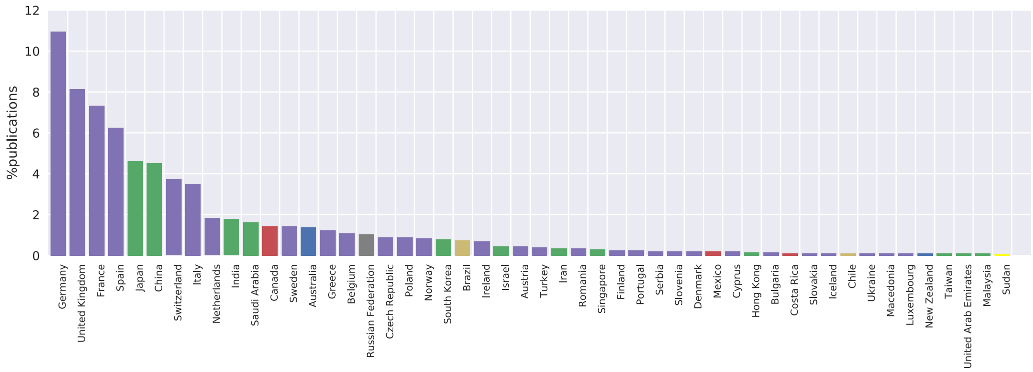


Fig. B.3. Number of publications per country (top 50), excluding United States (involved in 62% of all publications).

Table B.1. Top 10 Publications in the Citation Network (Figure 8)

Title	Authors	Year	Citations (within dataset)
The International Exascale Software Project Roadmap [54]	Dongarra et al.	2011	128
Exascale Computing Technology Challenges [162]	Shalf et al.	2010	80
Toward Exascale Resilience [32]	Cappello et al.	2009	72
Evaluating the Viability of Process Replication Reliability for Exascale Systems [68]	Ferreira et al.	2011	59
Fault Tolerance in Petascale/Exascale Systems: Current Knowledge, Challenges and Research Opportunities [31]	Cappello et al.	2009	49
Addressing Failures in Exascale Computing [166]	Snir et al.	2014	46
Toward Exascale Resilience: 2014 Update [33]	Cappello et al.	2014	43
Detection and Correction of Silent Data Corruption for Large-Scale High-Performance Computing [70]	Fiala et al.	2011	35
Exascale Computing and Big Data [151]	Reed et al.	2015	32
Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing [141]	Patki et al.	2013	26

The number of citations is measured as the number of publications within our dataset that reference the given publication. In other words, it is the number of *incoming* links in Figure 8.

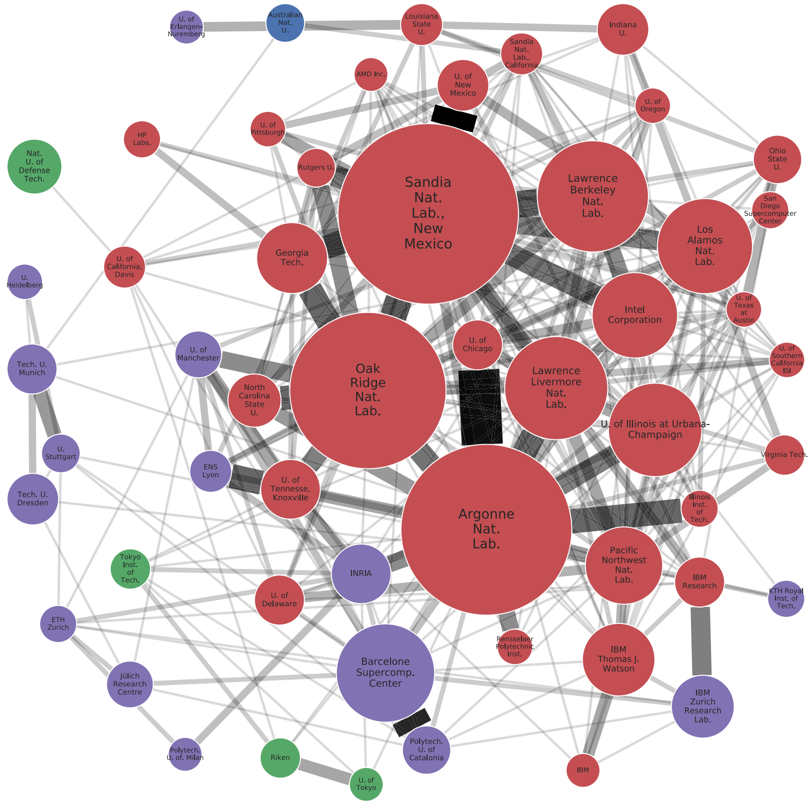
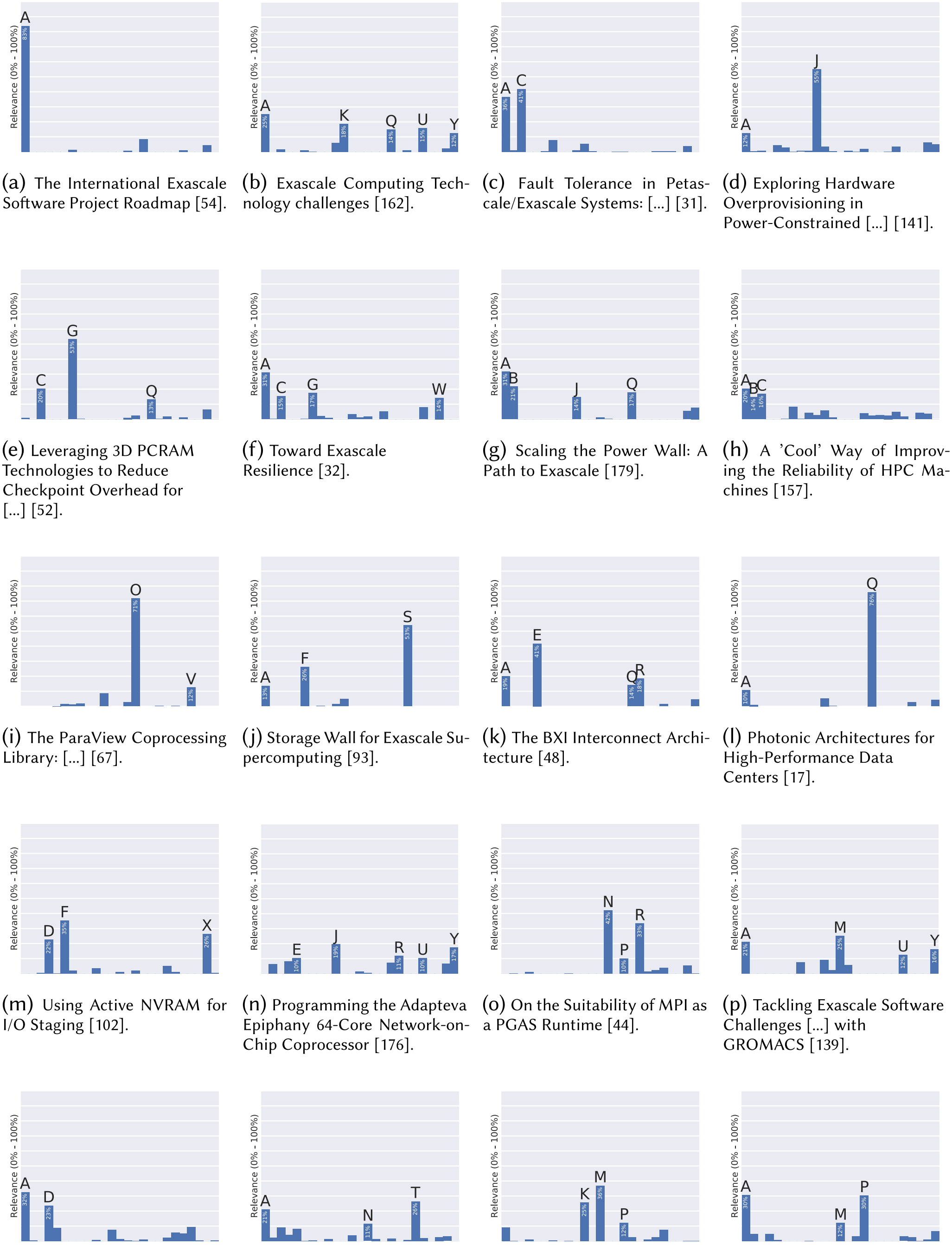


Fig. B.4. Visualization of the collaborations network (top 50). Nodes represent institutes with the number of shared publications indicated by the node size. Edges represent collaborations with the number of shared publications indicated by the edge width. Colors indicate continent as in Figure 5.

Table B.2. Dominant Words (Top 20) per Detected Topic from Figure 9

#	Dominant words
A	software, development, project, challenges, hardware, science, research, codesign, technology, community, design, architectures, supercomputers, future, years, infrastructure, scientific, tools, components, advanced,
B	energy, consumption, efficiency, savings, frequency, tuning, datacenter, techniques, optimization, runtime, consume, improve, aware, dvfs, dynamic, voltage, measurements, metrics, evaluation, tools,
C	failure, logging, recovery, fault-tolerance, resilience, message, overhead, techniques, protocol, event, nodes, prediction, execution, processing, tolerant, check, reliability, fail, mean, scale,
D	data, bigdata, compression, processing, analysis, scientific, science, distributed, analytics, volume, intensive, stream, query, centric, movement, analyze, techniques, cloud, staging, store,
E	network, topology, interconnect, routing, traffic, dragonfly, torus, nodes, links, latency, bandwidth, workloads, noc, router, design, packet, simulation, switch, radix, diameter,
F	io, nodes, collective, patterns, filesystem, burst, coordinated, optimization, access, tuning, buffer, parallel, staging, requests, adios, write, improve, layer, file, variability,
G	checkpoint, restart, overhead, compression, fault-tolerance, reduce, level, scheme, intervals, optimization, incremental, state, coordinated, mechanism, multilevel, uncoordinated, future, increasing, protocol, filesystem,
H	gpu, cpu, heterogeneous, accelerators, hybrid, nvidia, processingunit, kernels, graphics, cuda, code, supercomputers, openacc, devices, implementation, evaluation, nodes, important, research, fmm,
I	errors, detection, corruption, detectors, soft, sdc, correct, silent, resilience, recovery, bit, rates, reliability, latent, protection, dram, injection, false, overhead, flipping,
J	power, consumption, budget, capping, limited, job, supercomputers, efficiency, constraints, constrained, optimization, bound, management, overprovisioning, level, control, achieving, monitoring, study, improve,
K	algorithms, parallel, solver, methods, problem, matrix, scalability, linear, scale, implementation, sparse, multigrid, equations, grid, hierarchical, algebra, iterative, numerical, solve, efficiency,
L	scheduling, task, job, resources, allocation, management, distributed, heuristics, execution, utilization, dynamic, heterogeneous, workloads, aware, shared, many, optimization, dispatching, algorithms, stealing,
M	simulation, scale, code, physics, numerical, turbulence, mesh, flow, fidelity, resolution, framework, multiscale, plasma, particle, running, multi, cores, fusion, coupled, generation,
N	mpi, message, threads, processing, openmp, implementation, hybrid, interface, passing, cores, scalability, standard, manycore, rank, matching, millions, ulfm, benchmarks, comm, parallel,
O	visualization, in situ, analysis, post, processing, data, simulation, interactions, image, disk, extract, savings, rendering, explore, tools, damaris, scientists, raw, parallel, traditional,
P	model, prediction, climate, earth, analytics, parameters, atmospheric, resolution, cloud, weather, variability, accurate, different, e3sm, version, water, parameterization, ocean, author, execution,
Q	optical, photonic, interconnect, technology, chip, silicon, switch, integrated, bandwidth, low, circuit, devices, datacenter, density, electrical, enable, cmos, board, 3d, requirements,
R	communication, global, message, pgas, onesided, overlap, protocol, collective, trace, extrapolation, improve, operations, scheme, latency, distributed, characteristics, primitives, interface, potential, replay,
S	storage, metadata, filesystem, server, object, devices, tier, data, scalability, reliability, raid, file, write, management, disk, hierarchy, prototype, design, centric, stack,
T	kernels, os, linux, virtual, lwk, lightweight, operations, monitoring, nodes, cluster, environment, running, noise, runtime, enclave, provides, multi, machines, execution, host,
U	programming, parallel, runtime, code, language, model, task, programmers, grain, execution, abstraction, level, pgas, support, compiler, portable, fine, openmp, threads, development,
V	workflows, scientific, management, proven, tools, analysis, complex, metadata, analytics, science, end, services, execution, discovery, experiments, scale, environment, monitoring, online, optimization,
W	fault, resilience, tolerant, injection, fault-tolerance, redundancy, transient, presence, detection, techniques, tools, solution, propagation, grid, overhead, components, monitoring, blocks, expected, extensions,
X	memory, cache, bandwidth, hierarchy, access, dram, locality, shared, cores, architectures, capacity, improve, reuse, coherence, nvram, contention, processors, study, hardware, chip,
Y	architectures, heterogeneous, accelerators, intel, cluster, reconfigurable, xeon, manycore, platforms, processors, phi, deep, fpga, mic, cores, nodes, hardware, booster, project, efficiency,



(a) The International Exascale Software Project Roadmap [54].

(b) Exascale Computing Technology challenges [162].

(c) Fault Tolerance in Petascale/Exascale Systems: [...] [31].

(d) Exploring Hardware Overprovisioning in Power-Constrained [...] [141].

(e) Leveraging 3D PCRAM Technologies to Reduce Checkpoint Overhead for [...] [52].

(f) Toward Exascale Resilience [32].

(g) Scaling the Power Wall: A Path to Exascale [179].

(h) A 'Cool' Way of Improving the Reliability of HPC Machines [157].

(i) The ParaView Coprocessing Library: [...] [67].

(j) Storage Wall for Exascale Supercomputing [93].

(k) The BXI Interconnect Architecture [48].

(l) Photonic Architectures for High-Performance Data Centers [17].

(m) Using Active NVRAM for I/O Staging [102].

(n) Programming the Adapteva Epiphany 64-Core Network-on-Chip Coprocessor [176].

(o) On the Suitability of MPI as a PGAS Runtime [44].

(p) Tackling Exascale Software Challenges [...] with GROMACS [139].

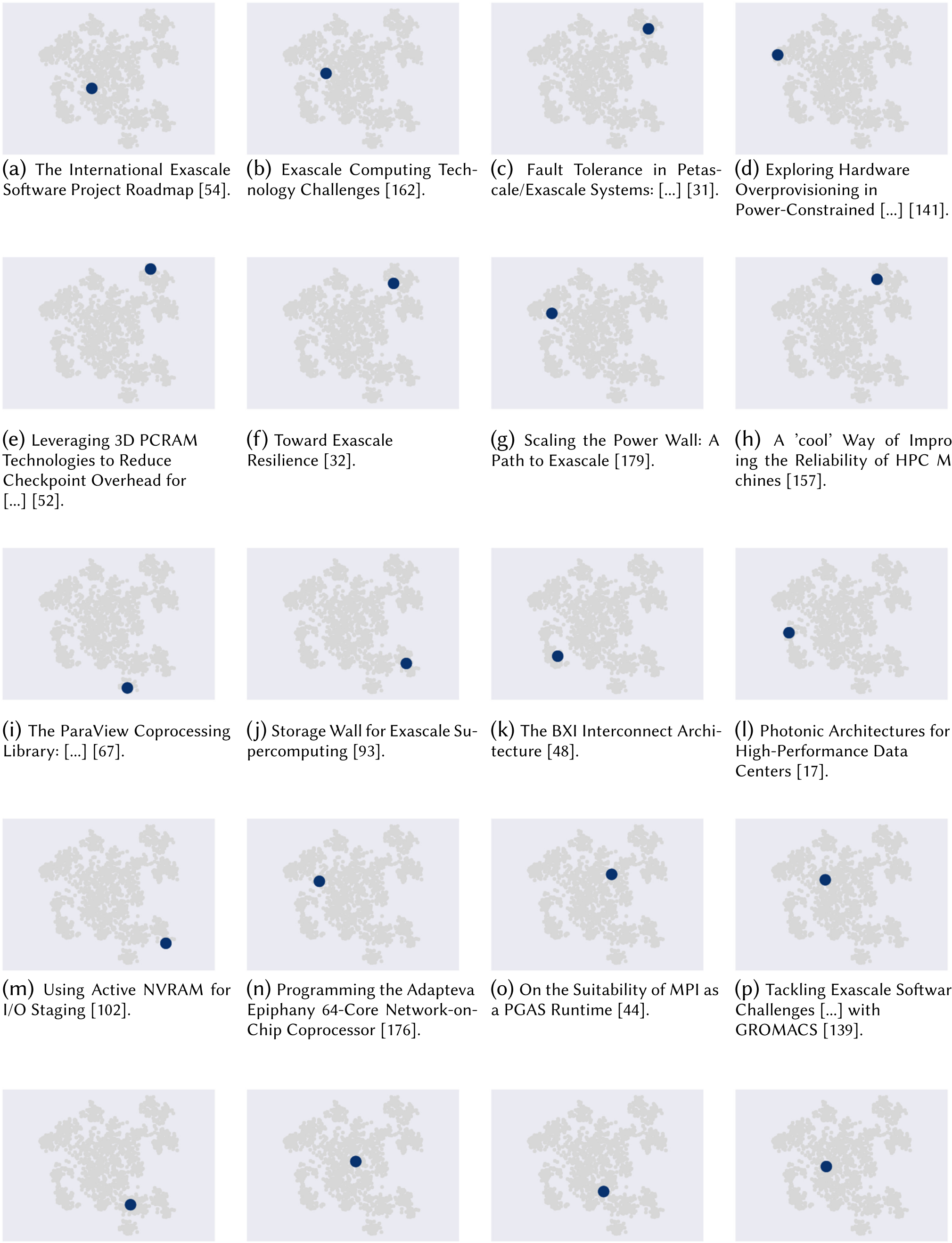
(q) Evolution of the ATLAS PanDA Workload Management System [...] [118].

(r) Application Migration in HPC - A Driver of the Exascale Era? [145].

(s) Alya: Multiphysics Engineering Simulation toward Exascale [177].

(t) ExaSAT: An Exascale Co-Design Tool for Performance Modeling [173].

Fig. B.5. Example of topic distribution for 20 randomly selected publications.



(a) The International Exascale Software Project Roadmap [54].

(b) Exascale Computing Technology Challenges [162].

(c) Fault Tolerance in Petascale/Exascale Systems: [...] [31].

(d) Exploring Hardware Overprovisioning in Power-Constrained [...] [141].

(e) Leveraging 3D PCRAM Technologies to Reduce Checkpoint Overhead for [...] [52].

(f) Toward Exascale Resilience [32].

(g) Scaling the Power Wall: A Path to Exascale [179].

(h) A ’cool’ Way of Improving the Reliability of HPC Machines [157].

(i) The ParaView Coprocessing Library: [...] [67].

(j) Storage Wall for Exascale Supercomputing [93].

(k) The BXI Interconnect Architecture [48].

(l) Photonic Architectures for High-Performance Data Centers [17].

(m) Using Active NVRAM for I/O Staging [102].

(n) Programming the Adaptева Epiphany 64-Core Network-on-Chip Coprocessor [176].

(o) On the Suitability of MPI as a PGAS Runtime [44].

(p) Tackling Exascale Software Challenges [...] with GROMACS [139].

(q) Evolution of the ATLAS PanDA Workload Management System [...] [118].

(r) Application Migration in HPC - A Driver of the Exascale Era? [145].

(s) Alya: Multiphysics Engineering Simulation toward Exascale [177].

(t) ExaSAT: An Exascale Co-Design Tool for Performance Modeling [173].

Fig. B.6. Location in embedding for the 20 publications from Figure B.5.

ACKNOWLEDGMENTS

The authors would like to thank W. A. Günther for many helpful discussions on natural language processing, R. S. Cushing for feedback during early stages of this work, and the anonymous reviewers for their time and valuable feedback.

REFERENCES

[1] 2018. China Reveals Third Exascale Prototype | TOP500 Supercomputer Sites. https://www.top500.org/news/china-reveals-third-exascale-prototype/.

[2] 2018. Frontier: OLCF's Exascale Future. https://www.olcf.ornl.gov/2018/02/13/frontier-olcfs-exascale-future/.

[3] 2018. Scopus - The Largest Database of Peer-Reviewed Literature. https://www.elsevier.com/solutions/scopus.

[4] 2018. TOP500 Supercomputer Sites. https://www.top500.org/. Accessed July 2018.

[5] 2019. BDEC: Big Data and Extreme-Scale Computing. https://www.exascale.org/bdec/.

[6] 2019. CAAR: Center for Accelerated Application Readiness. https://www.olcf.ornl.gov/caar/.

[7] 2019. EuroHPC: Europe's Journey to Exascale HPC. http://eurohpc.eu/.

[8] 2019. IESP: International Exascale Software Project. https://www.exascale.org/iesp.

[9] 2019. U.S. Department of Energy and Intel to Deliver First Exascale Supercomputer, Argonne National Laboratory. https://www.anl.gov/article/us-department-of-energy-and-intel-to-deliver-first-exascale-supercomputer.

[10] C. C. Aggarwal and C. X. Zhai. 2012. *Mining Text Data*. Springer Publishing Company, Inc.

[11] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber. 2009. HyperX: Topology, routing, and packaging of efficient large-scale networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (SC'09)*. ACM, New York, 41:1–41:11. DOI: https://doi.org/10.1145/1654059.1654101

[12] Y. Ajima, S. Sumimoto, and T. Shimizu. 2009. Tofu: A 6D mesh/torus interconnect for exascale computers. *Computer* 42, 11 (Nov. 2009), 36–40. DOI: https://doi.org/10.1109/MC.2009.370

[13] M. Asch et al. 2018. Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry. *The International Journal of High Performance Computing Applications* 32, 4 (July 2018), 435–479. DOI: https://doi.org/10.1177/1094342018778123

[14] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, et al. 2010. *The Opportunities and Challenges of Exascale Computing*. Technical Report. U.S. Department of Energy, Office of Science. Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee.

[15] J. A. Åström, A. Carter, J. Hetherington, K. Ioakimidis, E. Lindahl, G. Mozdzynski, R. W. Nash, P. Schlatter, A. Signell, and J. Westerholm. 2013. Preparing scientific application software for exascale computing. In *Applied Parallel and Scientific Computing*. Vol. 7782. Springer Berlin, Berlin, Germany, 27–42. DOI: https://doi.org/10.1007/978-3-642-36803-5_2

[16] J. Bachan, D. Bonachea, P. H. Hargrove, S. Hofmeyr, M. Jacquelin, A. Kamil, B. Van Straalen, and S. B. Baden. 2017. The UPC++ PGAS library for exascale computing. In *Proceedings of PAW 2017: 2nd Annual PGAS Applications Workshop - Held in Conjunction with SC 2017: The International Conference for High Performance Computing, Networking, Storage and Analysis*, Vol. 2017-January. 1–4. DOI: https://doi.org/10.1145/3144779.3169108

[17] R. G. Beausoleil, M. McLaren, and N. P. Jouppi. 2013. Photonic architectures for high-performance data centers. *IEEE Journal of Selected Topics in Quantum Electronics* 19, 2 (March 2013), 3700109–3700109. DOI: https://doi.org/10.1109/JSTQE.2012.2236080

[18] J. Bent, S. Faibish, J. Ahrens, G. Grider, J. Patchett, P. Tzelnic, and J. Woodring. 2012. Jitter-free co-processing on a prototype exascale storage stack. In *Proceedings of the 2012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*. 1–5. DOI: https://doi.org/10.1109/MSST.2012.6232382

[19] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. 2007. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis* 52, 1 (Sept. 2007), 155–173. DOI: https://doi.org/10.1016/j.csda.2006.11.006

[20] A. Bhatele, W. D. Gropp, N. Jain, and L. V. Kale. 2011. Avoiding hot-spots on two-level direct networks. In *SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–11. DOI: https://doi.org/10.1145/2063384.2063486

[21] A. Bhatele, P. Jetley, H. Gahvari, L. Wesolowski, W. D. Gropp, and L. Kalé. 2011. Architectural constraints to attain 1 exaflop/s for three scientific application classes. In *Proceedings of the 2011 IEEE International Parallel Distributed Processing Symposium*. 80–91. DOI: https://doi.org/10.1109/IPDPS.2011.18

[22] W. Bland. 2013. User level failure mitigation in MPI. In *Euro-Par 2012: Parallel Processing Workshops*. Vol. 7640. Springer Berlin, Berlin, Germany, 499–504. DOI: https://doi.org/10.1007/978-3-642-36949-0_57

[23] W. Bland, P. Du, A. Bouteiller, T. Herault, G. Bosilca, and J. Dongarra. 2012. A checkpoint-on-failure protocol for algorithm-based recovery in standard MPI. In *Euro-Par 2012 Parallel Processing (Lecture Notes in Computer Science).* Springer Berlin, 477–488.

[24] D. M. Blei, A. Y. Ng, and M. I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.

[25] D. Bodas, J. Song, M. Rajappa, and A. Hoffman. 2014. Simple power-aware scheduler to limit power consumption by HPC system within a budget. In *Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing (E2SC'14).* IEEE Press, Piscataway, NJ, 21–30. DOI: https://doi.org/10.1109/E2SC.2014.8

[26] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini. 2016. Predictive modeling for job power consumption in HPC systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* vol. 9697 (2016), 181–199. DOI: https://doi.org/10.1007/978-3-319-41321-1_10

[27] G. Bosilca, T. Herault, A. Rezmerita, and J. Dongarra. 2011. On scalability for MPI runtime systems. In *Proceedings of the 2011 IEEE International Conference on Cluster Computing.* 187–195. DOI: https://doi.org/10.1109/CLUSTER.2011.29

[28] K. W. Boyack, D. Newman, R. J. Duhon, R. Klavans, M. Patek, J. R. Biberstine, B. Schijvenaars, A. Skupin, N. Ma, and K. Börner. 2011. Clustering more than two million biomedical publications: Comparing the accuracies of nine text-based similarity approaches. *PLoS one* 6, 3 (2011). DOI: https://doi.org/10.1371/journal.pone.0018029

[29] P. C. Broekema, R. V. van Nieuwpoort, and H. E. Bal. 2012. ExaScale high performance computing in the square kilometer array. In *Proceedings of the 2012 Workshop on High-Performance Computing for Astronomy Date - Astro-HPC'12.* ACM, Delft, The Netherlands, 9. DOI: https://doi.org/10.1145/2286976.2286982

[30] A. Cabrera, F. Almeida, J. Arteaga, and V. Blanco. 2015. Measuring energy consumption using EML (energy measurement library). *Comput. Sci. Res. Dev.* 30, 2 (May 2015), 135–143. DOI: https://doi.org/10.1007/s00450-014-0269-5

[31] F. Cappello. 2009. Fault tolerance in petascale/exascale systems: Current knowledge, challenges and research opportunities. *The International Journal of High Performance Computing Applications* 23, 3 (Aug. 2009), 212–226. DOI: https://doi.org/10.1177/1094342009106189

[32] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. 2009. Toward exascale resilience. *The International Journal of High Performance Computing Applications* 23, 4 (Nov. 2009), 374–388. DOI: https://doi.org/10.1177/1094342009347767

[33] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir. 2014. Toward exascale resilience: 2014 update. *Supercomputing Frontiers and Innovations* 1, 1 (June 2014), 5–28–28. DOI: https://doi.org/10.14529/jsfi140101

[34] H. Carter Edwards, C. R. Trott, and D. Sunderland. 2014. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *J. Parallel and Distrib. Comput.* 74, 12 (Dec. 2014), 3202–3216. DOI: https://doi.org/10.1016/j.jpdc.2014.07.003

[35] C. Chan, D. Unat, M. Lijewski, W. Zhang, J. Bell, and J. Shalf. 2013. Software design space exploration for exascale combustion co-design. In *Supercomputing (Lecture Notes in Computer Science).* Springer Berlin, 196–212.

[36] R. R. Chandrasekar, A. Venkatesh, K. Hamidouche, and D. K. Panda. 2015. Power-check: An energy-efficient checkpointing framework for HPC clusters. In *Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* 261–270. DOI: https://doi.org/10.1109/CCGrid.2015.169

[37] A. Chien, P. Balaji, P. Beckman, N. Dun, A. Fang, H. Fujita, K. Iskra, Z. Rubenstein, Z. Zheng, R. Schreiber, J. Hammond, J. Dinan, I. Laguna, D. Richards, A. Dubey, B. van Straalen, M. Hoemmen, M. Heroux, K. Teranishi, and A. Siegel. 2015. Versioned distributed arrays for resilience in scientific applications: Global view resilience. *Procedia Computer Science* 51 (Jan. 2015), 29–38. DOI: https://doi.org/10.1016/j.procs.2015.05.187

[38] J. Choo, C. Lee, C. K. Reddy, and H. Park. 2013. UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (Dec. 2013), 1992–2001. DOI: https://doi.org/10.1109/TVCG.2013.212

[39] J. Chung, I. Lee, M. Sullivan, J. H. Ryoo, D. W. Kim, D. H. Yoon, L. Kaplan, and M. Erez. 2013. Containment domains: A scalable, efficient and flexible resilience scheme for exascale systems. *Scientific Programming.* https://www.hindawi.com/journals/sp/2013/473915/abs/. DOI: https://doi.org/10.3233/SPR-130374

[40] G. Congiu, S. Narasimhamurthy, T. Süß, and A. Brinkmann. 2016. Improving collective I/O performance using non-volatile memory devices. In *Proceedings of the IEEE International Conference on Cluster Computing, (ICCC).* 120–129. DOI: https://doi.org/10.1109/CLUSTER.2016.37

[41] G. Da Costa, T. Fahringer, J. A. R. Gallego, I. Grasso, A. Hristov, H. D. Karatza, A. Lastovetsky, F. Marozzo, D. Petcu, G. L. Stavrinides, D. Talia, P. Trunfio, and H. Astsatryan. 2015. Exascale machines require new programming paradigms and runtimes. *Supercomputing Frontiers and Innovations* 2, 2 (Aug. 2015), 6–27. DOI: https://doi.org/10.14529/jsfi150201

[42] P. Czarnul, J. Proficz, and A. Krzywaniak. 2019. Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments. *Scientific Programming* (2019). DOI: https://doi.org/10.1155/2019/8348791

[43] K. Czechowski, C. Battaglino, C. McClanahan, K. Iyer, P. Yeung, and R. Vuduc. 2012. On the communication complexity of 3D FFTs and its implications for exascale. In *Proceedings of the 26th ACM International Conference on Supercomputing (ICS'12)*. ACM Press, San Servolo Island, Venice, Italy, 205. DOI: https://doi.org/10.1145/2304576.2304604

[44] J. Daily, A. Vishnu, B. Palmer, H. van Dam, et al. 2014. On the suitability of MPI as a PGAS runtime. In *2014 21st International Conference on High Performance Computing (HiPC)*. 1–10. DOI: https://doi.org/10.1109/HiPC.2014.7116712

[45] A. Danalis, G. Bosilca, A. Bouteiller, T. Herault, and J. Dongarra. 2014. PTG: An abstraction for unhindered parallelism. In *Proceedings of WOLFHPC 2014: 4th International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing - Held in Conjunction with SC 2014: The International Conference for High Performance Computing, Networking, Storage and Analysis*. 21–30. DOI: https://doi.org/10.1109/WOLFHPC.2014.8

[46] D. Dauwe, R. Jhaveri, S. Pasricha, A. A. Maciejewski, and H. J. Siegel. 2018. Optimizing checkpoint intervals for reduced energy use in exascale systems. In *Proceedings of the 2017 8th International Green and Sustainable Computing Conference, (IGSC'17)*. 1–8. DOI: https://doi.org/10.1109/IGCC.2017.8323598

[47] W. Deconinck, P. Bauer, M. Diamantakis, M. Hamrud, C. Kühnlein, P. Maciel, G. Mengaldo, T. Quintino, B. Raoult, P. K. Smolarkiewicz, and N. P. Wedi. 2017. Atlas: A library for numerical weather prediction and climate modelling. *Computer Physics Communications* 220 (Nov. 2017), 188–204. DOI: https://doi.org/10.1016/j.cpc.2017.07.006

[48] S. Derradji, T. Palfer-Sollier, J. Panziera, A. Poudes, and F. W. Atos. 2015. The BXI interconnect architecture. In *Proceedings of the 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects*. 18–25. DOI: https://doi.org/10.1109/HOTI.2015.15

[49] S. Di, M. S. Bouguerra, L. Bautista-Gomez, and F. Cappello. 2014. Optimization of multi-level checkpoint model for large scale HPC applications. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*. 1181–1190. DOI: https://doi.org/10.1109/IPDPS.2014.122

[50] S. Di and F. Cappello. 2016. Adaptive impact-driven detection of silent data corruption for HPC applications. *IEEE Transactions on Parallel and Distributed Systems* 27, 10 (2016), 2809–2823. DOI: https://doi.org/10.1109/TPDS.2016.2517639

[51] J. Diaz, C. Muñoz-Caro, and A. Niño. 2012. A survey of parallel programming models and tools in the multi and many-core era. *IEEE Transactions on Parallel and Distributed Systems* 23, 8 (Aug. 2012), 1369–1386. DOI: https://doi.org/10.1109/TPDS.2011.308

[52] X. Dong, N. Muralimanohar, N. Jouppi, R. Kaufmann, and Y. Xie. 2009. Leveraging 3D PCRAM technologies to reduce checkpoint overhead for future exascale systems. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. 1–12. DOI: https://doi.org/10.1145/1654059.1654117

[53] X. Dong, Y. Xie, N. Muralimanohar, and N. P. Jouppi. 2011. Hybrid checkpointing using emerging nonvolatile memories for future exascale systems. *ACM Transactions on Architecture and Code Optimization* 8, 2 (July 2011), 1–29. DOI: https://doi.org/10.1145/1970386.1970387

[54] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J. Andre, D. Barkai, J. Berthou, T. Boku, B. Braunschweig, et al. 2011. The international exascale software project roadmap. *International Journal of High Performance Computing Applications* 25, 1 (2011), 3–60.

[55] J. Dongarra, M. Faverge, T. Hérault, M. Jacquelin, J. Langou, and Y. Robert. 2013. Hierarchical QR factorization algorithms for multi-core clusters. *Parallel Comput.* 39, 4 (April 2013), 212–232. DOI: https://doi.org/10.1016/j.parco.2013.01.003

[56] J. J. Dongarra. 2014. *Performance of Various Computers Using Standard Linear Equations Software.* Technical CS-89-85. University of Manchester. 110 pages.

[57] M. Dorier, G. Antoniu, F. Cappello, M. Snir, R. Sisneros, O. Yildiz, S. Ibrahim, T. Peterka, and L. Orf. 2016. Damaris: Addressing performance variability in data management for post-petascale simulations. *ACM Transactions on Parallel Computing* 3, 3 (2016). DOI: https://doi.org/10.1145/2987371

[58] M. Dorier, M. Mubarak, R. Ross, J. K. Li, C. D. Carothers, and K. Ma. 2016. Evaluation of topology-aware broadcast algorithms for dragonfly networks. In *Proceedings of the 2016 IEEE International Conference on Cluster Computing (CLUSTER)*. 40–49. DOI: https://doi.org/10.1109/CLUSTER.2016.26

[59] M. Dorier, R. Sisneros, T. Peterka, G. Antoniu, and D. Semeraro. 2013. Damaris/viz: A nonintrusive, adaptable and user-friendly in situ visualization framework. In *Proceedings of the 2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*. 67–75. DOI: https://doi.org/10.1109/LDAV.2013.6675160

[60] S. S. Dosanjh, R. F. Barrett, D. W. Doerfler, S. D. Hammond, K. S. Hemmert, M. A. Heroux, P. T. Lin, K. T. Pedretti, A. F. Rodrigues, T. G. Trucano, and J. P. Luitjens. 2014. Exascale design space exploration and co-design. *Future Generation Computer Systems* 30 (Jan. 2014), 46–58. DOI: https://doi.org/10.1016/j.future.2013.04.018

[61] S. T. Dumais. 2004. Latent semantic analysis. *Annual Review of Information Science and Technology* 38, 1 (2004), 188–230. DOI: https://doi.org/10.1002/aris.1440380105

[62] N. Dun, H. Fujita, J. R. Tramm, A. A. Chien, and A. R. Siegel. 2015. Data decomposition in Monte Carlo neutron transport simulations using global view arrays. *International Journal of High Performance Computing Applications* 29, 3 (2015), 348–365. DOI: https://doi.org/10.1177/1094342015577681

[63] N. Eicker, T. Lippert, T. Moschny, and E. Suarez. 2013. The DEEP project - Pursuing cluster-computing in the many-core era. In *Proceedings of the 2013 42nd International Conference on Parallel Processing*. 885–892. DOI: https://doi.org/10.1109/ICPP.2013.105

[64] N. El-Sayed and B. Schroeder. 2013. Reading between the lines of failure logs: Understanding how HPC systems fail. In *Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 1–12. DOI: https://doi.org/10.1109/DSN.2013.6575356

[65] C. Engelmann. 2014. Scaling to a million cores and beyond: Using light-weight simulation to understand the challenges ahead on the road to exascale. *Future Generation Computer Systems* 30 (Jan. 2014), 59–65. DOI: https://doi.org/10.1016/j.future.2013.04.014

[66] "European Exascale Software Initiative". 2015. Final Report on EESI2 Exascale Vision, Roadmap, and Recommendations. http://www.eesi-project.eu/ressources/documentation/.

[67] N. Fabian, K. Moreland, D. Thompson, A. C. Bauer, et al. 2011. The ParaView coprocessing library: A scalable, general purpose in situ visualization library. In *Proceedings of the 2011 IEEE Symposium on Large Data Analysis and Visualization*. 89–96. DOI: https://doi.org/10.1109/LDAV.2011.6092322

[68] K. Ferreira, J. Stearley, J. H. Laros, R. Oldfield, et al. 2011. Evaluating the viability of process replication reliability for exascale systems. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*. ACM, Seattle, Washington, 1. DOI: https://doi.org/10.1145/2063384.2063443

[69] R. Ferreira da Silva, S. Callaghan, T. M. A. Do, G. Papadimitriou, and E. Deelman. 2019. Measuring the impact of burst buffers on data-intensive scientific workflows. *Future Generation Computer Systems* 101 (2019), 208–220. DOI: https://doi.org/10.1016/j.future.2019.06.016

[70] D. Fiala. 2011. Detection and correction of silent data corruption for large-scale high-performance computing. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*. 2069–2072. DOI: https://doi.org/10.1109/IPDPS.2011.379

[71] M. Flajslik, E. Borch, and M. A. Parker. 2018. Megafly: A topology for exascale systems. In *High Performance Computing (Lecture Notes in Computer Science)*. Springer International Publishing, 289–310.

[72] K. Fürlinger. 2015. Exploiting hierarchical exascale hardware using a PGAS approach. In *Proceedings of the 3rd International Conference on Exascale Applications and Software (EASC'15)*. University of Edinburgh, Edinburgh, Scotland, UK, 48–52.

[73] H. Gahvari and W. Gropp. 2010. An introductory exascale feasibility study for FFTs and multigrid. In *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*. 1–9. DOI: https://doi.org/10.1109/IPDPS.2010.5470417

[74] M. Gamell, D. S. Katz, H. Kolla, J. Chen, S. Klasky, and M. Parashar. 2014. Exploring automatic, online failure recovery for scientific applications at extreme scales. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 895–906. DOI: https://doi.org/10.1109/SC.2014.78

[75] A. Geist and R. Lucas. 2009. Major computer science challenges at exascale. *The International Journal of High Performance Computing Applications* 23, 4 (2009), 427–436.

[76] B. Gerofi, Y. Ishikawa, R. Riesen, R. W. Wisniewski, Y. Park, and B. Rosenburg. 2016. A multi-kernel survey for high-performance computing. In *Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers (ROSS'16)*. ACM, New York, 5:1–5:8. DOI: https://doi.org/10.1145/2931088.2931092

[77] B. Gerofi, M. Takagi, Y. Ishikawa, R. Riesen, E. Powers, and R. W. Wisniewski. 2015. Exploring the design space of combining Linux with lightweight kernels for extreme scale computing. In *Proceedings of the 5th International Workshop on Runtime and Operating Systems for Supercomputers - ROSS'15*. ACM, Portland, OR, 1–8. DOI: https://doi.org/10.1145/2768405.2768410

[78] A. Gholami, D. Malhotra, H. Sundar, and G. Biros. 2016. FFT, FMM, or multigrid? A comparative study of state-of-the-art poisson solvers for uniform and nonuniform grids in the unit cube. *SIAM Journal on Scientific Computing* 38, 3 (Jan. 2016), C280–C306. DOI: https://doi.org/10.1137/15M1010798

[79] N. Gholkar, F. Mueller, and B. Rountree. 2016. Power tuning HPC jobs on power-constrained systems. In *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation (PACT'16)*. ACM, New York, 179–191. DOI: https://doi.org/10.1145/2967938.2967961

[80] P. Ghysels, T. J. Ashby, K. Meerbergen, and W. Vanroose. 2013. Hiding global communication latency in the GMRES algorithm on massively parallel machines. *SIAM Journal on Scientific Computing* (Jan. 2013). DOI: https://doi.org/10.1137/12086563X

[81] M. Giampapa, T. Gooding, T. Inglett, and R. W. Wisniewski. 2010. Experiences with a lightweight supercomputer kernel: Lessons learned from Blue Gene's CNK. In *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–10. DOI: https://doi.org/10.1109/SC.2010.22

[82] B. Giridhar, M. Cieslak, D. Duggal, R. Dreslinski, H. M. Chen, R. Patti, B. Hold, C. Chakrabarti, T. Mudge, and D. Blaauw. 2013. Exploring DRAM organizations for energy-efficient and resilient exascale memories. In *SC'13:*

Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 1–12. DOI:https://doi.org/10.1145/2503210.2503215

[83] L. A. B. Gomez and F. Cappello. 2015. Detecting and correcting data corruption in stencil applications through multivariate interpolation. In *Proceedings of the 2015 IEEE International Conference on Cluster Computing*. 595–602. DOI:https://doi.org/10.1109/CLUSTER.2015.108

[84] W. Gropp and M. Snir. 2013. Programming for exascale computers. *Computing in Science Engineering* 15, 6 (Nov. 2013), 27–35. DOI:https://doi.org/10.1109/MCSE.2013.96

[85] S. Gupta, T. Patel, C. Engelmann, and D. Tiwari. 2017. Failures in large scale systems: Long-term measurement, analysis, and implications. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, (SC'17)*. DOI:https://doi.org/10.1145/3126908.3126937

[86] A. Haidar, H. Jagode, P. Vaccaro, A. YarKhan, S. Tomov, and J. Dongarra. 2019. Investigating power capping toward energy-efficient scientific applications. *Concurrency and Computation: Practice and Experience* (March 2019). DOI:https://doi.org/10.1002/cpe.4485

[87] F. Hariri, T. M. Tran, A. Jocksch, E. Lanti, J. Progsch, P. Messmer, S. Brunner, C. Gheller, and L. Villard. 2016. A portable platform for accelerated PIC codes and its application to GPUs using OpenACC. *Computer Physics Communications* 207 (2016), 69–82. DOI:https://doi.org/10.1016/j.cpc.2016.05.008

[88] K. Hasanov, J. Quintin, and A. Lastovetsky. 2015. Hierarchical approach to optimization of parallel matrix multiplication on large-scale platforms. *The Journal of Supercomputing* 71, 11 (Nov. 2015), 3991–4014. DOI:https://doi.org/10.1007/s11227-014-1133-x

[89] A. Hayashi, S. R. Paul, M. Grossman, J. Shirako, and V. Sarkar. 2017. Chapel-on-X: Exploring tasking runtimes for PGAS languages. In *Proceedings of the 3rd International Workshop on Extreme Scale Programming Models and Middleware (ESPM2'17)*. ACM, New York, 5:1–5:8. DOI:https://doi.org/10.1145/3152041.3152086

[90] A. Heirich, E. Slaughter, M. Papadakis, W. Lee, T. Biedert, and A. Aiken. 2017. In situ visualization with task-based parallelism. In *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization (ISAV'17)*. ACM, Denver, CO, 17–21. DOI:https://doi.org/10.1145/3144769.3144771

[91] Stijn Heldens, Alessio Sclocco, and Henk Dreuning. 2019. NLeSC/automated-literature-analysis. DOI:https://doi.org/10.5281/zenodo.3386072

[92] T. Herault and Y. Robert. 2015. *Fault-Tolerance Techniques for High-Performance Computing*. Springer, Cham Heidelberg New York Dordrecht London.

[93] W. Hu, G. Liu, Q. Li, Y. Jiang, and G. Cai. 2016. Storage wall for exascale supercomputing. *Frontiers of Information Technology & Electronic Engineering* 17, 11 (Nov. 2016), 1154–1175. DOI:https://doi.org/10.1631/FITEE.1601336

[94] M. Huber, B. Gmeiner, U. Rüde, and B. Wohlmuth. 2016. Resilience for massively parallel multigrid solvers. *SIAM Journal on Scientific Computing* 38, 5 (Jan. 2016), S217–S239. DOI:https://doi.org/10.1137/15M1026122

[95] S. Hukerikar and R. F. Lucas. 2016. Rolex: Resilience-oriented language extensions for extreme-scale systems. *The Journal of Supercomputing* 72, 12 (Dec. 2016), 4662–4695. DOI:https://doi.org/10.1007/s11227-016-1752-5

[96] D. Ibtesham, D. Arnold, P. G. Bridges, K. B. Ferreira, and R. Brightwell. 2012. On the viability of compression for reducing the overheads of checkpoint/restart-based fault tolerance. In *Proceedings of the 2012 41st International Conference on Parallel Processing*. 148–157. DOI:https://doi.org/10.1109/ICPP.2012.45

[97] C. Iwainsky, S. Shudler, A. Calotoiu, A. Strube, M. Knobloch, C. Bischof, and F. Wolf. 2015. How many threads will be too many? On the scalability of OpenMP implementations. In *Euro-Par 2015: Parallel Processing*. vol. 9233. Springer Berlin, Berlin, Germany, 451–463. DOI:https://doi.org/10.1007/978-3-662-48096-0_35

[98] C. Jin, B. R. de Supinski, D. Abramson, H. Poxon, L. DeRose, M. N. Dinh, M. Endrei, and E. R. Jessup. 2017. A survey on software methods to improve the energy efficiency of parallel computing. *The International Journal of High Performance Computing Applications* 31, 6 (Nov. 2017), 517–549. DOI:https://doi.org/10.1177/1094342016665471

[99] H. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L. Huang, and B. Chapman. 2011. High performance computing using MPI and OpenMP on multi-core parallel systems. *Parallel Comput.* 37, 9 (Sept. 2011), 562–575. DOI:https://doi.org/10.1016/j.parco.2011.02.002

[100] H. Kaiser, T. Heller, B. Adelstein-Lelbach, A. Serio, and D. Fey. 2014. HPX: A task based programming model in a global address space. In *Proceedings of the 8th International Conference on Partitioned Global Address Space Programming Models (PGAS'14)*. ACM, Eugene, OR, 1–11. DOI:https://doi.org/10.1145/2676870.2676883

[101] D. A. Kane, P. Rogé, and S. S. Snapp. 2016. A systematic review of perennial staple crops literature using topic modeling and bibliometric analysis. *PLoS one* 11, 5 (May 2016). DOI:https://doi.org/10.1371/journal.pone.0155788

[102] S. Kannan, A. Gavrilovska, K. Schwan, D. Milojicic, and V. Talwar. 2011. Using active NVRAM for I/O staging. In *Proceedings of the 2nd International Workshop on Petascal Data Analytics: Challenges and Opportunities (PDAC'11)*. ACM, Seattle, Washington, 15. DOI:https://doi.org/10.1145/2110205.2110209

[103] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, et al. 2008. *Exascale Computing Study: Technology Challenges in Achieving Exascale Systems*. Technical Report. Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO).

[104] P. Kogge and J. Shalf. 2013. Exascale computing trends: Adjusting to the "new normal" for computer architecture. *Computing in Science Engineering* 15, 6 (Nov. 2013), 16–26. DOI: https://doi.org/10.1109/MCSE.2013.95

[105] J. M. Kunkel, M. Kuhn, and T. Ludwig. 2014. Exascale storage systems – An analytical study of expenses. *Supercomputing Frontiers and Innovations* 1, 1 (June 2014), 116–134–134. DOI: https://doi.org/10.14529/jsfi140106

[106] Oak Ridge National Laboratory. 2019. Summit. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[107] B. N. Lawrence, M. Rezny, R. Budich, P. Bauer, J. Behrens, M. Carter, W. Deconinck, R. Ford, C. Maynard, S. Mullerworth, C. Osuna, A. Porter, K. Serradell, S. Valcke, N. Wedi, and S. Wilson. 2018. Crossing the chasm: How to develop weather and climate models for next generation computers? *Geoscientific Model Development* 11, 5 (May 2018), 1799–1821. DOI: https://doi.org/10.5194/gmd-11-1799-2018

[108] S. Lee and J. S. Vetter. 2012. Early evaluation of directive-based GPU programming models for productive exascale computing. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* 1–11. DOI: https://doi.org/10.1109/SC.2012.51

[109] D. Li, J. S. Vetter, G. Marin, C. McCurdy, C. Cira, Z. Liu, and W. Yu. 2012. Identifying opportunities for byte-addressable non-volatile memory in extreme-scale scientific applications. In *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium.* 945–956. DOI: https://doi.org/10.1109/IPDPS.2012.89

[110] X. Liao, C. Yung, T. Tang, H. Yi, F. Wang, Q. Wu, and J. Xue. 2014. OpenMC: Towards simplifying programming for TianHe supercomputers. *J. Comput. Sci. Technol.* 29, 3 (May 2014), 532–546. DOI: https://doi.org/10.1007/s11390-014-1447-4

[111] J. Liu and G. Agrawal. 2016. Soft error detection for iterative applications using offline training. In *Proceedings of the 2016 IEEE 23rd International Conference on High Performance Computing (HiPC).* 2–11. DOI: https://doi.org/10.1109/HiPC.2016.011

[112] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, J. Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield, M. Parashar, N. Samatova, K. Schwan, A. Shoshani, M. Wolf, K. Wu, and W. Yu. 2014. Hello ADIOS: The challenges and lessons of developing leadership class I/O frameworks. *Concurrency and Computation: Practice and Experience* 26, 7 (May 2014), 1453–1473. DOI: https://doi.org/10.1002/cpe.3125

[113] J. Lofstead, I. Jimenez, C. Maltzahn, Q. Koziol, J. Bent, and E. Barton. 2016. DAOS and friends: A proposal for an exascale storage system. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* 585–596. DOI: https://doi.org/10.1109/SC.2016.49

[114] Los Alamos Lab. 2019. High-Performance Computing: Roadrunner. http://www.lanl.gov/roadrunner/.

[115] R. Lucas et al. 2014. *Top Ten Exascale Research Challenges.* Technical Report. U.S. Department of Energy, Office of Science. DEO ASCAC Subcommittee Report.

[116] J. Lüttgau, M. Kuhn, K. Duwe, Y. Alforov, E. Betke, J. Kunkel, and T. Ludwig. 2018. Survey of storage systems for high-performance computing. *Supercomputing Frontiers and Innovations* 5, 1 (April 2018), 31–58. DOI: https://doi.org/10.14529/jsfi180103

[117] K. Ma. 2009. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications* 29, 6 (2009), 14–19. DOI: https://doi.org/10.1109/MCG.2009.120

[118] T. Maeno, K. De, T. Wenaus, P. Nilsson, G. A. Stewart, R. Walker, A. Stradling, J. Caballero, M. Potekhin, D. Smith, and T. A. Collaboration. 2011. Overview of ATLAS PanDAWorkload Management. *J. Phys. Conf. Ser.* 331, 7 (2011), 072024. https://doi.org/10.1088/1742-6596/331/7/072024

[119] T. Maeno, K. De, T. Wenaus, P. Nilsson, G. A. Stewart, R. Walker, A. Stradling, J. Caballero, M. Potekhin, D. Smith, and T. A. Collaboration. 2011. Overview of ATLAS PanDA workload management. *J. Phys.: Conf. Ser.* 331, 7 (2011), 072024. DOI: https://doi.org/10.1088/1742-6596/331/7/072024

[120] J. Mair, Z. Huang, D. Eyers, and Y. Chen. 2015. Quantifying the energy efficiency challenges of achieving exascale computing. In *Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* 943–950. DOI: https://doi.org/10.1109/CCGrid.2015.130

[121] S. Markidis, I. B. Peng, J. Larsson Träff, A. Rougier, V. Bartsch, R. Machado, M. Rahn, A. Hart, D. Holmes, M. Bull, and E. Laure. 2016. The EPiGRAM project: Preparing parallel programming models for exascale. In *High Performance Computing.* Vol. 9945. Springer International Publishing, Cham, 56–68. DOI: https://doi.org/10.1007/978-3-319-46079-6_5

[122] C. D. Martino, W. Kramer, Z. Kalbarczyk, and R. Iyer. 2015. Measuring and understanding extreme-scale application resilience: A field study of 5,000,000 HPC application runs. In *Proceedings of the International Conference on Dependable Systems and Networks*, Vol. 2015-September. 25–36. DOI: https://doi.org/10.1109/DSN.2015.50

[123] G. Mathew, A. Agrawal, and T. Menzies. 2017. Trends in topics at SE conferences (1993-2013). In *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C).* 397–398. DOI: https://doi.org/10.1109/ICSE-C.2017.52

[124] T. G. Mattson, R. Cledat, V. Cavé, V. Sarkar, Z. Budimlić, S. Chatterjee, J. Fryman, I. Ganev, R. Knauerhase, M. Lee, B. Meister, B. Nickerson, N. Pepperling, B. Seshasayee, S. Tasirlar, J. Teller, and N. Vrvilo. 2016. The open community runtime: A runtime system for extreme scale computing. In *Proceedings of the 2016 IEEE High Performance Extreme Computing Conference (HPEC)*. 1–7. DOI: https://doi.org/10.1109/HPEC.2016.7761580

[125] E. Meneses, X. Ni, G. Zheng, C. L. Mendes, and L. V. Kalé. 2015. Using migratable objects to enhance fault tolerance schemes in supercomputers. *IEEE Transactions on Parallel and Distributed Systems* 26, 7 (July 2015), 2061–2074. DOI: https://doi.org/10.1109/TPDS.2014.2342228

[126] P. Messina. 2017. The exascale computing project. *Computing in Science & Engineering* 19, 3 (May 2017), 63–67. DOI: https://doi.org/10.1109/MCSE.2017.57

[127] M. R. Meswani, G. H. Loh, S. Blagodurov, D. Roberts, J. Slice, and M. Ignatowski. 2014. Toward efficient programmer-managed two-level memory hierarchies in exascale computers. In *Proceedings of the 2014 Hardware-Software Co-Design for High Performance Computing*. 9–16. DOI: https://doi.org/10.1109/Co-HPC.2014.8

[128] G. Mitra, E. Stotzer, A. Jayaraj, and A. P. Rendell. 2014. Implementation and optimization of the OpenMP accelerator model for the TI keystone II architecture. In *Using and Improving OpenMP for Devices, Tasks, and More (Lecture Notes in Computer Science)*. Springer International Publishing, 202–214.

[129] S. Mittal and J. S. Vetter. 2015. A survey of CPU-GPU heterogeneous computing techniques. *ACM Comput. Surv.* 47, 4 (July 2015), 69:1–69:35. DOI: https://doi.org/10.1145/2788396

[130] K. Moreland, U. Ayachit, B. Geveci, and K. Ma. 2011. Dax toolkit: A proposed framework for data analysis and visualization at extreme scale. In *Proceedings of the 2011 IEEE Symposium on Large Data Analysis and Visualization.* 97–104. DOI: https://doi.org/10.1109/LDAV.2011.6092323

[131] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns. 2017. Enabling parallel simulation of large-scale HPC network systems. *IEEE Transactions on Parallel and Distributed Systems* 28, 1 (2017), 87–100. DOI: https://doi.org/10.1109/TPDS.2016.2543725

[132] R. Nair et al. 2015. Active memory cube: A processing-in-memory architecture for exascale systems. *IBM Journal of Research and Development* 59, 2/3 (March 2015), 17:1–17:14. DOI: https://doi.org/10.1147/JRD.2015.2409732

[133] S. Narasimhamurthy, N. Danilov, S. Wu, G. Umanesan, S. W. D. Chien, S. Rivas-Gomez, I. B. Peng, E. Laure, S. De Witt, D. Pleiter, and S. Markidis. 2018. The SAGE project: A storage centric approach for exascale computing. In *Proceedings of the 2018 ACM International Conference on Computing Frontiers, (CF 2018)*. 287–292. DOI: https://doi.org/10.1145/3203217.3205341

[134] T. Naughton, G. Smith, C. Engelmann, G. Vallée, F. Aderholdt, and S. L. Scott. 2014. What is the right balance for performance and isolation with virtualization in HPC? In *Euro-Par 2014: Parallel Processing Workshops (Lecture Notes in Computer Science)*. Springer International Publishing, 570–581.

[135] V. P. Nikolskiy, V. V. Stegailov, and V. S. Vecher. 2016. Efficiency of the Tegra K1 and X1 systems-on-chip for classical molecular dynamics. In *Proceedings of the 2016 International Conference on High Performance Computing Simulation (HPCS)*. 682–689. DOI: https://doi.org/10.1109/HPCSim.2016.7568401

[136] Oak Ridge National Laboratory. 2019. Titan. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/.

[137] K. O'Brien, L. D. Tucci, G. Durelli, and M. Blott. 2017. Towards exascale computing with heterogeneous architectures. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. 398–403. DOI: https://doi.org/10.23919/DATE.2017.7927023

[138] J. Ouyang, B. Kocoloski, J. R. Lange, and K. Pedretti. 2015. Achieving performance isolation with lightweight co-kernels. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (HPDC'15)*. ACM, Portland, Oregon, 149–160. DOI: https://doi.org/10.1145/2749246.2749273

[139] S. Páll, M. J. Abraham, C. Kutzner, B. Hess, and E. Lindahl. 2015. Tackling exascale software challenges in molecular dynamics simulations with GROMACS. In *Solving Software Challenges for Exascale*. Vol. 8759. Springer International Publishing, Cham, 3–27. DOI: https://doi.org/10.1007/978-3-319-15976-8_1

[140] J. A. Pascual, J. Lant, A. Attwood, C. Concatto, J. Navaridas, M. Lujan, and J. Goodacre. 2017. Designing an exascale interconnect using multi-objective optimization. In *Proceedings of the 2017 IEEE Congress on Evolutionary Computation, (CEC 2017)*. 2209–2216. DOI: https://doi.org/10.1109/CEC.2017.7969572

[141] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. 2013. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing (ICS'13)*. ACM, Eugene, Oregon, 173. DOI: https://doi.org/10.1145/2464996.2465009

[142] V. Pauca, F. Shahnaz, M. Berry, and R. Plemmons. 2004. Text mining using non-negative matrix factorizations. In *Proceedings of the 2004 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 452–456. DOI: https://doi.org/10.1137/1.9781611972740.45

[143] S. Perarnau, R. Thakur, K. Iskra, K. Raffenetti, F. Cappello, R. Gupta, P. Beckman, M. Snir, H. Hoffmann, M. Schulz, and B. Rountree. 2015. Distributed monitoring and management of exascale systems in the Argo project. In *Distributed*

Applications and Interoperable Systems (Lecture Notes in Computer Science). Springer International Publishing, 173–178.

[144] S. Perarnau, J. A. Zounmevo, M. Dreher, B. C. V. Essen, R. Gioiosa, K. Iskra, M. B. Gokhale, K. Yoshii, and P. Beckman. 2017. Argo NodeOS: Toward unified resource management for exascale. In *Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 153–162. DOI: https://doi.org/10.1109/IPDPS.2017.25

[145] S. Pickartz, S. Lankes, A. Monti, C. Clauss, and J. Breitbart. 2016. Application migration in HPC — A driver of the exascale era? In *Proceedings of the 2016 International Conference on High Performance Computing Simulation (HPCS)*. 318–325. DOI: https://doi.org/10.1109/HPCSim.2016.7568352

[146] M. F. Porter. 1980. An algorithm for suffix stripping. *Program* 14, 3 (1980), 130–137. DOI: https://doi.org/10.1108/eb046814

[147] A. K. Porterfield, S. L. Olivier, S. Bhalachandra, and J. F. Prins. 2013. Power measurement and concurrency throttling for energy reduction in OpenMP programs. In *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*. 884–891. DOI: https://doi.org/10.1109/IPDPSW.2013.15

[148] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, and T. Hoefler. 2014. Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks. In *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing - HPDC'14*. ACM Press, Vancouver, BC, Canada, 129–140. DOI: https://doi.org/10.1145/2600212.2600225

[149] N. Rajovic et al. 2016. The Mont-Blanc prototype: An alternative approach for HPC systems. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 444–455. DOI: https://doi.org/10.1109/SC.2016.37

[150] C. Rasmussen, M. Sottile, S. Rasmussen, D. Nagle, and W. Dumas. 2016. CAFe: Coarray fortran extensions for heterogeneous computing. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 357–365. DOI: https://doi.org/10.1109/IPDPSW.2016.140

[151] D. A. Reed and J. Dongarra. 2015. Exascale computing and big data. *Commun. ACM* 58, 7 (2015), 56–68.

[152] P. M. Reed and D. Hadka. 2014. Evolving many-objective water management to exploit exascale computing. *Water Resources Research* 50, 10 (2014), 8367–8373. DOI: https://doi.org/10.1002/2014WR015976

[153] L. Rokach and O. Maimon. 2005. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*. Springer, Boston, MA, 321–352. DOI: https://doi.org/10.1007/0-387-25465-X_15

[154] S. Rumley, M. Bahadori, R. Polster, S. D. Hammond, D. M. Calhoun, K. Wen, A. Rodrigues, and K. Bergman. 2017. Optical interconnects for extreme scale computing systems. *Parallel Comput.* 64 (May 2017), 65–80. DOI: https://doi.org/10.1016/j.parco.2017.02.001

[155] G. Salton and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.* 24, 5 (Aug. 1988), 513–523. DOI: https://doi.org/10.1016/0306-4573(88)90021-0

[156] V. Sarkar, W. Harrod, and A. E. Snavely. 2009. Software challenges in extreme scale systems. In *Journal of Physics: Conference Series*, Vol. 180. IOP Publishing, 012045.

[157] O. Sarood, E. Meneses, and L. V. Kale. 2013. A "cool" way of improving the reliability of HPC machines. In *SC'13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–12. DOI: https://doi.org/10.1145/2503210.2503228

[158] O. Sarood, P. Miller, E. Totoni, and L. V. Kalé. 2012. "Cool" load balancing for high performance computing data centers. *IEEE Trans. Comput.* 61, 12 (Dec. 2012), 1752–1764. DOI: https://doi.org/10.1109/TC.2012.143

[159] K. Sato, N. Maruyama, K. Mohror, A. Moody, T. Gamblin, B. R. De Supinski, and S. Matsuoka. 2012. Design and modeling of a non-blocking checkpointing system. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*. DOI: https://doi.org/10.1109/SC.2012.46

[160] M. J. Schulte, M. Ignatowski, G. H. Loh, B. M. Beckmann, W. C. Brantley, S. Gurumurthi, N. Jayasena, I. Paul, S. K. Reinhardt, and G. Rodgers. 2015. Achieving exascale capabilities through heterogeneous computing. *IEEE Micro* 35, 4 (July 2015), 26–36. DOI: https://doi.org/10.1109/MM.2015.71

[161] F. Shahnaz, M. W. Berry, V. P. Pauca, and R. J. Plemmons. 2006. Document clustering using nonnegative matrix factorization. *Information Processing & Management* 42, 2 (March 2006), 373–386. DOI: https://doi.org/10.1016/j.ipm.2004.11.005

[162] J. Shalf, S. Dosanjh, and J. Morrison. 2010. Exascale computing technology challenges. In *Proceedings of the International Conference on High Performance Computing for Computational Science*. Springer, 1–25.

[163] J. Shalf, D. Quinlan, and C. Janssen. 2011. Rethinking hardware-software codesign for exascale systems. *Computer* 44, 11 (Nov. 2011), 22–30. DOI: https://doi.org/10.1109/MC.2011.300

[164] H. Shoukourian, T. Wilde, A. Auweter, and A. Bode. 2014. Monitoring power data: A first step towards a unified energy efficiency evaluation toolset for HPC data centers. *Environmental Modelling & Software* 56 (June 2014), 13–26. DOI: https://doi.org/10.1016/j.envsoft.2013.11.011

[165] A. Sidorova, N. Evangelopoulos, J. S. Valacich, and T. Ramakrishnan. 2008. Uncovering the intellectual core of the information systems discipline. *MIS Quarterly* 32, 3 (2008), 467–482. DOI: https://doi.org/10.2307/25148852

[166] M. Snir et al. 2014. Addressing failures in exascale computing. *The International Journal of High Performance Computing Applications* 28, 2 (May 2014), 129–173. DOI: https://doi.org/10.1177/1094342014522573

[167] D. Stroobandt et al. 2016. EXTRA: Towards the exploitation of eXascale technology for reconfigurable architectures. In *Proceedings of the 2016 11th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC).* 1–7. DOI: https://doi.org/10.1109/ReCoSoC.2016.7533896

[168] V. Subotić, S. Brinkmann, V. Marjanović, R. M. Badia, J. Gracia, C. Niethammer, E. Ayguade, J. Labarta, and M. Valero. 2013. Programmability and portability for exascale: Top down programming methodology and tools with StarSs. *Journal of Computational Science* 4, 6 (Nov. 2013), 450–456. DOI: https://doi.org/10.1016/j.jocs.2013.01.008

[169] S. Syed and C. T. Weber. 2018. Using machine learning to uncover latent research topics in fishery models. *Reviews in Fisheries Science & Aquaculture* 26, 3 (July 2018), 319–336. DOI: https://doi.org/10.1080/23308249.2017.1416331

[170] C. A. Thraskias, E. N. Lallas, N. Neumann, L. Schares, B. J. Offrein, R. Henker, D. Plettemeier, F. Ellinger, J. Leuthold, and I. Tomkos. 2018. Survey of photonic and plasmonic interconnect technologies for intra-datacenter and high-performance computing communications. *IEEE Communications Surveys Tutorials* 20, 4 (Fourthquarter 2018), 2758–2783. DOI: https://doi.org/10.1109/COMST.2018.2839672

[171] E. Totoni, N. Jain, and L. V. Kalé. 2013. Toward runtime power management of exascale networks by on/off control of links. In *Proceedings of the 2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Ph.D Forum.* 915–922. DOI: https://doi.org/10.1109/IPDPSW.2013.191

[172] R. Trobec, R. Vasiljević, M. Tomašević, V. Milutinović, R. Beivide, and M. Valero. 2016. Interconnection networks in petascale computer systems: A survey. *ACM Comput. Surv.* 49, 3 (Sept. 2016), 44:1–44:24. DOI: https://doi.org/10.1145/2983387

[173] D. Unat, C. Chan, W. Zhang, S. Williams, J. Bachan, J. Bell, and J. Shalf. 2015. ExaSAT: An exascale co-design tool for performance modeling. *The International Journal of High Performance Computing Applications* 29, 2 (May 2015), 209–232. DOI: https://doi.org/10.1177/1094342014568690

[174] L. van der Maaten and G. Hinton. 2008. Visualizing data using T-SNE. *Journal of Machine Learning Research* 9, (Nov. 2008), 2579–2605.

[175] R. F. Van der Wijngaart, A. Kayi, J. R. Hammond, G. Jost, T. St. John, S. Sridharan, T. G. Mattson, J. Abercrombie, and J. Nelson. 2016. Comparing runtime systems with exascale ambitions using the parallel research kernels. In *High Performance Computing (Lecture Notes in Computer Science).* Springer International Publishing, 321–339.

[176] A. Varghese, B. Edwards, G. Mitra, and A. P. Rendell. 2014. Programming the Adapteva Epiphany 64-core network-on-chip coprocessor. In *Proceedings of the 2014 IEEE International Parallel Distributed Processing Symposium Workshops.* 984–992. DOI: https://doi.org/10.1109/IPDPSW.2014.112

[177] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arŕs, D. Mira, H. Calmet, F. Cucchietti, H. Owen, A. Taha, E. D. Burness, J. M. Cela, and M. Valero. 2016. Alya: Multiphysics engineering simulation toward exascale. *Journal of Computational Science* 14 (May 2016), 15–27. DOI: https://doi.org/10.1016/j.jocs.2015.12.007

[178] J. S. Vetter and S. Mittal. 2015. Opportunities for nonvolatile memory systems in extreme-scale high-performance computing. *Computing in Science Engineering* 17, 2 (March 2015), 73–82. DOI: https://doi.org/10.1109/MCSE.2015.4

[179] O. Villa, D. R. Johnson, M. Oconnor, E. Bolotin, D. Nellans, J. Luitjens, N. Sakharnykh, P. Wang, P. Micikevicius, A. Scudiero, S. W. Keckler, and W. J. Dally. 2014. Scaling the power wall: A path to exascale. In *SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* 830–841. DOI: https://doi.org/10.1109/SC.2014.73

[180] Y. Wang and Y. Zhang. 2013. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* 25, 6 (June 2013), 1336–1353. DOI: https://doi.org/10.1109/TKDE.2012.51

[181] C. Weinhold, A. Lackorzynski, J. Bierbaum, M. Küttler, M. Planeta, H. Härtig, A. Shiloh, E. Levy, T. Ben-Nun, A. Barak, T. Steinke, T. Schütt, J. Fajerski, A. Reinefeld, M. Lieber, and W. E. Nagel. 2016. FFMK: A fast and fault-tolerant microkernel-based system for exascale computing. In *Software for Exascale Computing - SPPEXA 2013-2015 (Lecture Notes in Computational Science and Engineering).* Springer International Publishing, 405–426.

[182] S. Werner, J. Navaridas, and M. Luján. 2017. A survey on optical network-on-chip architectures. *Comput. Surveys* 50, 6 (Dec. 2017), 1–37. DOI: https://doi.org/10.1145/3131346

[183] W. Xu, X. Liu, and Y. Gong. 2003. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR'03).* ACM, New York, NY, USA, 267–273. DOI: https://doi.org/10.1145/860435.860485

[184] C. Yang, J. C. Pichel, A. R. Smith, and D. A. Padua. 2014. Hierarchically tiled array as a high-level abstraction for codelets. In *Proceedings of the 2014 Fourt Workshop on Data-Flow Execution Models for Extreme Scale Computing.* IEEE, Edmonton, AB, Canada, 58–65. DOI: https://doi.org/10.1109/DFM.2014.11

[185] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K. Ma. 2010. In situ visualization for large-scale combustion simulations. *IEEE Computer Graphics and Applications* 30, 3 (May 2010), 45–57. DOI: https://doi.org/10.1109/MCG.2010.55

[186] M. Zakarya and L. Gillam. 2017. Energy efficient computing, clusters, grids and clouds: A taxonomy and survey. *Sustainable Computing: Informatics and Systems* 14 (June 2017), 13–33. DOI: https://doi.org/10.1016/j.suscom.2017.03.002

[187] G. Zheng, X. Ni, and L. V. Kalé. 2012. A scalable double in-memory checkpoint and restart scheme towards exascale. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN 2012)*. 1–6. DOI: https://doi.org/10.1109/DSNW.2012.6264677

[188] Q. Zheng, K. Ren, G. Gibson, B. W. Settlemyer, and G. Grider. 2015. DeltaFS: Exascale file systems scale better without dedicated servers. In *Proceedings of the 10th Parallel Data Storage Workshop (PDSW'15)*. ACM, Austin, Texas, 1–6. DOI: https://doi.org/10.1145/2834976.2834984

[189] J. A. Zounmevo, S. Perarnau, K. Iskra, K. Yoshii, R. Gioiosa, B. C. V. Essen, M. B. Gokhale, and E. A. Leon. 2015. A container-based approach to OS specialization for exascale computing. In *Proceedings of the 2015 IEEE International Conference on Cloud Engineering*. 359–364. DOI: https://doi.org/10.1109/IC2E.2015.78

Received January 2019; revised October 2019; accepted November 2019