Big data and extreme-scale computing: Pathways to Convergence-Toward a shaping strategy for a future software and data ecosystem for scientific inquiry

Article *in* International Journal of High Performance Computing Applications · July 2018

DOI: 10.1177/1094342018778123

Mark Asch
Université de Picardie Jules Verne
50 PUBLICATIONS **346** CITATIONS
SEE PROFILE

Terry Moore
University of Tennessee
58 PUBLICATIONS **1,232** CITATIONS
SEE PROFILE

Rosa M. Badia
Barcelona Supercomputing Center
270 PUBLICATIONS **4,342** CITATIONS
SEE PROFILE

Franck Cappello
Argonne National Laboratory
262 PUBLICATIONS **6,379** CITATIONS
SEE PROFILE

International Journal of
HIGH PERFORMANCE
COMPUTING APPLICATIONS

Research Paper

The International Journal of High Performance Computing Applications
2018, Vol. 32(4) 435–479
© The Author(s) 2018
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/1094342018778123
journals.sagepub.com/home/hpc
SAGE

Big data and extreme-scale computing: Pathways to Convergence-Toward a shaping strategy for a future software and data ecosystem for scientific inquiry

M Asch, T Moore, R Badia, M Beck, P Beckman, T Bidot, F Bodin, F Cappello, A Choudhary, B de Supinski, E Deelman, J Dongarra, A Dubey, G Fox, H Fu, S Girona, W Gropp, M Heroux, Y Ishikawa, K Keahey, D Keyes, W Kramer, J-F Lavignon, Y Lu, S Matsuoka, B Mohr, D Reed, S Requena, J Saltz, T Schulthess, R Stevens, M Swany, A Szalay, W Tang, G Varoquaux, J-P Vilotte, R Wisniewski, Z Xu and I Zacharov

Abstract

Over the past four years, the Big Data and Exascale Computing (BDEC) project organized a series of five international workshops that aimed to explore the ways in which the new forms of data-centric discovery introduced by the ongoing revolution in high-end data analysis (HDA) might be integrated with the established, simulation-centric paradigm of the high-performance computing (HPC) community. Based on those meetings, we argue that the rapid proliferation of digital data generators, the unprecedented growth in the volume and diversity of the data they generate, and the intense evolution of the methods for analyzing and using that data are radically reshaping the landscape of scientific computing. The most critical problems involve the logistics of wide-area, multistage workflows that will move back and forth across the computing continuum, between the multitude of distributed sensors, instruments and other devices at the networks edge, and the centralized resources of commercial clouds and HPC centers. We suggest that the prospects for the future integration of technological infrastructures and research ecosystems need to be considered at three different levels. First, we discuss the convergence of research applications and workflows that establish a research paradigm that combines both HPC and HDA, where ongoing progress is already motivating efforts at the other two levels. Second, we offer an account of some of the problems involved with creating a converged infrastructure for peripheral environments, that is, a shared infrastructure that can be deployed throughout the network in a scalable manner to meet the highly diverse requirements for processing, communication, and buffering/storage of massive data workflows of many different scientific domains. Third, we focus on some opportunities for software ecosystem convergence in big, logically centralized facilities that execute large-scale simulations and models and/or perform large-scale data analytics. We close by offering some conclusions and recommendations for future investment and policy review.

Keywords

Big data, extreme-scale computing, future software, traditional HPC, high-end data analysis

1. Executive summary

Although the "big data" revolution first came to public prominence (circa 2010) in online enterprises like Google, Amazon, and Facebook, it is now widely recognized as the initial phase of a watershed transformation that modern society generally—and scientific and engineering research in particular—are in the process of undergoing. Responding to this disruptive wave of change, over the past 4 years,

Innovative Computing Laboratory, University of Tennessee, Knoxville, TN, USA

Corresponding author:
J Dongarra, University of Tennessee, Knoxville, TN 37996, USA.
Email: dongarra@icl.utk.edu

the big data and exascale computing (BDEC) project organized a series of five international workshops that aimed to explore the ways in which the new forms of data-centric discovery introduced by this revolution might be integrated with the established, simulation-centric paradigm of the high-performance computing (HPC) community. These BDEC workshops grew out of the prior efforts of the International Exascale Software Project (IESP)—a collaboration of US, European Union (EU), and Japanese HPC communities that produced an influential roadmap for achieving exascale computing early in the next decade. It also shared the IESP's mission to foster the codesign of shared software infrastructure for extreme-scale science that draws on international cooperation and supports a broad spectrum of major research domains. However, as we argue in more detail in this report, subsequent reflections on the content and discussions of the BDEC workshops make it evident that the rapid proliferation of digital data generators, the unprecedented growth in the volume and diversity of the data they generate, and the intense evolution of the methods for analyzing and using that data are radically reshaping the landscape of scientific computing.

At a time when science is, and needs to be, more international and interdisciplinary than ever, this data-driven upheaval is exacerbating divisions, both recent and longstanding, in the cyberecosystem of science; it is thereby raising a host of conceptual, political, economic, and cultural challenges to future cooperation. The BDEC community's sustained examination of these changes focused on the problems in two different divisions in the ecosystem that the big data revolution has either produced or destabilized.

1. The split between the traditional HPC and high-end data analysis (HDA): The divide between HPC and HDA software ecosystems emerged early this century when software infrastructure and tools for data analytics that had been developed by online service providers were open sourced and picked up by various scientific communities to solve their own data analysis challenges. Major technical differences between the HPC and the HDA ecosystems include software development paradigms and tools, virtualization and scheduling strategies, storage models (local vs cloud/storage area network (SAN)), resource allocation policies, and strategies for redundancy and fault tolerance. These technical differences, in turn, tend to render future cross-boundary collaboration and progress increasingly problematic.
2. The split between stateless networks and stateful services provided by end systems: The division between stateless datagram-delivery networks (e.g. the Internet) and stateful services provided by network-attached end systems (e.g. supercomputers, laptops, sensors, and mobile telephones) has been fundamental to the cyberinfrastructure paradigm that proved remarkably successful for well over three decades. However, the exponential growth in data volumes over the same period of time forced users and service providers to repeatedly find new workarounds (e.g. FTP mirror sites, web caches, web cache hierarchies, content delivery networks (CDNs), and commercial clouds) in order to manage the logistics of ever larger data flows. Unfortunately, such workarounds seem to have reached their limits. The most explosive growth in data generation today is taking place in "edge environments" (i.e. across the network from both HPC and commercial cloud machine rooms). These new data sources include major scientific instruments, experimental facilities, remote sensors (e.g. satellite imagery), and the myriad of distributed sensors with which the plans for "smart cities" and the Internet of Things (IoTs) are replete. The environment in which many of these data torrents originate lacks the processing and buffer/storage resources to manage the logistics of such immense flows, and yet the conventional strategy of backhauling all data across a fast link to the cloud or data center is no longer a viable option for many applications. Hence, the intense commercial and research interest in "fog" or "edge" computing infrastructure—in one way or another—is supposed to solve this fundamental problem by combining processing, storage/buffering, and communication in a converged distributed services platform (DSP) that can be deployed in edge environments.

Looking toward the future of cyberinfrastructure for science and engineering through the lens of these two bifurcations made it clear to the BDEC community that, in the era of big data, the most critical problems involve the logistics of wide-area, multistage workflows—the diverse patterns of when, where, and how data are to be produced, transformed, shared, and analyzed. Consequently, the challenges involved in codesigning software infrastructure for science have to be reframed to fully take account of the diversity of workflow patterns that different application communities want to create. For the HPC community, all of the imposing design and development issues of creating an exascale-capable software stack remain, but the supercomputers that need this stack must now be viewed as the nodes (perhaps the most important nodes) in the very large network of computing resources required to process and explore rivers of data flooding in from multiple sources.

Against that background, we suggest that the prospects for integration of technological infrastructures and research ecosystems need to be considered at three different levels—or from three different perspectives. First, we discuss opportunities for convergence of research applications and workflows, where, despite the impediments of the ongoing cyberecosystem Balkanization, progress toward a research paradigm that combines both HPC and HDA is already being made. Such success with application-workflow communities both orients and motivates efforts at the other two levels. Second, we offer an account of some of the problems involved with creating a converged infrastructure for distributed edge, or "peripheral," environments (i.e. shared infrastructure that can be deployed throughout the network in a scalable manner to meet the combined data processing, communication, and buffer/storage requirements of massive data workflows). Third, we focus on some opportunities for software ecosystem convergence in large, logically centralized facilities that execute large-scale simulations and models and/or perform large-scale data analytics. Finally, we offer some conclusions and recommendations for future investment and policy review. We briefly summarize each of these parts of the report in the following sections.

1.1. Emerging convergence in large-scale scientific applications

Despite the manifest challenges of a converged cyberinfrastructure for multidisciplinary research, the scientific and engineering opportunities are compelling. Many science communities are combining HPC and HDA applications and methods in large-scale workflows that orchestrate simulations or incorporate them into the stages of large-scale analysis pipelines for data generated by simulations, experiments, or observations. Communities that would benefit substantially from application and workflow convergence are abundant, spanning all of science and engineering. To provide intellectual context for an analysis of application-workflow level convergence, we describe a unified model of the inference cycle for the process of scientific inquiry that locates "computational science" and "data science" as different phases of that cycle. Next, to capture the complexity of application workflows, which typically involve changing configurations of software, hardware, and data flows, we structure our discussion around three computational archetypes that provide broader categories than specific traditional formulations and that furnish a convenient grouping of collaborations by the "culture" of the users. We illustrate these application workflows with exemplars that "follow the data." In that context, we address three modalities of data provenance: (1) data arriving from the edge (often in real time), never centralized; (2) federated multisource archived data; and (3) combinations of data stored from observational archives with a dynamic simulation. These exemplars make it evident that understanding application workflows, and especially application workflows that involve far-flung data sources, is pivotal for developing a new application-level convergence paradigm that combines HDA and HPC.

1.2. Challenges to converged infrastructure in edge environments

The proliferation of huge and heterogeneous flows of data generated outside of commercial clouds and HPC centers (i.e. across the wide area network (WAN) in peripheral environments), as well as the need to distribute large data sets from such centralized facilities to the edge, represents a highly disruptive development that makes the way forward in many different areas of research uncertain. At the core of this uncertainty is the fact that the explosive growth of digital data producers at the edge, or in the periphery, creates problems that are highly multidimensional. Looking at the properties of the data flows being generated, they exhibit a range challenging characteristics, including their volume, velocity, value, variety, variability, and veracity. The most general designation for the field that must study and understand the problems that these data flows present is "data logistics" (i.e. the management of the time-sensitive positioning and encoding/layout of data relative to its intended users and the computational resources that they can apply). But problems of data logistics affect not only wide area workflows, but also workflows in the machine room. For example, whether you are talking about the output of a major instrument or a large HPC simulation, some form of data reduction has to be applied locally before any further data movement can be attempted. Hence, one can think of data logistics as defining a continuum, with I/O issues inside the Internet data center (IDC) or supercomputing facility falling at one end, and data-intensive workflows that begin at remote and/or distributed data sources—possibly scattered across edge environments—falling at the other.

Creating a common, shared software infrastructure that can address the logistical challenges of application workflows along this entire continuum is a critically important challenge for the scientific community in the coming decade. We use DSP as a generic characterization of the infrastructure/ecosystem that the community must develop in order to support such compute- and/or data-intensive workflows between the ingress/egress of the IDC or HPC center and the network edge. We present two pieces of common context for this discussion. First, we preserve continuity with the traditional TCP/IP + Unix/Linux paradigm by reiterating the importance of the hourglass architecture, and the "spanning layer" at its waist, as the foundation for ecosystem interoperability. Second, since these workflows are inherently stateful processes, it is clear that an adequate DSP must be not only wide-area capable, but must also offer processing, memory/storage, and communication as shareable resources. Satisfying these requirements in a common, open, and interoperable way, which is essential for the broad science and engineering community but would also benefit society as a whole, will be no mean feat. We conclude this section by briefly reviewing some of the strategies (e.g. stream processing, CDNs, and edge computing) that are currently being deployed to address different problem areas on the data-logistics continuum.

1.3. Pathways to convergence for large, logically centralized facilities

Today, most scientific research programs are striving to integrate both advanced computing and data analytics, and the drive to fuse these two methodologies strongly motivates the integration of the associated software with hardware infrastructures and ecosystems. However, this desired fusion raises a number of challenges: overcoming the differences in cultures and tools; dealing with shifting workforce skills; adopting new infrastructure; ensuring the coexistence of stream and batch models; computing at the edge; and implementing virtualization for sharing, resource allocation, and efficiency. We have also identified two major cross-cutting challenges: energy sustainability and data reduction. As far as architectures are concerned, radically improved resource management is indispensable for next-generation workflows, and—in this context—containerization is a promising candidate for the narrow waist, or spanning layer, of the hourglass model.

As the new era of big data and extreme-scale computing continues to develop, it seems clear that both centralized systems (e.g. HPC centers and commercial cloud systems) and decentralized systems (e.g. any of the alternative designs for edge/fog infrastructure) will share many common software challenges and opportunities. Software libraries for common intermediate processing tasks need to be promoted, and a complete software ecosystem for application development is needed. Finally, the divergence of programming models and languages poses a convergence issue—not only with regard to interoperability of the applications but also to the interoperability between data formats from different programming languages.

1.4. Conclusions and recommendations

Following the abovementioned structure of the document, we have divided our findings and recommendations into three categories: (1) global recommendations, (2) recommendations for edge environments, and (3) recommendations for centralized facilities. However, our ultimate goal is to lay the ground work for the kind of community-driven "shaping strategy" (Hagel and Brown, 2017; Hagel et al., 2008) that we believe would be both more appropriate and more successful overall. Consequently, the conclusions as they appear subsequently may have to be refactored to serve the shaping strategy model.

Our primary global recommendation would be to address the basic problem of the two paradigm splits: the HPC/HDA software ecosystem split and the wide area data logistics split. For this to be achieved, new standards are needed to govern the interoperability between the data paradigm and the compute paradigm. These new standards should be based on a new common and open *DSP* that offers programmable access to shared processing, storage, and communication resources and that can serve as a universal foundation for the component interoperability that novel services and applications will require.

We make the following five recommendations for decentralized edge and peripheral ecosystems:

1. converge on a new hourglass architecture for a common DSP;
2. target workflow patterns for improved data logistics;
3. design cloud stream processing capabilities for HPC;
4. promote a scalable approach to content delivery/distribution networks; and
5. develop software libraries for common intermediate processing tasks.

We make the following five actionable conclusions for centralized facilities:

1. Energy is an overarching challenge for sustainability.
2. Data reduction is a fundamental pattern.
3. Radically improved resource management is required.
4. Both centralized and decentralized systems share many common software challenges and opportunities:
    a. leverage HPC math libraries for HDA;
    b. more efforts for numerical library standards;
    c. new standards for shared memory parallel processing; and
    d. interoperability between programming models and data formats.
5. Machine learning is becoming an important component of scientific workloads, and HPC architectures must be adapted to accommodate this evolution.

2. Introduction

This report, and the series of BDEC workshops that it summarizes, is part of the response from the HPC community to two major inflection points in the growth of scientific computing in the 21st century. The first marks the disruptive changes that flowed from the end of Dennard Scaling, c. 2004 (Dennard et al., 1974), which gave rise to the era of multi-core, many-core, and accelerator-based computing, as well as a variety of other complex and closely related problems of energy optimization and software complexity. The second, occurring nearly simultaneously, was the relatively rapid emergence (c. 2012) of "big data" and large-scale data analysis as voracious new consumers of computing power and communication bandwidth for a wide range of critical scientific and engineering domains. Because the BDEC community has its roots in traditional

HPC, the marked difference in the ways in which this community has struggled to absorb and adapt to these two watershed transitions, with varying degrees of success, provides essential context that informs the reading of this report.

To those who witnessed previous transitions from vector and shared multiprocessor computing, the response to the end of Dennard scaling was comparatively familiar and straightforward. Indeed, the dramatic effects of the end of Dennard scaling on processor and system designs were very much on the mind of the HPC leaders who gathered at the 2009 International Conference for High-Performance Computing, Networking, Storage and Analysis (SC09) to form what would become the IESP (Dongarra et al, 2011). Along with the European Exascale Software Initiative[1] and a parallel effort in Japan, these grass roots exascale efforts were the progenitors of the BDEC.

Although the challenges that confronted the IESP's vision of exascale computing were unquestionably formidable—orders of magnitude more parallelism, unprecedented constraints on energy consumption, heterogeneity in multiple dimensions, and resilience to faults occurring at far higher frequencies—they fit within the problem space and the ethos that defined traditional HPC. Participants in that planning effort worked successfully over the next 3 years to draft a technology roadmap leading to a common, high-quality computational environment for exascale systems (Attig et al., 2011; Dongarra et al., 2011). What the HPC community did not adequately anticipate was the eruption of interest in infrastructure and tools for doing large-scale data analysis in cloud computing environments. In the United States, this fact was revealed in the surprise that accompanied the Presidential strategic initiative Executive Office of the U.S. President (2015a, 2015b) that emphasized the importance of big data and HPC ecosystem *convergence.* This has been echoed in recent European Commission/EU communications on the "European Data Infrastructure" and the "European Open Science Cloud,"[2] where HPC has been completely "absorbed" into a broader "digital single market" and only appears as a component in this global system. In Japan, as well as in China, the latest national roadmaps are focused on combining HPC with artificial intelligence (AI) that itself is tightly linked to the big data revolution. The Japanese program provides for a 107.8 billion yen (US$1 billion) commitment over the next 10 years on inter-ministry AI-related research that encompasses big data, machine learning, and the IoTs. In China's 5-year plan for exascale systems, big-data analytics has been considered as a major application category. As clearly stated in the requirements for their exascale pilot systems, deep learning benchmarks are an important metric for evaluating the capability and efficiency of the proposed new hardware architectures.

The rapid transition of the earlier IESP roadmap activities to the BDEC effort shows how quickly the community recognized that high-end data analysis (HDA) and HPC needed to have equal status in an integrated computing research and development agenda. However, although the BDEC effort aims to expand the general mission of the IESP—to foster the codesign of software infrastructure for extreme-scale science drawing on international cooperation and supporting a broad spectrum of major research domains—subsequent experience has shown that adding the HDA dimension to the scientific computing problem space radically transforms it. As we argue in the following section, it raises a host of conceptual, political, economic, and cultural problems and places several of the existing paradigms and assumptions underlying traditional scientific computing and computational science into question.

2.1. Disruptive partitions in two current paradigms

The BDEC community's sustained examination of the changes wrought by the ongoing big data revolution has revealed at least two different—and somewhat orthogonal—ways that the cyberinfrastructure on which science and engineering communities depend is becoming—or has long been—bifurcated. One of these splits—between the traditional HPC approach and strategies that leverage or model commercial cloud computing—emerged early this century as an unexpected byproduct of the explosive growth of data associated with online commerce and social media. The second split—between stateless datagram delivery networks (e.g. the Internet) and stateful services provided by network-attached end systems (e.g. supercomputers, laptops, sensors, and mobile telephones)—is fundamental to the cyberinfrastructure paradigm that has been in use for well over three decades. Unfortunately, as we explain subsequently, the big data revolution has also made this entrenched bipartite cyberinfrastructure paradigm that is highly problematic. As reflected in the analysis subsequently, we believe that any planning for cyberinfrastructure convergence today has to take into account the partitioning of both types.

2.1.1. First partition: Recent split in software ecosystems.

The first split came to the attention of the BDEC community early in the process. The "two software ecosystems" diagram (Figure 1), which was introduced at the second BDEC workshop, quickly became emblematic of the ongoing bifurcation of the software ecosystems that were being deployed for data analytics and scientific computing. The computational science ecosystem developed and flourished over the course of roughly four decades (primarily) to increase the capabilities of scientists to model and simulate (i.e. to enable scientists and engineers to project, in more detail, the consequences of theories that had been—or could be—expressed mathematically). Meanwhile, the rapid growth of the data analytics ecosystem has occurred largely during the last 15 years. For the most part, however, it is not being developed by the scientific computing community to explore the rising flood of data from new instruments and sensor systems, but rather by an equally thriving group of academic and commercial software developers to

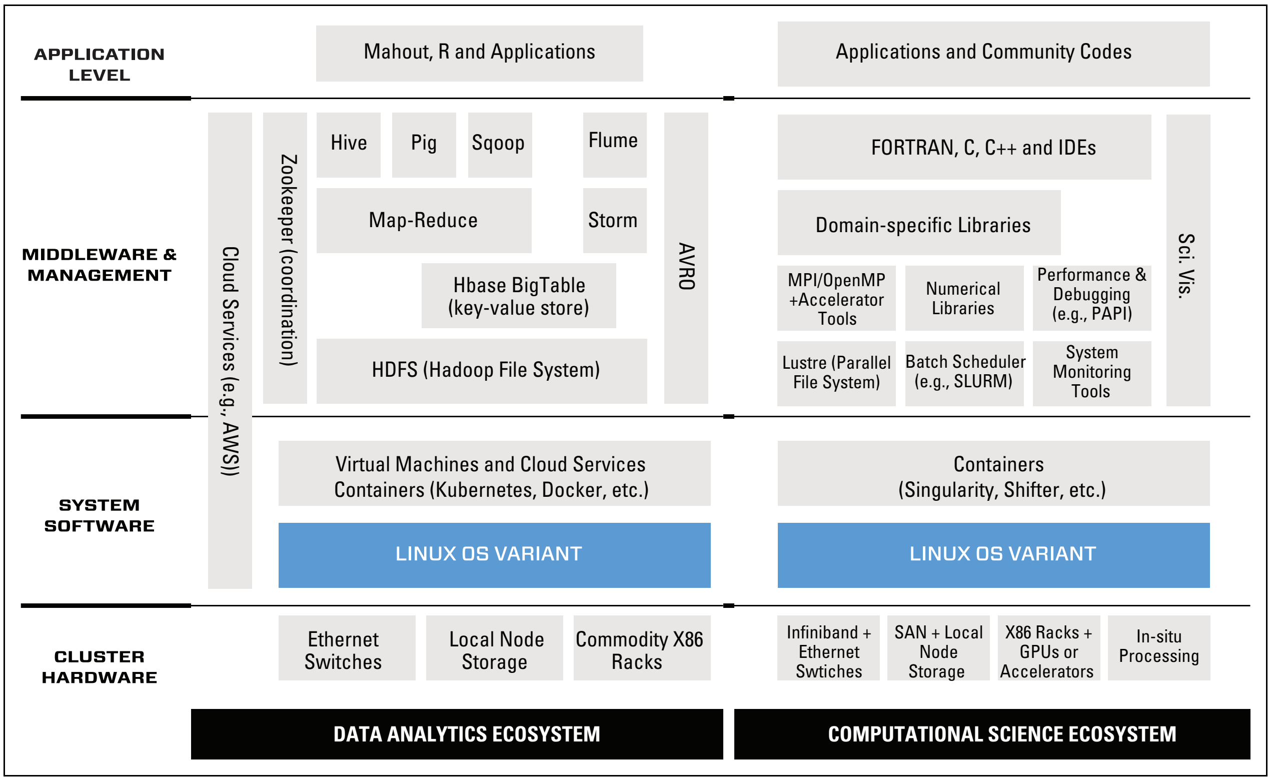
APPLICATION LEVEL
MIDDLEWARE & MANAGEMENT
SYSTEM SOFTWARE
CLUSTER HARDWARE
Mahout, R and Applications
Applications and Community Codes
Cloud Services (e.g., AWS)
Zookeeper (coordination)
Hive
Pig
Sqoop
Flume
Map-Reduce
Storm
Hbase BigTable (key-value store)
HDFS (Hadoop File System)
AVRO
FORTRAN, C, C++ and IDEs
Domain-specific Libraries
MPI/OpenMP +Accelerator Tools
Numerical Libraries
Performance & Debugging (e.g., PAPI)
Lustre (Parallel File System)
Batch Scheduler (e.g., SLURM)
System Monitoring Tools
Sci. Vis.
Virtual Machines and Cloud Services Containers (Kubernetes, Docker, etc.)
Containers (Singularity, Shifter, etc.)
LINUX OS VARIANT
LINUX OS VARIANT
Ethernet Switches
Local Node Storage
Commodity X86 Racks
Infiniband + Ethernet Swtiches
SAN + Local Node Storage
X86 Racks + GPUs or Accelerators
In-situ Processing
DATA ANALYTICS ECOSYSTEM
COMPUTATIONAL SCIENCE ECOSYSTEM

Figure 1. Different software ecosystems for HDA and traditional computational science. Credit: Reed and Dongarra (2015). HDA: high-end data analysis.

process the torrents of business, industrial processes, and social network data now being generated by consumer devices and the burgeoning IoT. The pace of change in the data analytics ecosystem is extraordinary, already rendering obsolete some of the elements in the figure above.

Thus, at least some of the major differences between the HPC and the HDA ecosystems—software development paradigms and tools, virtualization and scheduling strategies, storage models (local vs cloud/SAN), resource allocation policies, strategies for redundancy, and fault tolerance—can be accounted for by the fact that each evolved during a distinctly different phase of the ongoing digital revolution, driven by distinctly different optimization criteria.[3] For example, it can be reasonably argued that scientific "big data" has existed for more than a decade, but that it remained essentially "dark" (i.e. unavailable for analysis) until commercial cloud technology and content distribution networks began to provide broader access to the computing power and data logistics needed by the communities who wanted to analyze it.[4] By contrast, the HPC infrastructure model—a system of regional and national supercomputing centers connected together by high-performance research networks—was already fully mature at the beginning of the century and serving the needs of the modeling and simulation-centered parts of the scientific community relatively well.

But even the ultimate convergence of the HPC and HDA ecosystems, could it be achieved, would not help with the ongoing breakdown of the other, more basic paradigm, namely, the one in which networks only forward datagrams, while all other storage and computation is performed outside the network.

The problem is that much, if not most, of the explosive growth in data generation today is taking place in "edge environments" (i.e. outside of—and across the network from—both HPC data centers and commercial cloud machine rooms (Figure 2)). This includes not only major scientific instruments, experimental facilities, and remote sensors (e.g. satellite imagery), but even more importantly, the incredible welter of digital data generators with which the plans for "smart cities" and the IoT (Gorenberg et al., 2016) are replete. For example, a recent National Science Foundation workshop on the future of wireless networking concluded that the ubiquitous deployment of sensor technologies that are a standard element in such plans will "... generate massive data inflows [that produce] as much if not more data and network traffic than the World Wide Web" and will therefore "... reverse current loads, where most data is produced in the cloud and consumed at the edge" (Banerjee and Wu, 2013). Likewise, the authors of the 2017 European Network on High Performance and Embedded Architecture and Compilation report concluded that

> ... to stem the flood of data from the Internet of things, we must employ intelligent local data processing on remote devices that use minimal energy.... This may well require

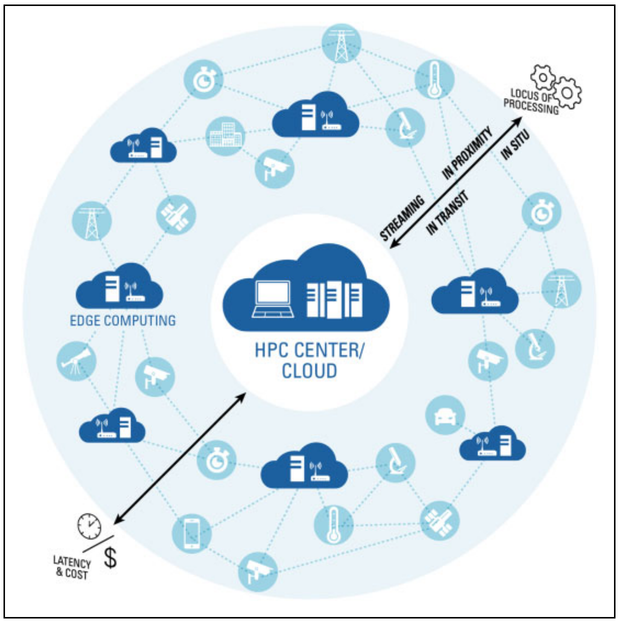

Figure 2. Current problem of data logistics: The highest concentrations of computing power and storage are in the "center" (i.e. in commercial clouds or HPC centers), but much of the rapid increase in data volumes and the dramatic proliferation of data generators is occurring in edge environments, where the processing and storage infrastructure needed to cope with this rising flood of data is *ad hoc* and under provisioned at best. HPC: high-performance computing.

> us to break away from the traditional von Neumann architecture and to rethink device technology. (Duranton et al., 2017)

Thus, this remarkable reversal of direction of the data tide, which turns the familiar "last mile problem" into a multidimensional "first mile problem," represents a challenge for which neither cloud-based HDA nor center-based HPC have a solution. In fact, explosive growth in data generation in edge environments seems to clearly indicate that revolutionary innovation in distributed computing systems is becoming an increasingly urgent requirement (Calyam and Ricart, 2016; Chen et al., 2014; Fox et al., 2016; Nahrstedt et al., 2017). As argued in the following section, we believe this represents the breakdown of the bipartite cyberinfrastructure paradigm that has been dominant for nearly three decades, making the problem of convergence substantially more complex and momentous.

2.1.2. Second partition: Inherent split in the legacy paradigm. Some historical perspective is required to understand the other "divergence" or paradigm breakdown that the proliferation of data generators seems to be causing. If one were to try to mark the year in which the two parts of the dominant research cyberinfrastruture paradigm of the last 25 years—TCP/IP and Unix/Linux—were first fused together, 1989 would make a very plausible candidate. In June of that year, the Berkeley Software Distribution (BSD) of Unix, including a Defense Advanced Research Projects Agency-approved version of the TCP/IP protocol stack, was made freely available under an open-source license. Their combination in that form was especially well received by the research community:

> As the Internet evolved, one of the major challenges was how to propagate the changes to the software, particularly the host software . . . . [T]he incorporation of TCP/IP into the Unix BSD system releases proved to be a critical element in dispersion of the protocols to the research community. Much of the [computer science] research community began to use Unix BSD for their day-to-day computing environment. Looking back, the strategy of incorporating Internet protocols into a supported operating system for the research community was one of the key elements in the successful widespread adoption of the Internet. (Leiner et al., 2009)

Although TCP/IP and Unix/Linux became complementary parts of one integrated package, they supported two different software ecosystems for two separate but complementary infrastructures. The former laid the software foundation for a globally scalable data network that offered one fundamental service: unicast datagram delivery to move/copy data from one location (buffer/host/end system) to another. The latter, by contrast, was designed to provide an application interface to end systems of all kinds, ranging from personal computers to "middle boxes" to supercomputers; its purpose was to enable applications and services that required the full complement of basic resources—processing power, storage, and networking. But to understand our current dilemma, we should recognize at least three additional factors that helped foster the rapid spread of this composite paradigm through the global research community.

1. *Both components were open source, public domain, and designed to support broad experimentation and rapid evolution*. The TCP/IP-Unix/Linux platform was introduced when the highly competitive (not to say ferocious) battles between different proprietary computing strategies (e.g. mainframe, minicomputer, and PC), desktop operating systems (e.g. MS-DOS, Windows, OS2, and macOS), and local area networking technologies (e.g. Novell, Apple, DECnet, and systems network architecture) were still ongoing. Avoiding high costs, loss of control of one's own tools, barriers to collaboration, and other perils of "vendor lock in" made the open source and public domain character of this new paradigm especially attractive. At the same time, as we discuss in Section 4.1, both halves of the dominant paradigm were designed in a way that facilitated freedom of innovation and speedy growth, making it especially well suited to the inclinations and the aptitudes of the science and engineering communities.
2. *Each component embodies a different model of resource sharing*. Finding ways to share

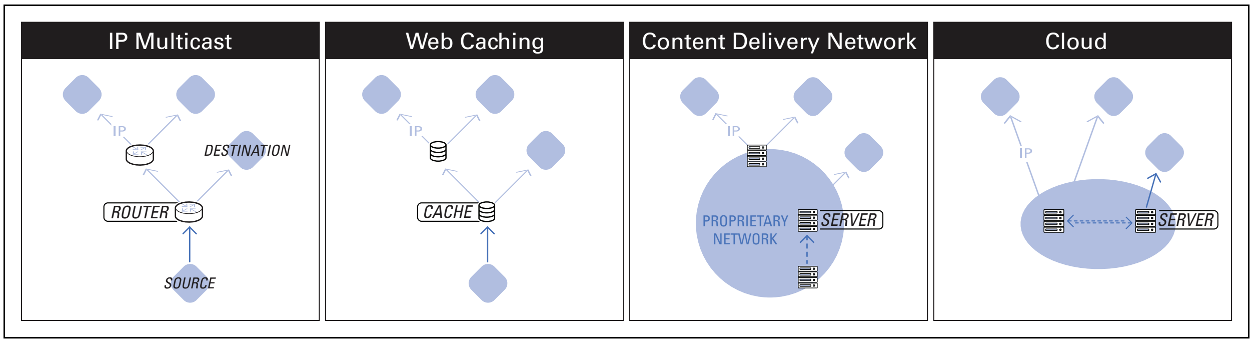

Figure 3. Evolution of the network to deal with growth in data volumes and rapid fluctuations in their popularity. The need to be able to use storage and computation at intermediate nodes (e.g. for asynchronous point-to-multipoint transfer) stimulated the development of content delivery networks and clouds with proprietary networks and server replication using standard IP to reach clients at the network edge. Credit: Beck et al. (2017).

computational resources has long been a fundamental part of the cyberinfrastructure problem, and time-sharing operating systems and digital data networking have long been essential parts of the solution. But the two parts of the dominant paradigm under discussion support resource sharing in different ways and with different consequences for their overall design (Bassi et al., 2002). In the context of large, shared systems, Unix/Linux was designed to allow the sharing of valuable processor, storage, and other system resources with authentication and access controls that created a secure environment supporting free sharing of most resources. Enforcement of storage quotas in Unix was a later adaptation necessary in more highly shared/public data center environments, and CPU metering and rationing has been implemented in specialized high-value supercomputing systems. By contrast, the TCP/IP was designed to facilitate communications among an enormous variety of devices used by a global community of indefinite size. With maximum deployment scalability as its goal, it shares available bandwidth in limited lightweight allocations that are easy to access and that typically lack significant admission or accounting controls. The incredible power of this model of resource sharing, especially in terms of its network effects, is manifest.

3. *Each component has a different approach to state management*. Perhaps the most significant asymmetry between the components of the dominant paradigm lies in the different ways that they handle process state. The Internet is based on a stateless (or virtually stateless) datagram delivery model, where the intermediate nodes of the network are stateless packet switches (Clark, 1988). By contrast, the end systems that attach to the Internet, ranging from smart phones to supercomputers, are typically general purpose computing devices that regularly have to manage a great deal of computational or process states (much of it hidden), and a wide range of mechanisms have been implemented to serve that purpose; these end points manage all the state information required to use the network. The Internet community has traditionally held this design, which keeps the core of the network simple and passive by keeping services other than datagram delivery (e.g. a storage service for caching and prefetching) out of network layer of the TCP/IP stack. This design has been essential to the remarkable scalability that the Internet has demonstrated. And so long as applications and services could be factored into two phases—using the datagram service to move data between end systems and then applying end system resources to achieve complex application or service goals—this symbiotic union of stateless and stateful systems was able to support an immense range of applications.

But problems with this bipartite paradigm, especially with respect to distributed state management in the wide area, have been both evident and growing for more than 25 years. In particular, lacking any general ability to do state management at its intermediate nodes (e.g. lacking native asynchronous point-to-multipoint capabilities), the Internet has long been plagued by bottlenecks or “hotspots” for high-volume and/or high-popularity data traffic. Since data volumes have been growing at exponential rates over the same period, users and providers have been forced to search for new workaround strategies on a recurring basis. Indeed, the Internet community has seen an evolving series of such workarounds, from collateral storage services in the 90s (e.g. FTP mirror sites, web caches, and hierarchical web caches) to full blown server replication in CDN and commercial cloud infrastructure beginning in this century (Figure 3). Today’s CDNs and clouds have more or less followed this path to its logical conclusion, using private or non-scalable mechanisms to implement internal communications among logically centralized, but physically distributed, nodes or machine rooms while using the Internet to implement selected parts of this scheme and to provide for end-user access.

Similarly, major HPC application communities have long since moved their high-volume traffic to non-TCP-friendly data transfer methods using dedicated bandwidth on over-provisioned research networks.

Unfortunately, we have entered an era—the era of big data—when such workaround strategies seem to have reached their limits. The cause of this exhaustion, as we argue earlier, is the unprecedented explosion in data generation in edge environments. The burgeoning torrents of data that are flowing from highly diverse and widely distributed sources originate in environments which, for the most part, lack the capacity to process or manage the logistics of such immense flows. Whether because of sheer volume and flow rate, or because of application-specific latency issues, the conventional strategy of backhauling all data across a fast link to a cloud service provider or an HPC data center (Figure 8) is no longer a viable option for many applications. Hence, there is intense commercial and research interest in "fog" and "edge" computing infrastructures (Bastug et al., 2014; Bonomi et al., 2012; Hu et al., 2015; Satyanarayanan et al., 2009; Wang et al., 2017). For perhaps obvious reasons, we view the creation of a future-defining DSP that meets the fundamental requirements of the scientific community to be—at best—a difficult problem full of unknowns and fraught with economic and political obstacles. More immediately, this assessment has influenced the structure of this report, as we describe next in our document overview.

2.2. Pathways overview: Prospects for convergence at three different levels

As noted at the outset of this report, the transition from the IESP to the BDEC project forced a radical shift in perspective on the part of HPC community participants. In contrast to the "big machine" focus of traditional HPC, the center stage in the era of big data has to be given to the many unsolved problems surrounding wide-area, multistage workflows—the diverse patterns of when, where, and how all that data are to be produced, transformed, shared, and analyzed. Consequently, the challenges involved in codesigning software infrastructure for science have to be reframed to fully take account of the diversity of workflow patterns that different communities want to create. All of the imposing design and development issues associated with creating an exascale-capable software stack remain; however, the supercomputers that need this stack must now be viewed as the nodes (perhaps the most important nodes) in the very large network of computing resources required to process and explore gigantic rivers of data.

As discussed previously, the dominant cyberinfrastructure paradigm that has been the foundation of such workflows for decades is now eroding—if not collapsing—under the onslaught of this growing data deluge. Unfortunately, this fact requires the BDEC community to find a new way to express its results. IESP participants were able to successfully produce an influential technology roadmap (Dongarra et al., 2011) for creating a software stack to support science applications on extreme-scale platforms. However, the IESP had the advantage of targeting a shared goal that was different in scale, but not different in kind, from less extreme goals that had been achieved before. For BDEC, the shared goal is not so clear. Arguably, the main cyberinfrastructure challenge of the big data era is to adapt or replace the legacy paradigm with a new type of DSP—one that combines computing, communication, and buffer/storage resources in a data processing network that is far more integrated than anything hitherto available. But since there is no widely agreed upon model for this platform, traditional technology road mapping techniques seem to be inapplicable.

Instead, we suggest that the development of a community-driven "shaping strategy" (Hagel and Brown, 2017; Hagel et al., 2008) would be a far more appropriate goal to pursue. A shaping strategy is a plan for changing the infrastructure/ecosystem of a market, industry, or community by proposing a well-defined concept of a technical platform that can support many kinds of applications or enterprises, and combining that platform with an inspiring vision of the mutually beneficial future that could be created through its widespread adoption and use. By offering a range of possible incentives to all the stakeholders who would benefit from such convergence, shaping strategies seek to coordinate and direct the creativity and energy of participants who might build on this platform in a way that leverages network effects and produces positive externalities. Shaping strategies are thought to be especially appropriate when the absence or breakdown of an established infrastructure/ecosystem paradigm has disrupted or unsettled some large field of endeavor. With the era of big data, this is precisely the situation that the scientific community now confronts.

Against that background, the reflections of BDEC participants over the course of successive workshops have suggested that the prospects for integration of technological infrastructures and research ecosystems need to be considered at three different levels, or from three different perspectives. The three major sections of this document, in turn, focus on one of these levels, as listed in the following.

1. *Opportunities for convergence of research applications and workflows (Section 3)*: We begin at the level of applications and workflows (i.e. composite applications) for two reasons. First, operating on the general principle that the goal of any infrastructure is to serve its users, it seems appropriate to begin by examining the forms of inquiry, new and old, that a converged infrastructure for research is supposed to support. Second, pushing back against the popular idea that HDA represents a distinctly new paradigm of scientific methodology, we argue that HDA actually represents the computerization of two phases of the classic model of the scientific method, which

had heretofore been far less digitally empowered. Accordingly, we offer examples in which HPC and HDA applications are being composed in workflows that embody the entire inference cycle of scientific inquiry.

2. *Issues in the creation of a converged infrastructure for large-scale, logically decentralized facilities (Section 4)*: This section offers one account of some problems associated with creating a converged infrastructure for distributed edge environments, one which can be deployed—in a scalable manner—to meet the data processing, communication, and storage requirements of massive data workflows in the wide area. Developing such a "next-generation Internet" for the big data era in science and engineering is fraught with challenges in various dimensions. There is, however, a great deal of excitement and activity in this part of the infrastructure convergence space under the heading of "fog" and/or "edge" computing, but there are also very high levels of uncertainty. Consequently, although we survey and discuss some of the prominent ideas and strategies, our review is by no means exhaustive.
3. *Opportunities for convergence of infrastructures of large, logically centralized, resource-rich facilities (Section 5)*: The third major section of this report focuses on major facilities that execute large-scale simulations and models or that perform large-scale data analytics. We refer to such systems as "logically centralized" because, whether or not they are actually physically distributed, they present themselves as being under the control of a single administrative domain, and users are required to interact with them in that form. Such facilities are already supporting some of the converged applications and workflows previously discussed, and some convergence at this level can and is already occurring.

Drawing on discussions at the BDEC workshops,[5] we begin each of the major sections by suggesting some points of plausible consensus that are intended to provide common context for cyberinfrastructure planning, especially for the development of a community-driven shaping strategy. These points of convergent opinion include an integrated conception of the general process of scientific inquiry, the overarching issue of energy efficiency as critical for sustainability, the new importance of questions of "data logistics," and prevalence of the "hourglass" model as a paradigm of good infrastructure/ecosystem design. We have assigned these assumed pieces of common context to the major sections that seemed most appropriate, but admittedly some are of a general nature and certainly apply in other sections as well.

Since the idea developing a community-driven shaping strategy for a new DSP for science was arrived at late in the BDEC reporting process, we do not describe even a straw man version of such a plan here. Nonetheless, we believe that the various observations, arguments, and conclusions we record in what follows should feed directly into the dedicated community effort that will be required to develop such a plan. For example, the argument for focusing on the careful design of the "spanning layer" at the waist of the hourglass model (Section 4.1) dovetails perfectly with the need to define an attractive "shaping platform" that many stakeholder communities will build on and use, and such a platform definition is one of the primary constituents of a successful shaping strategy. Likewise, we believe that analyzing—and building on—the application-workflow exemplars of the kind we describe and classify in Sections 3.2 and 3.3 and is an essential step in developing the "shaping view" that successful strategies must have in order to incentivize participation and motivate adopters and users. Accordingly, as we note in Section 6, we offer our final conclusions and recommendations with the intention of preparing the groundwork for a new community effort to develop the kind of shaping strategy for future cyberinfrastructure that the scientific community must have in order to thrive in the ongoing data revolution.

3. Emerging convergence in large-scale scientific applications

Despite the manifest challenges of a converged cyberinfrastructure for multidisciplinary research, the scientific and engineering opportunities are compelling. Many science communities are driven by a combination of computing tasks and managing large-scale data volumes resulting from data flows that range from continuous to sporadic. The associated computations may come from large-scale workflows that orchestrate simulations or from different stages of large-scale analysis pipelines for data generated by simulations, experiments, or observations, including the transitions between these stages and provenance tracking. Communities that would benefit substantially from application and workflow convergence are abundant, spanning all of science and engineering. Examples include (1) multiscale materials science; (2) integrative environmental modeling and analysis; (3) astronomy and cosmology; (4) aerospace; (5) autonomous vehicles; (6) weather and climate prediction; (7) smart cities; (8) health and biomedicine; and (9) exploration geophysics and seismology. Several of these application communities are highlighted subsequently in our examples of convergence.

It is important to note that when we talk about "applications" and "workflows" in this context, we are not talking merely about isolated software tools or application codes but rather about complex (and changing) configurations of software, hardware, and data flows that support long-running science and engineering campaigns and research practices. For this reason, we prefer to structure the discussion subsequently around three computational archetypes that provide broader categories than specific traditional formulations (e.g. partial differential equations and principal component analysis) or approaches (e.g. implicit method and randomized singular value decomposition). These three archetypes will instead furnish a convenient grouping of collaborations by the users' "culture."

To provide the intellectual context for this analysis of application-workflow level convergence, we begin by describing a unified model of an inference cycle for the process of scientific inquiry, pushing back against the popular idea that computational science and data science represent disparate paradigms in the search for new knowledge (Section 3.1). After presenting the rationale for our division of application-workflow archetypes, we illustrate them with exemplars that *follow the data*, since it is the data that carry the costs (time, energy, and human labor needed to produce and manage the data). We address three modalities of data provenance: (1) data arriving from the edge (often in real time), never centralized; (2) federated multisource archived data; and (3) combinations of data stored from observational archives with a dynamic simulation. These exemplars make it evident that the concept of workflows has become pivotal for understanding the convergence paradigm between data and simulation. Notably, however, we leave out of this account—but discuss later (Section 4)—many issues surrounding the data logistics infrastructure that would, and will be, needed to support our exemplar workflows.

3.1. Common context: A shared model of scientific inquiry for infrastructure planning

One notable objection to pursuing software infrastructure convergence for the scientific community draws on the idea that, along with traditional forms of experiment and theory, the emergence of digital information technology (IT) and the explosive growth in computing power have combined to produce two distinctly new paradigms of how science can be done: (1) modern computational science and (2) data-intensive science. Commenting on the latter, Turing award winner Jim Gray, who apparently originated this way of narrating the transformation of science in the digital age, asserted that, "The techniques and technologies for such data-intensive science are so different that it is worth distinguishing data-intensive science from computational science as a new, fourth paradigm for scientific exploration" (Hey et al., 2007). Sorting sciences into these different methodological boxes has become conventional wisdom in the HPC community, and this supposition, in turn, makes it plausible to argue that the HPC and HDA software ecosystems have separated because each is adapted to the peculiarities of a very different paradigm of scientific research. If this were true, it would seem to put a significant obstacle in the path of software ecosystem convergence.

A recent white paper from the Computing Community Consortium addressed this question, presenting a generic account of the "scientific process" that accommodates a

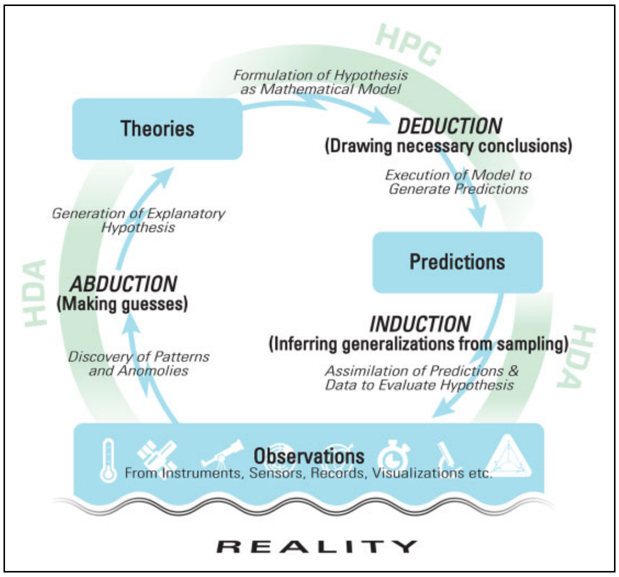
HPC
Formulation of Hypothesis as Mathematical Model
Theories
DEDUCTION
(Drawing necessary conclusions)
Execution of Model to Generate Predictions
Generation of Explanatory Hypothesis
Predictions
HDA
ABDUCTION
(Making guesses)
INDUCTION
(Inferring generalizations from sampling)
HDA
Discovery of Patterns and Anomolies
Assimilation of Predictions & Data to Evaluate Hypothesis
Observations
From Instruments, Sensors, Records, Visualizations etc.
REALITY

Figure 4. The inference cycle for the process of scientific inquiry. The three distinct forms of inference (abduction, deduction, and induction) facilitate an all-encompassing vision, enabling HPC and HDA to converge in a rational and structured manner. HPC: high-performance computing; HDA: high-end data analysis.

more unified point of view (Honavar et al., 2016). Following that suggestion, and for the moment leaving aside some important socioeconomic aspects of the problem, Figure 4 shows a simplified version of the key logical elements of that model.[6] It expresses the classic view that the scientific method is a complex inferential process that seeks to improve our predictive understanding of nature by building on a foundation of thorough and carefully controlled observation. The three distinct forms of inference it identifies, (1) abduction (i.e. guessing at an explanation), (2) deduction (i.e. determining necessary consequences of a set of propositions), and (3) induction (i.e. making a sampling-based generalization), are the chief elements in the "logic of discovery" first articulated by the American scientist and polymath Charles S. Peirce (Bellucci and Pietarinen, 2017).

Likewise, in a more contemporary treatment of the same topic, Richard Feynman's account of the logic of discovery for physical laws can be summarized as having essentially the same three steps or stages (Feynman, 1967).

1. We guess at a law that would explain what is currently inexplicable.
2. We derive the consequences of the law that we guessed.
3. We make further observations to see if the consequences predicted match the reality we find.[7]

On this analysis, all three steps taken in combination are required to increase our predictive understanding of the world (i.e. to really make our knowledge grow).

Now, if we accept this model as a plausible hypothesis, then we can draw at least two conclusions that are directly relevant to the problem of cyberinfrastructure convergence.

First, by viewing the transformation of science in the era of digital computing through the lens of this model, we can see that there is an alternative way to account for these changes—one that does not require us to posit substantially new paradigms of scientific inquiry. Namely, rather than forking new methodological branches, one can explain the more rapid emergence of computational science, and the slower and later emergence of data-intensive science, by examining the factors that made it possible to apply vast amounts of computing power to the deductive stage of the inference cycle far earlier than to the abductive and inductive stages.

As conventionally defined, computational science primarily covers the deductive part of the inferential process: Computer-based simulation and modeling shows what a given theory, expressed mathematically, predicts for a given set of input data. This could equally be viewed as mapping complex models to data. By the middle of the 20th century, when the power of microprocessors began to follow their amazing exponential arc upward, many physical sciences already had well established mathematical models of their target phenomena but had access to only enough computing power to solve them for relatively small problems and/or for cases where inherent uncertainties were not take into account. The impact of Moore's law was, therefore, immediate and explosive.

By contrast, the gathering and analysis of observational data have always been the focus of the abductive and inductive stages of scientific inquiry. Whether data are being explored to discover novel patterns or anomalies that might initiate new inquiry or being carefully examined for some predicted phenomena that would test (i.e. further confirm or potentially falsify) some well-formulated theory, data analysis strives to put the reasoning of scientists and engineers into relatively direct contact with the realities they are trying to understand. When the data are digital, such analyses obviously require computational power. But in a mirror image to the HPC situation, so long as the data volumes remained tractable, concerns about the amount of computational power required to do the necessary data analysis could be sidelined: either a small cluster could be used to analyze data where it was collected (e.g. at the site of the instrument), or, when necessary, the raw data could be transferred via a high-performance network to a supercomputing center for analysis.

Of course, since the turn of the century, these flows have become progressively more intractable, as remarkable improvements in sensor technology and other data gathering capabilities produce exponential growth in research data flows. This data deluge was spotted on the horizon early in this century's first decade (Hey and Trefethen, 2003), with projects like the Large Hadron Collider (LHC) serving as a harbinger of unprecedented increases in data rates across a wide variety of fields. Yet despite the burgeoning quantities of data that needed to be processed, the big data revolution was not really launched until the end of the decade, when the explosion of consumer/customer usage being collected and utilized by online companies (e.g. Google, Amazon, and Facebook) motivated the build out of massive, private, cloud computing infrastructures to store and process it all.

As we discuss in more detail in Sections 5 and 4.2, this proliferation of distributed data sources means that scientific cyberinfrastructure design must focus as never before on issues of workflow and data logistics, thereby covering the full path of data use from its collection to its use as a decision-making aid. Moreover, this focus is required in no small part by the new efforts, in line with the cycle of inquiry illustrated in Figure 4, to synthesize elements of HPC and HDA in new application-workflow hybrids. In short, the concept of a scientific application, familiar from the heyday of traditional computational science, is being gradually redefined. To set the stage for our application "exemplars," we first briefly discuss the new perspective on application workflows that is now emerging.

3.2. Classification of application-workflow archetypes

A good first step in understanding any new area typically consists of working out some rough classification of the different types of objects that make it up. It is no surprise, then, that every BDEC meeting has dedicated substantial amounts of effort to categorize the various data-intensive, compute-intensive, and hybridized applications and application workflows. Viewed from a relatively traditional standpoint, the categories or classes most frequently discussed included simulations, database and data management problems, scientific data analysis, and commercial data analytics. The last three have many examples in the National Institute of Standards and Technology's (NIST's) collection of 51 big data use cases,[8] and simulations are illustrated in many HPC benchmarks (e.g. the NASA Advanced Supercomputing Parallel Benchmarks) and the Berkeley Dwarfs. The ways in which such applications are driving new software infrastructure developments was a frequent point of interest. For example, we noted that MapReduce was originally introduced to parallelize basic database operations as seen in Apache Hive, but Andrew Ng from Stanford[9] observed that many machine learning algorithms exhibited the "summation form" and could be parallelized with MapReduce. It was later noted that this could be optimized with so-called Iterative MapReduce—a model supported by Apache Spark and Flink.

One BDEC white paper by Fox G, et al. (2016) noted that the comparison of simulation and big data problems can be made more precise by distinguishing data and models for each use case and carefully making model-to-model and data-to-data comparisons and not confusing them. This article introduced a common set of 64 properties or facets, divided into four views, that could be used to characterize and compare use cases within all the application classes defined earlier. As an example, the first "problem architecture" view includes four very common facets

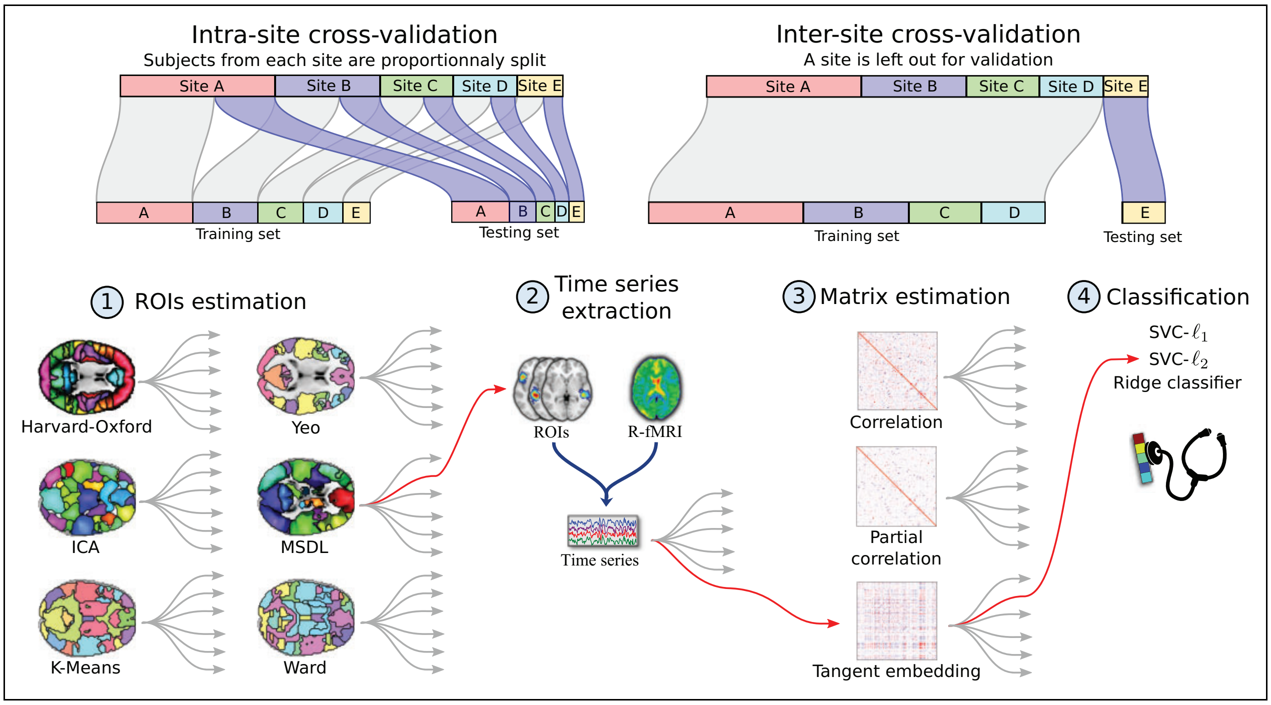

Figure 5. A functional magnetic resonance imaging analysis pipeline with cross-validation schemes. Regions of interest are deduced from predefined (Harvard-Oxford, Yeo) or data-driven atlases (*K*-means, independent component analysis (ICA), maximum distance secant line (MDSL)) and result in dimension reduction. Extracted time series are fed into a correlation matrix estimator that finally feeds into the classifier that distinguishes between autism spectrum disorder and typical controls. Credit: Abraham et al. (2017).

describing the synchronization structure of a parallel job: (1) pleasingly parallel, (2) MapReduce, (3) Iterative Map-Collective, and (4) Map Point-to-Point. Another important problem architecture facet is "Map Streaming," seen in recent approaches to processing real-time data. Other well-known facets in this view are single program, multiple data and bulk-synchronous processing. An execution view includes relevant facets like model size, variety, data volume, velocity, and veracity. The third data view includes storage and access models and relevant information on IoT, geospatial information systems, and need for provenance. The final processing view includes algorithm characterizations like optimization, alignment, N-body method, spectral method, and so on. One can look at use cases, assign facets, and use this information to understand commonalities and differences and how this is reflected in the hardware and software.

Complicating this algorithmic perspective of the classification problem is the fact that—from a strictly data/workflow point of view—data analysis is usually an iterative process. The model is often unknown beforehand, since principled statistical model selection is not solved in a general setting. Moreover, data must be interactively understood, and often "debugged," that is, we need to find anomalies in the data and detect wrong assumptions on the data. In this context, schemes of distributed computing are evolving, but users often do not rely on distributed computing, hence the data analysis operations are not expressed in a language that renders distributed computing suitable. Typical code is a succession of operations written in an imperative way. Here, it is useful to think of the computing as a *dataflow* graph, since access to data (and related bandwidth) can often be a bottleneck.

Three dataflow schemes are often encountered:

1. Extract, transform, and load (ETL): Pipelines that apply transformations known beforehand often encountered in data-input preprocessing. This is already done in distributed environments (e.g. with Hadoop). It can lead to data-parallel pipelines.
2. Database-centric distributed computing, as with Spark: heavily out-of-core, simple operations. Operations are then specified in a data-oriented language, often not imperative, such as the structured query language (SQL) or Scala. Such a language requires more training to perform complex processing pipelines.
3. Machine-learning on streams: stochastic gradient descent, as in deep learning. Here, distributing the model fit needs to be tackled with a different paradigm. However, applications often require fitting many models independently, which is embarrassingly parallel.

A typical dataflow is illustrated in Figure 5, where several hundred gigabytes of data from the autism brain imaging data exchange are treated and analyzed to extract the most predictive biomarkers. It should be noted that I/O is critical in this pipeline, and data reduction dominates the

Table 1. A 3 × 3 matrix showing beneficial interactions in the six off-diagonal locations between large-scale simulation, data analytics, and machine learning.

	To large-scale simulation	To data analytics	To machine learning
Simulation provides:	—	Physics-based "regularization"	Data for training, augmenting real-world data
Analytics provides:	Steering in high-dimensional parameter space; in situ processing	—	Feature vectors for training
Learning provides:	Smart data compression; replacement of models with learned functions	Imputation of missing data; detection and classification	—

computational costs. The results obtained in the study by Abraham et al. (2017) required 2 months of computation on a high-end cluster with a dedicated, local network attached storage housing the data. Obviously, this does not scale, and the need for more efficient data communication and reduction, as well as convergence with an HPC environment, are clearly needed here.

Although it is somewhat natural to try to classify applications either (most commonly) on the basis of the algorithms they share or (more rarely) on the basis of the workflow patterns in which they are embedded, the fact that motivations abound for converging large-scale simulation and big-data workflows that are executed largely independently today suggests that we should try to classify *application workflows* that represent the combination or interaction of these two dimensions. The interaction matrix shown in Table 1 describes some benefits that are immediately available to each paradigm and includes the rise of machine learning. For this purpose, we distinguish workflows that are independent of the data and are throughput-oriented from those that are learning-oriented and transform themselves by means of the data. Lumping together many varieties of batch-oriented scientific simulations, we therefore consider three types of workflows—each of has its own niche of applications, its own software stack, and its own user community.

As indicated in the first column of the matrix, to take advantage of advances in analytics and learning, large-scale simulations should evolve to incorporate these technologies in situ, rather than as forms of postprocessing. This potentially reduces the burden on file transfers and on the runtime I/O that produces the files. In some applications at major computing centers (e.g. climate and weather modeling), I/O is well documented (Luu et al., 2015) as consuming more resources than the computation itself. In situ data analytics allows the simulation to avoid writing data that is needed only to advance the simulation, though this does not apply to post examination. However, the benefits go far beyond this and into the realm of steering. Smart steering may obviate significant computation—along with the I/O that would accompany it—in unfruitful regions of the physical parameter space, as guided by the in situ analytics. In situ machine learning offers additional benefits to large-scale simulation, beginning with smart data compression, which complements analytics in reducing I/O and storage use. Beyond assisting with the performance of the known implementation, machine learning has the potential to improve the simulation itself. This is because many simulations incorporate empirical relationships like constitutive parameters or functions that are not derived from first principles but are tuned from dimensional analysis, intuition, observation, or other simulations. For such components, common in multi-scale and multi-physics models, machine learning in the loop may ultimately (or already) be more effective than the tuning of expert scientists and engineers. Such learning would "probe" a simulation, necessitating an in situ process.

Next, turning to the first row of the interaction matrix, simulation potentially provides significant benefits to analytics and learning workflows once the software and hardware environments converge. Data science models have had limited penetration in systems representing complex physical phenomena. Theory-guided data science (Karpatne et al., 2016) is an emerging paradigm that aims to improve the effectiveness of data science models by requiring consistency with known scientific principles (e.g. conservation laws). Theory-guided data science is gaining attention in applications like turbulence, materials, cosmology, and climate modeling. It is analogous to "regularization" in optimization, wherein nonunique candidates are penalized by some physically plausible constraint (such as minimizing energy) to narrow the field. In analytics, among statistically equally plausible outcomes, the field could be narrowed to those that satisfy physical constraints, as checked by simulations. Simulation can also provide training data for machine learning, thereby complementing data that are available from experimentation and observation. A traditional advantage of simulation over experiment is emphasized in this context: Some interesting regions of the parameter space—physically realizable in principle or representing an extreme limit that is not realizable (e.g. flow with zero viscosity)—can be opened up to a learning algorithm.

The other two nontrivial elements of the matrix are between the two workflows within big data. For machine learning, analytics can provide feature vectors for training. In return, machine learning can provide analytics with the imputation of missing data and functions of detection and classification. For any of these "off-diagonal" fruits to be harvested efficiently, the corresponding workflows should be cohosted. The scientific opportunities are potentially important enough to overcome the enormous inertia (and spur convergence) of the specialized communities that have gathered around each of these tremendously successful paradigms.

3.3. Exemplars of application-workflow convergence: Science at the boundary of observation and simulation

Over the course of successive BDEC workshops, it quickly became apparent that application communities would be leading the drive toward convergence. Given the "converged" model of scientific inquiry described earlier (Section 3.1), this is more or less to be expected. Indeed, recent developments show that it is already happening. One contrast with commercial cloud computing that these developments highlight is based on the fundamental difference between the data generated by human interaction with computers (such as social media and other commercial sites) and data collected from sensors and observations. The latter is governed by some underlying physical process even if it is a nondeterministic one or one that has not been understood yet. The analysis performed on scientific data may, sometimes, resemble that performed on commercial data—that of drawing inferences without a hypothesis or conjecture. However, such analysis is limited to instances where there is little or no prior knowledge about the underlying physical processes. In those instances, inferences help to formulate hypotheses and ultimately the theory. More often some theory exists, and the observations collect data that are used to prove or falsify a derived prediction that helps refine the theory. Examples of the different, rough, and overlapping patterns with these research workflows include the following:

- *Patterns of interplay between simulation and data analysis*: High-energy physics has several examples where interplay between simulations and data analysis plays a critical role in scientific insight. One example is large experiments like European Organization for Nuclear Research (CERN's) LHC, which currently generates about 30 petabytes of data per year, which is expected to grow to 400 petabytes in a decade. Experiments need support from theory which is provided by the simulations. Confronting data generated from the simulations with the observational data helps tune both the model and the experiment design (see the inference cycle in Section 3.1). Another example of synergy between simulations and experimental data is in determining the placement of sensors in an experiment. This has been used in designing high-energy density physics laser experiments, where simulations can predict where to place the measuring instruments to have the best chance of detecting features being formed. Without this interaction between simulation and placement of instruments in experiments, entire features can be missed and have been known to have done so in the past. Use of simulations to determine the placement of instruments or sensors is now making its way into many areas, including environmental and climate observations.
- *Patterns of alternating data generation and consumption*: Another interesting class of workflows are ones that alternate between data generation and data consumption. These kinds of workflows occur in experimental and observational validation and uncertainty quantification. Here, the data volumes and patterns vary throughout the pipeline. The experimental data may require fine-grained "event" style analysis where data pipelines can be complex and need to be run many times. Demands on I/O can vary due to the nature of the simulations and the fine-grained nature of the outputs (many small files). An example of such complex workflows is in dark energy research where understanding type Ia supernovae is critical. A typical workflow requires running a model simulation, postprocessing it, and running it through another computational stage that results in generating the spectra and the intensity curves. On the other side, similar artifacts are generated for comparison from the observational data through various stages of analysis. An important challenge faced in this kind of workflow is data movement. Large volumes of data are generated both from observations and from computations—typically at different sites. It is not just the processing, but also the curation, migration, and archiving of data that becomes the scientists' concern. Good science requires provenance of data involved in scientific discovery, which makes the task of managing the data pipelining highly resource intensive for many scientists.
- *Patterns of multi-messenger scientific inquiry*: Several discussions at the BDEC workshops highlighted the way in which the application of different observational modalities, which can now be simultaneously applied to the same object, promises to dramatically increase our predictive understanding of those objects. Generalizing the idea "multi-messenger astronomy" (Wikipedia, 2017c), we refer to this application-workflow pattern as "multi-messenger scientific inquiry."[10] The ability to predict the effects that result from interactions between physical or biological systems, or to predict patient outcomes or treatment responses, hinges on the accurate multi-resolution characterization of the objects and systems involved. Happily, the dramatic reduction in costs required to obtain imaging and sensor data has made it increasingly feasible to capture the necessary data.

The general approach, which cuts across the many relevant application domains, is to carry out and synthesize the diverse data streams that record different types of sensor observations of the same object, or set of objects; and in many such scenarios, analysis and integration of data streams from different types of sensors is coupled with simulations. The earth sciences offer a number of outstanding examples, as in the study of the earth's magnetic field, where satellite data, ground instruments, paleo databases, and simulation are all combined. Aspects of this huge field are briefly surveyed in Sections 3.3.2 and 3.3.3. Likewise, biomedicine also contains a rich and growing collection of multi-messenger use cases, where—for example—integration of information from pathology, radiology, and molecular studies is becoming increasingly crucial in guiding cancer therapy (Section 3.3.4). Finally, light and neutron source beamline analyses constitute a third collection of multi-messenger use cases. Requirements have a great deal of commonality across, beamlines and scientific projects. Beamline experiments frequently generate data that, when reconstructed, can be used to characterize properties of materials, tissue, or chemical systems.

The examples of convergent application workflows shown here highlight the factors that will have to be taken into account when developing the platform infrastructure to support them. These considerations led to a number of pertinent questions that will feed into the following two sections and motivate our discussions in follow-up BDEC meetings.

3.3.1. Plasma fusion. Building the scientific foundations needed to accelerate the delivery of fusion power can best be accomplished by engaging the predictive capability of modern big-data-driven statistical methods featuring machine learning and deep learning. These techniques can be formulated and adapted to enable new avenues of data-driven discovery in key scientific applications areas such as the quest to deliver fusion energy. An especially time-critical and challenging problem facing the development of a fusion-energy reactor is the need to deal reliably with major, large-scale disruptions in magnetically confined tokamak systems like today's EUROfusion Joint European Torus (JET) and the upcoming International Thermonuclear Experimental Reactor (ITER). The JET team led the development of supervised machine-learning, support vector machine (SVM)–based predictive methods, which have achieved over 80% predictive capability for disruptions that occur 30 ms prior to damaging events—far exceeding current HPC "first principles" approaches. However, ITER will require approximately 95% or better predictive accuracy with less than 5% false positives at least 30 ms before disruptions. Accordingly, this requirement will demand the deployment of improved physics-based classifiers that encompass multidimensional features—a machine learning challenge for training that exceeds the current capabilities of SVM methods.

Very encouraging advances in the development and deployment of deep learning recurrent neural nets were recently demonstrated in the results obtained by the team at the Princeton Plasma Physics Laboratory. The results have already exceeded those of SVM methods (i.e. better than 90% predictive accuracy with 5% false positives at least 30 ms before the occurrence of JET disruptions). Moreover, scalability studies of the fusion recurrent neural net (FRNN) deep learning code—first on 200 GPUs on Princeton University's "Tiger" cluster and then on 6000 GPUs on Oak Ridge National Laboratory's (ORNL's) "Titan" leadership-class supercomputer—show encouraging results that indicate that sufficiently rapid training of higher physics fidelity classifiers is now feasible.

FRNN uses the well-known "Theano" and "Tensorflow" (from Google) back ends to train deep neural networks (DNNs; e.g. stacked, long short-term memory). With this approach, a replica of the model is kept on each worker, processing different mini-batches of the training data set in parallel. The results on each worker are combined after each epoch using standard message passing interface (MPI) methods, and the model parameters are synchronized using parameter averaging. The learning rate is adjusted after each epoch to improve convergence.

This project has direct access to the huge EUROfusion JET disruption database (over 500 terabytes of data) to drive these studies. Since the JET signal data come in sequences of variable length, the training of RNNs for time intervals as long as 2000 time steps is challenging but achievable in that FRNN uses a technique of patches and chunks that make it possible to capture physical patterns that are only visible for significantly long sequences (i.e. about 1–2 s). The FRNN project will explore the viability of modern deep learning code for deployment on leadership-class supercomputers that feature a very large number of modern Pascal P100 GPUs (e.g. "Piz Daint" at the Swiss National Supercomputing Centre and "Tsubame 3.0" at the Tokyo Institute of Technology) and future Volta GPUs (e.g. Summit at ORNL).

3.3.2. Earth, planetary, and universe sciences. The earth, planetary, and universe (EPU) sciences are exemplary in their sharing of scientific culture with observational, data-driven research practices that cover a wide spectrum of spatial and temporal scales. EPU research addresses fundamental problems associated with the understanding of the formation, structure, and evolution of EPU systems in their environment (e.g. stellar and fluid envelopes); transient events (e.g. stellar explosions, earthquakes, and volcanic eruptions) and their radiation (e.g. high-energy astro particles and gravitational, electromagnetic, acoustic, and seismic waves); and associated applications of societal and economic impact (e.g. prevention and risk mitigation of volcanic and seismic hazards, exploration and management of energetic resources, evaluation and monitoring of environmental changes, and spatial meteorology).

EPU communities are well organized and federated at the national and international levels around more and more complex space missions (e.g. Euclid, the laser interferometer space antenna, the International Gamma-Ray Astrophysics Laboratory, and the Advanced Telescope for High ENergy Astrophysics), large instruments (e.g. the large synoptic survey telescope, the Cherenkov Telescope Array, and the Square Kilometer Array (SKA)), and observation systems (e.g. cubic kilometer neutrino telescope, AstroMev, the Laser Interferometer Gravitational-Wave Observatory (LIGO)-Virgo, EarthCube, and the European Plate Observation System) often run by international intergovernmental consortia and agencies. The communities place a premium on internationally distributed and federated data resources for archiving and distributing data and have pioneered the prevailing philosophies of globally shared and open data with internationally approved data, metadata, and exchange standards—including provenance and interoperability—together with a growing commitment to open science.

Building federated computing and data analysis platforms with persistent storage to capture and aggregate the large volumes of diverse data (e.g. events, time series, and images) from the observations systems—which use distributed archive resources—and from large numerical simulations (e.g. virtual instruments) to accelerate the path of data use can best be accomplished by engaging a research-driven strategy shown outlined in the following.

- Design and exploit more and more complex space missions, large instruments, and federated observation systems that have complex, on-the-fly data processing and analysis workflows for instrument calibration and data compression, together with large HPC simulations of the experiments in their environment;
- Develop innovative, big data-driven methods to accelerate the full path of data use, with data-stream workflows orchestrating data processing and physics-based or data-driven statistical analysis methods, in a Bayesian inference framework featuring machine learning and deep learning;
- Use advanced, physics-based stochastic simulations of multi-physics and multi-scale natural systems that map complex model space of high dimensions (e.g. cosmology, magnetohydrodynamics, and seismology) into data space, exploring prior density probability information together with data assimilation; and
- Implement modern, big data-driven, high-resolution, and multi-scale imaging techniques featuring nonlinear inversion methods in the framework of new Bayesian approximate and variational inference—together with machine learning—to efficiently sample posterior probability distributions in high dimension and quantify statistical incertitude measurements and extreme events.

Recent developments in high-end statistical data analysis have been demonstrated with new results in the direct, physics-based detection of gravitational wave signals from two ground interferometers (LIGO), thereby opening new observational conduits in astronomy for dense and massive astrophysics objects (e.g. black holes and neutronic stars). These recent developments are also apparent in physics-based extraction of coherent seismic signals from statistical correlation-based analysis of noncoherent background signals—generated by environmental sources resulting from the coupling of the solid Earth with external fluid envelopes (i.e. atmosphere and ocean)—recorded on dense seismic arrays, thereby opening new imaging methodologies in Earth sciences and exploration geophysics. Finally, these high-end statistical data analyses have helped with data-driven detection from dense seismic arrays of new, unknown, long-period seismic signals (tremors) associated with transient deformation processes in volcanic and tectonic contexts, thereby opening a completely new area of research for the understanding and monitoring of natural hazards.

New discoveries in EPU sciences are critically dependent on innovative computing and data analysis platforms federating parallel computing and data resources with research-driven architectures and services that can integrate data-aware computing environments; data-aware architectures; collaboratively designed software with large, complex observation systems; and new virtualization techniques.

For EPU applications, recent advances in Bayesian approximate and variational inference methods—together with machine learning—have the potential to dramatically impact event detection, data analysis, and data modeling methods. New statistical analysis methods featuring machine learning also have an increasing role in efficient data assimilation and in situ data analysis of large-scale stochastic simulations of EPU systems. In parallel, an increasing number of new scientific problems require a combination of model and multi-type data from experiments and simulations across multiple domains.

3.3.3. Data assimilation in climate research. Data assimilation combines data coming from both observations and model outputs in an optimal way to provide a more complete and coherent description of the system and thus improve our predictive capabilities (Figure 6). An obvious requirement is that a model of said system must be available, whereby the increased and novel forms of data analysis helps drive the development of the system model. Climate and weather forecasting applications are among the most common and most important examples of this process, where observational data are combined with simulation data to create a better model than either could create separately.

Data assimilation can be used with two different aims: (1) reconstruct the actual state of a system by combining the two types of information; in this case, it could be said that observational data and model outputs complement

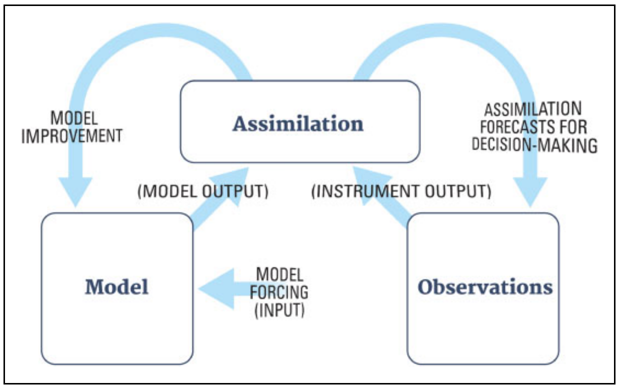

Figure 6. Data assimilation combines observations with model outputs in an optimal fashion, to improve predictive capabilities. This provides a concrete illustration of the Inference Cycle of Section 3.1.

each other by helping correct the largest errors from one or the other and provide respective interpolation/extrapolation tools to reconstruct the best possible description of the actual system. (2) Or use data assimilation for prediction in time; in this case, the best reconstruction refers to the initial time of the forecasting model, and data assimilation ensures both that this initial state is as good as possible from a physical point of view, and that it is also coherent with the model dynamics, therefore avoiding too-large transient effects in the simulation.

A technology complementary to simulation, namely, emulation, is rapidly evolving in the domain of geospatial statistics. Emulation is emerging as a reliable and less expensive computational alternative to simulation in its predictive power; however, it lies wholly within the HDA-driven approach and is not covered in this brief discussion.

The relationship of assimilation to the computational model and the inputs and outputs of the combined system is shown in Figure 6. Two major approaches can be used: (1) a variational approach based on optimization theory, and (2) a statistical approach based on optimal (Kalman) filtering and ensemble averaging. Today, the tendency is to combine the two into what are know as “hybrid ensemble-variational approaches” (Asch et al., 2017). These approaches constitute a major challenge for convergence of HPC with HDA.

The huge amounts of data required for weather prediction make forecasting the weather a highly data and compute-intensive exercise. For each forecast, data must be collected from multiple sources—including satellites, dropsondes, weather stations and buoys, current and historical observations, and simulations of the Earth’s physical patterns and processes. All of this information is then input into complex, nonlinear meteorological models that simulate weather patterns that are likely to occur, based on data assimilation, as shown in Figure 6.

When data assimilation is used for forecasting, the differences between the predicted values and the observed values are inputs to the assimilation engine. However, the observational data are sparse compared to the number of degrees of freedom required to specify the state of the system. Therefore, they are not used to directly reset individual state values. They go into an optimization process typically in the form of a Kalman filter. Data assimilation is perhaps best understood in the context of inverse problems—the latter being well developed in theory and algorithms (Asch et al., 2017).

With the availability of increasing computing power, data assimilation can rely on an ensemble consisting of a large number of independent simulations. Given a thousand instances of a petascale simulation, one has an exascale computation, with the last factor of 1000 in scaling coming in a different dimension than traditional weak-scaling that employs a finer resolution of a given space-time domain. Data assimilation is therefore a prime example of the immediate utility of exascale capability, since the software and workflows already exist. Their combination is not trivial but rides existing efforts to scale up individual simulations with a higher payoff factor.

Recent work by Miyoshi et al. (2016) addressed so-called “big data assimilation” for rapidly changing, severe weather forecasting. The aim is to provide early warnings to civilians and administrations of impending, high-magnitude weather events. By coupling a high resolution “Phased Array Weather Radar” system with RIKEN’s “K Computer,” they have been able to produce such early warnings for local, severe-weather events. One hundred simulated states are fed into a (local transform) ensemble Kalman filter, and each state is simulated in 30 s (only) on 3072 nodes of the 10 petaFLOP/s K Computer. This can be scaled up to the full 88,128 nodes (705,000 processors) of the K Computer, yielding a full data assimilation cycle that remains within the 30-s time window and provides a constantly updated 30 min forecast. It should be noted that data volumes are of the order of 1 terabyte of observation data and 3 petabytes of simulation-produced data. To achieve such spectacular execution times, this required extensive code optimization at all levels, from I/O to basic linear algebra routines.

3.3.4. Cancer diagnosis and treatment. The Cancer Moonshot[11] aims to accomplish 10 years of cancer research in only 5 years. With complex, nonlinear signaling networks, multi-scale dynamics from the quantum to the macro level, and giant, complex data sets of patient responses, cancer is a major challenge for HPC and big data convergence. In light of the difficulty and relative inefficiency of new drug development and screening,[12] the CANcer Distributed Learning Environment (CANDLE) project proposes to employ machine learning techniques to cancer biology, pre-clinical models, and cancer surveillance. The aim is to

> . . . establish a new paradigm for cancer research for years to come by making effective use of the ever-growing volumes and diversity of cancer-related data to build predictive models,

provide better understanding of the disease, and, ultimately, provide guidance and support decisions on anticipated outcomes of treatment for individual patients.[13]

The project's specific objective is to position deep learning at the intersection of Ras[14] pathways, treatment strategies, and drug response. Computational requirements associated with cancer diagnosis and treatment are extremely heterogeneous, and the work carried out in the CANDLE project is a very interesting subset of a much broader set of computational challenges.

Predictive modeling of drug response is a function of tumors (gene expression levels) and drug descriptors. These can be analyzed by deep, convolutional, supervised learning algorithms. But how can one search a trillion drug combinations to predict the response of a given tumor to a given drug combination? One possible answer is deep learning. Within scientific computing applications, hundreds of examples of this technique are emerging, including fusion energy (see above), precision medicine, materials design, fluid dynamics, genomics, structural engineering, intelligent sensing, and so on. To apply deep learning to drug response, modeling requires a hyper-parameter search of 10,000 dimensions that can only be done by effective parallelism at the level of 1,000,000–100,000,000 compute cores. In order to achieve this, the following questions need to be addressed

1. What are the key frameworks and workloads for deep learning?
2. What hardware and systems architectures are emerging for supporting deep learning?
3. Is deep learning a distinct class worthy of its own software stack in the BDEC Universe?

We will briefly address each of these in turn, noting that they remain largely open questions that are certainly application dependent.

The number of available frameworks is increasing every day with a concomitant risk of dispersion and lack of interoperability. The most well-known frameworks are Torch, Theano, Caffe, TensorFlow, and the Microsoft Cognitive Toolkit. Numerous languages are supported, and this can vary from framework to framework, but they are mostly interactive scripting-type languages such as Julia, Python, Lua, and R. In addition, the presence of interactive workflows is an important aspect of many deep learning projects. We are currently witnessing an exponential growth in the number of deep learning-based projects. All of this argues in favor of some type of standardization that could be motivated by convergence issues.

But to apply deep learning across a broad range of fields, a number of deep learning system architecture challenges must also be confronted. The first is "node centric" versus "network centric" architectures with either integrated resources on a node or disaggregated resources on a network. Then the issue of name space/address space across instances/stacks can be approached by either one integrated space across all stacks or each stack maintaining names and addresses. But are the technology components converging? Finally, the issue of "training" versus "inferencing" must be addressed. What balance should there be between online versus offline training, and what are the possibilities of embedding the training within simulation environments?

We are witnessing the emergence of numerous hardware and system architectures for supporting deep learning. Here we can point out CPUs (e.g. chips with advanced vector extensions, chips with vector neural network instructions, and Xeon Phi), GPUs (e.g. NVIDIA's P100, AMD's Radeon Instinct, and Baidu's GPU), application-specific integrated circuit (IC) chips (e.g. Nervana, DianNao, Eyeriss, GraphCore, tensor processing unit (TPU), and deep learning processing unit), field-programmable gate arrays (FPGAs; e.g. Arria 10, Stratix 10, and Falcon Mesa), and neuromorphic technologies (e.g. True North, Zeroth, and N1). But which solution or combination is best for a given application?

So, is deep learning a distinct class worthy of its own software stack in the BDEC Universe? The CANDLE project has developed a stack of its own (see also the abovementioned dataflow and plasma fusion discussions) made up of workflow, scripting, engine, and optimization layers. The workflow contains hyper-parameter sweeps and data management using the NVIDIA deep learning GPU training system or Swift. The scripting consists of a network description and an application programming interface (API) for execution (e.g. Keras or Mocha). A tensor/graph execution engine is based on Theano, Tensorflow, and so on. Finally, an architecture-specific optimization layer is added, based on the NVIDIA CUDA deep neural networks library or the Intel Math Kernel Library for DNNs.

To enable HPC convergence, parallelism options and I/O must be examined. Data parallelism (distributed training by partitioning training data) must be managed at an appropriate level within the stack. Model parallelism (parallel training by partitioning network) could be managed independently. At what level should streaming training data loaders be implemented? And what about main I/O?

As we have seen, numerous fundamental questions are raised here. These will require careful consideration for a successful convergence of deep learning approaches within the context of HPC.

3.3.5. Numerical laboratories. Large supercomputer simulations are rapidly becoming instruments in their own right. Scientists in many disciplines seek to compare the results of their experiments to numerical simulations based on first principles. This requires not only that we can run sophisticated simulations and models, but also that the results of these simulations are available publicly through an easy-to-use portal. We have to turn the simulations into "open numerical laboratories," in which anyone can perform their own experiments. Integrating and comparing experiments to simulations is a nontrivial data management challenge (Figure 7). Not every data set from a simulation has the same life cycle. Some results are transient and only need to

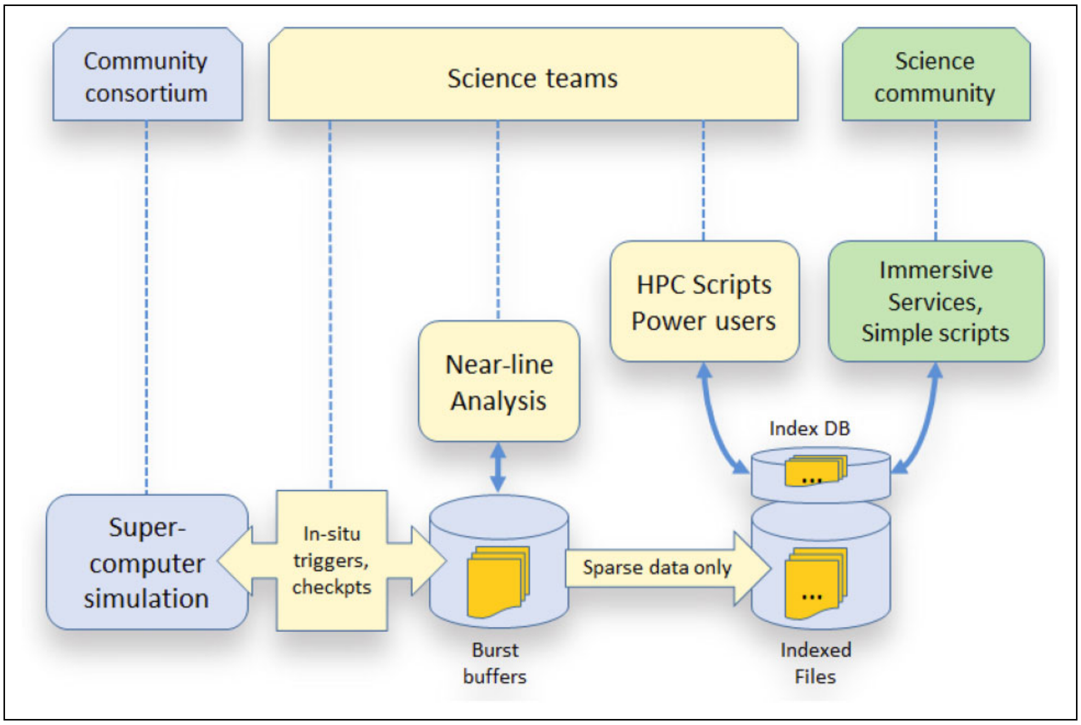

Figure 7. Future scenario for numerical laboratory analysis and data flows.

be stored for a short period for analysis, while others will become community references with a useful lifetime of a decade or more.

As we have learned over the years, once the data volume is too large, we have to move the analysis (computing resources) to the data rather than the traditional approach, which moved the data to the computing resources. With these large data volumes, one has to approach the data in a fully algorithmic fashion—manual exploration of small (or large) files is no longer feasible. Even though the largest simulations today are approaching hundreds of billions of particles or grid points, the total size of the output generated rarely exceeds 100 terabytes and almost never reaches a petabyte. As the interconnect speeds are not going to increase by a factor of 30–100, it is likely that this limitation will remain. Even with exascale machines, the publicly available outputs will likely remain in the range of a few petabytes. To date, the usual way of analyzing someone else's simulation is to download the data. With petabyte-scale data sets, this is obviously not going to work. For a scalable analysis, we need to come up with a data access abstraction or metaphor that is inherently scalable. For the user, it should not matter whether the data in the laboratory are a terabyte or a petabyte. The casual user should be able to perform very lightweight analyses without downloading much software or data. Accessing data through the flat files violates this principle: The user cannot do anything until a very large file has been physically transferred. This implies an access pattern drastically different from the sequential I/O of the checkpoint restart. We need to support random access patterns, where we have a "database" of the most interesting events. This approach is reminiscent of how physicists at the LHC are only storing one out of 10,000,000 events—still yielding tens of petabytes every year.

Supporting such localized access patterns can enable new metaphors for interacting with large numerical simulations. For example, one can create a so-called immersive environment in which the users can insert immersive virtual sensors into the simulation, which can then feed a data stream back to the user. By placing the sensors in different geometric configurations, users can accommodate a wide variety of spatial and temporal access patterns. The sensors can feed data back on multiple channels, measuring different fields in the simulation. They can have a variety of operators, like computing the Hessian or Laplacian of a field or applying various filters and clipping thresholds. Imagine how scientists could launch mini accelerometers into simulated tornadoes, emulating the movie *Twister*!

Scientists at Johns Hopkins have successfully implemented this metaphor for various turbulence data sets and are now porting it to cosmology simulations. This simple interface can provide a very flexible and powerful way to do science with large data sets from anywhere in the world. The availability of such a 4-D data set "at your fingertips" and the ability to make "casual" queries from anywhere is beginning to change how we think about the data. Researchers can come back to the same place in space and time and be sure to encounter the same values.

The *Twister* metaphor mentioned above has been implemented in the Johns Hopkins Turbulence Database (Li et al., 2008), the first space-time database for turbulent flows containing the output of large simulations made

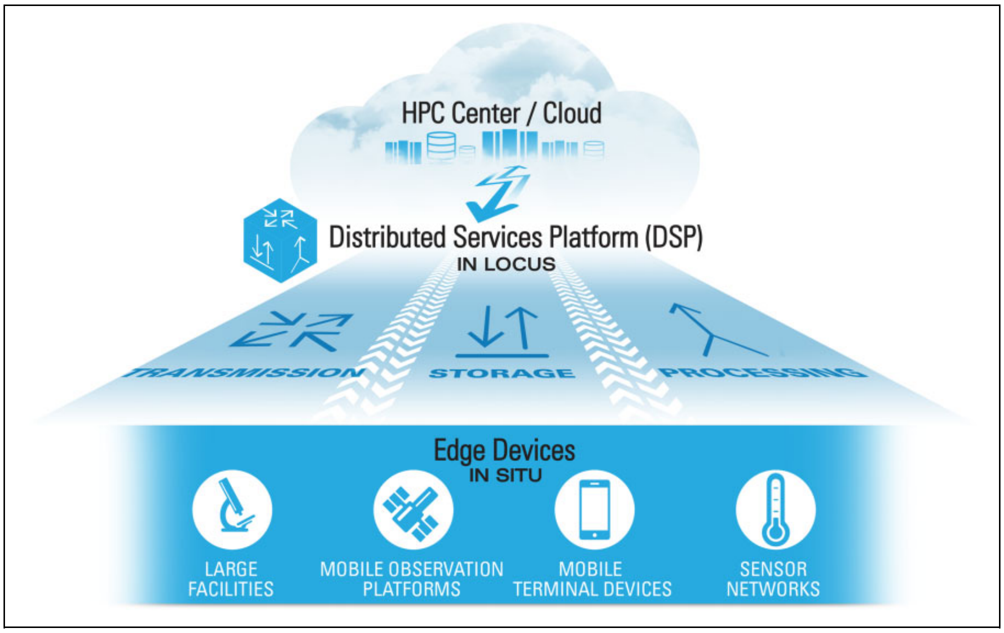

Figure 8. The general problem with multiple high-volume generators at the edge is that these "edge environments" (i.e. environments across the network from centralized facilities) are, or soon will be, experiencing unprecedented increases in data rates from diverse and rapidly proliferating sources. Everyone agrees that these data will have to be buffered/stored and processed in various ways and for various reasons; but there is currently no shared—open—interoperable infrastructure adequate for the task and no agreed-upon roadmap for developing it.

publicly available to the research community. It supports a web service that handles requests for velocities, pressure, various space derivatives of velocity and pressure, and interpolation functions. The data and its interface are used by the turbulence research community and have led to about 100 publications to date. As of this writing, the service has delivered over 44,000,000,000,000 (trillion) data points to the user community. In a recent paper on magnetohydrodynamics, trajectories were computed by moving the particles backward in time—impossible to do in an in situ computation and only made possible by interpolation over the database (Eyink et al., 2013).

A similar transformation is happening in cosmology. The Sloan Digital Sky Survey SkyServer framework was reused for the Millennium simulation database (Kuntschke et al., 2006). The database has been in use for over 10 years, has hundreds of regular users, and has been used in nearly 700 publications.

These large numerical data sets, analyzed by a much broader range of scientists than ever before, using all of the tools available in the computer age, are creating a new way to do science, one that we are just starting to grasp. We cannot predict where exactly it will lead, but it is already clear that these technologies will bring about dramatic changes in the way we do science and make discoveries and how we will use our exascale simulations.

4. Challenges to converged infrastructure in edge environments

Based on our report on the cumulative work of the BDEC community so far, one might reasonably draw at least two provisional conclusions. First, at the level of scientific applications and interdisciplinary research, the convergence of techniques and methods of compute-intensive simulation and data-intensive analytics is already underway. Several of the examples given in Section 3 show that a more complete and integrated vision of the general logic of scientific inquiry is already being implemented in multiphase workflows, and there is every reason to expect that many more dramatic and complex products of this process will be emerging in the near and medium term. Second, at the level of (logically) centralized infrastructure (i.e. on stand-alone HPC systems and distributed cloud facilities [Section 5]), promising avenues for integrating the HPC software stack and the HDA software stack are being opened and explored from both sides. If the ultimate success of these two developments, taken together, were sufficient to reestablish the shared foundation of interoperability and software portability that has proved so catalytic for scientific collaboration and progress over the past two decades, then the BDEC community—in concert with other ongoing infrastructure planning efforts from around the world—would be well positioned to draft a roadmap to get there. Unfortunately, the proliferation of huge and heterogeneous flows of data generated *outside* such centralized facilities (i.e. across the WAN in "edge environments"), as well as the need to *distribute* large data sets from the center to the edge, represents a third factor that makes the way forward uncertain (Figure 8).

The explosive growth and dispersion of digital data producers in edge environments is a highly multidimensional problem. Looking at just the properties of the data flows being generated, the published lists of the challenging characteristics they exhibit include their volume, velocity, value, variety, variability, and veracity (Bethel et al., 2016; Chang, 2015; Grady et al., 2014; Hashem et al., 2015; ur Rehman et al., 2016). The size dimension tends to be highlighted first, if only because the volumes are so striking. Big instruments—such as the LHC and the Argonne photon source (APS) mentioned in the Section 5.2—provide familiar illustrations, but other examples are plentiful. For instance, a review of the records of over two decades of storage usage for neuroimaging and genetics data shows that the influx is doubling every year, with some estimates reaching over 20 petabytes per year by 2019 (Dinov et al., 2014). In a different domain, light detection and ranging (LIDAR) survey technology (Wikipedia, 2017b), which has hundreds (if not thousands) of applications in mapping and monitoring for many fields of science, engineering, and technology, already routinely produces terabyte-level data sets, with cumulative volumes that are correspondingly immense (Cao et al., 2015). Autonomous vehicles will generate and consume roughly 40 terabytes of LIDAR data for every 8 h of driving, and LIDAR prices have come down three orders of magnitude in 10 years. Finally, taking a more comprehensive perspective, a recent Cisco Global Cloud Index (2015) report index asserts that mobile data traffic has increased more than three orders of magnitude over the last decade and will continue to grow at more than 50% annually; by 2020, this data traffic is projected to surpass 500 zettabytes in total (Shi et al., 2016; Wang et al., 2017).

Another set of factors that render edge environments complex is the diversity of the data collecting devices and the modes in which they can be used. LIDAR remote sensing can be used in several modes: static terrestrial (e.g. tripod), mobile (e.g. autonomous vehicles), and aerial (e.g. drones). Some, like the SKA and environmental sensor nets, are highly distributed, so that the data they produce generally need to be aggregated and appropriately coordinated or merged. Others, like the LHC, are centralized. Moreover, the power constraints under which some devices can or must operate obviously affect the schedule on which they can collect and transmit data. All of these factors will need to be taken into account in designing a DSP that will need to be deployed in order to support them.

Perhaps the most general designation for the field into which this large welter of problems falls is "data logistics" (i.e. the management of the time-sensitive positioning and encoding/layout of data relative to its intended users and the resources they can use) (Chard et al., 2012). Any user who wants to analyze data that are generated at one location or many locations, but is worked on somewhere else and possibly in many other places or in transit, confronts challenges of data logistics in this sense. High-profile examples for the international scientific community, including the LHC, the SKA, and the climate modeling communities (e.g. through the Earth System Grid Federation), illustrate how managing the movement and staging of data—from where it is collected to where it needs to be analyzed—can take up most of the time to solution.[15] Moreover, in many, if not most cases, some form of *data reduction* has to be applied locally before any further data movement can even be attempted (Section 5.2). However, other more mundane examples are plentiful, and, clearly, the concept of data logistics can be applied much more broadly. At the root of the problem is that the data and the computing resources needed to process it have to be co-located for work to proceed. From the point of view of "time to solution," the challenges of just moving massive data objects into and out of the memory of HPC systems can also be characterized as logistical in nature. Hence, one can think of data logistics as defining a continuum, with I/O issues inside the IDC or supercomputing facility falling at one end,[16] and big data workflows that begin at remote and/or distributed data sources—possibly scattered across edge environments—falling at the other.

Now, experience shows us that coping with the logistics of multiform workflows in the wide area is a nontrivial problem, especially when the only service the underlying network supplies is the routing of datagrams between end points. As noted earlier in Section 2.1.2, the Internet's lack of native support for point-to-multipoint content distribution, among other things, has forced the research community (and service providers generally) to deploy workarounds to problems of data logistics since the earliest days of the network. Figure 3 provides a snapshot summary of successive attempts to bypass this problem: first with FTP mirrors and web caching, which generally preserve the classic Internet article; and then via CDNs and cloud computing, which use standard IPs to connect to clients at the edge of the network but deploy proprietary networks in their core that incorporate storage and computing as services at the intermediate nodes. Historically, work on CDNs served as the origin of current efforts to develop computing infrastructure for edge environments (i.e. infrastructure to provide storage, computing, and networking at various places along the path from the cloud or data center to the edge (Satyanarayanan, 2017).

A variety of different approaches under a variety of different names have been proposed for edge computing infrastructure: the European Telecommunications Standards Institute's mobile edge computing (MEC) (Hu et al., 2015), fog computing (Bonomi et al., 2012), cloudlet (Satyanarayanan et al., 2009), and edge caching (Bastug et al., 2014), to name some prominent contenders (Wang et al., 2017). Wherever possible, we use the expression DSP as a more generically descriptive phrase for infrastructure designed to support compute-intensive and/or data-intensive work that must be carried out between the cloud (or data/HPC center) ingress/egress and the network edge. This is motivated partly by the fact that there appears to be little consensus at this point about what the right approach should be or about which of the contenders might come to dominate (Fox et al., 2016; Wang et al., 2017). In this regard, the current situation in edge computing infrastructure has plausibly been compared to the situation in networking "at the dawn of the Internet in the late 1970s to early 1980s" (Satyanarayanan, 2017). In any event, the unsettled state of the field means that our exploration below of some of the important issues to be addressed by, and some of the main strategies for using, such an infrastructure needs to focus on possible points of agreement upon which the views of different stakeholders might converge.

4.1. Common context: Converging on a new hourglass

If we want a new distributed infrastructure to support science and engineering research in the era of big data, an infrastructure with the kind of openness, scalability, and flexible resource sharing that has characterized the legacy Internet paradigm, then a recent community white paper on "intelligent infrastructure" for smart cities makes the nature of the challenge clear:

> What is lacking—and what is necessary to define in the future—is a common, open, underlying 'platform,' analogous to (but much more complex than) the Internet or Web, allowing applications and services to be developed as modular, extensible, interoperable components. To achieve the level of **interoperation** and innovation in Smart Cities that we have seen in the Internet will require *[public] investment in the basic research and development* of an analogous open platform for intelligent infrastructure, tested and evaluated openly through the same inclusive, open, consensus-driven approach that created Internet. (Nahrstedt et al., 2017, emphasis in source)

As argued above, the dominant cyberinfrastructure paradigm of the past three decades actually has two main components: the IP stack and the Unix operating system, the kernel interface of which became the foundation for an immense ecosystem of largely interoperable open-source tools and operating systems, such as Linux. If we are to go in search of a future-defining DSP, it seems reasonable to start from a prominent architectural feature that they share. Specifically, they both conform to the "hourglass" design model for layered software stacks (Beck, 2016). The hourglass image (Figure 9) represents the idea that an appropriately designed common interface can be implemented on a wide variety of technology platforms (yielding a wide "lower bell"), while at the same time supporting an equally wide variety of applications (yielding a wide "upper bell"). David Clark, one of the leading designers of the IP stack, called the common middle interface (i.e. the thin waist of the hourglass), the "spanning layer" because it bridges, through virtualization, a heterogeneous set of resources that lie below it (Clark, 1997). In the case of IP, these underlying resources consist of different types of local area networks (LANs). In the case of the Unix/Linux kernel interface, possible technology substrates include an enormous variety of software/hardware platforms—from massive compute clusters to handheld devices and wrist watches.

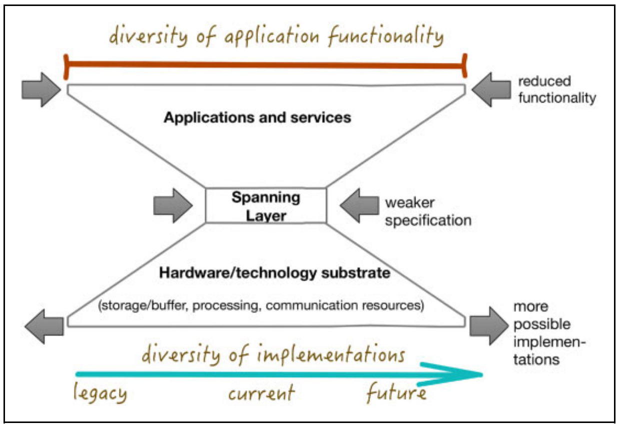
diversity of application functionality
reduced functionality
Applications and services
Spanning Layer
weaker specification
Hardware/technology substrate
(storage/buffer, processing, communication resources)
more possible implemen-tations
diversity of implementations
legacy
current
future

Figure 9. The "hourglass model" of the system software stack. The goal is to achieve deployment scalability while maximizing the diversity of applications. Credit: adapted from Beck (2016).

In terms of general cyberinfrastructure design, and cyberinfrastructure for science and engineering in particular, this combination of properties is critical. From the point of view of application communities, the ability to easily implement a spanning layer on new hardware is fundamental, because the underlying platform technologies are constantly evolving. Unless a spanning layer can be ported or reimplemented on each successive generation, the applications it supports will be stranded on some island of technical obsolescence. Yet new types of applications and application requirements are also constantly emerging. So no matter how portable a spanning layer is, if it cannot address innovative application demands, it will be abandoned for a software stack that can. The challenge, of course, is to provide a service specification (i.e. an API) for the spanning layer that can satisfy both of these requirements.

The waist of the hourglass is called "thin" or "narrow" because it needs to be minimal, restricted, or otherwise weak to be easily implementable on new software/hardware substrates. However, as shown in Figure 9, and contrary to what is commonly believed, squeezing the waist of the hourglass, in order to increase the number of possible implementations covered by the lower bell, will tend to have the reverse effect on the width of the upper bell (i.e. this will reduce the potential range of supported applications) (Beck, 2016). Hence, the design challenge for creating a specification for a scalable DSP spanning layer is to find one that optimally balances these two opposing goals. If we assume that we are talking about a software stack for workflows, which are inherently stateful processes, that would seem to imply that the spanning layer must include (orthogonal) primitives for communication, computation, and storage. So, striking the right balance is likely to prove challenging indeed.

Unfortunately, in seeking a new DSP spanning layer to address the challenges of the big data era, the science cyberinfrastructure community finds itself in something of a dilemma. On the one hand, at present, there does seem to be at least one plausible and widely touted candidate for the new spanning layer—operating system-level virtualization that supports software "containers" (Figure 13 and Section 5.3.2). Certainly, converging on a common interface for containerization would go a long way to achieving ecosystem convergence and do so in way that requires something closer to evolutionary, as opposed to revolutionary, changes to current modes of operation. Perhaps for that reason, and as participants in the BDEC workshops made clear, the potential of containerization is a very active area of research and experimentation across all the contexts that scientific cyberinfrastructure will have to address, including commercial clouds, HPC systems, and computing resources deployed in edge environments. Indeed, the "Two Ecosystems" picture (Figure 1) was recently updated to show that both HDA and HPC are pursuing containerization strategies. In Section 5.3.2, we explore the possible benefits and liabilities associated with sliding yet another layer of virtualization (i.e. for software containers) into the current software stack.

4.2. Common context: A common DSP for data logistics

Given the complex nature and broad scope of the data logistics problem space, the DSP label captures at least three features that any reasonably adequate infrastructure capable of meeting these challenges should have.

1. *Be wide area capable*: The intermediate nodes of the DSP must support services that can be deployed in a decentralized fashion across the wide area (i.e. outside the machine room and the LAN).
2. *Offer flexible compute, storage, and communication services*: These nodes must go beyond simple datagram forwarding to provide compute and storage/buffering services in the system's core as well. The need for this capability follows directly from what is perhaps the primary application requirement for edge computing. Namely, relatively low-latency processing (e.g. aggregation, analysis, compression, and distillation) of massive quantities of data flowing from edge devices and sensors) (Desprez and Lebre, 2016).
3. *Provide a scalable model of resource sharing*: Like the legacy cyberinfrastructure paradigm that is becoming increasingly problematic (Section 2.1.2), the DSP (which will replace the legacy paradigm) must—for both technical and practical reasons—enable a variety of different application communities to share a core set of services and the resources they require.

Yet, while several of the proposed alternatives for edge computing (e.g. Fog, cloudlet, and MEC) might provide a

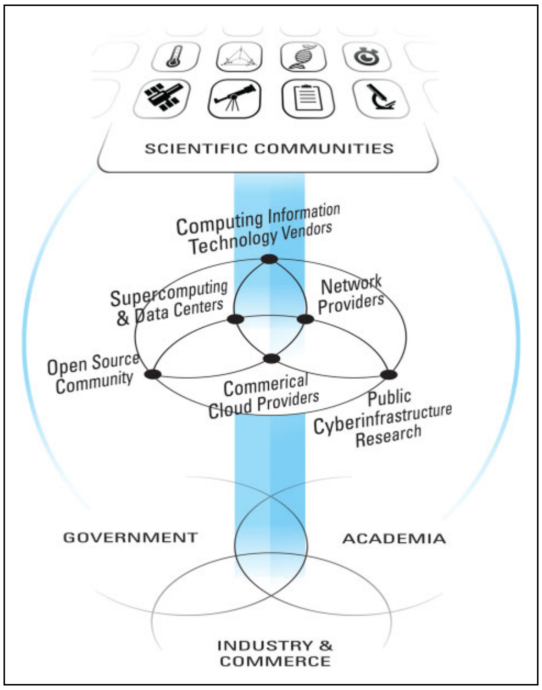
SCIENTIFIC COMMUNITIES
Computing Information Technology Vendors
Supercomputing & Data Centers
Network Providers
Open Source Community
Commerical Cloud Providers
Public Cyberinfrastructure Research
GOVERNMENT
ACADEMIA
INDUSTRY & COMMERCE

Figure 10. Stakeholders in the software/data ecosystem for science and engineering.

DSP that meets what seem to be the major technical requirements for this problem space—highly responsive and resilient cloud services, scalability through local data processing, flexible control of privacy, and computational offloading for client devices (Satyanarayanan, 2017)—the overarching goal of the BDEC community requires a broader perspective. In particular, the guiding purpose of the BDEC community has always been to foster the codesign of a software infrastructure for science and engineering research that supports the broadest possible spectrum of fields or domains and empowers international cooperation both within and among them.

Consequently, since the scientific community, thus broadly conceived, embraces an enormous network of organizations and actors, extending not just around the world but also across generations, the problem of designing a DSP that can be voluntarily adopted as foundational by this community raises unique software ecosystem considerations (Figure 10) that ought to frame our deliberations. Following both the suggestion of the Computing Community Consortium whitepaper quoted earlier (Nahrstedt et al., 2017) and the spirit of the original IESP (Dongarra et al., 2011) below, we briefly discuss three ecosystem design constraints that seem to be among the most important: commonality, openness, and interoperability.

1. *A Common DSP*: The legacy DSP paradigm—the Internet—enabled the scientific community to bridge geographic, organizational, political, and technological boundaries and fostered a profusion of interdisciplinary collaborations across a vast range of fields. The motivation for seeking a

"future-defining" DSP that provides a similarly common foundation for future scientific inquiry should, therefore, be evident. Of course, it should be equally evident that the problem of designing a common DSP that everyone in the community might voluntarily adopt, in the absence of coercive legal or economic power, is truly formidable.

The upper and lower bells of the "hourglass" software architecture model (see Section 4.1 and Figure 9) are intended to illustrate two critical conditions that any proposed universal solution would apparently have to satisfy. First, a wide lower bell for a given DSP means that its spanning layer (Clark, 1997; Kavassalis et al., 1996; Messerschmitt and Szyperski, 2005) (i.e. the common interface at the "narrow waist") can be implemented on a very large collection of heterogeneous hardware technologies, so that all the applications and services above the spanning layer can, through it, access and utilize the resources these technologies make available. The wider the lower bell, the stronger the assurance that hardware technology—both legacy and future—will support that DSP's spanning layer. Second, a wide upper bell means that the small set of primitive services that the spanning layer makes available on the system's nodes can be composed, in combination with end system resources, to support a very large diversity of higher-level services and applications. The wider the upper bell, the stronger the assurance that more specialized application communities, down to the level of individuals, can build on top of the shared infrastructure that the community provides. Now, if a given DSP can thus be implemented with reasonable ease on an extremely broad spectrum of hardware technologies, and can thus be used to create an enormous diversity of services and applications to satisfy multifarious user demands, those facts will tend to minimize barriers to its universal adoption. However, it is important to remember that defining a spanning layer that includes shared compute and/or storage services, and that is weak enough to achieve the former but strong enough to achieve the latter, is still an unsolved problem (Beck, 2016; Beck et al., 2017).

2. *An Open DSP*: Even if a DSP spanning layer with requisite technical properties can be specified, lack of *openness* would also present a major barrier to the kind of ubiquitous acceptance the scientific community needs to achieve. Absent proprietary legal constraints, such a software specification is paradigmatic of a public good (i.e. of something that no one is excluded from using and which can be exploited in a completely non-rivalrous fashion). At a minimum, then, the specification should be freely accessible for people to use as their circumstances require. Moreover, the standardization process should be equally nonproprietary and open and carried out with due process, broad consensus, and transparency (Fälström, 2016). We know that in the case of the IP stack such openness was critical to its relatively rapid acceptance by the global scientific community. We also know that this acceptance, working outward from the universities, government laboratories, and all institutional niches where research gets done, played an instrumental role in spreading the Internet to society generally. It is reasonable to expect that if the more general DSP that science now seeks is kept open, then similar—if not more powerful and more positive—spillover effects for society are likely to result.
3. *An Interoperable DSP*: Although both openness and community deployment are essential objectives for a next generation DSP for data logistics, the linchpin of any plan for achieving those goals is *interoperability* (i.e. the capacity for different modules or devices using shared protocols to exchange information with and/or invoke services on one another, despite differences in their implementations). For one thing, interoperability fosters openness by lowering or eliminating switching costs, which helps users avoid well-known problems like "vendor lock-in," while enabling well-known advantages like "fork-lift upgrades." More importantly, providing a foundation for widespread interoperability tends to catalyze community adoption and deployment through powerful direct and indirect network effects, in which increases in the number of adopters of platform-compatible systems drives up the platform's value for all of its users (Messerschmitt and Szyperski, 2005). The success of the Internet paradigm has been due in no small degree to these effects of interoperability, and so one should reasonably expect the same would be true of a new DSP paradigm that supported similar levels interoperability. But if we believe that "... [a]s a practical matter, real interoperation is achieved by the definition and use of effective spanning layers" (Clark, 1997; see also Kavassalis et al., 1996; Messerschmitt and Szyperski, 2005), then we must also acknowledge that the goal of creating a more general, future-defining DSP, with a primary spanning layer that the entire community is willing to adopt, has already shown itself to be very elusive. In the United States alone, over the past 20 years at least four well-supported efforts—Active Networking (Tennenhouse and Wetherall, 1996), Globus (Foster et al., 2001), PlanetLab (Anderson et al., 2005), and the Global Environment for Network Innovations (McGeer et al., 2016)—took up this challenge explicitly; and while all of them have been (and even continue to be) successful in many other respects, their proposed platforms, and the alternative spanning layers that they have tried to build on,

have not yet achieved the kind of broad acceptance and organic growth that the convergent DSP would require.

Although these interrelated design assumptions may not be exhaustive, they are sufficient for this report's discussion of the alternative strategies for data logistics and related challenges. There seem to be at least four nonexclusive alternatives for interfacing HPC to this new paradigm, in which no strong assumptions are made about where the data are: (1) data streaming, (2) in transit processing, (3) processing at the edge of the distributed system (i.e. as close to the data sources as possible), and (4) logically centered cloud-like processing. This last option is not of the same nature as the first three, and it could encompass them: The user could have a cloud-inspired vision of processing that is decoupled from physical resource distribution; under the hood, the actual processing can rely on options 1–3 (i.e. in situ processing, in transit processing, or streaming across multiple sites).

We briefly discuss each of these strategies in Section 4.3, but expect the BDEC and big data communities to substantially develop this discussion.

4.3. Strategies for processing and state management between the edge and the centralized computing facility

As the examples discussed in Section 3.3 suggest, various combinations of computations and data movement are required to integrate often noisy information from often spatially distributed sensors to create a high-quality, multi-scale description of the physical or biological system under study. For instance, a variety of commonly employed methods cut across many multi-messenger use cases, making this area both an important opportunity and a complex challenge for the design of shared software infrastructure. Most methods include the need to (1) carry out low-level reconstruction, filtering, and noise reduction processing to improve data quality; (2) detect and often segment pertinent objects to extract imaging features and classify objects and regions of images; (3) characterize temporal changes; and (4) use simulation, machine learning, or statistical methods to make predictions.

4.3.1. Strategy 1: Streaming. An emerging model of big data applications is that data are generated or collected at multiple places in the "system." The data may need to move from location to location within the system, and processing may be needed at different locations as the data move through the system. Further, unlike classical HPC, where the data source and sink are in the same location, in the new model, there may be multiple sources and sinks, and they may not be at the same location. An example is the SKA project. Scattered telescopes collect data, but because of the vast amounts of data collected, not all the data can be streamed to a single location for processing. Instead, there are advantages to doing some amount of processing at the instrument. Data are then streamed to intermediate sources, where once again the ability to aggregate and perform local processing allows for smarter (required) reduction of the data streams. While the SKA project will have a relatively static mapping of sources, sinks, and data streams to their processing locations, other applications may be more dynamic. Regardless, to support this emerging model, tools are being developed that let workflow designers abstractly describe the sources, sinks, streams, data, and processing elements and then dynamically connect them as the situation requires.

To accommodate the needs of such applications, stream processing languages have been developed to allow programmers to specify the needs of their workflows and applications to feed into the tools and runtimes being developed. Two common examples from the cloud space are Apache Spark and Flink. These data-centric libraries and runtimes enable the programmer to more easily describe the requirements laid out above (i.e. be able to specify where data are originating, ending, and how it needs to be processed as it moves from source to sink). Many of the languages and runtimes abstract the notion of processing elements, network connections, data sources, sinks, and processing elements. This provides for a separation of concerns, thereby enabling the programmers to focus on describing what connections need to exist and what processing needs to occur without specifying where it occurs and how the processing elements are connected. Some of the languages and runtimes allow the programmers to provide hints or specify where (on which nodes or computers) the processing should occur and what networking topology should be used to connect the processing elements.

The use of data streaming has three benefits: (1) low overhead, particularly for local coupling, means that it is efficient for scientists to compose low-cost steps for a "fine-grained workflow;" (2) the direct handling of streams means that scientists can develop methods for live, continuously flowing observations; and (3) the load placed on the increasingly limiting bottleneck of disk I/O is minimized.

Determining whether we can move to a common model, as we examine the need to converge the tools for HPC, big data, and analytics, is worth investigating. Because work on stream processing within HPC is somewhat untrammeled, this would—hopefully—provide an easier path toward convergence. However, there are challenges. Many existing cloud stream processing capabilities were not designed with HPC in mind, and developers are only more recently starting to examine the high-performance aspects of their runtimes (e.g. low latency). This represents a good area of focus for convergence.

4.3.2. Strategy 2: CDNs (processing in transit). Over the past 15 years, the relentless increase in demand for rich multimedia content has driven commercial providers to build or buy services from sophisticated CDNs to ensure the kind of quality of service (QoS) that their customers want and expect (Ni and Tsang, 2005; Papagianni et al., 2013). Unfortunately, commercial CDNs are expensive, difficult to operate, and are practical only for implementing web and media streaming sites that generate enough income to pay for their service. Consequently, a myriad of noncommercial research and education communities who need to distribute large amounts of data to numerous receivers are unable to make use of existing CDN services.

To see why, we need to briefly consider how CDNs work. A CDN delivers data from a collection of servers distributed in the wide area, either at a large number of collocation sites—like network interchange points, Internet service provider points of presence (e.g. Akamai, Layer3)—or at a smaller number of cloud data centers (e.g. Google or Amazon). While uniform resource locators are used to name content and services in a CDN, they are not interpreted in the same way as they are in the implementation of traditional web services. Instead of using the domain name system (DNS) to translate the domain component to an undifferentiated set of servers, a CDN chooses the best server according to its own resolution algorithm. CDNs use a proprietary resolution protocol that provides the network interface of an uncacheable DNS server, which is how it integrates with existing application layer clients like web browsers.

If a CDN's resolution algorithm is implemented using only the abstractions provided by the Internet's network layer, then the CDN can be viewed as an overlay network implemented at the application layer. However, it is difficult to obtain accurate and effective resolution using only the abstraction provided by the Internet's network layer. So, commercial CDNs instead reach down past the network layer to monitor the link layer network topology, which is not directly observable using the abstractions provided by the network layer. From an architectural point of view, commercial CDNs should be characterized as an alternative network layer that is implemented using a combination of mechanisms from the Internet's link, network, and application layers.

The implication of this strategy for implementing CDNs is that the mechanisms used to implement it as an alternative network layer are proprietary, non-interoperable, and not particularly scalable. Monitoring and even controlling the network at the link layer (by determining the topological location of replica servers) and maintaining replicas of complex application layer services are expensive and difficult engineering obstacles. This makes commercial CDNs expensive to run and creates barriers to interoperation, which—in any case—is not seen as congruent with CDN business strategies. These considerations explain why commercial CDN services are too expensive for large-scale use by the scientific community, in spite of the fact that the cost of the underlying servers and software is relatively modest. What is missing: (1) a scalable approach to a CDN implementation (i.e. suitably designed forms of storage and processing at the nodes of the distribution tree) and (2) the aggregate organizational will of the scientific community.

In the HPC context, a framework for in transit processing has been proposed by Bennett et al., where data generated by simulations are transferred from the supercomputer to intermediate storage for asynchronous processing before archiving. This method has enabled scientists who are dealing with the data deluge at extreme scale to perform analyses at increased temporal resolutions, mitigate I/O costs, and significantly improve the time to insight.

4.3.3. Strategy 3: Computing at the edge (at the source). Edge computing can be differentiated from in transit processing in two ways: (1) it may not need to support competing users, although it probably does need to support multiple applications on behalf of cooperating users, and (2) there may be fate sharing between the end user and the edge computation, meaning that fault tolerance is not required. In addition, the resource limitations of the node on which edge computing is implemented may be more stringent than the constraints on shared intermediate nodes.

Different edge scenarios may allow the constraints imposed on shared intermediate nodes (weak, simple, general, and limited) to be loosened in different ways. In some cases, such as indefinite localized storage, implementation at the edge is the only possibility.

In a network that only implements delivery of data between end systems, all computation and storage must be implemented at the edges of the network. However, certain shared infrastructure can be thought of as defining overlay networks that do support these services implemented at the application layer (e.g. CDNs). The limitation of this approach is the inability to provision and operate shared resources on behalf of a larger community and the inability to locate such resources at arbitrary points within the network topology.

5. Pathways to convergence for large, logically centralized facilities

The examples presented in Section 3 of application-workflow convergence, which integrate both HPC and HDA methods, promise to open new frontiers of scientific inquiry in almost all fields of science and engineering. Today, scientific discovery almost universally integrates both advanced computing and data analytics. This fusion also strongly motivates integrating the associated software with the hardware infrastructures and ecosystems. Moreover, there are equally strong political and economic motivations for such an approach, both within countries and across borders. Consequently, when the US National Strategic Computing Initiative Executive Office of the U.S. President (2015a, 2015b) expresses the goal of "... [driving] the convergence of compute-intensive and data-intensive systems," and this goal is echoed in the strategic plans of the EU, Japan, and China (Section 2), we can see that the question of what "convergence" might mean has at least one clear answer in this context: at minimum, "convergence" means a scientific software ecosystem that overcomes the current state of "Balkanization." However, this raises challenges and problem areas that include, but are not limited to, the following.

- *Differing, though converging, cultures and tools*: When the 1990s dot-com revolution began, the underlying hardware and software infrastructure used for e-commerce sites was strikingly similar to those used in HPC. Software developers could and did move readily across the two domains. Indeed, the web and the web browser originated at scientific computing facilities—CERN and the National Center for Supercomputing Applications, respectively. Today's cloud services run atop hardware strikingly similar to that found in HPC systems—x86 systems with accelerators and high-performance interconnects. Although the programming models and tools and the underlying software differed markedly just a few years ago (Figure 1), there are now signs of convergence. Container (virtualization) technology is now available on HPC systems, and machine learning tools and techniques are increasingly being integrated into HPC workflows.
- *Shifting workforce skills*: Over the past 20 years, the IT industry has expanded dramatically, driven by e-commerce, social media, cloud services, and smartphones, with the IoT, healthcare sensors, industrial automation, and autonomous vehicles further expanding the domain of big data analytics and services. In response to seemingly insatiable workforce demands, most students are now trained in software tools and techniques that target these commercial opportunities rather than scientific computing and HPC. Few students outside of scientific domains learn C, Fortran, or numerical methods, which could be considered the traditional "tools of the trade" in computational sciences and engineering. This trend is an extension of one that began in the 1990s and is irreversible. Consequently, the HPC community must and is beginning to embrace new tools and approaches while also encouraging students to learn both HPC and data analytics tools.
- *Adopting new infrastructure*: The HPC community neither can nor should attempt to replicate the vibrant big data and machine learning infrastructure and ecosystem. Simply put, the locus of investment and human resources in data analytics and machine learning now rests with the commercial sector, and it will drive ecosystem evolution. Lest this seem an insurmountable hurdle, it is worth remembering that the HPC community has long been dependent on, and a contributor to, the Linux and open-source software communities. Likewise, as core components of the HDA ecosystem become open source, they will continue to create benefits for the HPC community. In particular, the Apache software stack, R, and other machine learning toolkits are galvanizing a generation of faculty and students who see a multitude of scientific and engineering applications for big data and machine learning technology. Similarly, science and engineering researchers are increasingly applying machine learning technology to their own domains.
- *Stream and batch model coexistence*: Leading-edge scientific computing systems are, by definition, scarce resources. Thus, the HPC community has long relied on batch scheduling to maximize utilization of limited hardware resources and to serve multiple scientific communities. By contrast, cloud services and most sensor data, whether from small environmental monitors or large-scale observatories, are (soft) real-time streams that necessitate continuous processing with flexible workflow systems. Not surprisingly, these differences have profound implications for application programming models, software tools, and system software, as streaming data analysis requires that at least some of the resources be dedicated for days, months, and sometimes years. As new generations of scientific instruments and environmental sensors produce ever-larger streams of daily data, real-time data processing and statistical assimilation with computational models will drive fusion of batch and stream models and will provide a strong motivation (as well as an opportunity) for the emergence of a flexible common model to meet the needs of both HPC and HDA.
- *Computing at the edge*: Another consequence of real-time data streams is the need for computing and analysis at the edge (i.e. at the data sources). Whether for data reduction and conditioning or more rapid response than end-to-end network delays would permit for centralized analysis, workflows must increasingly span a continuum of device types, bandwidth differences, and computing capabilities. The shift of intelligence to the edge, coupled with central computing support, is a new model that combines elements of HPC and HDA.
- *Virtualization for sharing*: Traditional virtual machine (VM) environments based on hypervisors and replicated operating systems imposed large overheads and latencies, thereby rendering them less suitable for tightly coupled scientific applications. However, new technology (e.g. Linux "containers" like Docker) has far less overhead for such workloads and is increasingly being deployed for scientific computing. This enables developers to shape their application's computing environment and enables the provider to simultaneously run many such environments. Containerization also brings scientific application portability, allowing workflows and software to be packaged, shared, and redeployed without complex and often arduous configuration. In turn, this enables users and communities to evaluate approaches and software easily and rapidly.

- *Resource allocation and efficiency*: The value of a large, centralized resource rests in part on its scale. By allocating a substantial portion of the total resource, one can achieve results not possible with small-scale infrastructure. This is equally true for commercial cloud infrastructure and the largest HPC systems. On large HPC systems, this resource management uses work queuing (i.e. batch processing). Policies control resource allocation (e.g. to select only the most worthwhile applications and to match the resource funders' priorities). Permission for use depends on the quality of an application's optimization (for a given platform). Implicit in such an approach is the need for efficiency, lest a precious resource be consumed unnecessarily. Efficiency may refer to application execution efficiency, system utilization rate, and/or energy efficiency (Section 5.1). The cloud and big data communities tend to emphasize user experience and scalability far more than efficiency, though this is also changing rapidly as parallel computing techniques are being applied to machine learning. This is an additional convergence opportunity, as both the HPC and big data application communities learn from each other.

The *raison d'être* for BDEC is to help plan and create, in a coordinated and international way, a new generation of software infrastructure for scientific research. However, our discussion in this section—of strategies for integrating the infrastructures and ecosystems of large HPC computing platforms and facilities, on the one hand, and of commercial cloud facilities, on the other—can address only some of them, and those only partially. In particular, in this section, we largely abstract away from the large set of difficult issues surrounding the explosive growth of data in edge computing environments, which is discussed in Section 4.

5.1. Common context: Energy as an overarching challenge for sustainability

One challenge probably deserves special attention: making the scientific enterprise sustainable given current technology would seem to demand that the total energy consumption of a given investigation be minimized or at least kept within reasonable limits. Given that information and computing technology (ICT) required about 4.7% of the world's electricity in 2012[17] and continues to grow (Gelenbe and Caseau, 2015), its role in this problem cannot be assumed to be negligible, especially if we include all aspects of ICT supporting a scientific endeavor. These statistics are not limited to the energy used for simulations and statistical analyses. Data movement from the instrument to quality assurance and integration organizations, to archival and curation sites, to each scientific step that accesses the data, to visualizations, to researchers' work spaces, and even to archives, can all represent a very significant use of energy.

With this in mind, we identified four steps toward energy minimization: (1) reduce computational costs by using platforms that are well-matched to the stage within the scientific method; (2) reduce data-movement costs by using collocation, compression, and caching; (3) encourage reuse of calculations and data through effective sharing, metadata, and catalogs—a strategy that a provenance system supports well; and (4) reduce computing system entropy (e.g. workload interference, system jitter, tail latency, and other noise) through on-demand isolation, noise-resistant priority, cache QoS, and novel uncertainty bounding techniques. The cyberinfrastructure itself has the task of taking care of energy minimization as it has access to the required information; leaving this burden to the domain scientists is undesirable, since it would divert them from their scientific goals.

Moving information is fundamentally energy-consuming, more so if it is moved quickly. Moving it across boundaries (e.g. from site to site, from an HPC system to an analysis platform system within the same site, from node to node, or from one storage medium to another) incurs even greater costs. Avoiding unnecessary repetitious transfers (e.g. by caching, reducing distances and boundary crossing by collocation, reducing speed requirements by prefetching, and reducing volume by compression) can all help reduce these costs. Reducing data transport costs and delays also encourages collocation of simulation platforms with data-analysis platforms and with archival storage sites holding reference data.

However, challenges exist because of the limits of what can be achieved as environmental and social science observation systems are inevitably a globally distributed endeavor. Furthermore, the many approaches, different timescales, and different manifest effects make for growing diversity of primary data collection and quality assurance. This is undertaken by a wide range of organizations, in many locations, that are managing a wide variety of globally dispersed instruments. Data quality control also has to be done close to the acquisition system. It is crucial that, as methods for caching data are developed, they do not detract from the recognition of the value of data collectors, quality assurance teams, and data curation organizations.

Hence, a political and accounting model is needed to ensure energy savings and to sustain the respect for the value of contributing institutions. For example, institutions responsible for emergency response information services and hazard estimation, as well as research, will need to recruit the relevant experts and have demonstrable resources that are needed in a major emergency. Major research universities and national research centers need to have a sufficiently powerful computational resource acting as a "totem" so that they can continue to attract leading researchers, projects, contracts, and funding. Therefore, there are pressures to maintain the visibility of independent resources and to sustain their diversity. A cyberinfrastructure that spans autonomous resources needs to minimize energy consumption to reduce environmental impact.

However, due consideration for the organizational and social issues must be given. Encouraging sharing and efficient use of simulation runs and their results may have two benefits: (1) a reduction in environmental impact and (2) an improvement in the quality and pace of scientific discovery—as has been demonstrated in cosmology and climate modeling.

Publishing and sharing the models and the results of simulation runs, for an appropriate period, could establish virtual numerical laboratories (Section 3.3.5), where many researchers could explore interpretations and comparisons of simulation results with primary data. This amortizes the costs of the simulation runs and data gathering over more investigations and over time.

5.2. Common context: Data reduction as a fundamental pattern in HPC and big data convergence

Data reduction is a common issue for both centralized and edge computing infrastructures and facilities. In this section, we look at the "centralized" viewpoint; though the examples and arguments here also apply to the outer edge, where—as explained in Section 4—data logistics are a key issue with the concomitant need for reduction to prevent saturating the upstream (toward the center) communication channels.

Scientific simulations and instruments are already generating more data than can be stored, transmitted, and analyzed. One of the most challenging issues in performing scientific simulations, running large-scale parallel applications, or performing large-scale physics experiments today, is the vast amount of data to store on disks, to transmit over networks, or to process in post analysis. The Hardware/ Hybrid Accelerated Cosmology Code, for example, can generate 20 petabytes of data for a single one trillion particle simulation. And yet a system like the "Mira" supercomputer at the Argonne Leadership Computing Facility (ALCF) has only 26 petabytes of file storage, and a single user cannot request 75% of the total storage capacity for a simulation. Climate research also deals with a large volume of data during simulation and post analysis. As indicated by Gleckler et al. (2016), nearly 2.5 petabytes of data were produced by the Community Earth System Model for the Coupled Model Intercomparison Project (CMIP) Phase 5, which further introduced 170 terabytes of postprocessing data submitted to the ESGF (Williams et al., 2008). Estimates of the raw data requirements for CMIP Phase 6 exceed 10 petabytes (Baker et al., 2014).

Scientific experiments on large-scale instruments also require significant data reduction, and updates of these instruments will produce orders of magnitudes more data. For example, the National Institutes for Health's Brain Initiative focuses on high-throughput x-ray tomography of whole mouse brains using the upgraded APS source. Researchers need to generate high-throughput mosaic tomography with a total reconstruction volume of about 40 teravoxels (corresponding to 160 terabytes) per specimen, followed by automated transport, cataloging, analysis, and comparison of a very large number of specimens in order to understand disease-correlated changes in brain structure. Another example is the IC imaging study under Intelligence Advanced Research Projects Activity's Rapid Analysis of Various Emerging Nanoelectronics program, still at the APS. Every IC will involve about 8 petavoxels (corresponding to 32 petabytes) in the reconstructed image obtained via X-ray ptychography. Today's detector can acquire up to 500 frames/s, and each frame is about 1000 $\times$ 1000, at 16 bits (0.98 gigabytes/s). With the APS-U, researchers expect to acquire data at a 12-KHz frame rate or 23 gigabytes/s, for a total volume of data reaching 70 exabytes. In these two cases, the data produced will be transferred to a supercomputing facility for data analysis. Even upgrading the connection between the APS and the ALCF will not allow one to transfer the data quickly enough to keep up with the flow produced by the instrument. For example, a 20 gigabits/s connection would still require months to move the data from the source to its analysis location.

The communication, analysis, and storage of data from these experiments will only be possible through aggressive data reduction that is capable of shrinking data sets by one or more orders of magnitude. Aggressive data reduction techniques already exist but in a very ad hoc form. The LHC, for example, already reduces the data produced by the detectors and plans to reduce the data even more in Run 3 (Albrecht, 2016). The raw data per event is about 1 megabyte for Atlas and the Compact Muon Solenoid (CMS), and 100 kilobytes for Large Hadron Collider beauty (LHCb). Atlas currently produces events at 100 MHz for Run 2, CMS currently produces events at 100 MHz for Run 2, and LHCb currently produces events at 1 GHz for Run 2. For Run 3, Atlas will produce events at 0.4 MHz, CMS will produce events at 0.5 MHz, and LHCb will produce events at 40 MHz.

These detectors will produce a gigantic amount of data at an extraordinary rate: 60 terabytes/s for ATLAS and CMS and 2 terabytes/s for LHCb. To tackle this unprecedented data flow, the Alice project has defined a new combined offline–online framework called "O2" that supports data flows and processing. It performs online compression of events to reduce the data rate of storage to approximately 20 gigabytes/s. For Run 3, the O2 framework design features 463 FPGA detectors for readout and fast cluster finding; 100,000 CPU cores to compress 1.1 terabytes/s data streams; 5000 GPUs to speed up the reconstruction; and 50 petabytes of disk storage to enable more precise calibration.

Aggressive data reduction is already used in the consumer environment, and the consumer big data domain is preceding the scientific domain on the systematic use of lossy data reduction. Most of the photos taken by smartphones or digital cameras are stored in a lossy compressed version. This is also true for music files, and digital music is stored in lossy compressed format on most devices. The projection made by Cisco about future Internet traffic is striking: in 2025, 80% of Internet traffic will be video streaming, which means that more than 80% of the data transiting on the Internet will be lossy compressed. Microsoft has already deployed FPGAs into its data center to accelerate Lempel–Ziv–Markov chain algorithm and JPEG compression as well as other frequent operations, such as encryption.

An important distinction between scientific and consumer big data domains is the specificity of the data reduction techniques. As mentioned previously, aggressive data reduction techniques in the scientific domain are currently ad hoc. Conversely, the consumer big data domain relies on generic compressors (e.g. JPEG for images, MP3 for audio, and MPEG4 for video). This also reveals a certain advance of the consumer big data domain over the scientific domain. An important push toward the use of generic lossy compressors in scientific applications is the trend toward a generalization of its usage. The United States' Exascale Computing Project helped identify and better quantify these needs. Many scientific applications at extreme scale already need aggressive data reduction. Spatial sampling and decimation in time are used to reduce data, but these techniques also significantly reduce the quality of the data analytics performed on the sampled or decimated data sets. Advanced lossy compression techniques provide a solution to this problem by enabling the user to control the data reduction error. Another important distinction between scientific and consumer big data domains is the difference in quality assessment of the reduced data set. JPEG, MP3, and MPEG4 are not only generic, they are also universal: All users have the same perception of images and sound. This allows for defining compression quality criteria that correspond to a very large population of users. This is not the case in the scientific big data domain, where each combination of application and data set may lead users to define different sets of quality criteria. One open question is the relevant set of quality criteria for scientific data sets. Users have already expressed the need to assess spectral alteration, correlation alteration, the statistical properties of the compression error, the alteration of the first-order and second-order derivatives, and more. As the domain of lossy data reduction for scientific data sets grows, the community will learn what metrics are relevant and needed.

Although compression is critical for enabling the evolution of many scientific domains to the next stage, the technology of scientific data compression and the understanding on how to use it are still in their infancy. The first piece of evidence is the lack of results in this domain: over the 26 years of the prestigious Institute of Electrical and Electronics Engineers (IEEE) Data Compression Conferences, only 12 papers identify an aspect of scientific data in their title (e.g. floating-point data, data from simulation, numerical data, and scientific data). The second piece of evidence is the poor performance on some data sets.

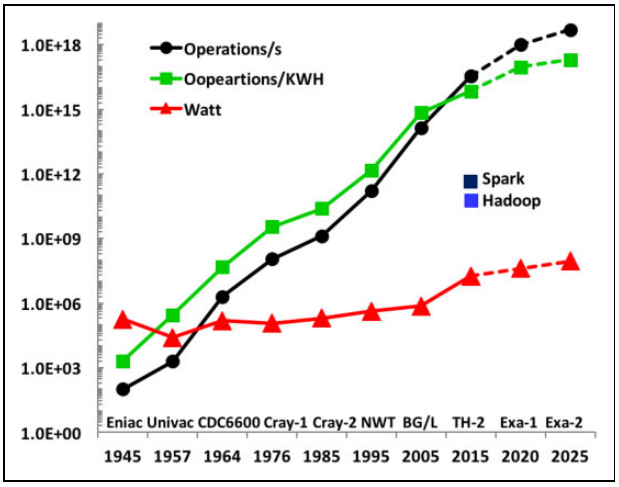

Figure 11. Trends of speed, energy efficiency, and power consumption of the world's fastest computers (Lu et al., 2014; Xu et al., 2016).

Beyond the research on compression, scientists also need to understand how to use lossy compression. The classic features of compressors (e.g. integer data compression, floating-point data compression, fast compression and decompression, and error bounds for lossy compressors) do not characterize compressors, specifically, with respect to their integration into an HPC and HDA workflow. For example, in the APS's IC imaging application, assuming a lossy compressor capable of 100× compression, can we perform the tomography and the data analytics directly from the compressed data? Obviously, if the subsequent analysis steps can only work from decompressed data, large storage and significant decompression time will be needed. If the data needs to be decompressed, can we decompress it only partially to allow for pipelined decompression, reconstruction, and analytics? Note that partial decompression requires random access to the compressed data set—a capability that is not systematically considered a priority today in aggressive data reduction techniques. The same set of questions applies to large-scale simulations: if we can avoid data sampling and decimation and compress the raw data set by a factor of 100, can the following data analytics steps be performed on the compressed data?

5.3. Architecture

5.3.1. Radically improved resource management for next-generation workflows. As large HPC systems become major nodes in data-intensive workflows, which encompass not just classical HPC applications, but also big data, analytics, machine learning, and more, it becomes important to provide both the hardware and software support to run those workflows as seamlessly as possible. It is certainly clear that the diverse application communities described in Section 3.3 would benefit from a combination of HPC, data-intensive, and high-throughput computing resources. Many

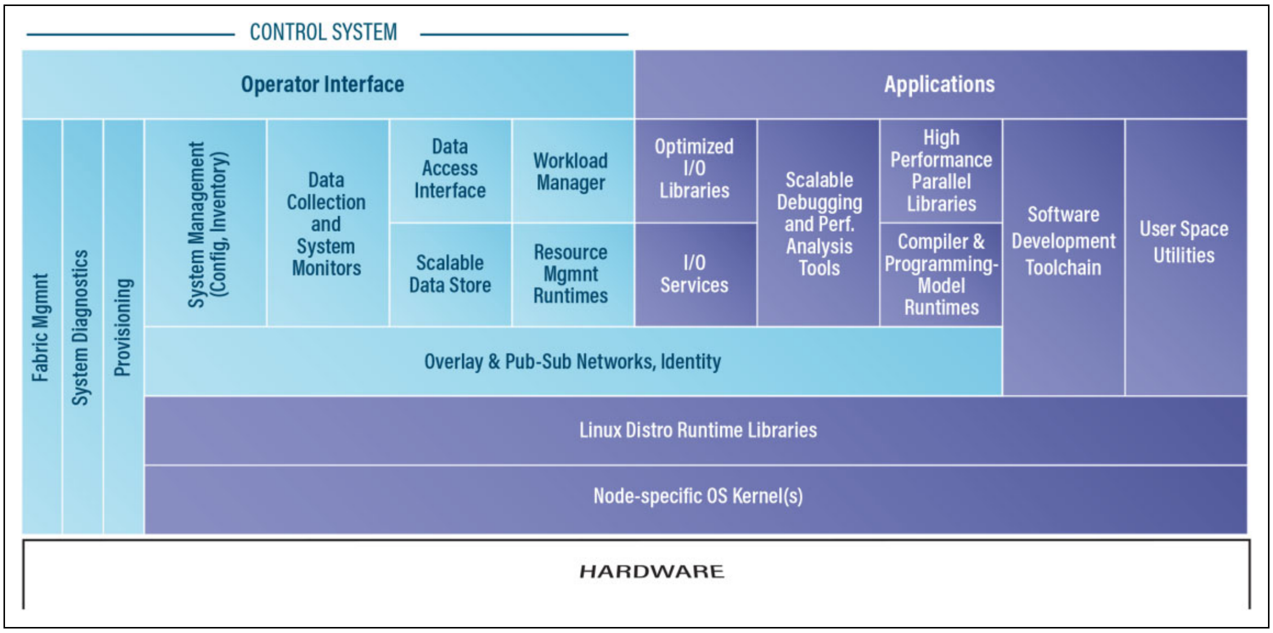
CONTROL SYSTEM
Operator Interface
Applications
Fabric Mgmnt
System Diagnostics
Provisioning
System Management (Config, Inventory)
Data Collection and System Monitors
Data Access Interface
Workload Manager
Scalable Data Store
Resource Mgmnt Runtimes
Optimized I/O Libraries
I/O Services
Scalable Debugging and Perf. Analysis Tools
High Performance Parallel Libraries
Compiler & Programming-Model Runtimes
Software Development Toolchain
User Space Utilities
Overlay & Pub-Sub Networks, Identity
Linux Distro Runtime Libraries
Node-specific OS Kernel(s)
HARDWARE

Figure 12. The OpenHPC modular software stack, with the components of the control system, required for a flexible workflow configuration, on the left.

of the requirements of such communities are summarized as follows:

- *Software stack*: Ability to run an arbitrarily complex software stack on demand.
- *Resilience*: Ability to handle failures of job streams and ability to checkpoint/restart applications (ideally using application-based methods) for dynamically allocating resources to urgent computing/postprocessing workloads (e.g. massive amount of data coming from large-scale instrumentation, clinical data from hospitals)
- *Resource flexibility*: Ability to run complex workflows with changing computational "width."
- *Centralized intelligence of the resource manager*: Knowing the behavior/request of each application, the policies of the center, the current status of the systems, and the number of jobs running/to be executed, with the ability to "smart schedule" jobs on the system with the appropriate level of resource allocation (in terms of energy, memory, and storage placement).
- *Wide-area data awareness*: Ability to seamlessly move computing resources to the data (and vice versa where possible); ability to access remote databases and ensure data consistency.
- *Automated workloads*: Ability to run automated production workflows.
- *End-to-end, simulation-based analyses*: Ability to run analysis workflows on simulations using a combination of in situ and offline/co-scheduling approaches.

Although the notion of being able to launch multiple application workflows from the exemplars presented in Section 3.3 in a converged manner is attractive, having to launch them on a single converged hardware platform is not. Fortunately, however, this is a place where the hourglass model can be beneficial. We want to be able to support workflows consisting of multiple and divergent applications (e.g. classical HPC, machine learning, and visualization), the top portion of the hourglass. We also envision, as described in the hardware trends section, an increasing proliferation of hardware—be it in processors, accelerators, FPGAs, and so on—to target these divergent applications. This is positive, as this hardware will best meet (in terms of performance, power, and cost) the needs of the different applications. This is the bottom portion of the hourglass. However, in an ideal world, the application would just be built and execute on the correct hardware. Thus, the middle, narrow portion of the hourglass allows application developers to just be concerned with their code. Between the ideal world and what is implementable in an early version, there is a continuum of system software options that allow the application programmer to focus less on how to run their workflow and more on what interactions between different applications in their workflow need to occur. Thus, the goal of this section is to describe, from a system management perspective, the different levels we can take to move us toward the ideal world, where the users need only to care about their application.

We define "system management" as how a machine (or collection of machines) is controlled via system software to boot, execute workflows, and allow administrators or users to interact with and control the system. The "control system" is comprised of a set of components in the software stack that enable this functionality. From the OpenHPC[18] software stack depicted in Figure 12, we define the "control system" to be those components shown on the left side in blue. Many of these components need to be extended to provide a converged view and operation of the machine—especially in the face of divergent hardware. As described earlier, there is a continuum from a manually (by application effort) converged machine to a fully automated one.

We view the roadmap to successful convergence as moving along this continuum, freeing the user from the responsibility of managing the underlying machines themselves. This accomplishes two things. It makes the user wishing to leverage a converged system more productive. More importantly, it opens up such an architecture to a much broader user base. Without system software efforts in moving us toward the converged hourglass model, it will be difficult to gain widespread use of the increasingly complex machines.

So we can clearly describe the different capabilities needed, we divide the continuum into the following discrete points.

C1: The user workflow (via scripting) knows about the different underlying machines and architectures and launches its individual application on the various machines, interacts with the various resource management and monitoring components, and collates the data coming from the different machines and applications. The user must manage the individual steps in the workflow and ensure that the data is correctly transferred and is where it needs to be.

C2: The user workflow knows about the different underlying machines and architectures and indicates where and how different applications in the workflow should run. The underlying control system manages the launching of the different applications and the staging of the data before and between the application execution. It returns to the user when the workflow is finished.

C3: The user specifies the workflow by describing each application. The compiler, runtimes (e.g. MPI, partitioned global address space, and Open Multiprocessing (OpenMP)), and control system automatically construct the workflow, launch the individual applications, shepherd the data to the applications, collate the results, and present the collated results of the workflow to the user. Note: there is probably a reasonable point between C2 and C3, but given that C2 is a significant piece and likely challenging enough in the short to middle term, we leave the various aspects of moving from C2 to C3 as good topics for advanced research without presupposing how they would be divided.

We now describe the system software including the control system work that needs to be accomplished for each of the points. Even though C1 appears to place all of the burden on the user, there is still control system work that needs to occur to obtain a reasonable C1. In particular, today's accelerators, FPGAs, and nonstandard core computing elements are not well understood by the operating system, resource manager, or monitoring and control systems. More work needs to be put into enhancing these components to comprehend the new types of hardware that will be available. Much, but probably not all, of this work will be undertaken by vendors producing new hardware. However, there will still be open-source work (e.g. Slurm in the RM space or the performance API in the monitoring space) that either the vendors would need to fund or the community will need to contribute.

As indicated, C2 is the desired intermediate step, where we believe efforts should be directed to achieve widespread use of the converged system. Significant control system work must be undertaken to achieve C2. First, high-level architecture work should determine the best way to bring machines together. We use resource management as an example, but it applies for many of the other components (e.g. provisioning and fabric management). Work should also be done to determine whether there should be a single resource manager that spans each of the underlying machines or whether there should be an overarching (new) resource manager that knows how to communicate with the resource managers on each machine. Either path is a nontrivial effort. It is likely the latter can be more easily achieved because individual resource managers will be developed to handle each machine—by the open-source community, by the vendor producing a given machine, or by a combination of both. However, the downside of this overarching resource manager approach is that there will likely be inefficiencies and potentially missing functionality in each individual manager. The positive side of this approach is that the effort to build even a single resource manager and keep it current with the broadening architecture is very large, so keeping the sum total effort reduced is a plus. The overarching approach also has the advantage of being able to more readily leverage particular features put in by a particular resource manager for a given hardware platform. A hybrid approach is also possible. There are likely to be resource managers that work on many of the underlying hardware platforms, and some additional capability could be added with modest cost to enable them to recognize that they are managing a machine in a group of machines—where another of those machines is managed by another copy of the same resource manager.

In addition to this, the control system components must be made aware of the different types of machines and be able to map between the user-specific "where and how" to the underlying machine. In this model, the components also (given a sufficiently rich enough description) need to be able to ensure that the data are available to a particular application of a workflow at the right time in the right place. This could be as simple as a single, globally shared file system. However, given the impending memory hierarchy architecture, it is much more likely to mean that many of the components (e.g. provision and resource managers) need to be connected to a data manager that knows in what nonvolatile memory (NVM) a particular data set resides and also has the capability to move the data around and pre-stage it for efficiency. In this model, the user still needs to have knowledge of the various applications in their workflow and where they would like those applications to run.

Third, C3 represents an ideal state, where the user writes the applications in the workflow and describes how they are related. During the compilation phase, information is extracted to determine the best underlying architecture on which to run. The various runtimes move the computations dynamically between the different hardware types on a given machine, and the control system components move the computation between different machines at the appropriate times. Data are moved around between the NVM connected to each of the computing components. This removes the burden of determining the best type of hardware on which to run, and it removes the burden of creating the workflow specifying where the individual applications need to run each application (or even subsets or phases of applications). While there is considerable and unknown research work that needs to be accomplished, some aspects of this work are already underway. For example, the OpenMP runtime manages systems with standard cores and GPUs, or cores and accelerators, and moves computation automatically based on monitoring information from the application. For C3, necessary components will likely be researched in the medium term, with the overall puzzle being filled in based on research trends and availability of funding.

Having described the steps for system software along the path toward providing a converged machine, including the different possible points along the path and the work needed to achieve them, it is important to note that OpenHPC is not the only implementation strategy. In particular, OpenStack[19] provides a set of alternative implementations to many of the control system components depicted in Figure 12. For example, OpenHPC provides validated recipes using bare-metal provisioners like Warewulf and xCAT, while OpenStack uses Glance and Ironic for image management and provisioning. OpenStack has focused on the cloud, while OpenHPC has focused on classical HPC. As this report exemplifies, there is interest from both sides for bringing the stacks together. There are two approaches. It is possible to use an OpenStack environment to deploy OpenHPC runtimes and development tools across a set of compute resources. The nodes would run with the high performance of OpenHPC and provide HPC performance in a cloud environment. The other approach would be to run elements of OpenStack within an OpenHPC environment. The former approach was demonstrated as a proof of concept at SC16. Other work in this vein has been to run an instance of OpenHPC in Azure. These examples demonstrate the value of having a more unified approach for some types of workflows.

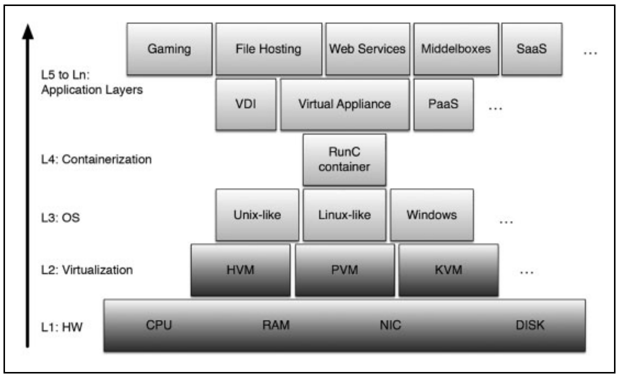

Figure 13. One vision of containerization as the narrow waist of the system software stack. Credit: Fu et al. (2016).

5.3.2. Considering containerization as the new "narrow waist".

A typical HPC application code is composed of source code (Fortran, C, C++, and Python), which is explicitly developed and managed by the application development team, and of links to reusable component libraries. Commonly used libraries like MPI and basic linear algebra subprograms (BLASs) are often obtained from pre-built object code that is installed and managed by a system administrator. However, other libraries like solvers, I/O tools, and data analysis packages are often downloaded from a website supported by the library development team and built from source code.

This model of building from source is accepted and common in the HPC community, but unfamiliar to much of the HDA community. Although software components developed by the HPC community may have much to offer the HDA community in terms of scalable computational and data management capabilities, the build-from-source assumption is a nonstarter.

One of the first places where this spanning layer analysis is likely to be used is in the consideration of "container" technology (Figure 13). Containerization has emerged as a technology that, already used in HDA, has strong potential to serve as a software bridge to the HPC community. Containers avoid the runtime overhead of VMs by being integrated into the host kernel, thereby making execution times within a container essentially the same as those from a native installation on a given system. This fact has spurred interest on the part of HPC system vendors to start supporting containers on leadership-computing platforms. For this reason and others, and as shown in Figure 13, containers are already being considered as a potential spanning layer for cloud infrastructure (Fu et al., 2016). Since containers have emerged as a viable delivery platform for HPC software, and are already widely used in HDA, they have become an attractive target for convergence between HDA and HPC. Packaging reusable HPC software components in containers dramatically reduces the usage barrier for the HDA community (and the HPC community), providing a large collection of high-performance capabilities never seen before by the HDA community. For the HPC community, containers also dramatically reduce the usage barrier, thereby making previously challenging workflow setups almost trivial, and providing a portable software environment on many HPC systems. Also, once containers are adopted by the HPC community, HDA components become available to the HPC community. Thus, the bidirectional benefits of containerization are very compelling.

The HPC community is just starting its exploration of the potential for containerization. Further HPC opportunities include isolation to reduce performance variability due to kernel interrupts, coexistence of distinct software stacks on the same hardware platform, launch-time detection of special compute devices such as GPUs, and more. Issues to be explored are listed as follows:

- The use of containers promises greater portability and isolation. The former reduces or eliminates sensitivity to differences in the underlying platform, thereby increasing flexibility when multiple platform versions are in use; and the latter provides better protection from performance variability by reducing interrupts in multi-kernel environments.
- The enhanced portability provided by containers suggests that they might be used to create a spanning layer that provides a bridge between the data center/cloud and edge systems. In that context, two issues will need to be explored. First, loading the entire runtime environment of an indeterminate number of processes on an edge system—and making available the CPU, memory, and other resources to enable them to run to completion—is a heavy lift. A thin edge node may not be able to support many, or even a single large container. The second issue is that the runtime environment in such a scenario is indivisible. A process invoked on one processor may be tied to that processor until it completes. A container that has allocated, and is using, resources within the host operating system may have to reside on that host until its entire cohort of processes has completed. If changing or unexpected events prevent the successful completion of the container's execution, it may or may not be possible to checkpoint and migrate the environment, depending on the characteristics of the host operating system.
- While the HDA community's container ecosystems are already very valuable to the HPC community, notable security features—essential for any effective deployment in a multiuser environment—are currently missing. But HPC environments are always multi-job, multiuser environments. Furthermore, sensitive system and user data are typically present on the system at any given time. Common container environments such as Docker [cite] effectively permit root access from the container into the host environment, which is clearly unacceptable on an HPC platform. Augmented environments such as Shifter [cite] can encapsulate a standard container and improve the security and device detection capabilities needed for HPC.

5.4. Software

As the new era of big data and extreme-scale computing continues to develop, it seems clear that both centralized systems (i.e. HPC centers and commercial cloud systems) and decentralized systems (i.e. any of the alternative designs for edge/fog infrastructure) will share many common software challenges and opportunities. For example, continued exponential growth in data volumes will make data reduction of one form or another indispensable on all fronts. None the less, we locate much of the discussion of software issues and possibilities in this section on centralized infrastructure, because—despite the recognized differences and uncertainties—architectural and hardware questions for HPC and cloud computing environments are still far clearer than they are for edge/fog environments, where proposals, blueprints, and promises now prevail. Thus, many of the software issues that are discussed in this section will need to be revisited for edge/fog environments as their platform models acquire more definition and begin to stabilize.

5.4.1. Software libraries for common intermediate processing tasks. One common theme in the workflow descriptions at the BDEC workshop was the amount of "intermediate" (or pre) processing that data require before the more substantial analysis and visualization processes occur. For instance, we describe "multi-messenger" forms of inquiry in Section 3.3, which tend to exhibit a relatively common set of requirements, listed as follows:

- Identify and segment trillions of multi-scale objects from spatiotemporal data sets.
- Extract features from objects and spatiotemporal regions.
- Support queries against ensembles of features extracted from multiple data sets.
- Use statistical analyses and machine learning to link features to physical and biological phenomena.
- Use feature-driven simulation; extracted features used as simulation initial, boundary conditions, and to assimilate data into simulations.
- Develop and enrich in situ/in transit framework with machine/deep learning capabilities for on-the-fly automatic pertinent structures detection/extraction for both static and dynamic analyses.
- As a reverse loop, perform smart computational steering of the application—also seen in Section 3.3.5.

Many of the intermediate transformations in this list are normally described in generic terms: cleaning, subsetting, filtering, mapping, object segmentation, feature extraction, registration, and so on. The question is whether or not some of these operations are generic enough that a common set of software tools—appropriately layered and modularized—could be developed to serve the diverse purposes of a number of different communities at the same time. For example, image-driven workflows from fields such as medical imaging, microscopy, and remote (satellite) sensing, utilize all of the operations given earlier. Although the codesign effort that would probably be necessary to produce it would be challenging to organize, common software infrastructure that (suitably configured) could satisfy intermediate processing needs in a wide variety of fields would be a boon to data-driven research.

- *A common and visible data model*: One major obstacle to creating shared software infrastructure for intermediate processing is the absence of interoperable data object models, or, just as importantly, a way of making the object model being used visible. The effort to develop a common model achieved limited success when object-oriented databases were introduced in the 1990s. However, that success was largely restricted to tightly coupled systems, and these tools did not succeed for many more loosely coupled situations, which are typical of today's many emerging BDEC domains and workflows. Web-based approaches (e.g. the representational state transfer API) are likely to be viable for only a relatively small segment of these big data applications. Common object models have been established in some application domains (e.g. multi-physics applications and climate modeling), but creating a common software stack that supports more general interoperability has proved elusive. Moreover, for any such model to succeed, it will need to be flexible enough to provide data layout distribution options to support the kind of parallelism that applications and I/O services will require.
- *Shared software infrastructure for intermediate processing*: The digitization of all scientific data has opened up a major opportunity space for research methods that integrate or synthesize data of multiple types and/or from multiple sources or sensor modalities. This is particularly true for application areas, now common, that utilize and combine multidimensional, spatial-temporal data sets. Examples include radioimaging and microscopy imaging combined with "omic" data; simulation data (e.g. for oil fields, carbon sequestration, and groundwater pollution/remediation) combined with seismic and earth sensor data; and weather prediction based on the real-time integration of data from simulations, satellites, ground sensors, and live video feeds. The Google self-driving car provides a more practical consumer illustration of real-time integrated analysis of correlative data from multiple sensor modalities and sources. The multidimensional data space that these applications define tends to be high resolution in each of their correlative dimensions, so that, when even a modest number of data steps are involved, extremely large volumes of data need to be accessed and processed in a coordinated way.

5.5. Math libraries, software ecosystems for application development

5.5.1. Leveraging HPC math libraries in HDA. The HPC community has a large and growing collection of high-performance math libraries for scientific computing. Many of these libraries have been carefully designed and implemented to exploit scalable parallel systems, from multi-core and GPU-enabled laptops to the largest computing systems in the world. At the same time, these libraries are typically used by the HPC community in the form of user-compiled source code. Furthermore, the interfaces to these libraries are complex, with many options that require substantial experience in mathematical algorithms and parallel computing.

The advent of component-based software ecosystems like Docker enables pre-compilation and predefined parameterization of HPC math libraries for dedicated problem solving in a component software ecosystem. For example, the Trilinos eigensolver package, "Anasazi," has many algorithmic and parameter options that can be adapted to provide capabilities for a variety of problems. Furthermore, Anasazi has a complex build environment that supports optimized compilation on a variety of platforms. While attractive to experts, the sheer range of choices can be a nonstarter for someone who simply wants a solver for large sparse eigenvalue problems.

Containers provide an opportunity to encapsulate the complexity of a solver like Anasazi by supporting pre-compilation of the source and a simplified interface, such that the resulting container can be considered a filter that takes a sparse matrix as input and produces eigenvalues and eigenvectors as output; executes on a laptop, parallel cluster, or supercomputer; and provides a portable workflow environment for the user. This capability will enable turnkey use of sophisticated solvers for both HPC and HDA users.

5.6. New efforts for dense linear algebra standards

The emergence of new HDA markets has created, renewed, and expanded interest in standard functionality and interfaces for dense linear algebra. The so-called "batched BLAS" is a new standards effort looking to efficiently compute dense linear operations on a large collection of matrices at the same time. These kernels have always been of interest in finite element computations, part of the HPC community, but have never had the market potential to drive a standard. The emergence of "deep learning" algorithms in HDA provides new incentives, and the linear algebra community is now working toward a standard.

Batched BLAS and other potential standards that can benefit both HPC and HDA represent a synergistic opportunity that would not be otherwise easy to exploit.

5.7. Challenges in the HPC software ecosystem

While in many ways the HPC software ecosystem is rich, stable, and provides a ready-made environment for attaining good performance, there are some challenges that only time, exploration, and "coopetition" can resolve. Of particular importance are standards for shared memory parallel programming. While MPI is the ubiquitous and acknowledged standard for internode (across node) parallel programming and execution, intra-node parallel programming and execution environments are not nearly as stable. Presently, there are two dominant efforts, OpenMP and Open Accelerators (OpenACC), that are targeting two distinct approaches to node-level parallelism.

OpenMP and its predecessors have been available for more than three decades, and these programming models are particularly suitable for multi-core CPUs and many-core parallel processors (e.g. Intel Xeon Phi or GPUs). OpenACC started as a forked effort of OpenMP, with the intent to focus more specifically on accelerators like NVIDIA GPUs. While there is resolve on the part of the HPC community to bring OpenMP and OpenACC back together, the obvious self-interest of specific vendors leads to competitive concerns that fundamental design choices could bias the community for or against a particular architecture. This "coopetition" is healthy and necessary to ensure a truly portable standard that can support the all-important intra-node parallelism approaches. Even so, the present state of uncertainty makes the writing of portable intra-node parallel code particularly challenging at this time.

5.7.1. Interoperability between programming models and data formats. While HPC programming applications have traditionally been based on MPI to support parallel and distributed execution, and based on OpenMP or other alternatives to exploit the parallelism inside the node, big data programming models are based on interfaces like Hadoop, MapReduce, or Spark. In addition to different programming models, the programming languages also differ between the two communities—with Fortran and C/C++ being the most common languages in HPC applications, and Java, Scala, or Python being the most common languages in big data applications.

This divergence between programming models and languages poses a convergence issue, not only with regard to interoperability of the applications but also to the interoperability between data formats from different programming languages. In this scenario, we need to consider how to build end-to-end workflows, providing a coordination layer that enables the management of dynamic workflows composed of simulations, analytics, and visualizations—including I/O from streams. In such a scenario, simulations can be MPI applications written in Fortran or C/C++, and the analytics codes can be written in Java or Python (maybe parallelized with Spark). To enable the efficient exchange of data between the simulations and analytics parts of the workflows, other means beyond traditional Portable Operating System Interface (POSIX)-based (POSIX) files should be considered. Alternatives for implementing this are being considered, with approaches like dataClay or Hecuba, which provide persistent storage libraries and tools. However, further research still remains to better support data interoperability between different programming languages.

6. Conclusions and recommendations

The goal of the BDEC workshops has been to develop an ICT planning document for science and engineering that articulates an analysis and vision of the conjoint evolution of data-intensive research and extreme-scale computing. As we argued previously, however, since there is no widely agreed upon model for the new kind of DSP that data-intensive workflows of the big data era seem to require, traditional technology road mapping techniques may be inappropriate. Following the structure of the document, we divided our findings and recommendations into three categories: (1) global recommendations, (2) recommendations for edge environments, and (3) recommendations for centralized facilities. However, our ultimate goal is to prepare the ground for the kind of community-driven "shaping strategy" (Hagel and Brown, 2017; Hagel et al., 2008) approach that we believe would be both more appropriate and more successful (Section 2.2). Consequently, the conclusions as they appear in the following section may have to be refactored to serve the shaping strategy model.

6.1. Global recommendations

The major recommendation is to address the basic problem of the two paradigm splits: the HPC/HDA software ecosystem split and the wide-area data logistics split. For this to be achieved, new standards are needed to govern the interoperability between data and compute. However, if we want a new distributed infrastructure to support science and engineering research in the era of big data—an infrastructure with the kind of openness, scalability, and flexible resource sharing that has characterized the legacy Internet paradigm—then we will have to define a new, common, and open DSP, one that offers programmable access to shared processing, storage, and communication resources, and that can serve as a universal foundation for the component interoperability that novel services and applications will require. As the data revolution continues, such well-designed DSP infrastructure will be necessary to support such compute-intensive and/or data-intensive work that many application areas will have to carry out between the ingress/egress to the cloud (or data/HPC center) and the network edge. As the history of the Internet shows, the scientific community is, with appropriate public investment in basic research and development, uniquely positioned to create and develop the kind of DSP that the emerging era of extreme-scale data and computing requires, building on the kind of open, consensus-driven approach that helped establish the Internet.

6.2. Recommendations for decentralized facilities for edge and peripheral ecosystems

1. *Converging on a New Hourglass Architecture for a Common DSP*: The "hourglass" represents the idea that an appropriately designed common interface can be implemented on an ever-increasing variety of technology platforms (yielding a wide "lower bell"), while at the same time supporting an equally diverse and growing variety of applications (yielding a wide "upper bell"). The common interface, or "thin waist of the hourglass," is called the "spanning layer" because it bridges, through virtualization, a heterogeneous set of resources that lie below it but leaves the application and services above it free to evolve independently. This point clearly ties in with the global recommendation above. Unfortunately, in seeking a new spanning layer to address the challenges of the big data era, the science cyberinfrastructure community finds itself in something of a dilemma. On the one hand, at present, there does seem to be at least one plausible and widely touted candidate for the new spanning layer—operating system–level virtualization that supports software "containers." Certainly, converging on a common interface for containerization would go a long way to achieving ecosystem convergence and do so in way that requires something closer to evolutionary, as opposed to revolutionary, changes to current modes of operation.

   Containerization should thus be a very active area of research and experimentation across all contexts that scientific cyberinfrastructure will have to address, including commercial clouds, HPC systems, and computing resources deployed in edge environments. At the same time, the fact that containers preserve legacy silos for storage, processing, and communication at a low-level, and may therefore bring with them unexpected impediments to interoperable convergence, suggests that other ideas for a new spanning layer should also be aggressively pursued.
2. *Target Workflow Patterns for Improved Data Logistics*: There seem to be at least four nonexclusive alternatives for interfacing HPC to this new DSP paradigm, in which no strong assumptions are made about where the data are located: (1) data streaming, (2) in transit processing, (3) processing at the edge of the distributed system (i.e. as close as possible to the data sources), and (4) logically centered cloud-like processing. These should be the basis for new research funding with an applications-oriented objective.
3. *Cloud Stream Processing Capabilities*: Stream processing in cloud computing was not designed with HPC in mind, and there is a need to examine the high performance aspects of the runtimes used in this environment.
4. *Content Delivery/Distribution Networks*: Commercial CDNs are expensive to run and create barriers to interoperation. To resolve, this requires (1) a scalable approach to CDN implementation (i.e. suitably designed forms of storage and processing at the nodes of the distribution tree) and (2) the aggregate organizational will of the scientific community.
5. *Software Libraries for Common Intermediate Processing Tasks*: One common theme in the workflow descriptions is the amount of "intermediate" (or pre-) processing that data require before the more substantial analysis and visualization processes can occur. Some of these operations are generic enough that a common set of software tools—appropriately layered and modularized—could be developed to serve the diverse purposes of a number of different communities at the same time.

6.3. Recommendations for centralized facilities

1. *Energy as an Overarching Challenge for Sustainability*: We can identify four steps toward energy minimization: (1) reduce computational costs by using platforms that are well-matched to the stage within the scientific method; (2) reduce data-movement costs by using collocation, compression, and caching; (3) encourage reuse of calculations and data through effective sharing, metadata, and catalogs—a strategy that a provenance system supports well; and (4) reduce computing system entropy (e.g. workload interference, system jitter, tail latency, and other noise) through on-demand isolation, noise-resistant priority, cache QoS, and novel uncertainty bounding techniques. The cyberinfrastructure itself has the task of taking care of energy minimization as it has access to the required information; leaving this burden to the domain scientists is undesirable, since it would divert them from their scientific goals.
2. *Data Reduction as a Fundamental Pattern*: The communication, analysis, and storage of data from large scientific experiments will only be possible through aggressive data reduction that is capable of shrinking data sets by one or more orders of magnitude. Although compression is critical to enabling the evolution of many scientific domains to the next stage, the technology of scientific data

compression and the understanding of how to use it are still in their infancy. Beyond the research on compression, scientists also need to understand how to use lossy compression. If the data need to be decompressed, can we decompress it only partially to allow for pipelined decompression, reconstruction, and analytics? The same set of questions applies to large-scale simulations: If we can avoid data sampling and decimation and compress the raw data set by a factor of 100, can the data analytics be performed on the compressed data?

3. *Radically Improved Resource Management*: As HPC workflows start encompassing not just classical HPC applications, but also big data, analytics, machine learning, and more, it becomes important to provide both the hardware and software support to run those workflows as seamlessly as possible. We define "system management" to be how a machine (or collection of machines) is controlled via system software to boot, execute workflows, and allow administrators or users to interact with and control the system. The roadmap to successful convergence requires freeing the user from the responsibility of managing the underlying machines themselves.
4. *Software Issues*: As the new era of big data and extreme-scale computing continues to develop, it seems clear that both centralized systems (e.g. HPC centers and commercial cloud systems) and decentralized systems (e.g. any of the alternative designs for edge/fog infrastructure) will share many common software challenges and opportunities. Eminent among these are the following needs.
   - Leverage HPC math libraries for HDA;
   - Increase efforts for dense linear algebra standards;
   - Develop new standards for shared memory parallel processing; and
   - Ensure interoperability between programming models and data formats.
5. *Machine Learning*: Machine learning is emerging as a general tool to augment and extend mechanistic models in many fields and is becoming an important component of scientific workloads. From a computational architecture standpoint, DNN-based scientific applications have some unique requirements. They require high compute density to support matrix–matrix and matrix–vector operations, but they rarely require 64-bit or even 32-bit precision arithmetic, thus architects should continue to create new instructions and new design points to accelerate the training stage of the neural network. Most current DNNs rely on dense, fully connected networks and convolutional networks and are thus reasonably matched to current HPC accelerators (i.e. GPUs and Xeon Phi). However, future DNNs may rely less on dense communication patterns. In general, DNNs do not have good strong-scaling behavior. So, to fully exploit large-scale parallelism, they rely on a combination of model, data, and search parallelism.
   Deep learning problems also require large quantities of training data to be made available or generated at each node, thus providing opportunities for nonvolatile random access memory. Discovering optimal deep learning models often involves a large-scale search of hyperparameters. It is not uncommon to search a space of tens of thousands of model configurations. Naive searches are outperformed by various intelligent searching strategies, including new approaches that use generative neural networks to manage the search space. HPC architectures that can support these large-scale intelligent search methods, and also support efficient model training, are needed.

Acknowledgments

The authors would like to acknowledge David Rogers for his work on the illustrations, Sam Crawford for editing support and the creation of the Appendix, and Piotr Luszczek for technical support. They would also gratefully acknowledge all the following sponsors who supported the big data and exascale computing workshop series: *Government Sponsors*: US Department of Energy, the National Science Foundation, Argonne National Laboratory, European Exascale Software Initiative, and European Commission; *Academic Sponsors*: University of Tennessee, the National Institute of Advanced Industrial Science and Technology (AIST), Barcelona Supercomputer Center, Kyoto University, Kyushu University, Riken, The University of Tokyo, the Tokyo Institute of Technology, and the University of Tsukuba Center for Computational Sciences; *Industry Sponsors*: Intel, Cray, Data Direct Networks, Fujitsu, Hitachi, Lenovo, and NEC. The authors would also like to thank Jean-Claude André and Gabriel Antoniu for their valuable contributions to this work.

Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

Notes

1. http://www.eesi-project.eu.
2. COM(2016) 178 of 19/4/2016—European Cloud Initiative—Building a competitive data and knowledge economy in Europe.
3. The system and application software of the original dot. com revolution had much in common with

contemporary high-performance computing infrastructure. That is far less true today.

4. One early exception was data from the Large Hadron Collider experiments in CERN; the global high-energy physics community had both the organization and the clout to command sufficient government funding to support a content distribution network and distributed computing infrastructure, purpose built to meet their needs. Astronomy and astrophysics is another example. The International Virtual Observatory Alliance provides a good example of an international organization that nurtured web-services standards; data representation standards like the flexible image transport system; and other standards for accessing astronomic data, exchanging data between virtual observatories, tools to communicate, and analysis software. The result is that 90% of the world's astronomy data is reachable, and real science can be done by using the tools and software that are being built all around the world. The same is true for seismology with the international federated digital seismic network organization.
5. Please see http://www.exascale.org/bdec/ for a list and descriptions.
6. Adapted from illustration in *Abduction and Induction: Essays on their relation and integration* (Flach and Hadjiantonis, 2013).
7. Summarized version of Feynman's account provided by Victor Baker in "The pragmatic roots of American Quaternary geology and geomorphology" (Baker, 1996).
8. http://dsc.soic.indiana.edu/publications/NISTUseCase.pdf.
9. http://www.andrewng.org/portfolio/map-reduce-for-machine-learning-on-multicore/.
10. Just as multi-messenger astronomy is "the coordinated observation and interpretation of disparate 'messenger' signals" from the same astronomical objects (Wikipedia, 2017c), multi-messenger scientific inquiry combines observations and interpretations of disparate streams of sensor or instrument data of the same objects in an integrated, inferential process.
11. https://www.cancer.gov/research/key-initiatives/moonshot-cancer-initiative.
12. Between 2009 and 2013, the United States' Food and Drug Administration approved 20 new oncologic drugs, having a treatment costs of US$100,000 per year and an average progression-free survival improvement of less than 6 months.
13. http://candle.cels.anl.gov.
14. A family of proteins whose over activity can ultimately lead to cancer.
15. In the 6th Annual Earth System Grid Federation conference report (see http://esgf.llnl.gov/media/pdf/2017http://-ESGFF2FConferenceReport.pdf), a major concern expressed was to minimize the time spent finding, using, and storing data.
16. This point reminds one of Ken Batcher's well-known quip that, "A supercomputer is a device for turning compute-bound problems into I/O-bound problems" (Wikipedia, 2017a), the consequential truth of which is destined to be reinforced by emerging exascale systems.
17. Recent estimates place this as high as 10%.
18. http://www.openhpc.community/.
19. https://www.openstack.org/.

References

Abraham A, Michael P, Milham A, et al. (2017) Deriving reproducible biomarkers from multi-site resting-state data: an autism-based example. *NeuroImage* 147: 736–745.

Albrecht J (2016) Challenges for the LHC Run 3: computing and algorithms. Presentation at *International workshop on Advanced Computing and Analysis Techniques in physics research*, January 2016, UTFSM, Valparaso, Chile.

Anderson T, Peterson L, Shenker S, et al. (2005) Overcoming the Internet impasse through virtualization. *Computer* 38(4): 34–41.

Asch M, Bocquet M and Nodet M (2017) *Data Assimilation: Methods, Algorithms and Applications*. Philadelphia, PA: SIAM.

Attig N, Gibbon P and Lippert T (2011) Trends in supercomputing: the European path to exascale. *Computer Physics Communications* 182(9): 2041–2046.

Baker VR (1996) The pragmatic roots of American quaternary geology and geomorphology. *Geomorphology*, 16(3): 197–215.

Baker AH, Xu H, Dennis JM, et al. (2014) A methodology for evaluating the impact of data compression on climate simulation data. In: *Proceedings of the 23rd international symposium on high-performance parallel and distributed computing*, HPDC '14, pp. 203–214. New York: ACM. ISBN 978-1-4503-2749-7.

Banerjee S and Wu DO (2013) Final report from the NSF workshop on future directions in wireless networking. Washington, USA : National Science Foundation.

Bassi A, Beck M, Fagg G, et al. (2002) The Internet backplane protocol: a study in resource sharing. In: *Cluster computing and the grid, 2nd IEEE/ACM international symposium on*, pp. 194–194, May 2002.

Bastug E, Bennis M and Debbah M (2014) Living on the edge: the role of proactive caching in 5G wireless networks. *IEEE Communications Magazine*, 52(8):82–89.

Beck M (2016) On the hourglass model, the end-to-end principle and deployment scalability. Available at: http://philsci-archive.pitt.edu/12626/ (accessed march)

Beck M, Moore T and Luszczek P (2017) Interoperable convergence of storage, networking and computation. *arXiv preprint arXiv:1706.07519* 03 July 2017. https://dblp.org/rec/bib/journals/corr/BeckML17 (accessed 01 March 2018).

Bellucci F and Pietarinen AV (2017) *Charles Sanders Peirce: Logic*, chapter Charles Sanders Peirce: Logic. Internet Encyclopedia of Philosophy. Internet Encyclopedia of Philosophy, s.v. “Charles Sanders Peirce,” by Francesco Bellucci and Ahti-Veikko Pietarinen, http://www.iep.utm.edu/peir-log/.

Bennett JC, Abbasi H, Bremer P-T, et al. (2012) Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In: *Proceedings of the international conference on high performance computing, networking, storage and analysis (SC ‘12)*, pp. 1–9. Salt Lake City, UT, USA: IEEE Computer Society Press.

Bethel EW, Greenwald M, van Dam KK, et al. (2016) Management, analysis, and visualization of experimental and observational data—the convergence of data and computing. In: *e-Science (e-Science), 2016 IEEE 12th international conference on*, Baltimore, MD, USA, 23–27 October 2016. pp. 213–222. Piscataway: IEEE.

Bonomi F, Milito R, Zhu J, et al. (2012) Fog computing and its role in the Internet of things. In: *Proceedings of the first edition of the MCC workshop on mobile cloud computing*, Helsinki, Finland, pp. 13–16. New York: ACM.

Calyam P and Ricart G (2016) In: NSF Workshop on Applications and Services in the Year 2021. Washington, DC, 2016. Available at: https://asw2016.wordpress.com/

Cao VH, Chu KX, Le-Khac NA, et al. (2015) Toward a new approach for massive LiDAR data processing. In: *Spatial data mining and geographical knowledge services (ICSDM), 2015 2nd IEEE international conference on*, Fuzhou, China, pp. 135–140. Piscataway: IEEE.

Chang WL (2015) *NIST Big Data Interoperability Framework: Volume 5, Architectures White Paper Survey*. Gaithersburg: Special Publication NIST SP-1500-5.

Chard K, Caton S, Rana O, et al. (2012) A social content delivery network for scientific cooperation: vision, design, and architecture. In: *High performance computing, networking, storage and analysis (SCC), 2012 SC companion*, Salt Lake City, UT, pp. 1058–1067. Piscataway: IEEE.

Chen M, Mao S and Liu Y (2014) Big data: a survey. *Mobile Networks and Applications* 19(2): 171–209.

Cisco Global Cloud Index (2015) Forecast and methodology, 2014-2019. *Cisco White Paper*.

Clark D (1988) The design philosophy of the DARPA Internet protocols. *ACM SIGCOMM Computer Communication Review* 18(4): 106–114.

Clark DD (1997) *The unpredictable certainty: Information infrastructure through 2000*. Washington, DC: National Academy Press.

Dennard RH, Gaensslen F, Yu H-N, et al. (1974) Design of ion-implanted MOSFET’s with very small physical dimensions. *IEEE Journal of Solid State Circuits* 9(5): 38–50.

Desprez F and Lebre A (2016) Research issues for future cloud infrastructures: Inria position paper. Available at: https://fdesprez.github.io/research/papers/PDF/2016/report-cloud-2016.pdf (accessed 01 March 2018).

Dinov ID, Petrosyan P, Liu Z, et al. (2014) The perfect neuroimaging-genetics-computation storm: collision of petabytes of data, millions of hardware devices and thousands of software tools. *Brain Imaging and Behavior* 8(2): 311.

Dongarra J, Beckman P, Moore T, et al. (2011) The international exascale software project roadmap. *International Journal of High Performance Computing Applications* 25(1): 3–60. ISSN 1094-3420.

Duranton M, Bosschere KD, Gamrat C, et al. (2017) *HiPEAC Vision 2017. Technical Report, H2020 HiPEAC CSA*. ISBN 978-90-9030182-2. HiPEAC network of excellence, pp.138.

Executive Office of the U.S. President (2015a) National strategic computing initiative executive order. Available at: https://www.whitehouse.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative. (accessed 01 March 2018)

Executive Office of the U.S. President (2015b) National strategic computing initiative fact sheet. Available at: https://www.whitehouse.gov/sites/default/files/microsites/ostp/nsc_fact_sheet.pdf (accessed 01 March 2018).

Eyink G, Vishniac E, Lalescu C, et al. (2013) Flux-freezing breakdown in high-conductivity magnetohydrodynamic turbulence. *Nature* 0(497): 466–469.

Fälström P (2016, May) *Market-Driven Challenges to Open Internet Standards*. Online. GCIG Paper No. 33, Global Commission on Internet Governance Paper Series.

Feynman RP (1967) *The Character of Physical Law*, Vol. 66. Cambridge, UK: MIT press.

Flach PA and Hadjiantonis AM (2013) *Abduction and Induction: Essays on their Relation and Integration*, Vol. 18. Berlin, Germany: Springer Science & Business Media.

Foster I, Kesselman C and Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications* 15(3): 200–222.

Fox G, Qiu J, Jha S, et al. (2016) White paper: Big data, simulations and HPC convergence. Online. Available at: http://dsc.soic.indiana.edu/publications/HPCBigDataConvergence.Summary_IURutgers.pdf (accessed 01 March 2018).

Fox G, Shantenu J and Ramakrishnan L (2016) Final report: first workshop on streaming and steering applications: Requirements and infrastructure. Available at: http://streamingsystems.org/finalreport.pdf (accessed 01 March 2018).

Fu S, Liu J, Chu X, et al. (2016) Toward a standard interface for cloud providers: the container as the narrow waist. *IEEE Internet Computing* 20(2): 66–71.

Gelenbe E and Caseau Y (June 2015) The impact of information technology on energy consumption and carbon emissions. *Ubiquity New York*: ACM. ISSN 1530-2180.

Gleckler PJ, Durack PJ, Stouffer RJ, et al. (2016) Industrial-era global ocean heat uptake doubles in recent decades. *Nature Climate Change* 6: 394–398.

Gorenberg M, Schmidt E and Mundie C (2016) *Report to the President: Technology and the Future of Cities*. Washington, DC, USA: President’s Council of Science and Technology Advisors, pp. 1–99.

Grady NW, Underwood M, Roy A, et al. (2014) Big data: challenges, practices and technologies: NIST big data public working group workshop at IEEE big data 2014. In: *Big data (big data), 2014 IEEE international conference on*, Washington, DC, USA, pp. 11–15. Piscataway: IEEE.

Hagel J and Brown JS (2017) Shaping strategies for the IoT. *Computer* 50(8):64–68.

Hagel J, Brown JS and Davison L (2008) Shaping strategy in a world of constant disruption. *Harvard Business Review* 86(10): 80–89.

Hashem IAT, Yaqoob I, Anuar NB, et al. (2015) The rise of "big data" on cloud computing: review and open research issues. *Information Systems* 47: 98–115.

Hey T and Trefethen A (2003) The data deluge: an e-Science perspective. In: Berman F, Fox GC and Hey AJG (eds) *Grid Computing: Making the Global Infrastructure a Reality*. Hoboken: Wiley and Sons, pp. 809–824.

Hey T, Tansley S and Tolle KM (2007) Jim Gray on eScience: a transformed scientific method. Online. Available at: https://www.semanticscholar.org/paper/Jim-Gray-on-eScience-a-transformed-scientific-meth-Hey-Tansley/b71ce8fa2d7795acc4b03df8691184ff722fc7a1 (accessed 01 March 2018).

Honavar VG, Hill MD and Yelick K (2016) Accelerating science: a computing research agenda. *arXiv preprint arXiv:1604.02006.*

Hu YC, Patel M, Sabella D, et al. (2015) Mobile edge computing? A key technology towards 5G. *ETSI White Paper* 11(11): 1–16.

Karpatne A, Atluri G, Faghmous J, et al. (2016) Theory-guided data science: a new paradigm for scientific discovery. IEEE Transactions on Knowledge and Data Engineering, 29(10): 2318–2331.

Kavassalis P, Solomon RJ and Benghozi PJ (1996) The Internet: a paradigmatic rupture in cumulative telecom evolution. *Industrial and Corporate Change* 5(4): 1097–1126.

Kuntschke R, Scholl T, Huber S, et al. (2006) Grid-based data stream processing in e-Science. In: *Second international conference on e-Science and grid technologies (e-Science 2006)*, pp. 4–6 December 2006, Amsterdam, The Netherlands, pp. 30.

Leiner BM, Cerf VG, Clark DD, et al. (2009) A brief history of the Internet. *SIGCOMM Computer Communication Review* 39(5): 22–31.

Li Y, Perlman E, Wan M, et al. (2008) A public turbulence database cluster and applications to study Lagrangian evolution of velocity increments in turbulence. *Journal of Turbulence* 9: N31.

Lu XY, Liang F and Wang B (2014) DataMPI: extending MPI to Hadoop-like big data computing. In: *International parallel and distributed processing symposium*, Phoenix, AZ, USA, 19–23 May 2014, pp. 829–838.

Luu H, Winslett M, Gropp W, et al. (2015) A multiplatform study of I/O behavior on petascale supercomputers. In *Proceedings of HPDC'15,* 15-19 June, 2015. DOI: 10.1145/2749246.2749269http://.

McGeer R, Berman M, Elliott C, et al. (eds) (2016) *The GENI Book*. Berlin, Germany: Springer. ISBN 978-3-319-33767-8.

Messerschmitt DG and Szyperski C (2005) *Software Ecosystem: Understanding an Indispensable Technology and Industry*. Cambridge, UK: MIT Press. ISBN 9780262633314. Available at: https://books.google.com/books?id=6ipSHAAACAAJ. (accessed 01 March 2018)

Miyoshi T, Kunii M, Ruiz J, et al. (2016) Big data assimilation revolutionizing severe weather prediction. *Bulletin of the American Meteorological Society* 97(8): 1347–1354.

Nahrstedt K, Cassandras C and Catlett C (2017a) City-scale intelligent systems and platforms. Online. Available at: http://cra.org/ccc/wp-content/uploads/sites/2/2017/03/City-Scale-Intelligent-Systems-and-Platforms.pdf (accessed 01 March 2018).

Ni J and Tsang DHK (2005) Large-scale cooperative caching and application-level multicast in multimedia content delivery networks. *Communications Magazine, IEEE* 43(5): 98–105.

Papagianni C, Leivadeas A and Papavassiliou S (2013) A cloud-oriented content delivery network paradigm: modeling and assessment. *Dependable and Secure Computing, IEEE Transactions on* 10(5): 287–300.

Reed DA and Dongarra J (2015) Exascale computing and big data. *Communication. ACM* 58(7): 56–68.

Satyanarayanan M (2017) The emergence of edge computing. *Computer* 50(1): 30–39.

Satyanarayanan M, Bahl P, Caceres R, et al. (2009) The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing* 8(4): 14–23.

Shi W, Cao J, Zhang Q, et al. (2016) Edge computing: vision and challenges. *IEEE Internet of Things Journal* 3(5): 637–646.

Tennenhouse DL and Wetherall DJ (1996) Towards an active network architecture. *Computer Communication Review* 26: 5–18.

ur Rehman MH, Liew CS, Abbas A, et al. (2016) Big data reduction methods: a survey. *Data Science and Engineering* 1(4): 265–284.

Wang S, Zhang X, Zhang Y, et al. (2017) A survey on mobile edge networks: convergence of computing, caching and communications. *IEEE Access* 5: 6757–6779.

Wikipedia (2017a) Ken batcher — Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Ken_Batcher (accessed 16 October 2017).

Wikipedia (2017b) Lidar — Wikipedia, the free encyclopedia. Available at: https://https://en.wikipedia.org/wiki/Lidar (accessed 07 July 2017).

Wikipedia (2017c) Multi-messenger astronomy — Wikipedia, the free encyclopedia. Available at: https://en.wikipedia.org/wiki/Ken_Batcher (accessed 16 October 2017).

Williams DN, Ananthakrishnan R, Bernholdt DE, et al. (2008) Data management and analysis for the earth system grid. *Journal of Physics: Conference Series* 1250(1): 012072. Available at: http://stacks.iop.org/1742-6596/125/i=1/a=012072 (accessed 01 March 2018).

Xu ZW, Chi XB and Xiao N (2016) High-performance computing environment: a review of twenty years of experiments in China. *National Science Review* 3(1): 36–48.

Author biographies

M. Asch is full professor of Applied Mathematics at the University of Picardy Jules Verne (France) and is currently on secondment to Total, where he coordinates group-level research programs on Uncertainty and Data Science.

T. Moore is the Associate Director of the Innovative Computing Laboratory in the Tickle College of Engineering at the University of Tennessee. He earned his PhD in Philosophy from the University of North Carolina, Chapel Hill. His research interests include collaboration technologies, distributed computing, logistical networking, and the history and philosophy of science.

R. Badia leads the Workflows and Distributed Computing research group at the Barcelona Supercomputing Center. The group focuses its research on the development of PyCOMPSs/COMPSs, a task-based programming model for distributed computing that is used in several European projects.

M. Beck began his research career in distributed operating systems at Bell Laboratories and received his PhD in Computer Science from Cornell University (1992) in the area of parallelizing compilers. He then joined the faculty of the Computer Science Department at the University of Tennessee, where he is currently an Associate Professor working in distributed and high-performance computing, networking, and storage.

P. Beckman is the co-director of the Northwestern University / Argonne National Laboratory Institute for Science and Engineering. As a Senior Computer Scientist at Argonne, he leads research into exascale system software and intelligent sensor networks. He received his PhD in Computer Science from Indiana University in 1993.

T. Bidot is a consultant in high-performance computing for Neovia Innovation. He has an MS in Numerical Analysis from the University Pierre et Marie Curie (Paris VI) and has been teaching Numerical Analysis in France and abroad. He has over 30 years of experience within European and US companies and has been involved in more than 50 European funded research projects.

F. Bodin is a Professor at the University of Rennes I. He was the founder and CTO of Caps-Enterprise, which specialized in programming tools for high-performance computing.

F. Cappello is a Senior Computer Scientist at Argonne National Laboratory and an Adjunct Associate Professor in the Department of Computer Science at the University of Illinois at Urbana Champaign. He is the Director of the Joint-Laboratory on Extreme Scale Computing, which brings together seven of the leading high-performance computing institutions in the world: ANL, NCSA, Inria, BSC, JSC, Riken CCS, and UTK-ICL. He is a fellow of the IEEE.

A. Choudhary is a Henry and Isabel Dever Professor of Electrical Engineering and Computer Science at Northwestern University. He is a fellow of IEEE, ACM, and AAAS.

B. de Supinski Supinski is CTO for Livermore Computing at Lawrence Livermore National Laboratory (LLNL), where he formulates LLNL's large-scale computing strategy and oversees its implementation. In addition to his work with LLNL, Bronis is also a Professor of Exascale Computing at Queen's University of Belfast and an Adjunct Associate Professor in the Department of Computer Science and Engineering at Texas A&M University.

E. Deelman is a Research Professor at the USC Computer Science Department and a Research Director at the USC Information Sciences Institute. Dr. Deelman's research interests include the design and exploration of collaborative, distributed scientific environments, with particular emphasis on automation of scientific workflow and management of computing resources, as well as the management of scientific data.

J. Dongarra holds appointments at the University of Tennessee, Oak Ridge National Laboratory, and the University of Manchester. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced computer architectures, programming methodology, and tools for parallel computers. He was awarded the IEEE Sid Fernbach Award in 2004; in 2008 he was the recipient of the first IEEE Medal of Excellence in Scalable Computing; in 2010 he was the first recipient of the SIAM Special Interest Group on Supercomputing's award for Career Achievement; in 2011 he was the recipient of the IEEE Charles Babbage Award; and in 2013 he received the ACM/IEEE Ken Kennedy Award. He is a Fellow of the AAAS, ACM, IEEE, and SIAM, a foreign member of the Russian Academy of Science, and a member of the US National Academy of Engineering.

A. Dubey is a Computer Scientist in the Mathematics and Computer Science Division at Argonne National Laboratory. She also holds a joint appointment in Astronomy and Astrophysics at the University of Chicago. Her primary research interests are design, architecture, and productivity issues related to multicomponent scientific software.

G. Fox is a Distinguished Professor of Engineering, Computing, and Physics at Indiana University, where he is director of the Digital Science Center and Department Chair for Intelligent Systems Engineering at the School of Informatics, Computing, and Engineering.

H. Fu is the Deputy Director of the National Supercomputing Center in Wuxi and an Associate Professor in the

Ministry of Education's Key Laboratory for Earth System Modeling and Department of Earth System Science at Tsinghua University. Dr. Fu works on computational solutions for geoscience problems. His research has led to two consecutive ACM Gordon Bell Prizes (fully-implicit atmospheric dynamic solver in 2016 and non-linear earthquake simulation in 2017), Significant Papers of FPL (27 out of 1,765 publications in 25 years of FPL), and Best Paper Award (3 out of 278 submissions) of ICTAI 2015.

S. Girona, PhD in Computer Science, has been the Director of the Barcelona Supercomputing Center's Operations Department since 2004 and is the manager of the Spanish Supercomputing Network (RES). He has also held several managing positions in PRACE since its creation.

W. Gropp is Director of the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and holds the Thomas M. Siebel Chair in the Department of Computer Science. He is a Fellow of ACM, IEEE, and SIAM and an elected member of the National Academy of Engineering.

M. Heroux is a Senior Scientist at Sandia National Laboratories, Director of SW Technologies for the US Department of Energy's Exascale Computing Project, and Scientist in Residence at St. John's University, MN. His research interests include all aspects of scalable scientific and engineering software for new and emerging parallel computing architectures.

Y. Ishikawa leads the FLAGSHIP 2020 project at the Riken Center for Computational Science.

K. Keahey is one of the pioneers of infrastructure for cloud computing. She created the Nimbus project, recognized as the first open-source Infrastructure-as-a-Service implementation. Kate also leads the Chameleon project, a deeply reconfigurable, large-scale, and open experimental platform for computer science research. Kate is a scientist at Argonne National Laboratory and a Senior Fellow at the Computation Institute at the University of Chicago.

D. Keyes is a Professor of Applied Mathematics and Computational Science and Director of the Extreme Computing Research Center at the King Abdullah University of Science and Technology (KAUST). He is also an Adjunct Professor and former Fu Foundation Chair in Applied Physics and Applied Mathematics at Columbia University. He has co-authored over a dozen reports on aspects of high-performance computing and computational science and engineering for the US DOE, NSF, NASA, and SIAM.

W. Kramer leads the National Center for Supercomputing Applications' (NCSA's) @Scale Science & Technologies program and is the principal investigator for NCSA's Blue Waters project, which deployed and supports the highest sustained-performance computational and data analysis system available to the nation's open research community. Bill is also a Research Professor in the Computer Science Department at the University of Illinois at Urbana-Champaign.

J.-F. Lavignon now CEO of Technology Strategy, is involved in the development of high-performance computing in Europe. He managed high-performance computing at Bull/Atos from 2001 to 2007 and was the chairman of the European Technology Platform for High Performance Computing (ETP4HPC) from its creation until 2016.

Y. Lu is a Professor of Computer Science at both Sun Yat-sen University and the National University of Defense Technology (NUDT). She is also Director of the National Supercomputing Center in Guangzhou. She received her BS, MS, and PhD from NUDT. Her extensive R&D experience has spanned several generations of domestic supercomputers in China, and she is deputy chief designer of the Tianhe-2 system. Her continuing research interests include parallel operating systems, high-speed communications, global file systems, and advanced programming environments converging high-performance computing and big data.

S. Matsuoka is the director of the RIKEN Center for Computational Science, the organization that oversees the K Computer and its upcoming exascale successor, the Post-K supercomputer.

B. Mohr is a Senior Scientist at Forschungszentrum Juelich, Germany, serving since 1996. Starting in 2007, he also serves as deputy head for the Juelich Supercomputing Centre's division of Application Support.

D. Reed is the Senior Vice President for Academic Affairs as well as a Professor of Computer Science and Electrical & Computer Engineering at the University of Utah.

S. Requena is Chief Technology Officer of GENCI and the French HPC agency. She is in charge of formulating GENCI's HPC, HPDA, and AI converged facilities/services strategy and implementing it with the support of the three national centers. In addition, Stȗphane is also a member of the PRACE Board of Directors and has past expertise in HPC applications development and optimization for the oil and gas, energy, and automotive industries.

J. Saltz is a Digital Pathology pioneer having worked for the past twenty years in the development of digital Pathology whole slide image software, algorithms, and tools. He has also made substantial contributions to high-end computing system software, having developed the inspector/executor framework and the innovative filter/stream middleware with concepts that found their way into many commercial systems. He is a boarded Clinical Pathologist, holds an MD-PhD in Computer Science from Duke, completed a Clinical Pathology residency from Hopkins, and has founded Biomedical Informatics departments at Stony Brook, Emory, and Ohio State. Dr. Saltz is Chair and Professor of Biomedical Informatics at SUNY Stony Brook, is Vice President for Clinical Informatics, Stony Brook Medicine, Associate Director for Informatics at Stony Brook Cancer Center, and holds the Cherith endowed chair.

T. Schulthess is Director of the Swiss National Supercomputing Centre (CSCS) at Manno. As CSCS director, he is also a full Professor of Computational Physics at ETH Zurich.

R. Stevens is a professor at the University of Chicago and an Associate Laboratory Director at Argonne National Laboratory. He is internationally known for work in high-performance computing, collaboration and visualization technology, and for building computational tools and web infrastructures to support large-scale genome and metagenome analysis for basic science and infectious disease research. He is also recognized for his role in developing the national initiative for exascale computing, and for AI-based cancer research that is defining exascale computing requirements.

M. Swany is Associate Chair and Professor in the Intelligent Systems Engineering Department in the School of Informatics and Computing at Indiana University. His research interests include high-performance parallel and distributed computing and networking.

A. Szalay is the Bloomberg Distinguished Professor at Johns Hopkins University. He has been the Architect of the science archive for the Sloan Digital Sky Survey. His research interests include scalable data-intensive architectures and algorithms.

W. Tang is the Principal Research Physicist at the Princeton University Plasma Physics Laboratory, Lecturer with Rank & Title of Professor in the University's Department of Astrophysical Sciences, member of the Executive Board, and PI for the Intel Parallel Computing Center at the University's interdisciplinary Princeton Institute for Computational Science and Engineering. He has an "h-index" or "impact factor" of 45 on the Web of Science, including over 8,000 total citations, and was recently named recipient of the NVIDIA 2018 Global Impact Award.

G. Varoquaux is a tenured computer science researcher at INRIA. He works on statistical machine learning and is a core contributor to the scikit-learn, joblib, Mayavi, and nilearn software. Varoquaux has a PhD in quantum physics and is a graduate of Ecole Normale Superieure, Paris.

J.-P. Vilotte (res.id H-1552-2017) is Professor at the Institut de Physique du Globe de Paris (IPGP), and is Director of the High-performance Computer and Data Analysis Centre (S-CAPAD) of IPGP. He is part of the External Advisory board of the European project AENEAS for designing the distributed, federated European Science Data Centre (ESDC) to support the astronomical community in achieving the scientific goals of the Square Kilometre Array (SKA).

R. Wisniewski is an ACM Distinguished Scientist, the Chief Software Architect for Extreme Scale Computing, and a Senior Principal Engineer at Intel Corporation. He has published over 74 papers, filed over 56 patents, and given over 53 invited presentations. Prior to working at Intel, he was the chief software architect for Blue Gene Research.

Z. Xu is a Professor and CTO at the Institute of Computing Technology, Chinese Academy of Sciences and holds a PhD degree from the University of Southern California.

I. Zacharov occupied leading positions in industry as a Senior Solution Architect specializing in energy-efficient, high-performance computers. Before joining Skoltech, Igor worked at Eurotech designing liquid-cooled computers. He served on the European ETP4HPC industry panel leading the Energy Efficiency & Resiliency track.