Machine Learning and Understanding for Intelligent Extreme Scale Scientific Computing and Discovery

15-CS-1768

DOE Workshop Report

January 5-7, 2015
Rockville, MD

U.S. DEPARTMENT OF ENERGY
Office of Science

BERKELEY LAB

Machine Learning and Understanding for Intelligent Extreme Scale Scientific Computing and Discovery

DOE Workshop Report

January 7–9, 2015
Rockville, MD

Workshop Organizing Committee

Michael Berry (Chair), University of Tennessee

Thomas E. Potok, Oak Ridge National Laboratory

Prasanna Balaprakash, Argonne National Laboratory

Hank Hoffmann, University of Chicago

Raju Vatsavai, North Carolina State University

Prabhat, Lawrence Berkeley National Laboratory

DOE ASCR Point of Contact

Robinson Pino

Cover: Machine learning techniques can be applied to a wide range of DOE research areas, such as automatically identifying weather phenomena in massive simulation datasets..

Contents

1 Executive Summary

1.1 Self-Aware Operating and Runtime Systems

Large-scale parallel simulations and data analysis drive scientific discovery across many disciplines. To drive larger and more detailed simulations, and deal with larger data volumes, exascale machines capable of 10^{18} operations per second are expected within the next five to ten years. However, the complexity of developing and adapting modern simulation codes for these architectures is increasing rapidly. Workloads on exascale machines will be billion-way parallel. Exploiting the full capability of the machine will require carefully assigning application tasks to cores, accelerators, deep memories, and other heterogeneous compute resources, while simultaneously optimizing for time to solution, data movement, power, and resilience. Optimizations that improve performance on one machine may slow down another. Worse, applications themselves are dauntingly complex. Production simulations comprise millions of lines of code and use sophisticated, adaptive algorithms whose performance is input-dependent. Complex workflows can couple multi-physics simulation with data preprocessing and post-processing modules.

Without wholesale change, this complexity will become unmanageable, scientific advancement will slow, and the cost of new scientific discoveries will increase dramatically. Currently, human performance experts work with application teams to tune their codes. Through painstaking effort, repeated measurement, and manual parameter space exploration, they can wring performance out of benchmarks and simple codes. However, integrated production codes have multiple physics modules with distinct bottlenecks and scaling behaviors. Simultaneously optimizing all of these components together is too difficult even for performance experts, and the most sophisticated simulations often run at less than 10 percent of peak performance on *today's* machines. **If current trends continue, achieving even 1 percent of peak on an exascale machine may become an exceptional accomplishment.** Similarly, it takes many months for vendors and system operators to stabilize new systems and have them run in a reasonable regime. As system complexity increases, time to full production will continue to increase. The unexpected side-effects of any software change result in increasing reluctance to evolve platforms, hence increasingly slow evolution of software.

For scientific discovery to continue unabated, high performance computing (HPC) optimization must be done much faster than is possible with humans in the loop. **Exascale compute facilities must be *self-aware*, improving themselves over time without human intervention.** To handle complexity at this scale, the system must accumulate knowledge and act on it to tune performance, power, and resilience parameters at runtime. Optimization concerns must be separated from application logic to hide complexity and allow computational scientists to focus on producing new scientific insights.

With this holistic approach, the SAOSR will enable performance portability to ***any*** machine, while increasing developer productivity and speeding the advance of science.

The Self-Aware Operating System/Runtime (SAOSR) will provide the features needed to make exascale optimization tractable, and to allow applications to achieve high performance on any machine without human intervention. The SAOSR will monitor itself, leveraging machine learning techniques to increase the rate of scientific discovery over time. The SAOSR will use learning techniques to understand expected performance and to build models of data-dependent adaptive codes. These models will be used to predict performance and resource usage, and also to make online optimization decisions. Most importantly, The SAOSR will handle *global* decisions, optimizing across coupled applications and operating system interfaces. **With this holistic approach, the SAOSR will enable performance portability to *any* machine, while increasing developer productivity and speeding the advance of science.**

1.2 Machine Learning

The DOE charter includes broad support for scientific innovation in the areas critical to the US national security and competitiveness. Progress in these areas is predicated on the ability to conduct simulations and computational experiments generating vast amounts of data, and subsequently processing that data into information. Significant DOE investments in state-of-the-art experimental and computing facilities have resulted in scientific data being generated in unprecedented volumes and velocities. Machine learning techniques have proven to be invaluable in the commercial world; and we believe that such techniques can be an indispensable tool for extracting insights from current and future scientific datasets; thereby enhancing scientific productivity and providing maximal science impact from existing DOE investments.

Exascale-class systems will feature applications, operating system and runtime frameworks running at unprecedented concurrencies. At such scales, human supervision and control of faults will become impossible. Intelligent, dynamic, resilient runtime systems that are capable of predicting, detecting and recovering from faults will become the norm. We believe that machine learning (ML) methods will play a critical role in ushering in this new age of embedded intelligence. Such augmented systems will perform real-time allocation of power and other resources consistent with user-defined constraints and priorities. At the same time, it is important to design novel ML algorithms, which extract the best performance of many-core systems, while considering the pivotal issues of reliability and data movement—the primary impediments in designing exascale systems.

We believe that sustained long-term investment in the field of ML has the potential for a large payoff, both in the short and long-term. Resulting advances, both in the field of machine learning, as well as science domains, can potentially catalyze and revolutionize new breakthroughs. We believe that the long-term impact of these investments will benefit the DOE and other federal and commercial organizations in the US.

1.3 Resilience and Trust

This report presents the case for a machine learning research program to ensure that the scientific integrity and discovery from exascale computing and other DOE user facilities is significantly enhanced by dramatically improving the resiliency of scientific data, applications, software, and hardware systems, as well as the trustworthiness of the scientific results produced. We believe this goal can be achieved by:

- **Fault characterization**: Characterize HPC faults using existing and extended machine learning techniques to analyze fault data that is produced by HPC systems. This fault data will require fusing disparate data from many different aspects of the computer systems, including environmental, hardware, software, and application performance. This data is produced at very high volumes, in very different time scales, and is currently not being fused today. Machine learning will be used to develop automated fault classifiers based on this fused fault data. We expect this to be an offline process, where labeled and unlabeled fault data is used to develop fault classifiers. For example, machine learning may show that when the temperature of one part of the HPC system is more than 3 degrees different from another part, there will be increased I/O faults.
- ***In situ* fault detection**: Use classifier models developed offline to detect faults and learn new patterns *in situ*. There are several challenges with this approach, one is gaining access to data that can reasonably predict faults, which may require dynamically enabling log information, another challenge is the scale and speed which is required to analyze the data, which will likely require some means of high performance computing. If a fault is detected, then processing needs to be very quickly shifted to systems unaffected by the fault. This requires near real-time classification of potential faults, their locations, and the system affected, a capability that does not exist today.
- **Fault prediction**: Predict faults before they actually occur so that the system can be repaired or faults avoided when running an application. This will require machine learning algorithms that can determine precursors of faults, similar to the way certain medical tests can predict the onset of diseases. Finding these precursors requires capturing large volumes of data over a long periods of time, and tracing back through a fault to find early warning signs of problems. Unsupervised machine learning methods combined with time-frequency representations of the data,

such as wavelets, can be used to find common patterns that occur before a fault. A model can then be built to detect and act on these precursors before a fault occurs.

- **Trusted results**: The first three areas above focus on creating an HPC environment that is resilient to failure, which enables a scientist to have trust in an application's results. However, there will always be a set of faults or combination of faults that do not produce fault data, yet the resulting output of an application is not correct. To ensure the output of an application is producing accurate results, the output of an application must be examined for accuracy. This is quite a challenging task, given the extremely large amounts of output data, the normal variability associated with stochastic simulations, and the computational expense of running these models. There may be a handful of reference output data from which to determine an accurate result. Quite likely, sections of a single run may need to be used as training data to determine normal variability across a range of simulated parameters. From this analysis, we will develop a measure of trustworthiness based on the similarity of previous or earlier results that needs to be reported back to the domain scientist, so that he or she can assess the trustworthiness of the results.

  Expanding on this, the trustworthiness of this type of output is not limited to exascale computing. Any of the DOE user facilities that produces scientific data faces the potential of untrustworthy results for a wide variety of reasons ranging from equipment problem, to manual error, malicious intent, to even sabotage. Machine learning algorithms can be used to train on trustworthy data from a variety of user facilities, and help provide scientists a better understanding of the trustworthiness of their results.

- **Extensions to trust**: Given the need to examine scientific output using machine learning methods, the natural extension would be to help the domain scientist understand more than just the trustworthiness of the results. It would be valuable to use machine learning methods to train on interesting sections of output data, such as a severe storm in a climate simulation, to see if these methods can be used to detect similar phenomenon with new output results. In this way, a scientist can have the confidence that their scientific data havw been reviewed computationally, and the most promising areas have been highlighted.

Such a program will help ensure the scientific integrity of these results produced from DOE user facilities and help further scientific discovery from the wealth of data produced at these facilities.

In summary, given the volume and complexity of data generated from DOE user facilities and exascale computing, we believe than an ASCR ML program focused on machine intelligence is needed to ensure the resiliency and trust of scientists in simulation and experimental results. Such a program will help ensure the scientific integrity of these results produced from DOE user facilities and help further scientific discovery from the wealth of data produced at these facilities.

2 Acknowledgements

2.1 Self-Aware Operating and Runtime Systems

We thank all the participants of the workshop for submitting high quality abstracts that led to this report. The self-aware operating and runtime systems topic group would like to acknowledge Hank Hoffmann (Univ. Chicago), Prasanna Balaprakash (ANL), Dorian C. Arnold (Univ. New Mexico), Pete Beckman (ANL), Marc Snir (ANL), Ron Brightwell (SNL), Todd Gamblin (LLNL), Steven Hofmeyr (LBNL and UCB), Una-May O'Reilly (MIT), Abhinav Sarje(LBNL), and Stephen P. Crago (USC). Finally, we thank Robinson Pino and Michael Berry for their feedback throughout the workshop.

2.2 Deep Learning

We would like to thank Stefan Hau-Reige from Lawrence Livermore National Laboratory for his helpful discussion regarding the opportunities for machine learning to revolutionize data analysis for the Linear Coherent Light Source (LCLS) at SLAC. Thanks also to SLAC and Kwei-Yu Chu for the LCLS illustrations.

2.3 Resilience and Trust

The resiliency and trust topic group expresses gratitude to the many contributors who have helped craft this section of the report.

We thank Robinson Pino and Michael Berry for their outstanding leadership of the machine learning workshop, all the participants of the workshop who helped broaden and shape our ideas, and to those who helped write this report: Tom Potok (ORNL), Raju Vatsavai (NCSU), Franck Cappello (ANL and UIUC), Kurt B. Ferreira (SNL), Mahantesh Halappanavar (PNNL), Ignacio Laguna (LLNL), Hoony Park (ORNL), Arvind Ramanathan (ORNL), Arjun Shankar (ORNL), Rangan Sukumar (ORNL), and Elizabeth Whitaker (Georgia Tech).

3 Introduction

3.1 Self-Aware Operating and Runtime Systems

Physical constraints are increasingly affecting scientists' interactions with HPC systems. Scientific researchers are no longer tasked with just maximizing performance, but now must deal with maximizing performance in the face of physical constraints. Scaling to deep sub-micron transistors is increasing hardware failure rates. Power consumption and heat dissipation limit the hardware resources that can be brought to bear at one time. Yet the increasing size of datasets demands ever increasing compute performance. These constraints create an environment where scientific application programmers must reason about not just performance, but also power constraints and resilience to hardware failures.

The problem is compounded due to the fact that computer systems are increasingly complex, dynamic, and highly nonlinear. The performance of an application is influenced by the interaction of many different subsystems, each with its own feedback loops and control mechanisms. These include instruction scheduler, cache controller, memory controller, power controller, task scheduler and memory manager at the node level; and network routing and I/O services at the global level. Application programmers have limited knowledge on the behavior of these various subsystems, and even vendors seem be unable to understand the complex interactions between these subsystems. Furthermore, their behavior changes dynamically in response to temperature changes, faults, or changes in the environment, in ways that cannot be anticipated. Small changes (in temperature or in the behavior of a code) can lead to large changes in performance.

Application codes are also becoming more complex, often involving the tight coupling of modules that were developed independently. Even if the behavior of each module is well understood, the emergent behavior of the composed application may be hard to predict.

Even if the behavior of computer systems were predictable, it would be unrealistic to expect application developers working at exascale to be both experts in their application domain and have the deep systems knowledge required to maximize performance under power and resilience constraints and react to dynamic fluctuations.

Machine learning represents a broad class of techniques that can help provide well-founded solutions to many of these exascale challenges.

Instead, exascale systems should put more intelligence in the operating system and runtime (OS/R) to help alleviate the burden of programming these machines. Smarter operating systems could take on the challenge of constrained optimization, reacting to dynamic events and providing guarantees.

The very nonlinear behavior of compute systems, and the complexity of the interactions, limit the applicability of classical control theory. Machine learning represents a broad class of techniques that can help provide well-founded solutions to many of these exascale challenges. In this report, we detail some of the ways in which ML techniques can address specific challenges of exascale OS/R.

3.2 Introduction to Machine Learning

3.2.1 What is Machine Learning?

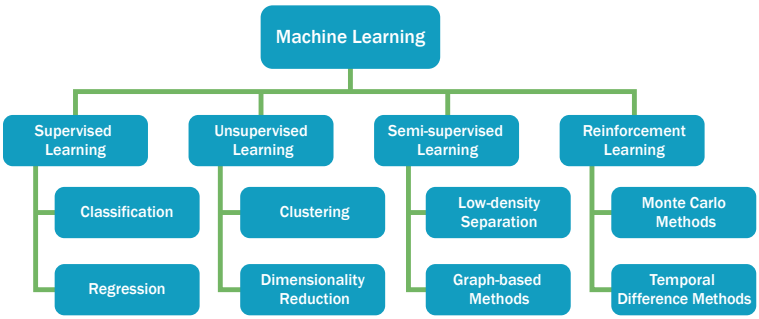


Figure 1. Landscape of Machine Learning approaches and tasks

Machine learning tasks are often grouped into several broad categories, depending on the nature of the data and information available to the algorithm. One categorization is based on what information is provided to the algorithm.

- In **supervised learning**, the algorithm is presented with example inputs and "ground truth" outputs, and the goal is to learn a general rule that maps inputs to outputs.
- In **unsupervised learning**, the algorithm is presented with example inputs with no "ground truth" labels, and the goal is to find structure, e.g., hidden patterns, in its input.
- In **reinforcement learning**, the algorithm interacts with a dynamic environment, e.g., a robot interacting on its environment, and the task is to perform a certain goal, e.g., having the robot perform a given task, without explicit information on how close the algorithm has come to the goal.

Of course, there are many variants. For example, between supervised and unsupervised learning there is semi-supervised learning. Here, only a small fraction of the inputs given to the algorithm are provided ground truth labels. Another categorization of ML tasks arises when one considers the desired output of an ML algorithm. For example, a support vector machine is a classifier that divides its input space into two regions, separated by a linear boundary.

- In **classification**, the input data is divided into two or more classes, with labels, and the machine learning algorithm outputs a model that assigns unseen inputs to one or more of these classes.
- In **regression**, the setup is similar except that the outputs are continuous rather than discrete.
- In **clustering**, one is typically not given labels for the input data, and the output consists of a grouping of the data.
- In **dimensionality reduction**, one outputs a lower-dimensional space that describes important properties of the data.

Although machine learning grew out of the quest for artificial intelligence, its scope and potential is much more general. For example, it has strong connections with statistical modeling and data analysis, and attempts to learn and apply concepts from neuroscience, or neuromorphic computing. While many view machine learning as focusing on prediction, based on known properties learned from the training data, related methods in data mining focus on the discovery of previously unknown properties in the data. Similarly, inference and learning are two extremely related topics that are referred to differently by researchers with different backgrounds.

While the methods depend on areas of computer science and statistics out of which they grew (e.g., data mining uses many machine learning methods, but often with somewhat different goals in mind), many of these techniques are tools that could be used by downstream scientists in different ways, albeit in somewhat modified ways or with somewhat modified objectives. For example: quantitative prediction versus qualitative understanding. Most ML methods are developed for applications where one does not have a strong understanding of the data, and thus the models have to do more of the work, and the evaluation metric is some sort of prediction task. This is very different than using ML methods to obtain understanding of the data, and in general there is a big tradeoff between the two and the latter is probably more relevant for DOE scientific applications.

3.2.2 DOE-Relevant Challenges in Machine Learning

In the exascale timeframe, scientific progress will be predicated on our ability to process vast, complex datasets from extreme scale simulations, experiments and observational facilities. Even at present, scientific data analysis is becoming a bottleneck in the discovery process; we can only assume that the problem will become intractable in a decade. At the moment, scientists are often forced to create their own ad hoc solutions where a lack of scalable analytic capabilities means that there are large-scale experimental and simulation results that cannot be fully and quickly utilized. Moreover, the scientists lack dynamic insight into their analyses, unable to modify the experiment or simulation on the fly. In this section, we comment on challenges that the DOE scientific and research community will face in the area of machine learning.

- **New capabilities**: Machine learning has a rich history, and a successful track record in the commercial world, but several challenges in DOE are unique and require novel methods. DOE needs to develop expertise in applying existing ML methods and focus research capabilities to develop, extend and customize methods and apply them successfully to scientific problems relevant to DOE. Some of these problems are documented in Section 4.1 and Table 1.
- **ML frameworks for HPC platforms**: The field of computational modeling and simulation has made major progress in the past decades due to robust scientific software architectures developed by DOE (MPI, PETSc, Trilinos, etc.). Similarly, in order to make major advances in machine learning, and data analytics more broadly, we need to develop a robust, production framework for deployment on DOE's extreme scale systems. The framework should support DOE computer architectures, file systems, and data formats. Individual researchers and developers can utilize the framework for deploying their state-of-the-art research, and more effectively utilize large scale computational resources across the DOE complex.
- **Usability and out-of-the-box learning**: In order for machine learning to have broad impact across the DOE complex, it is imperative that the methods work well out-of-the-box. In other words, we can't expect domain scientists and interested users to become experts on various methods and how parameters associated with methods might impact the accuracy of the prediction. In order to address this challenge, we will need machine learning methods to incorporate meta-level heuristics

(such as cross-fold validation and auto-tuning) to provide good performance.

- **Scaling ML methods for exascale**: In order to keep pace with the increasing volume of scientific big data, novel scalable ML methods will need to be developed and implemented. It is likely that these implementations will run on portions of the exascale hardware, where considerations such as power consumption, cost of data movement across the memory hierarchy and resilience will become important to address. Current implementations are largely ignorant of such considerations; we will require ML research to be more closely connected with the exascale programs.

3.2.3 NRC/NIST Big Data reports

Prior to delving into the details of science drivers and resulting machine learning requirements unique to DOE, it is important to acknowledge two major efforts that have been conducted recently in the big data analytics space: the NRC Frontiers in Massive Data Analysis report [31] and the use cases gathered by NIST [2, 106]. These cover industry, government, and academic applications and illustrate many facets of big data systems. While not exhaustively comprehensive, these two studies give us a way to identify particular challenges and the overall context in which they sit.

NRC studied in detail seven application areas: Earth and planetary observations, astronomy, biological and medical research, large numerical simulations, social network analysis, national security (command and control, cybersecurity, remote surveillance), and telecommunications and networking (managing a global network). NIST collected, at a finer granularity, 51 specific big data use cases covering (partially) the first six of the NRC areas, as well as instruments (particle sources, light sources), commercial (search, commerce, finance), government operations, science and commodity sensors, and energy. Both studies distilled general features with the NRC's insightful Seven Computational Giants of Massive Data Analysis covering Basic Statistics, Generalized N-Body Problems, Graph-Theoretic Computations, Linear Algebraic Computations, Optimizations (e.g. Linear Programming), Integration, Alignment Problems. Aspects highlighted in Ogre analysis [51] of NIST use cases include categories such as pleasingly parallel (26), MapReduce (25), iterative MapReduce (23), graph (9), fusion (11), streaming/dynamic data-driven application systems (41), classification (30), search/query (12), collaborative filtering (4), local machine learning (36), global machine learning (23), workflow (51), geographic information systems (16), simulation results (5), and agents (2). The numbers in parentheses give an estimate of number of the 51 use cases exhibiting different features.

3.2.4 Success and Limitations of Machine Learning in the Commercial World

Machine learning is used extensively in the commercial world, for instance, to improve targeted advertisements (Google Ads), to predict user behavior (Twitter), to develop improved recommendation systems (Netflix, Amazon) and to suggest friends (Facebook/LinkedIn). Typically, the success of machine learning in the commercial world stems from the capability of gathering and processing a large amount of data in various forms in a timely manner. This has prompted the development of big data analytics tools such as Apache Hadoop MapReduce (Apache Big Data Stack) and more recently Apache Spark (Berkeley Data Analytics Stack). Also, many industry machine learning applications have considered supervised learning because they rely on data that mostly comes from system logs or user activity logs, which can be used to supply the label of each training sample in an automatic way. Originally, many industry applications of machine learning could be considered as data mining, or describing/representing data, which is in some sense a simpler task than inference.

An open research challenge for the DOE machine learning community is whether common analytics frameworks can be developed that can be applicable to both environments.

Based upon the success in applying simpler data analysis and machine learning algorithms for relatively simple tasks, in recent years industry has begun more aggressively attempting to address more challenging tasks, e.g., fine-scale user modeling, recognizing faces in pictures, real time speech recognition, advertisement placement and optimization, and automatically generating descriptive captions in plain English for images.

Some leading examples include Microsoft research on a deep learning system that demonstrated a top-1 result that classifies 22,000 categories of pictures at 29.8 percent of accuracy, (a random guess will result in 1⁄22,000 = 0.004 percent). They also demonstrated real-time speech translation between Mandarin Chinese and English [125]. Although this task involves relatively long training time compared with other ML algorithms, predicting a new instance based upon a trained model can be done in a real time with a relatively small amount of compute resources. For instance, the Siri application uses a deep learning trained model, and the voice recognition in the Google Android phone also uses a deep learning trained model. A recent study between Carnegie Mellon University and Google showed that the software system for deep learning can be exploited for other relatively computationally intensive machine learning algorithms such as Latent Dirichlet Allocation (LDA) for topic modeling. In this particular application, the training of deep learning requires thousands of CPU cores (Google, Microsoft Research studies) or requires a few tens of GPUs, for a few days. Industry researchers estimate that a petaflop-class system will be required in order to complete training of deep learning networks within a reasonable amount of time. Industry data centers are throughput oriented, providing interactive response to accommodate a large number of users. This is in contrast to the performance-oriented nature of HPC facilities run by DOE. An open research challenge for the DOE machine learning community is whether common analytics frameworks can be developed that can be applicable to both environments.

3.3 Resilience and Trust

The current increase in speed of high performance computers from petaflops to exaflops is not driven by increases in processor speed, as it has been in the past, but instead by the combination of processors. The number of cores in an exascale computer will easily be in the tens of millions, as will the other components necessary to create such a large machine. The science produced from simulation and analysis on these machines will help guide worldwide climate policy, provide new understanding in quantum and astrophysics, provide breakthroughs in biology, and new ways of analyzing massive data.

With the dramatic increase in the number of components in an exascale computer comes a dramatic increase in the potential amount of diagnostic data that these computers will produce. In the past, this data had been used to alert an operator to the failure of a given component; however, with the volume and speed with which fault data is produced, it will be impossible to process it manually [36, 22].

The scientists using these computers expect them to reliably run very large and complex applications producing trustworthy results. Numerous studies have shown this goal may be in jeopardy, as very rare component-level faults may produce frequent system-level faults as the number of components within an HPC system dramatically increases. The common methods of dealing with system faults, such as restarting from a known good state (checkpoint restart on file system), or manual root cause analysis to find the cause of faults may no longer be effective or even doable at such scales and complexities. The trustworthiness of results may be compromised due to undetected or silent faults. Such faults give the impression that the application completed without issue, but with incorrect results, and potentially incorrect scientific interpretation. At best, this is an unacceptable result for the scientific community, at worst; this can result in a misinterpretation of climate projections or design flaws in nuclear power systems. Given the scale and complexity of the computers and scientific applications, determining a trustworthy result is an open question.

Trust

We define trust as a relationship between two entities and characterized by the confidence that an entity will work according to expectations, and that if a task is delegated to this entity, it will successfully perform the task. Trust is involved in a situation in which a system or agent is vulnerable or at risk based on the actions of another entity upon which it depends for a necessary service. We can represent components of the scientific computing process as entities, including computation (or algorithms), users of these computations, and the system that executes them. Trust in an entity is context-dependent, that is the degree of trustworthiness of an entity is dependent on what we trust that agent to do. An entity's trustworthiness may be computed based on performance history on similar tasks and its reputation based on information gathered from other entities, as well as knowledge of the characteristics of the entity.

Resilience

Resilience in HPC generally encompasses the collections of techniques and mechanisms that allow an application to complete despite the presence of faults which may manifest as incorrect/lost execution or system state. The faults and their associated errors and failures are dealt with by a number of techniques, including prevention, prediction, detection, diagnosis, and recovery (correction, tolerance). Lastly, these faults can occur at all levels of the HPC system, and thus these mechanisms may exist in the hardware, firmware, system/runtime levels, libraries, and application software. Therefore, coordination is typically required for effective and efficient resilience.

The ultimate goal of enabling ML for trust and resilience of HPC systems is to support the scientific computing user. The intent is to increase the degree of trustworthiness of the system to increase the quality of the results. This quality of results is dependent upon the trustworthiness of the entire HPC system and may be enabled by using two complementary approaches. First, we want to understand the trustworthiness of components in an intelligent HPC environment to enable the operating system to make dynamic, adaptive decisions. This intelligent operating system may reason about resource management, power usage, resource needs, and costs, while monitoring and controlling resources that can come from a variety of sources, for example logs and/or sensor data. Second, we want to use ML for verification of simulation data to ensure expected properties are guaranteed. This verification might be carried out through checking intermediate results as the simulation progresses.

We further believe that the trustworthiness of results highlighted in HPC systems are also found in other DOE user facilities. The challenge of trusting the results of large-scale experiments, may well be found in sophisticated analysis of these results themselves. Additionally, this analysis of results may be valuable in helping scientists discover new phenomena within their dataset that may have been otherwise missed with manual methods.

We believe that machine learning is a critical technology that will further the fourth paradigm of modern scientific discovery, and complement and support other models of scientific inquiry.

4 Motivating Science

4.1 DOE Science Drivers

DOE supports the four fundamental paradigms of scientific discovery: we have made substantial investments in theoretical research; a large number of experimental facilities are shedding light on natural phenomena from the subatomic to the astronomical scale; computational modeling and simulation are providing us with valuable insights into scientific phenomena and their impact on society; and finally the paradigm of data-driven discovery is taking root across numerous domains.

We believe that machine learning is a critical technology that will further the fourth paradigm of modern scientific discovery, and complement and support other models of scientific inquiry. In this section, we elaborate on a few sample applications, challenges, and the potential role of ML methods.

4.1.1 Pattern Classification for HPC Climate Simulations

Modern petascale and future exascale platforms will further the progress of computational science and support fundamental breakthroughs in a broad array of scientific disciplines. As a leading example of the importance and success of computational modeling, climate simulations provide us with an unprecedented view of the state of the Earth's present and potential future climate under global warming. Climate modeling faces a large number of fundamental challenges: scaling applied mathematical techniques to operate at kilometer-scale models, scaling code implementations to run efficiently on exascale platforms, representing improved physical and chemical processes both at the sub-grid and global scales, and accounting for various sources of uncertainty. Several of these challenges are being addressed by various programs in DOE. For the purpose of this report, we will comment on data analysis challenges resulting from the massive datasets generated by climate simulations.

Contemporary climate codes, such as CAM5 [3], when run in 25-km spatial resolution with 6-hour data multi-variate dumps, produce over 100TB from a 25-year integration period. The current CMIP-5 archive [4], consisting of international contributions from a number of climate modeling groups, consists of over 5PB of data; this dataset was mined extensively for the IPCC AR5 report [5]. It is anticipated that CMIP-6 dataset [7] will cross the exabyte threshold with 25-km model runs being the norm. Faced with this massive deluge of complex, spatio-temporal data, it is inevitable that the data analytics community will need to develop sophisticated pattern detection tools which can extract scientifically meaningful information. One example of

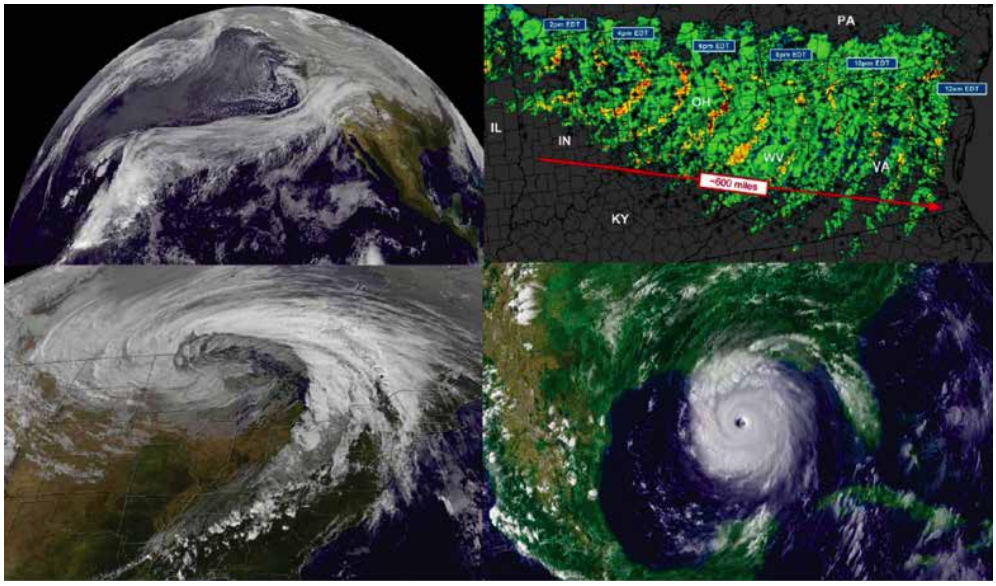
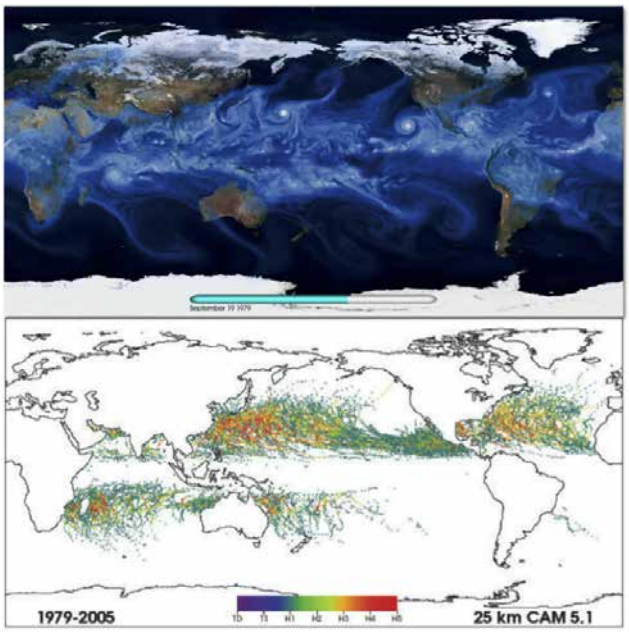


Figure 2: Examples of extreme weather phenomena observed through satellite and radar. Clockwise from bottom-left: extra-tropical cyclone, atmospheric river, derecho and tropical cyclone events.

the types of climate data analytics of societal relevance is the characterization of extreme weather. Figure 2 illustrates the types of extreme weather observed in the natural climate system. Phenomena such as cyclones and atmospheric rivers can have widespread and long-lasting impact on national economies. Understanding how extreme weather will change as a result of future climate changes is an important open question.

Figure 3: Pattern detection tools can help in quantifying and assessing model performance. In this example, a highly scalable pattern detection tool TECA [105] was applied to CAM5 simulation output (top) to extract tropical cyclones tracks (bottom).



Climate codes are now able to reproduce the initiation and development of such extreme weather phenomena. The challenge is finding the phenomena of interest in petabytes of data! A limited class of extreme weather phenomena (such as tropical cyclones, Figure 3) has been studied extensively by meteorologists, and have a procedural definition which can be implemented efficiently. However, a broad class of single events (e.g., extra-tropical cyclones, derechos, weather fronts, blocking events) and multiple interacting events (e.g., Hurricane Sandy system, teleconnections) do not have a clear statistical or quantitative description.

We envision that ML techniques, which have been highly successful in pattern detection of for computer vision problems, can be adopted to find and localize spatio-temporal phenomena in climate datasets. We also note that climate model output is a surrogate for a broad class of HPC simulation output. The need to extract patterns in terabyte- or petabyte-sized datasets is a fairly generic and cross-cutting requirement, and is applicable to all major applied science programs with an HPC component (e.g., combustion, astrophysics, high-energy physics, plasma physics). Successful demonstration of ML techniques to find known and unknown patterns in climate simulations will go a long way in the adoption of such methods by the broader research community.

4.1.2 Regression and Clustering for Material Science and Computational Chemistry

Predicting the properties of new material/molecules. Historically, the methods for discovering new materials have been trial-and-error based. Significant effort is now being invested into the development of high-throughput approaches to quickly survey and focus work on a smaller pool of candidates. The Materials Genome Initiative, launched in 2011, strives to create broad and open databases of materials data generated both experimentally and computationally. These databases [75] are being designed to support data mining and to interface with workflow software that will enable rapid prototyping of new materials *in silico*. Thus the question "Can we predict the property of a material that has never been synthesized?" will remain important to supporting innovations in materials research.

Using state-of-the-art first-principle techniques (theory of quantum mechanics), many material properties can be approximated *in silico*, albeit at significant computational expense. Even small molecules can require hours to hundreds of hours of CPU time to compute chemical properties using quantum chemistry computational methods like density functional theory (DFT). One promising avenue of research exploits a database of already-calculated properties to substitute inexpensive ML-generated predictions for expensive first-principles calculations for novel molecules [65, 96, 111]. In theory, as illustrated in Figure 4, predicting the electronic properties of the billion possible molecules containing 13 heavy atoms could be accelerated by 3–4 orders of magnitude by training a learning machine on a small fraction of the database. While initial work

shows this avenue of research to be very promising, many open questions remain, such as how to design good feature vectors to describe molecules and choose robust learning methods ,and how to extend methods to more complex molecules and to a broader set of molecular properties. However, massively expanding the predictive reach of first-principles computations via ML techniques may well be the pivotal step for actually achieving a comprehensive molecular property database.

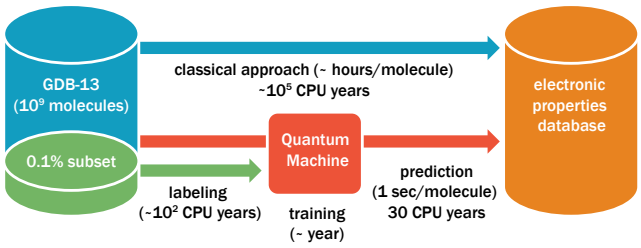


Figure 4: Using machine learning, the electronic properties of new molecules can be predicted from an existing database in a fraction of the time it would take to complete a first-principles computation. (Used with permission from von Lillienfeld)

Another important application is using ML to bridge the gap between atomistic and mesoscale simulations. Here, the exact details of how atoms interact with their nearest neighbors require, again, expensive quantum mechanics calculations that are computationally limited to tens to hundreds of atoms. In order to scale from thousands to millions of atoms, the scale at which many phenomena of interest occur, mesoscale simulations of materials using classical molecular dynamics rely on fast empirical interaction calculations that are typically 10,000 times more computationally efficient than their quantum counterparts. These fast functions are, in effect, predictions of the output of the expensive quantum calculations over a continuous high-dimensional space. In the past, these have been human-designed functions, based on physical principles and a small set of data. ML methods, in contrast, use an expansive training set of quantum-accurate calculations to exceed human intuition in constructing a predicting function from a larger possible function space [15, 116, 124].

Even further along the nanoscale-mesoscale axis, many DOE-funded researchers are investigating how to design and control materials whose structure and behavior depend on mesoscale interactions, such as polymers, colloids, complex fluids, membranes, and granular materials. These materials, ubiquitous in nature, are found in many engineering applications. Generally speaking, the emergent mesoscale behaviors of these materials cannot be approached by first-principles atomistic calculations at all. For example, thermodynamically stable colloidal crystals can be self-assembled from nanoparticles, each of which is composed of thousands of atoms and the stress-strain relationship of granular materials is an aggregate behavior that is determined by the local structural details of macroscopic particles that are visible to the eye. Determining the aggregate behavior of such materials is determined largely by experiment or computationally expensive simulations, where researchers seek to identify the rules that determine how the properties of the sub-units of such a material determines the aggregate behavior [13, 61, 34, 91, 16, 88]. The end goal is to be able to design new materials with engineered microstructures and desired properties. Finding the connection between the design of material sub-units and the aggregate behavior of the particles is still largely dependent on human cognition-based pattern detection, with some successes transcending human cognition by using evolutionary algorithms to find optimal solutions [94]. We identify this as an area that has yet to take advantage of ML techniques, but where ML may well be key to initiating the design era of mesoscale nanoengineered materials via self-assembly.

For problems of material design ranging from molecular to mesoscale, the parameter space is very large, even combinatorially increasing, and the interaction between parameters can be complex. ML, thus, is a promising method for expanding the predictive power of experimental and computational datasets to predict the properties of novel materials and accelerate the material design.

4.1.3 Classifying Large Sets of Data

In select areas of material science and chemistry, high-throughput experimental and computational techniques are leading to the creation of massive databases. The defining characteristic of these high-throughput methods is that data is generated faster or exists in larger volumes than can be analyzed through traditional methods and new automated learning techniques are required [33, 82]. Figure 5 shows an illustration of how ML was applied [103] to computationally generated data to group noisy simulation data by the crystal type that self-assembled. Researchers had previously studied the dataset extensively by hand, but the automated learning technique was able to discover a new crystal structure for the first time. Figure 6 shows an example of how ML was applied to experimentally generated high-throughput data. Researchers discovered a novel rare-earth-free permanent magnet by performing an unsupervised cluster analysis of the diffraction data from a composition spread wafer that provided a combinatorial library of ternary alloy compositions [82].

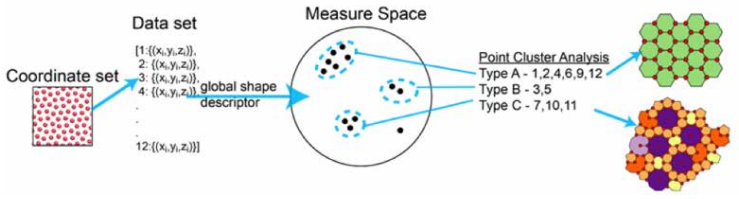


Figure 5: Illustration of unsupervised clustering algorithm finding similar crystalline structures in a dataset.

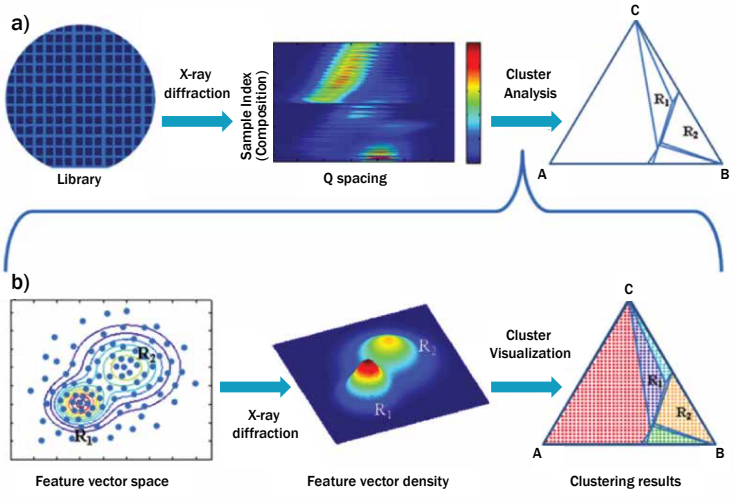


Figure 6: A cluster analysis of a high-throughput X-ray diffraction dataset is used to generate a structural phase distribution diagram [82].

The need for better methods for learning from large datasets is especially applicable to computational material science where improving computational resources and algorithms are accelerating the rate at which data is generated. However, as the rate at which data can be amassed through both computational simulation and experiment continues to accelerate, new methods of automatically recognizing and cataloging patterns in data based on machine learning will be necessary to alleviate the analysis bottleneck. These problems can be addressed via robust unsupervised ML techniques that can discover the common patterns in a dataset [103].

Feature learning from large datasets: A common problem across the fields of computational material science, chemistry, and physics is the need for robust feature methods to encode the high-dimensional data of an N-body system into a minimal vector representation, or feature vector. In the examples above, [96, 127, 124, 116, 65, 111, 45, 103], where ML has been successfully applied to a research area, the key to good performance was the selection of a good descriptor/feature vector. These feature vectors are used not just for ML, but also to support powerful metamodeling techniques that allow chemical and mechanical properties to be accurately quantified. The effectiveness of any ML method depends first on having a feature vector that captures the right invariances (rotational, translational, index permutations). Indeed, all successful application of ML to material systems discussed relies on careful initialization of the right feature vector. These feature vectors are referred to alternatively as collective variables, order parameters, and local or global descriptors. In general, the power of metamodeling techniques is directly limited by the correctness of the choice of feature vector and by the length of vector required.

Initial research efforts have been applied to designing ML methods for taking a large volume of simulation data and extracting lower dimensional coordinates [87, 101, 113] or learning the lower dimensional manifold that the higher dimensional simulation data occupies [45]. In general these are numerically intensive calculations that have been successfully applied to only a handful of small systems. Machine learning algorithms, such as deep learning, that could take a large body of data and automatically extract a minimal feature vector that separates ordered states from other ordered states as well as disordered states could have an immense impact on the field.

4.1.4 Pattern/Anomaly Detection for Light Sources

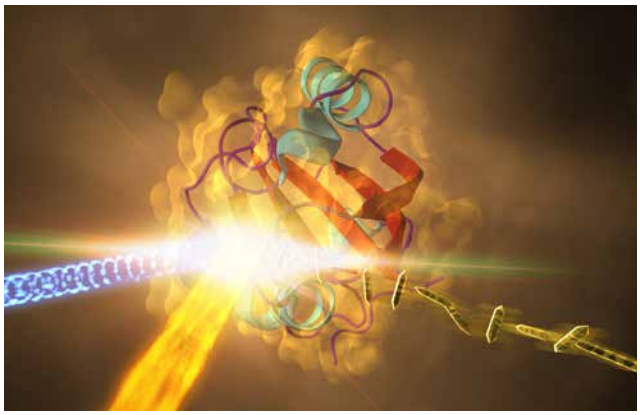


Figure 7: An artist's rendering depicts an experiment at SLAC that revealed how a protein from photosynthetic bacteria changes shape in response to light. Samples of the crystallized protein (right), called photoactive yellow protein or PYP, were sprayed into the path of SLAC's LCLS X-ray laser beam (bottom left). Some of the crystallized proteins had been exposed to blue light (left) to trigger shape changes. Diffraction patterns created when the X-ray laser hit the crystals allowed scientists to re-create the 3-D structure of the protein (center) and determine how light exposure changes its shape. Image courtesy of SLAC National Accelerator Laboratory [119].

As new scientific instruments and sensors are developed and used to collect an ever-increasing amount of data, ML techniques will play a critical role in helping scientists to triage and automatically find patterns of interest. The DOE Office of Science light sources: the National Synchrotron Light Source at BNL, the Stanford Synchrotron Radiation Lightsource at SLAC, the Advanced Light Source at LBNL, the Advanced Photon Source at ANL, and the Linac Coherent Light Source (LCLS) at SLAC, are all examples of powerful scientific instruments that can generate large amounts of experimental data. LCLS, for example, has enabled scientists unlock the mysteries of materials at the atomic and molecular level. As depicted in Figure 7, using the LCLS's X-ray Free Electron Laser's (XFEL) unprecedented brightness, spatial and temporal resolution, scientists recently imaged the highest-resolution protein snapshots ever taken with an X-ray laser, revealing how a key protein in a photosynthetic bacterium changes shape when hit by light [69]. Figure 8 illustrates how LCLS captures X-ray diffraction patterns resulting from the laser colliding with injected nanoparticles.

Deriving scientific insights using LCLS is a complex process that is hampered by several challenges that ML can help overcome. First, the massive amounts of rapidly collected X-ray diffraction imagery data is beyond human ability to fully inspect, let alone annotate. Totaling 20 terabytes per day, the imagery acquisition rates can exceed 10 GB/second for experiments running at 120 Hz. By 2017, next-generation XFEL facilities like LCLS-II and the European XFEL will offer pulse rates at several of orders magnitude beyond that of LCLS (tens of kHz to several MHz), leading to dramatically increased data rates. Second, of the millions of images collected each day, only a small fraction are useful, i.e., very few images actually contain well-resolved diffraction patterns from the particles of interest. Finally, LCLS is an exquisite instrument with many sensitive settings that allow the laser to have properties ideal for probing different samples leading to better images. Controlling these settings in real time would improve the yield of good images and improve the scientific productivity of LCLS.

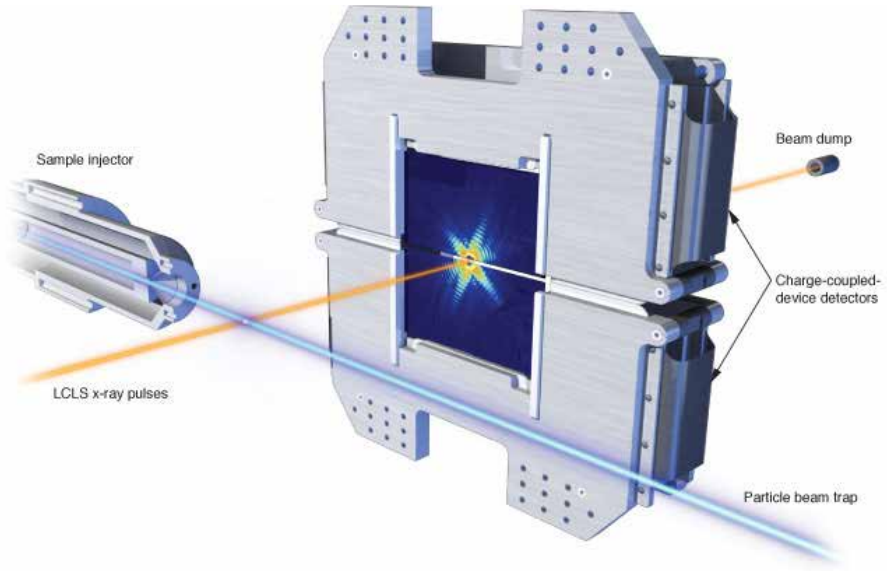


Figure 8: Nanoparticles are injected into the LCLS X-ray beam as an aerosol using a sample injector. Diffraction patterns are recorded with a pair of X-ray CCD detectors. Space between the CCD detectors allows the X-ray beam to pass through to a beam dump. (Rendering by Kwei-Yu Chu, LLNL.)

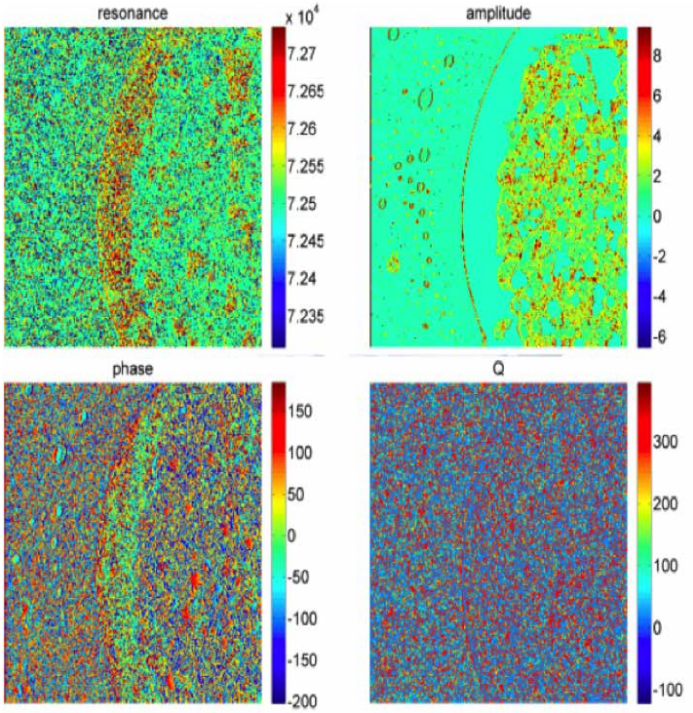
Recent advances in ML, specifically unsupervised feature learning with deep learning neural networks (UFLDL), hold the potential to overcome these aforementioned challenges and help accelerate the

scientific discovery process on LCLS. Given the massive amounts of unlabeled imagery, UFLDL can be used to learn fundamental basis patterns of the images. UFLDL is able to effectively learn basis patterns or features that help the neural network to reconstruct the original imagery without any supervision. These features can then be used to build a classifier layer of the deep neural network for automatically partitioning the images into useful and not useful categories offering LCLS scientists the ability to automatically identify good images. Additionally, the UFLDL features can be used for clustering the imagery into sets of similar images, which can help scientists to automatically sort their data. An added benefit to this approach comes from the fast speed with which the evaluation of new imagery is performed by deep neural networks. Deep neural networks accelerated by GPU hardware will be able to keep up with the image acquisition rates of the next generation XFELs. Finally, deep learning has shown some successes in the feedback control of systems. In reference [95], a deep learning model successfully learned control policies directly from high-dimensional sensory input using reinforcement learning. This deep convolutional neural network was able to learn how to automatically play several video games, even beating a human expert on several of the games. Applied to the control of an LCLS experiment, this approach can be used to improve the yield of scientifically valuable imagery.

Pattern Detection in Images

While the light sources present a range of daunting image processing challenges; there is a broader class of scientific domains that involve processing of data from microscopes, telescopes, satellites and so on. Identifying objects (shapes of patterns) in the image (e.g., unsupervised classification of cells in multidimensional XFM datasets [127]); classifying objects into categories and associating labels and descriptions are important tasks.

The computer vision community has made tremendous progress on related problems in object recognition and classification (CIFAR10 [80], ImageNet [112], Labeled Faces in the Wild [73]). Further research is needed into whether techniques applicable for finding readily labeled object categories can be translated to finding scientific objects of interest. One of the unique challenges posed by scientific datasets stems from the high dimensionality of data: a typical ImageNet dataset might be 256 x 256 x 3 in size; scientific image data often includes higher spatial resolution (1000 x 1000) and frequency measures (1K–1M) [127]. In these cases, applying relatively simple ML techniques like principal component analysis can fail.



(a) An example of scientific image data from the Center for Nanophase Materials Sciences at ORNL.

(b) An example from ImageNet.

Figure 9: Sample image data analysis problems: In (a) we are tasked with detecting a subtle edge in the middle of figures, while in (b) we need to detect a cat. Also (a) is a visualization of $256 \times 256 \times 16K$ dimensional data, while (b) is a $256 \times 256 \times 3$ dimensional data.

4.1.5 Analysis of Text Data

Scientific articles have been one of the primary sources for text analysis. With text analysis, we can determine the most frequent terms in each text document and the most distinctive terms of a text document from other documents in a set. In such a way, we can classify documents such as web pages, news articles, and scientific articles, with minimal human labors. Text analysis can help scientists to survey a huge amount of scientific literature across domains. This will be more effective when we are able to perform interdisciplinary research. For instance, with text analysis, we can analyze scientific articles from major outlets of each domain, without necessarily understanding all the backgrounds of each article. Thus, before creating a team of scientists on material science/chemistry/physics/ and pharmacy, we can set reasonable hypotheses on exploring new chemical compounds with certain pharmaceutical effects. After setting up hypotheses, scientists can organize a team to check if generated hypotheses can be meaningful or not. In that way, scientists can perform their studies in a less laborious and more timely manner. Or, perhaps, more unbiased and breakthrough hypotheses can be explored.

For instance, as shown in Figure 10, a survey paper in 1987 described a few factors associated with migraine and magnesium. Today, however, using a text analysis of the entire PubMed online library, we can identify 133,193 connections between migraine and magnesium. Digesting the entire PubMed online library would be a time-consuming task, but if we can accurately index documents and connect relationships between documents, scientists can more easily determine what will be an interesting hypothesis to test or what would be a proper experimental setting.

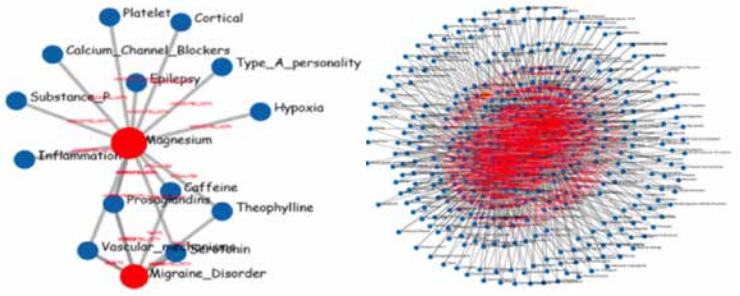


Figure 10: Comparison of the connections between migraine and magnesium. Eleven connections were found in 1987, while the present day count stands at 133,193.

4.1.6 Embedded Intelligence for Exascale Systems

In Topic 1, the distributed performance monitoring of the computer cluster itself was proposed as a streaming application to which online machine learning-based feedback methods could be used to tune the system. In a similar way, the distributed large-scale applications may themselves need to be treated as streaming applications. Here we propose that, in the future, to manage the I/O limitations relative to velocity of data produced, ML-based feedback methods will need to be applied locally to the data streams emitting from resources such as large-scale scientific simulations, large-scale experimental apparatuses, and distributed sensor systems such as Smart Buildings to triage and prioritize data.

It is already the case that limitation of I/O and data storage means that large-scale scientific applications and experimental systems can only store a small fraction of the data that is produced. For exascale computing, the fraction that will be stored will be even smaller. This will drive the need for more dynamic I/O decision making by applications; that is, making choices on-the-fly to store data based on its importance or significance rather than using the more common static models for storing data at fixed frequencies. This model of triaging data may well be similar to how raw data is handled from the Large Hadron Collider, where data is filtered and compressed locally based on its predicted significance, so as not to overwhelm I/O resources. For example, in large multi-scale, multi-physics scientific computations, there is often a desire to detect rare/fast/local events in the simulation, such as a rare nucleation event in a simulated supercooled liquid, or a computational event, such as a numerical error caused by too large of a time step, or the numerical evidence of a soft error (e.g., bit flip) on a node. Identifying and capturing the details of these rare and distributed events, whether to record statistics, rollback calculations, or divert resources, is likely to be an important part of minimizing I/O, handling hardware or software failures, and managing extremely large calculations. Often the exact form of these anomalous events may not be known ahead of time.

Machine learning-based methods that process streaming data from scientific applications will be an essential part of detecting recordable events or anomalies in the data stream (e.g., symptoms that the application or experiment is not performing as expected due to hardware with soft errors, algorithms, or a bug in the calculation), to allow a real-time, vice post-processing, response from system or scientist.

Such sampling and detection will need to be performed locally, without significantly impacting the application performance, and only lightly using the communication network. The detection of anomalies from streaming data in such a distributed environment without significantly impacting the performance of the system represents an interesting open problem.

4.1.8 Summary

We have presented vignettes documenting requirements from some of the leading scientific applications from simulations, large scale facilities, smaller scale instruments and HPC platforms. Table 1 presents a broader range of analytics tasks from all DOE applied offices, and how they map onto flavors of machine learning methods. Clearly, investments made in core ML research and production tools for deploying novel capabilities will pay off across the board for DOE. We now elaborate on specific challenges for the field of machine learning when faced with DOE's scientific requirements.

Table 1: Relevance of various ML techniques to a broad spectrum of DOE problems; Supervised Learning (SL), Unsupervised Learning (UL), Semi-supervised Learning (SSL)

DOE Science Domain	Analytics Problem	SL	UL	SSL
Climate/BER	Extreme Weather Detection	X	X	X
Astrophysics/HEP	Halo Finding		X	
Plasma Physics/HEP+FES	Tracking Magnetic Reconnections			X
Material Science/BES	Predicting Synthetic Materials	X		
Light Sources/BES	Pattern/Anomaly Detection	X	X	
Particle Physics Detectors/HEP	Pattern/Anomaly Detection	X	X	
Tokamaks/FES	Pattern/Anomaly Detection	X		X
Telescopes/HEP	Transient Detection, Data Fusion	X		
BioImaging/BER	Clustering		X	
Genome Sequencers/BER	Sequencing, Assembly		X	
Smart Buildings/ARRA			X	X
HPC systems/ASCR	Fault Detection, Perf. Prediction	X	X	X

4.2 Self-Aware Operating and Runtime Systems

The traditional operating and runtime (OS/R) system runs in an open loop where the mapping phase is static, the system runs the application in the same way repeatedly, irrespective of the state of the system and other applications running in the system. The mapping decisions are performed in an *ad-hoc* manner at design time. In contrast, SAOSR is an empirical, evidence-based system that observes the system and running applications to understand, model, predict, and optimize application behavior by making model-informed decisions. A high-level overview of the differences is shown in Figure 11. In summary, traditional OS/Rs:

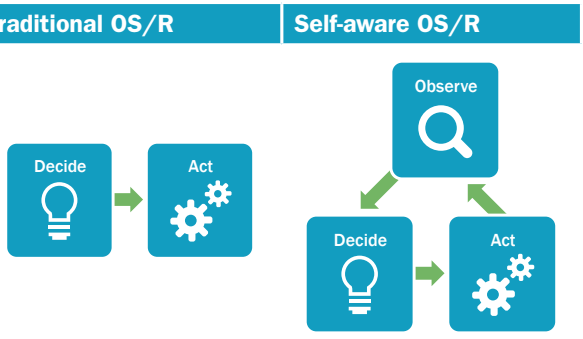
1. Force the application programmer to optimize for the system; and
2. Cannot respond to unpredictable runtime changes.

A self-aware OS/R will:

1. Optimize the system for the current application; and
2. Flexibly adapt to runtime fluctuations. Where the traditional OS/R is rapidly becoming a barrier to scientific discovery, the SAOSR will increase the rate and decrease the cost of future discoveries.

The SAOSR will adapt at different spatial scales, including the node, machine, and facility level. At the node level, OS/R will use multi-metric performance models (e.g., for run time, power, energy, memory footprint) based on the hardware/software/application knobs. At the machine level, OS/R will use models for communication, load balancing, and I/O. More importantly, it will try to understand how other applications are using the shared communication and I/O resources, how they affect the performance of the given application, and finally figure out the best way to run the given application. At the facility level, OS/R will use multi-metric performance models for facility-wide resource utilization, power constraints, user satisfaction, and time to solution to optimize job scheduling, staging phases of the application, and file transfers. Model-informed decisions will be made at every level with multiple, coordinated feedback loops. In particular, the coordination will happen both bottom up (node to facility) and top down (facility to node). The decisions made at the node-level should propagate up to the facility level and vice versa via interfaces that allow decision-making components (actors) to communicate.

The crucial component of the SAOSR is a closed loop and/or feedback loop that informs the SAOSR's own decision-making process. The SAOSR will refine and update its own performance models to take into account dynamic changes in the system and model inaccuracies. This feedback loop will help the OS/R to constantly adapt itself (hence the "self" portion of self-aware OS/R) and improve the system's overall effectiveness in aiding scientific discovery. This use of feedback and self-monitoring is the key distinguishing feature of the self-aware OS/R. In summary, the responsibility of efficiently mapping the application to the extreme-scale system will be the responsibility of the self-aware OS/R through accumulation and use of knowledge gained from the application and underlying hardware.



	Traditional OS/R	Self-aware OS/R
When Decisions Made	Design Time	Runtime
How Decisions Made	Ad-hoc, based on guesses about future	Evidence-based
Understanding User Goals	No	Yes
Optimizes For	System Metrics (Utilization)	Application Metrics (Science accomplished)
Performance	Static	Improves without user action

Figure 11: Comparison of a self-aware OS/R with a traditional approach.

4.3 Example of Future OS/R

Almost all technical reports on exascale systems identify the power consumption of the computers as the single largest hardware research challenge [12, 18, 37, 72]. This challenge increases when we couple simulations with each other or with *in situ* analysis. In this section we illustrate the benefits of a future SAOSR through an example which optimizes the energy consumption of a clustering application which would analyze the data from a scientific simulation. Specifically, we look at the K-means clustering benchmark from MineBench [98].*

* We emphasize that this example shows what is possible. We do not intend to limit the scope of the self-aware OS/R to this application, techniques, or even the general power/performance optimization problem.

4.3.1 Challenges

Optimizing K-means clustering is difficult for several reasons. First, it will be coupled with a simulation. Optimizing a coupled application is a distinct challenge from traditional HPC optimization problems. Instead of making K-means run as fast as possible, our self-aware OS/R should make it run at the same speed as the application it is analyzing while minimizing its power consumption. Second, this optimization problem requires a great deal of knowledge to solve. More than knowledge of the single fastest or most energy-efficient system configuration, solving this problem requires knowledge of the power and performance available in all system configurations and the extraction of those configurations that represent Pareto-optimal tradeoffs. Acquiring this knowledge is additionally complicated by the fact that these power/performance tradeoffs are often application-dependent or even input-dependent.

4.3.2 Approaches

Machine learning techniques represent a promising approach to addressing this problem. Offline approaches collect profiling data for known applications and use that to predict optimal behavior for unseen applications (example systems using offline approaches include [129, 115, 85, 83, 26]). Online approaches use information collected while an application is running to quickly estimate the optimal configuration (example systems using online approaches include [86, 102, 117, 104, 11, 84]). Offline methods require minimal runtime overhead, but cannot adapt to particulars of the current application. Online methods customize to the current application, but cannot leverage experience from other applications. In a sense, offline approaches are dependent on a rich training set that represents all possible behavior, while the online approaches generate a statistically weak estimator due to small sample size.

The strength of this approach is that it quickly matches the behavior of the current application to a subset of the previously observed applications.

We can combine the strengths of both offline and online learning using a Hierarchical Bayesian Model (HBM) model (the implementation is called LEO) to estimate power and performance tradeoffs [93]. We assume that there is some set of applications for which the power and performance tradeoffs are gathered offline. HBM-based approaches use that set of known applications to form prior beliefs about the probability distributions of the power and performance achievable in different system configurations. Given that information, the HBM takes a small number of observations of the given application and uses a hierarchical model to estimate the power and performance for that application in all the other configurations. The strength of this approach is that it quickly matches the behavior of the current application to a subset of the previously observed applications. For example, if the HBM has previously seen an application that only scales to 8 cores, it can use that information to quickly determine if the current application will be similarly limited in its scaling.

4.3.3 Results

For this example, we run on a 16-core Linux x86 server with hyperthreading (allowing up to 32 cores to be allocated). The system has 16 different clock speeds and 2 memory controllers. With these three parameters, there are 1,024 possible system configurations. For any given speed of the simulation, we would like K-means to match it and minimize power consumption. If the self-aware OS/R can quickly estimate the power and performance tradeoffs for this application, it can determine the optimal performance configuration for any power limit.

To illustrate the benefits of HBMs, we compare it with three other approaches: heuristic, offline learning, and online learning. The heuristic uses the well-known race-to-idle strategy—simply allocating all resources (cores, clock speed, etc.) to K-means and then idling the system once the application completes. The offline learning approach builds a statistical model of performance and power for each configuration based on prior measurements of other applications. The online approach uses quadratic regression to learn the tradeoffs for each configuration while K-means is running.

The heuristic approach simply assumes that the most energy-efficient configuration is the one where all the system resources are in use, but that has been shown to be a poor assumption for this type of application [71, 90]. The offline approach predicts average behavior for a range of applications, but it may be a poor predictor of specific applications (K-means, in this case). The online approach will produce a good prediction if it takes a sufficient number of samples, but the required number of samples may be prohibitive.

HBMs combine the best features of both the offline and online methods. At runtime, it changes system configurations, observes the power and performance, and combines this data with that from previously seen applications to obtain the most probable estimates for other unobserved configurations. The key advantage of the HBM approach is that it quickly finds similarities between K-means and previously observed applications. It builds its estimation not from every previous application, but only those that exhibit similar performance and power responses to system resource usage. This exploitation of similarity is the key to quickly producing a more accurate estimate than either strictly online or offline approaches.

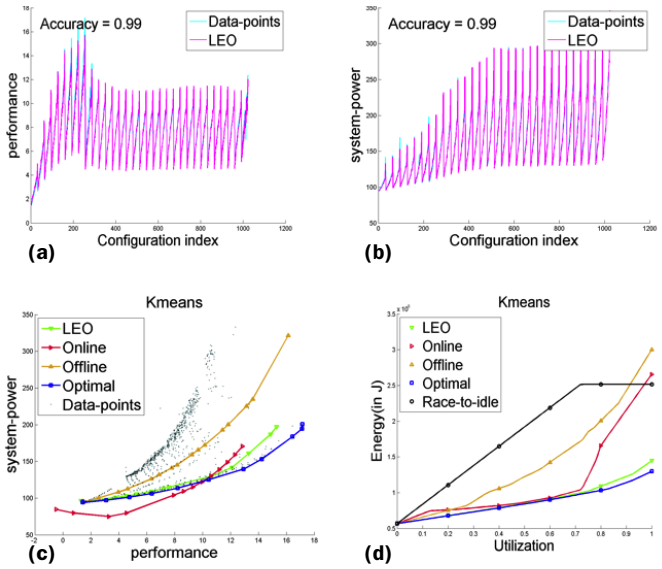


Figure 12: Estimation for K-means clustering using LEO, an HBM implementation. LEO accurately estimates performance **(a)** and power **(b)** for 1024 possible system configurations, allowing LEO to construct the Pareto-optimal frontier of power and performance tradeoffs **(c)**. These frontiers are then used to determine minimal energy configurations for various system utilizations **(d)**.

Figure 12 shows the results for this example. Figure 12 **(a)** shows LEO's performance estimates as a function of system configuration, while Figure 12 **(b)** shows the power estimates. These runtime estimates are then used to reconstruct the Pareto-optimal frontier of the power/performance tradeoff space shown in Figure 12 **(c)**. This last figure shows the estimates produced by the offline and online approaches as well. Finally, Figure 12 **(d)** shows the effects of using these estimates to determine optimal system configuration for various utilization levels. As can be seen in the figures, LEO is the only estimation method that captures the true behavior of the application and this results in significant energy savings across the full range of utilizations.

Learning the performance for K-means is difficult because the application scales well to 8 cores, but its performance degrades sharply after that. In addition, the consideration of clock speeds and memory controllers mean that performance is a non-linear function of the configuration. It is, in general, difficult to find the true optimal of a non-linear function without exploring every possible configuration. The offline learning method predicts the highest performance at 32 cores because that is the general trend over all applications. The online method predicts peak performance at 24 cores, so it learns that performance degrades, but would require many more samples to correctly place the peak. LEO, in contrast, leverages its prior knowledge of an application whose

performance peaks with 8 cores. Because LEO has previously seen an application with similar behavior, it is able to quickly realize that K-means follows this pattern and produce accurate estimates with just a small number of observations. Furthermore, using LEO to allocate for energy efficiency produces an energy consumption just 6 percent above optimal compared to 50 percent increased energy for the online approach, over 2× for offline and over 3.5× for the race-to-idle heuristic.

We now outline three technical approaches key to developing the self-awareness necessary to accelerate scientific findings while decreasing cost per insight.

While this example illustrates a few of the potential benefits of SAOSR and gives reason to believe that this is a promising area of research, there is much research to be done to explore such techniques in the context of exascale systems. This example was carried out for one application on a small-scale, homogeneous system. The research program for /sas will be done in the context of extreme-scale systems, a wider range of DOE applications, and multiple-user systems with heterogeneous hardware, where the dimensionality of the optimization problems is much larger.

5 Machine Learning

5.1 Machine Learning

Enabling Approaches for a Self-aware OS/R
Given the need for an extreme scale system that autonomously improves over time, we have described a SAOSR by illustrating its key properties and examples. We now outline three technical approaches key to developing the self-awareness necessary to accelerate scientific findings while decreasing cost per insight.

Self-awareness sets up an extensive set of OS/R requirements:

- The analytic models within the OS/R that we are accustomed to developing manually must be automated because they do not suffice. The SAOSR has more components and more interactions than are tractable for a human to manage. Therefore automated models are needed. Sometimes the parameters of these models will need to be adaptively tuned. Other times the parameters of the models themselves will need to be revealed from within large candidate sets.
- Optimizations will be performed on behalf of an application or performance engineer. The objectives will be application-specific and the SAOSR will need to respond to the "knob selections" of the performance engineer. For this, automated optimization is needed because this information has to be used at execution time.
- The SAOSR will optimize resources, recognize patterns, load balance and re-adjust in multiple ways. Some of the logic for this re-adjustment is dependent on what applications will be executing–other parts of the logic are dependent upon information supplied by the application or performance engineer while yet others are architecturally specific. Some tasks may call for a single strategy covering the use cases, others for different strategies for different ones, or the SAOSR may yet need to, on the fly, devise a strategy "online."
- For runtime decision making, the SAOSR should exploit multi-objective optimization, where the search algorithm should optimize multiple conflicting objectives simultaneously, which yields a Pareto-optimal set of solutions for exploring potential tradeoffs among multiple objectives. For example, using the offline models, select the Pareto front for different runtime scenarios and metrics; identify the operating condition at runtime and choose the appropriate solution from the Pareto front; when the models require significant

calibration, use the offline Pareto front as a (warm) starting point to potentially refine the solution. Subsidiary methods need to be developed to address the variability and noise inherent in many of the metrics of interest.

- In general, the SAOSR needs to gather and exploit measurements of the complex inter-relationships between its own components, among coupled applications and between the application and its components. This process requires automated system identification and variable sensitivity analysis.
- With the help of instrumentation and interfaces that communicate key measurements and objectives, the SAOSR's actors need support for the decisions that guide their actions.

Machine learning offers a way to approach many of these requirements. Whereas some of them seem to require very explicit information about a system that is too complex to specify explicitly, machine learning allows experiments; i.e., proposed models, candidate optimizations of complex configurations to be empirically tested. It uses the collected feedback (*post hoc* or immediately, with or without labeling) as its exemplars for inferring general solutions. Whereas some of the requirements demand information that doesn't become available until execution, machine learning allows the SAOSR, at that point, to experiment with solutions that it can improve upon once their merit or the gradient of improvement is determined.

Machine learning can also accommodate the requirements for awareness-based intelligence to be acquired on different temporal scales: offline, online, and in hybrid circumstances. For offline requirements, the SAOSR can collect experiential data in the aggregate and use machine learning to infer a model or actor logic that can then be transferred to the SAOSR. Here both supervised and unsupervised techniques can be exploited. For online requirements, the OS/R is faced with noisy, limited data and, because it is important to be sensitive to learning overhead, reinforcement learning techniques are appropriate.

In general, machine learning allows the SAOSR to convert the empirical data of experiments into knowledge that it accumulates and uses to support continuous improvement. At the same time, it is important to recognize that machine learning will not solve every problem. The SAOSR requires advancements in the state-of-the-art machine learning algorithms from outside the system's context so that it has a bigger and better set of general techniques at its disposal. We anticipate that many challenges will arise in the customization of techniques to the extreme scale or SAOSR context. Indeed, we anticipate that SAOSR will motivate and be the proving ground for DOE research generating advances in fundamental machine learning techniques.

5.2 Data Collection, Management and Integration

Self-aware operating systems and runtimes must sense the (software and hardware) environmental features that impact the performance and behavior of the platform and incident applications. Such environmental data includes static features, such as processor/core configurations and network performance capabilities, and dynamic features, such as current system load and observed network performance. In the context of collecting and managing this data, we identify several research challenges:

- **Identifying what data to collect**: machine learning components will facilitate feature extraction to identify what data is relevant to the various OS/Rs optimization challenges. Additionally, application users and system administrators should be able to specify user and system goals. For example, a user may specify behaviors or characteristics that denote application progress, such as loop indices. Accordingly, user and administrator input may also dictate what data should be collected.
- **Identifying data collection points**: Once the desired set of data is established, we must identify the necessary (or perhaps best) data collection points. These points can occur anywhere along the runtime software stack from the application code to auxiliary libraries to the middleware and runtime to the operating system.
- **Instrumenting data collection points**: Data collection points must be instrumented to collect desired data. Such instrumentation may be static (encoded *a priori* in the application or OS/R software, compile-time) or dynamic (online) during the system's execution.
- **Controlling raw data**: As a system executes, the amount of raw data needed may change over time. It is important to consider mechanisms that can throttle data collection rates and data filtration mechanisms that can sift collected raw data.
- **Managing data volumes**: data aggregation mechanisms can be useful for reducing data volume often without loss of information. Aggregation mechanisms should be flexible and customizable since they may target different types of data or different usage scenarios.

- **Establishing holistic views**: In order to gain system-wide awareness and make holistic adaptation decisions, the SAOSR must either harness distributedly collected data to singleton learning and decision processing elements or the system should allow individual elements to use local data to make local decisions that meet global constraints and goals. The former is conceptually easier but requires efficient and highly scalable data propagation mechanisms. The latter does not require data movement, but is conceptually more difficult to realize.

Additionally, to establish holistic views, data collected from different elements (different levels of the software stack or different scopes with the same level of the stack) may need to be integrated and reconciled. For instance, it may be necessary to align or correlate performance-counter data collected within the OS kernel with functions at the application level. Another example may be correlating the observed application performance data with a particular configuration of the SAOSR.

5.2.1 Interfaces

One of the key approaches to building a self-aware OS/R will be defining and implementing the interfaces that enable different components to transfer relevant information. We identify three main components that will be present in any self-aware OS/R implementation. The first is a sensor that can be used to monitor the current state of the system and application. The second is an actuator which can be configured to change the behavior of the system. The third is an actor, or an individual self-aware component which monitors the sensors and determines actuator settings.

Given these components, a self-aware OS/R should include the following interfaces: one to communicate monitoring information from sensors to actors, one to communicate settings from actors to actuators, and one to communicate between independent actors. Each is discussed in more detail below.

- **Actor to Actor**: A large exascale system will not be managed by one monolithic actor because of the scale of the system and because of the need to accommodate separately developed resource managers that manage different resources. This leads to two problems. The lesser one is to avoid duplication in the collection of data from the same sensors. The more significant problem is to ensure that different actors do not act at counter purpose: Even if two actors, in isolation, manage resources in an efficient manner, their coupled action may result in a system that is unstable and operates far from optimum. To avoid this problem we need a good theory on the composition of feedback loops and interfaces that support such a composition. We expect that actor-to-actor communication will support the hierarchical and distributed management of resources, their coordination, as well as composition of strongly interaction feedback loops at the same level.
- **Sensor to Actor**: We envision that sensor to actor will manage the communication and will have additional intelligence for handling noise, sensor failure detection, and data aggregation.
- **Actor to Actuator**: Controller for actor to actuator needs methodologies for distributed communication settings.

5.3 Metrics

In order to motivate research on a SAOSR, we need to establish high-level metrics. Our intent is to establish metrics that can be used to establish the value of the research, but some of these metrics may also be thought of as objectives of the run-time system. The overarching metric is that the self-aware OS/R should optimize application progress that matters to scientists. We focus on application-level progress because traditional system utilization metrics can be misleading. For example, a system may be highly utilized doing "busy work" that is less productive than an alternative. We also recognize that this application progress must be determined in the context of constraints, which could be either fixed or negotiable. Relevant constraints include power consumption and hardware resources used. This single overall goal can be broken into sub-goals that address performance, performance portability, and productivity and cost issues.

Performance portability will become a first-order goal for exascale systems because of the extreme complexity of these systems. In the past, performance portability has been a second-order goal because scientific application owners were often willing to invest the time required to get the necessary performance for an application on a specific system. For exascale systems, it will be infeasible for application owners or even performance engineers or programmers to meaningfully optimize an application for a system of that complexity.

An envisioned SAOSR will do this performance optimization automatically and will do the optimization better over time, so that programmers will not have to. As a side effect, the application will have performance portability and will be expected to achieve good performance on any system that

implements the SAOSR. The performance portability applies to an application running on a single system with different resources available over time (for example, as other applications are run or more resources become available or as resources fail or are taken away). Performance portability also applies to running an application on different instantiations of a given architecture, potentially with different numbers of nodes or different mixes of heterogeneous resources. Performance portability potentially applies to completely different architectures too, as long as the system implements the SAOSR interfaces. A potential secondary metric is to characterize how stable application performance is across different systems, where stability may be defined as a function of system resources (which may be complicated to define).

We have also considered a number of cost metrics, where cost is defined broadly to include metrics that should be reduced to minimize the cost of producing scientific discovery. These cost metrics include application programmer time, system administrator time, performance engineering time (which may or may not overlap with application programmer and system administrator time), and operating costs (such as power consumption). Measuring the productivity of programmers and engineers is notoriously difficult. Nevertheless, we believe it is important that we recognize these metrics and the improvement in these areas that a SAOSR would bring, and the problems in these areas that the DOE will have if a SAOSR is not developed. Since a SAOSR will optimize the performance of applications automatically, the application programmer will not be expected to consider performance in the specification of the application program, so this will reduce the programmer's time, and the metric of application programmer time will capture that savings. With today's systems, system administrators must manually examine log files to try to determine the root cause of performance issues. With a SAOSR, this functionality will also be performed automatically, so the metric of time that system administrators use to investigate performance issues will also be reduced. Finally, there is a whole class of programmers and engineers that work on optimizing performance of applications. We hope that a SAOSR will allow us to repurpose those programming and engineering cycles more directly toward scientific discovery.

5.4 Motivating Science from Topic 3

The research results supported by DOE's scientific user facilities will shape national and worldwide policy in climate change and energy production and usage. These scientific results must be accurate and trustworthy for science to drive these policies.

Much of the scientific discovery will be done with exascale computing, with the main scientific areas driving the need for improved resilience and highly trustworthy results being:

- **Combustion science**: Creating a fundamental understanding of combustion to increase efficiency by 25–50 percent and lower emissions from internal combustion engines using advanced fuels and new, low-temperature combustion concepts.
- **Climate change science**: Understanding the dynamic ecological and chemical evolution of the climate system with uncertainty quantification of impacts on regional and decadal scales.
- **Energy storage**: Gaining a fundamental understanding of chemical reaction processes at the atomic and molecular level required for predictive design of new materials for energy storage and predictive engineering of safe, large-format, durable, rechargeable batteries.
- **Nuclear power**: Enabling reactor-scale simulations to allow safe, increased nuclear fuel burn times, power upgrades, and reactor lifetime extensions, and in doing so reduce the volume of spent fuel.

Very sophisticated data analysis methods are needed to ensure that an HPC system is running correctly, despite faults, and that the output of an application is accurate, reproducible, and trustworthy. There are a wide variety of data analysis methods ranging from statistical to sampling to machine learning approaches. While all can be very effective for understanding a dataset, statistical and sampling approaches typically are driven by hypothesis, while machine learning approaches are driven by data. Since only a small percentage of faults in an HPC system are clearly understood, data-driven machine learning appears to be the best approach. Machine learning approaches have been highly effective in understanding high-dimensional data, and discovering hidden associations and higher-order effects within the data.

Beyond exascale computing, there is the question of the trustworthiness of scientific results from DOE user facilities. There are a number of potential problems when running a scientific experiment, from instrument errors to recording errors to analysis errors. A major challenge to the scientific community is the ability to reproduce published results. This drives the need for methods that can quickly and accurately determine if scientific results are self-consistent, and consistent across experiments. Unsupervised machine learning methods have been shown to be effective in comparing results and show promise in addressing these types of problems.

The final motivation is for scientific discovery; the vast majority of scientific results are never analyzed due to the sheer volume of data. Supervised and semi-supervised machine learning methods can be used by creating training sets from desired simulation and experimental results, and use these training sets to automatically find desired results within new results. This will give scientists confidence that the entire result sets have been reviewed, and that interesting data have been highlighted.

6 Challenges of Machine Understanding and Learning

6.1 Challenges of ML for Scientific Discovery from Topic 2

6.1.1 Four V's of Scientific Big Data: Volume, Variety, Velocity, and Veracity

DOE science application areas listed in the previous section require research into a range of ML methods. Table 2 summarizes the unique challenges pertaining to these applications. This is important to appreciate, especially in the context of commercial big data analytics, wherein active research and software development is being conducted, albeit for applications of a different flavor. As Table 2 indicates, the volume aspects of scientific big data is substantial. Single simulation runs or experimental acquisitions may result in datasets in the range of O (100GB–100TB). We only expect this number to increase with the availability of exascale-class systems. Thanks to a Moore's Law-like improvement in electronic fabrication, we expect experimental devices to also increase their pace of data acquisition. The variety challenges in scientific big data is tremendous: simulation output is typically multi-variate and spatio-temporal in nature. The phenomena of interest might

DOE Application	ML Challenge	Volume	Velocity	Variety	Veracity
Climate/BER	Pattern Detection	O(100)TB	N/A	MV, ST	Simulation accuracy
Material Science/BER	Prediction, Regression	O(100)GB	N/A		
Light Sources/BES	Pattern Detection	O(10)TB	100 GB/s	MM	Noisy sensors, missing data
Telescopes/HEP	Pattern Detection	O(100)TB	10GB/s	MM, MI	Sensor noise, acquisition artifacts
HPC systems/ASCR	Log Analysis, Fault Detection	O(1)TB	100 GB/s	HC, EM	Missing data

Table 2: Characterization of scientific big data problems along volume, velocity, variety and veracity dimensions. Size/velocity estimates are for problem sizes and configurations circa 2014; multi-variate (MV), spatio-temporal (ST), multi-modal (MM), multi-instrument (MI), hardware counters (HC), and environmental monitors (EM).

be multi-scale, spanning a range of temporal durations. For experimental acquisitions, procedures requiring inference across multiple instruments might require data fusion across multiple devices with varying characteristics. Finally, the veracity challenges in scientific data are worth highlighting. Any reasonable statistical inference procedure needs to account for the quality of the dataset, the properties of the noise inherent in the sampling process, and any missing data. While simulation datasets typically do not suffer from quality issues; these concerns become paramount when processing observational products.

Scarcity of Labeled Training Data

Scientific analytics faces the recurring challenge of relative scarcity of annotated data available for supervised learning. There is, however, a nearly infinite supply of unlabeled data that is constantly being collected, and with the proliferation of new sensors and the growth of inexpensive data storage, the rate of data collection is likely to continue to outpace that of human hand-annotation. The commercial world, buoyed by the success of purely supervised training models like Google's deep convolutional network [81] for image classification and Baidu's deep recurrent network for speech recognition [64], has chosen the approach of collecting more annotated training data to improve ML performance. Utilizing crowd-labeling tools like Amazon Mechanical Turk, companies can obtain plenty of hand-annotated data for supervised learning. Crowd-labeling, however, is often not a viable approach for many DOE mission-relevant datasets which require more advanced understanding of the data for accurate annotation. Additionally, budgetary and policy considerations may make crowd-labeling infeasible due to monetary or data sharing constraints. For these reasons, it is important to continue research on improving purely unsupervised or semi-supervised machine learning algorithms that can learn with none or few annotated training data.

6.1.2 Broad Applicability to Multiple Science Domains

Feature Engineering

For the vast majority of machine learning techniques, data is not fed to the algorithm in a raw form for the analysis. Rather, data is fed to the algorithm in a pre-processed form known as a feature vector, which is a lower-dimensional representation of the data. While there are some techniques to assist in reducing the dimensionality of complex data, generally speaking, the best feature vectors are handcrafted, that is based on expert knowledge of what variables are likely to be important. In many types of complex data, it remains to be seen whether automated techniques can recover the same powerful expert handcrafted features.

While ML techniques may have broad applicability across domains, how to generate robust feature vectors does not. Generally, the predictive power of an ML technique is directly dependent on how well the feature vector has been constructed (e.g., Are there too many or too few features? Are they invariant to details of the problem that are irrelevant?). Therefore, often, the majority of the work in generating good predictions from machine learning lies in engineering the best vector of features for a given problem. Determining how to dimensionally reduce the complex domain data so that ML techniques can be used is an open research problem in every scientific domain and, by far, the largest research obstacle.

Interpretability

One difference between developing predictive models in industry versus by scientists is the latter have a vested fundamental interest in understanding why X predicts Y. While sometimes a robust predictive ML model is sufficient, generally scientists are more interested in extracting insight from the output of data analysis algorithms. Standard ML methods are designed to do the former over the latter. Thus a major challenge in extending existing machine learning tools for scientific applications is to develop them into more interpretable data analysis methods.

We point out that improved interpretability is a problem common to many powerful abstract mathematical techniques, where advances in interpretability directly lead to better and wider application to scientific domains. For example, in contrast with Principal Component Analysis and SVD-based data analysis methods, CX/CUR matrix decomposition [89, 100], are low-rank matrix

decompositions that are explicitly constructed from actual data elements and thus are interpretable in terms of quantities that domain scientists understand. These methods have been applied to large-scale data problems in genetics, astronomy, and mass spectrometry imaging [89, 100].

Usability

Machine learning is rapidly becoming a critical tool for performing science, moreover a tool that will be wielded increasingly widely by non-experts. By out-of-the-box applicability, we mean that the degree of ML-specific expertise that a domain scientist should need to master to successfully apply ML should be minimal. Here we describe the common obstacles faced by a domain scientist in successfully deploying ML techniques.

- Since many machine learning algorithms offer similar capabilities, choosing the best algorithms to induce the appropriate model requires an expertise in individual algorithms. Which ML algorithm or set of algorithms should be selected?
- Each ML algorithm has parameters that need to be tuned to achieve best performance. Often choosing these parameters is more art than science, and involves trial-and-error selection or a deep knowledge of the workings of the algorithm.
- Ultimately, choosing the right volume of training data, combination of features, algorithm, and parameters is determined by the complexity and details of the underlying model, which is in turn, what the scientist is seeking to discover. Thus, successfully implementing a ML model is inherently an iterative process.

A robust machine learning framework should:

- Allow different algorithms to be easily interchanged for rapid experimentation;
- Provide tools designed for scientific applications;
- Provide diagnostic tools to aid the selection of tuning parameters;
- Provide tools to assist with cross-validation of model choices;
- Provide scalable versions of algorithms; and
- Be provided in a library that can be easily implemented in an HPC environment.

This definition of usability highlights the dual and sometimes conflicting goals: develop high-performance methods; and develop methods that are more consistent with user productivity. For example, computing even basic statistical regression diagnostics, such as the diagonal elements of the hat matrix, on a terabyte- or petabyte-sized matrix is currently not possible in general. Interactive analytics tools need not be the highest performance, but integrating them in a high-performance environment such as is supported by DOE labs is an important challenge.

6.1.3 Deep Learning Frameworks for Science

It is well understood that successfully applying the most proven and robust ML algorithms to a dataset involves first crafting the best low-dimensional representation of the data for applying the ML algorithm to, or feature engineering. Part of the attraction of deep learning algorithms (versus shallow learning algorithms) is that they promise to avoid the need to handcraft a feature set. This is a powerful claim that, in theory, could revolutionize the application of ML to new problems.

First, however, this capability has yet to be demonstrated outside of a handful of problems more related to industry applications than science applications. Second, deep neural networks have an especially large number of tuning parameters for controlling model complexity, which ultimately affect how well the model fits the data. Finding the best parameter set via a cross-validation optimization procedure is a computationally intense effort.

Here we see a significant opportunity for the DOE to make a unique contribution to the field of machine learning where the contribution could have a significant impact on DOE scientific applications and where we cannot count on industry to take the lead. What especially differentiates the DOE from other science-funding organizations with respect to this are DOE's extensive HPC resources. An HPC environment may be highly suitable and even necessary for applying and tuning deep learning methods, first for accelerating the core neural network learning process, and, second for parallelizing the training of models with different parameter settings.

6.2 Challenges of ML for High Performance Computing

We now review the current state of production analytics stacks in the commercial world and comment on requirements for developing and deploying ML software on extreme-scale systems.

6.2.1 Production ML Software Frameworks for HPC systems

We consider the challenges of creating an HPC ML software framework in the context of the current HPC-ABDS (High Performance Computing - Apache Big Data Stack) software stack. ABDS-Stack [50] presents a comprehensive list of 289 data processing software from either HPC or commercial sources. Many critical components of the commodity stack (such as Hadoop and HBase) come from Apache projects. We note that data systems constructed from this software can run inter-operate on virtualized or non-virtualized environments aimed at key scientific data analysis problems. In Figure 13 we further layer some of the HPC-ABDS subsystems and contrast HPC and ABDS. We believe there are many opportunities for DOE to leverage the rich ABDS software ecosystem by evaluating the software on the left side of Figure 13 and selectively incorporating capabilities, for instance.

In some cases like orchestration, there are new approaches like Apache Crunch that should be compared with the mature HPC solutions. In areas like streaming, there is no well-established HPC approach and direct adaptation of ABDS to DOE's requirements is appealing. An in-depth investigation needs to be conducted for various layers in the complete ABDS stack [50] and Figure 13.

More generally, since linear algebra methods are at the heart of many machine learning algorithms, there is an important need—and one to which DOE capabilities synergize very well—to develop novel linear algebra theories and frameworks that go beyond optimizing gigaglops and wall-clock time to considering metrics such as implicit regularization, computing to different levels of precision, considering power and performance in conjunction with other metrics, developing tools as uniformly as possible in extreme scale applications, and developing tools for ill-structured sparse matrices.

We list the following specific challenges:

- **Providing high performance in the context of ABDS software:** Since most of ABDS emphasizes scalability over performance, an important objective is to determine how to also produce high performance environments. This requires addressing better node performance and support of accelerators like Xeon Phi and GPUs.

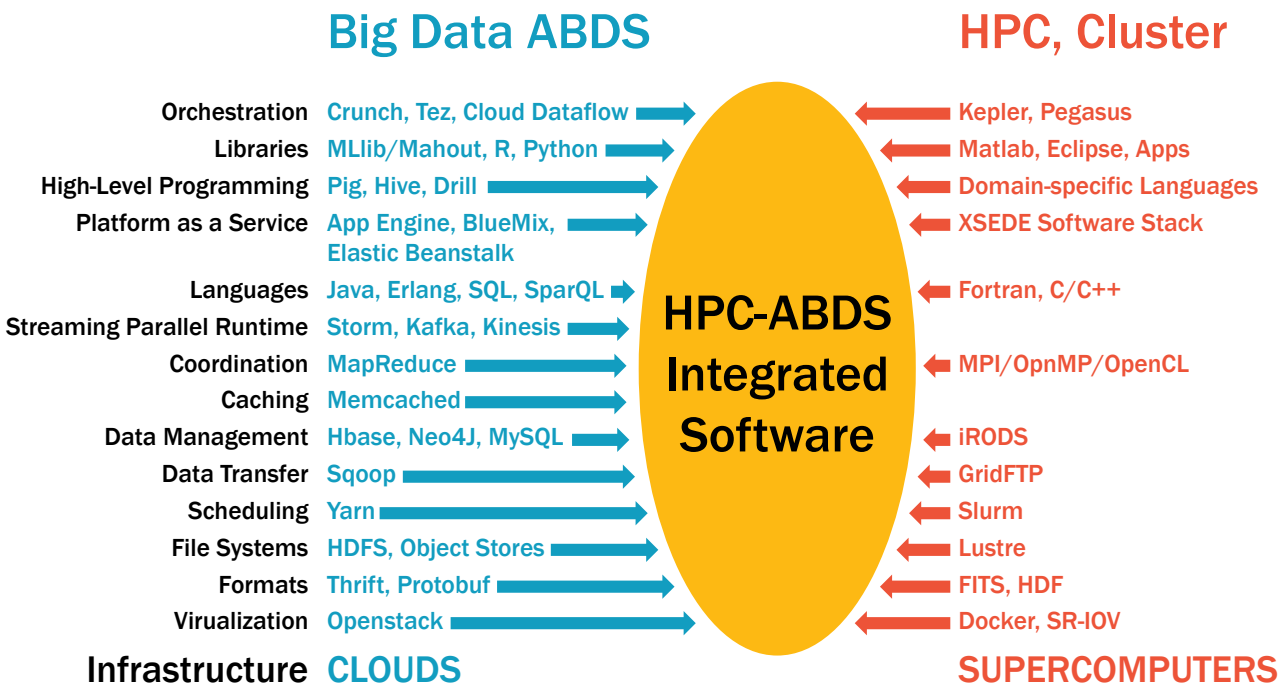


Figure 13: Comparison of current data analytics stack for cloud and HPC infrastructure.

- **Data movement and resilience:** Most commercial ML software is geared towards throughput-oriented performance on commodity clusters, which are at a different design point compared to HPC systems. Considerations such as extreme concurrency, data movement, and resilience will become important on exascale-class platforms and need to be considered.
- **Storage and data management:** Currently, most scientific data analysis is centered on files. However, we expect that in the future, scientific data analysis will expand to integrate approaches of Object stores, SQL and NoSQL. HPC distributed and parallel storage environments need to be reconciled with the data parallel storage seen in HDFS in many ABDS systems.
- **Communication, (high level or basic) programming, analytics and orchestration:** These areas have seen rapid commodity/commercial innovation. Reconciling these layers with an HPC environment will be challenging, although there is substantial development here to leverage off.

6.2.2 ML Algorithms at Exascale

State-of-the-art research and software for scalable machine learning has focused on scaling the algorithms for multi-core systems and clusters with commercial off-the-shelf processors. However, with exascale systems on the horizon, power consumption of individual components and cost of data movement are expected to be the primary deciding factors in achieving sustained performance. Thus, power consumption has resulted in the advent of revolutionary architectures, which are expected to execute at near threshold voltage (NTV). This includes several many-core architectures (NVIDIA GPUs, Intel Xeon Phi, AMD APUs), in addition to several more on the horizon. Much of the existing ML software has focused on using multi-core systems, but is not suitable for lightweight many-core architectures. As a first step, there is a need to design programming frameworks and candidate algorithms which can seamlessly leverage a combination of multi-core and many-core systems.

An undesirable impact of NTV execution is the tremendous increase in soft errors, which can possibly result in silent data corruption (SDC). A few case base studies exist on the impact of soft errors on the accuracy of respective applications. In many cases, iterative applications—which should self-correct themselves—have been shown to converge incorrectly. It is imperative to study the effect of soft errors on ML algorithms, and to design fault-tolerant algorithms. In addition to silent errors, permanent faults are expected to increase sharply due to combining individual components at massive scales. A performance-only optimization of machine learning algorithms *de facto* in the machine learning community is insufficient in solving the big data challenge on exascale systems using machine learning. As a result, it is critical to understand and address the delicate balance of power, performance, and reliability in designing machine learning algorithms, which has been largely ignored in the machine learning community.

There are several complementary efforts in DOE (X-Stack2, Co-Design centers, FastForward2) addressing the triage of power, performance, and reliability. However, these efforts are necessary, but insufficient in addressing the challenges imposed by ML algorithms. Machine learning algorithms feature a richer variety of data structures and communication patterns.

Resilience is an open issue for machine learning algorithms, while it is being addressed for legacy algorithms such as PDEs. Similar to PDEs, machine learning algorithm data structures are susceptible to soft errors, possibly resulting in SDC. Hence, it is critical to design algorithms, which are resilient to soft errors. Similarly, little to no research has been done in addressing the cost of data movement in machine learning algorithms. Novel algorithms for machine learning, which would provide bounds on accuracy loss, to address the data movement challenges are critical in tackling the primary impediments for exascale.

6.2.3 Specialized Hardware for Machine Learning—Neuromorphic Computing

The ending of Dennard scaling [35] is causing on-chip power densities to increase as transistor dimensions shrink. This is expected to lead to a phenomenon known as "dark silicon," [44] where different parts of a chip will need to be turned off in order limit temperatures and ensure data integrity. Thus, in future multi-core processors, all the processing cores may not available at time. It is expected that heterogeneous processing architectures with specialized processing cores will become more common to counter this problem. The specialized cores can process their specific classes of applications very efficiently, thus consuming much lower power. In order to warrant their use, specialized cores need to be reprogrammable to ensure a sufficiently large application base. At present, specialized processing is already in use with systems having GPUs in addition to CPUs.

Reprogrammable specialized neuromorphic computing hardware for ML would have significant applications in

exascale systems if they can significantly accelerate the algorithms and reduce the power consumed. The scope of applications would be particularly broad if *in situ* training can be carried out directly in the hardware. *In situ* analysis of system sensor data using conventional computing systems can be have a heavy overhead, thus this analysis is typically done either offline or with very simple online learning algorithms. Specialized hardware for machine learning would be able to aid in this analysis with little performance and power overheads. Other applications include finding patterns in large datasets and the other applications outlined in this report.

Research is needed in the design of specialized neuromorphic computing hardware and software for ML algorithms as this can have a transformative effect on exascale systems and applications. Processing systems that can learn *in situ* are especially important to examine. New classes of ML algorithms that are better suited to this hardware should also be investigated to make the best use of the specialized hardware. In particular, neuromorphic computing and biologically inspired algorithms that build on the collective behavior of a group of neurons can have strong information processing capabilities.

Several new classes of ML circuits promise significantly lower area and power overheads. In particular, memristor-based processing systems have been shown to reduce power consumption by over 100,000 times and chip area by over 1,000 times compared to existing mainstream computing systems [1]. Within a neuromorphic computing architecture, memristor devices can inherently mimic the behavior of biological synapses [120].

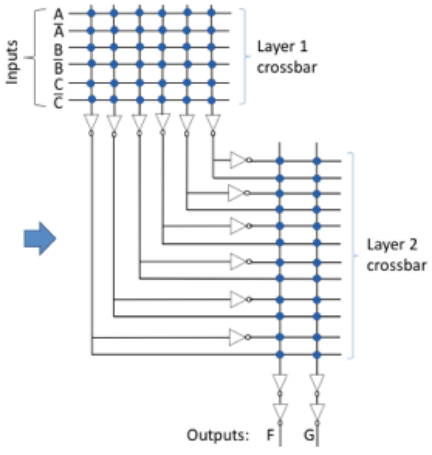


Figure 14: Two layer network for learning three-input, odd-parity function [1].

The basic neuromorphic computing architecture involves memristor devices arranged into programmable arrays (crossbars) to model a large set of neurons in a low area footprint (see Figure 14). In this type of circuit, a pair of memristors models a synapse (see Figure 15) based on the conductance of the memristors. Inputs to the synapse in a neuron are modulated by the memristor conductance resulting in current flows that are added to generate the neuron outputs. Thus, the memristors are not only storing data, but also computing a multiply-add operation in the analog domain. This leads to their significant low area and power consumption. One particular benefit of memristive systems is their capability for high-performance *in situ* learning using parallel, high-speed circuits [67]. The low-power, high-performance, and *in situ* learning capability of memristor-based neural network processing, or neuromorphic computing, systems make them highly attractive for extreme acceleration of ML algorithms.

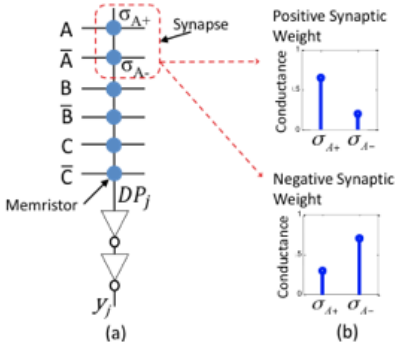


Figure 15: Circuit diagram for a single memristor-based neuron [1].

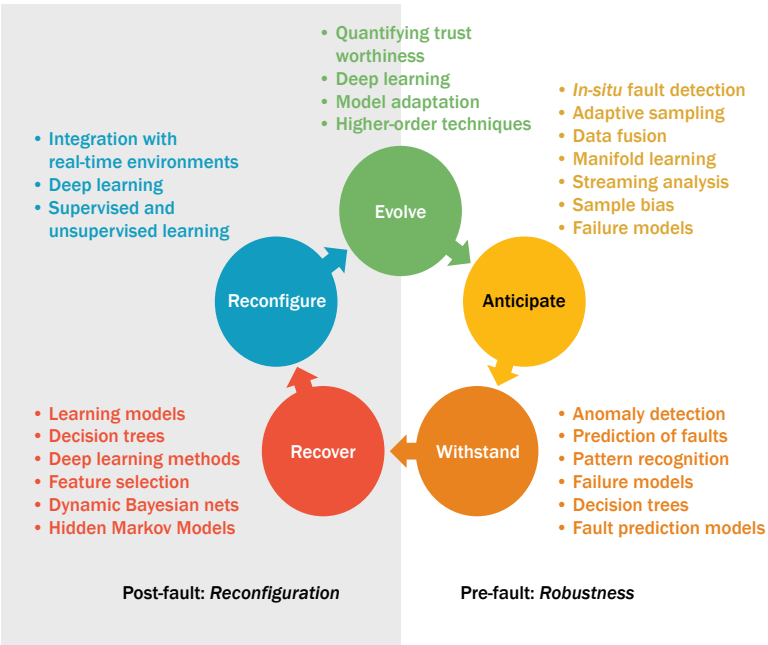
Several recent studies have proposed using neuromorphic processors to approximate applications at very low power consumption [118]. Chen *et al.* have examined the mapping of several applications from the PARSEC benchmark suite into neural form [27], while St. Amant *et al.* have shown that by mapping applications to neuromorphic form, energy savings of up to 30 times can be seen for general purpose applications [118].

6.3 Challenges at Exascale from Topic 3

First, hardware will see more frequent faults due to increased scale. With the stagnation of CPU clock rates, a system 1,000× more powerful than today's petascale systems will likely need 1,000× more components to deliver this increased performance [23]. This 1,000-fold increase in component count will likely lead to a 1,000-fold increase in the failure rate. This is compounded by the fact that shrinking transistor feature sizes and near-threshold voltage logic needed to address energy concerns may further increase the hardware failure rates.

Second, software errors at each level of the system will likely be more frequent due to increased complexity of the software stack. At the application level, dramatic increases in concurrency, emerging programming models, and increasingly complex workflows are likely to lead to increased errors. At the system and runtime levels, the diverse hardware technologies expected on future systems (heterogeneous architectures, deeper memory hierarchies, etc.), will demand richer and more complex sets of system services than observed on today's systems, further increasing failure potential.

Resilience is a crosscutting concern for exascale systems as these systems must be capable of predicting, detecting, informing, and isolating errors and failures at all levels of the system, from low-level hardware to application-level software. Each of these hardware and software levels are expected to exhibit higher fault rates than observed on current systems for reasons outlined below.



6.4 Challenges for ML towards Resilience and Trust from Topic 3

The goal of machine learning is to be able to use data or expert experience to create a computer model to solve a specific problem potentially as illustrated in Figure 16. Problems where machine learning has been successfully used include predicting future searches based on past searches, clustering documents by content, answering questions, automatically translating documents to other languages, recognizing faces from images, recommending products and services, and recognizing objects within videos, to name a few.

A type of problem that machine learning is ideally used for is to classify data into groups based on the features or known aspects of the data. Unsupervised learning relies solely on the data and not on prior information; supervised learning relies on training examples or previously known information. An example of unsupervised learning is document clustering, where a collection of documents is converted to term-weight vectors, then document similarity vectors, then a cluster showing the similarity relationship among the documents [110]. An example of supervised learning is recognizing handwritten letters. Training examples from handwritten letters are used to train machine learning classifiers to recognize specific letters of the alphabet. Once trained, these classification models can then be used to classify previously unseen data.

Traditionally, the major challenges in machine learning algorithms have been the lack of ability to scale to large datasets given the computational complexity of many of the algorithms, the need for impractically large numbers of training examples, and the need to retrain the model if the domain changes. These three areas are being addressed with highly efficient and parallel algorithms, semi-supervised methods that require a fraction of the

Figure 16: An overview of the different aspect of resilience for extreme-scale systems and the associated machine learning tasks.

number of training sets, and online learning methods that adapt to the data in near-real time. However, several challenges still remain, such as:

- **Sample bias**—where data sampling methods cause models to be skewed;
- **Anomalies detection**—misclassification of faults and anomalies;
- **Domain changes**—given the wide range of HPC applications, can a generalized model be developed to work across applications;
- **Adaptive sampling**—how to gather more information when it is needed;
- **Streaming analysis**—given the volume and speed of the data, how to predict outcomes without offline storage and analysis of the data;
- **Tight integration with real-time environments**—how to efficiently run models on HPC platforms to enhance resilience and trust, but without impacting applications; and
- **Quantify the trust worthiness of the HPC application**—what is an appropriate measure of trust for a scientific application and data? What are the dimensions of trust which will be incorporated into this measure (is this measure a vector of values rather than a single number)? What are the sources of information upon which to base our reasoning about the trustworthiness?

We believe that recent advances in scalable ML, when combined with the understanding and subject matter expertise from HPC operations and HPC event-log datasets, can enable the development of proactive failure management methods. Until recently, the application of machine learning algorithms on HPC event logs faced three major challenges in the iterative process of pattern discovery and pattern recognition to predict occurrence, coverage and extent of failures at the leadership computing facilities. The three challenges were:

1. **Data science**—do we need another supercomputer to analyze the terabytes of event-log data generated by leadership computing infrastructure? Are data analysis algorithms going to be fast enough to crunch terabytes for HPC operations personnel to be proactive;
2. **Science of trust and resilience in HPC computations**—the understanding of the event logs (What is collected and how useful is it for predicting failures and provenance?), and understanding applications with respect to hardware interactions, the inter-application dependencies, etc.; and
3. **Library of models**—that learn and infer from increasing data size, more data sources (logs, sensors, users, etc.), fewer examples of failures and interdependent relationships.

Machine learning algorithms need to be designed to operate online and at scale. There are two reasons why online learning is necessary: the configuration of executions in HPC systems is never the same and different applications are executed concurrently and the state of the different system software is evolving with time. So information needs to be acquired in a permanent regime. Since there is not enough storage space to store all events coming from all sources, filtering is necessary before storage. Outliers need to be detected from the flow of apparently normal events. Only a selected subset of all events will be stored.

As mentioned in Section 9, many sources of errors (we use this term in a generic way. It could be replaced by disruption, corruptions, etc.) can reduce the level of trust that we have in simulation results. Applied mathematics already offer techniques to mitigate errors coming from several sources. Others sources like bugs, malicious attacks, and silent data corruptions still need to be mitigated. Trust and resilience raise similar questions: what is the cost that a user is ready to pay for them (What is an acceptable overhead in terms of execution time and additional hardware resources, energy?). Trust also raises new questions: since trust is not necessarily binary, can we provide at the end of the execution a trust level (or confidence level) of the simulation results? Conversely, could users express a requirement in level of trust before the execution, in order to guide the execution toward this objective? This could be seen as the level of risk that a user is ready to accept. Another question concerns the coverage for trust: does trust need to cover all sources of errors or can we consider levels of trust with respect to each source of errors.

7 Current and Future Research Directions

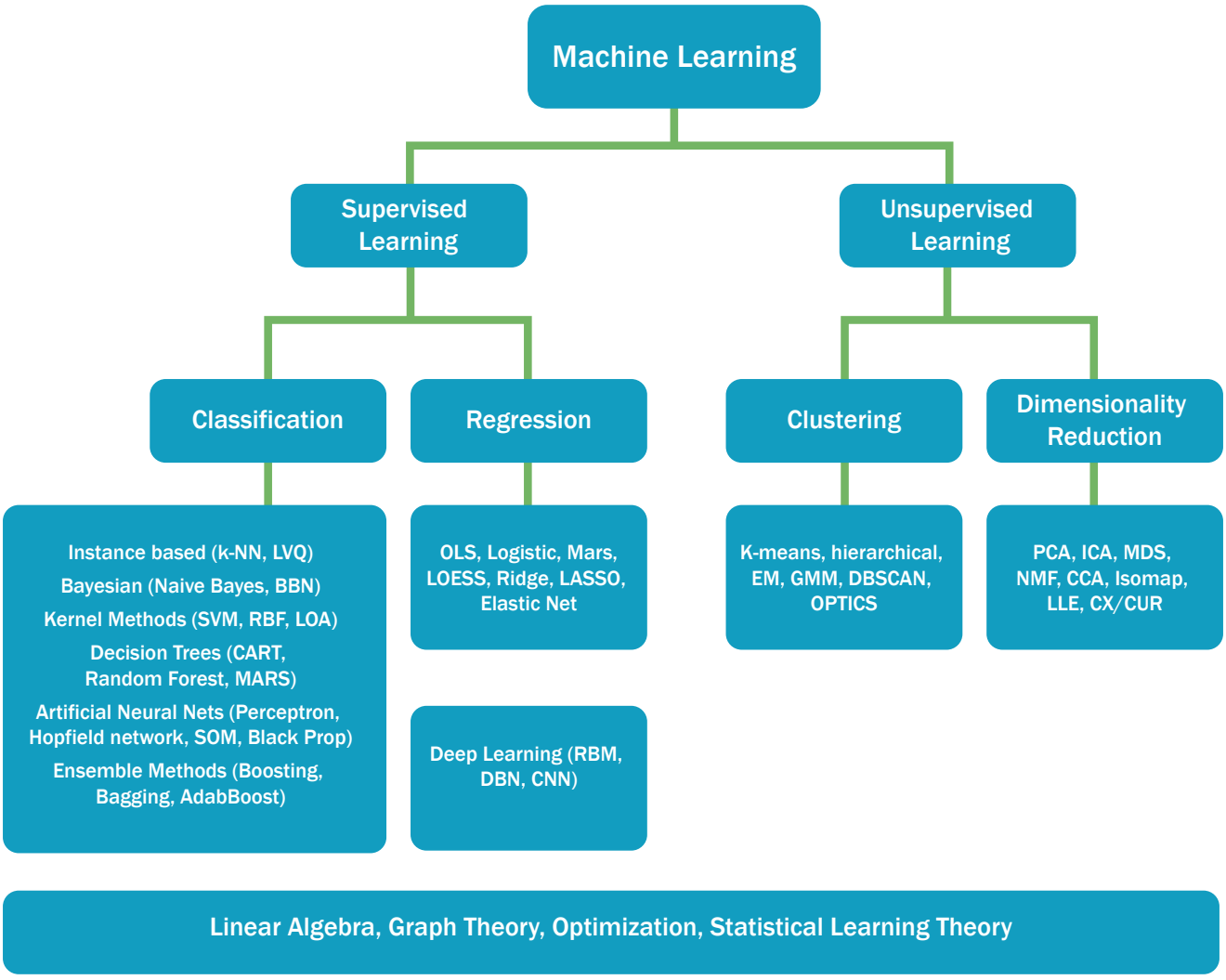
7.1 Research Directions in ML from Topic 2

In this section, we will identify the major research directions of the field of machine learning in industry and academia. Figure 17 provides the taxonomy for the traditional high-level ML areas and provides examples of the well-known techniques that fall into the various categories. Much research still lies in improving the generality and robustness of the techniques in these areas. However, we also would point out that some of the most interesting and applicable research directions in ML do not neatly fit into the traditional taxonomy. These research areas are noted in the box at the bottom of Figure 17.

Figure 17: Taxonomy of various ML approaches and methods.

Machine Learning-built Models to Understand the Past and to Predict the Future

For example, one would like to understand how the data was generated and to use that knowledge to make predictions. When labeled data is not available, the process of creating this understanding is often referred to as unsupervised learning. If multiple types of variables are available, one of which may be interpreted as providing labels, then the task of building a prediction model, which allows one to predict the future value of a target variable as a function of the other variables, is often referred to as supervised learning. There are many variations. For example, one may have only a small number of labels, one may receive labels iteratively, etc. These are often called semi-supervised learning or reinforcement learning. Important to the use of all of these is the famous observation from the late George Box that all models are wrong, but some are useful. While the model-building literature presents a vast array of approaches and spans many disciplines,



model building with massive data is relatively uncharted territory. For example, most complex models are computationally intensive and algorithms that work perfectly well with megabytes of data may become infeasible with terabytes or petabytes of data, regardless of the computational power that is available. Thus, for large-scale problems and in different types of applications such as scientific applications, one must re-think the trade-offs between complexity and computational efficiency. Many approaches are used to address many of these questions, but many challenges remain. There are two types of challenges here. The first is scaling up existing algorithms. In some cases, this means scaling up on architectures not optimized for ML. The second challenge is identifying objectives and algorithms that are more appropriate for particular applications. In this case, the goals of the user of machine learning might be very different if one is interested in high-quality quantitative prediction versus obtaining qualitative understanding of the data.

Unsupervised Learning

Unsupervised learning or data analysis aims to find patterns in the data. This can include the following:

- **Clustering**. This is partitioning data into groups so that data items within each group are similar to each other and items across different groups are not similar. For example, K-means, hierarchical clustering, and mixture models are popular algorithmic approaches.
- **Dimension reduction**. This represents high-dimensional data points by points in a lower-dimensional space so that some properties of the data can be preserved. For example, one approach might be to preserve enough information to fully reconstruct the data, and another may be to preserve only enough information to recover distances among data points.
- **Anomaly detection**. This is determining whether a data point is an outlier (e.g., is very different from other typical data points). One general approach is to use a statistical model to characterize the data, and an outlier is then an unlikely point.
- **Characterizing the data through basic statistics**, such as mean, variance, the frequency distribution of node degrees in a graph, etc. Although simple, a challenge here is to find computational algorithms that can efficiently work with massive data.
- **Testing whether a probability model of the data is consistent with the observed statistics,** e.g., whether the data can be generated from a Gaussian distribution, or whether a certain statistical model of a random graph will produce a graph with observed characteristics.

Existing approaches to address these questions include probabilistic modeling approaches, non-probabilistic approaches based on optimization, and procedures that try to find desired structures. For example, a mixture model can be used as a statistical model for addressing the clustering problem, while an optimization model does not. There are also clustering procedures that are not based on optimization or statistical models. For example, in hierarchical agglomerative clustering, one starts with each single data point as a cluster, and then iteratively groups the two closest clusters to form a larger cluster; this process is repeated until all data is grouped into a single cluster. In a loose sense, it also builds a useful model for the data that describes similarity relationship among observations; but the model is not detailed enough to generate the data in a probabilistic sense.

Supervised Learning

Supervised learning is sometimes called predictive modeling. In this case, one typically has a response or output variable Y, and the goal is to build a function $f(X)$ of the inputs X for predicting Y. Basic prediction problems involving simple outputs include classification (Y is a discrete categorical variable) and regression (Y is a real-valued variable). Statistical approaches to predictive modeling can be generally divided into either generative models or discriminative models. In a generative model, the joint probability of X and Y is modeled; that is, $P(X|Y)$. The predictive distribution $P(Y|X)$ is then obtained via Bayes' theorem. In a discriminative model, the conditional probability $P(X|Y)$ is directly modeled without assuming any specific probability model for X. An example of generative model for classification is linear discriminant analysis. Its discriminative model counterpart is linear logistic regression, which is also widely used in practice. The maximum likelihood estimation (MLE) is a common parameter estimation method in these cases, but one can define other criteria to optimize. For example, one may consider a geometric concept such as a margin and use it to define an optimization criterion for classification that measures how well classes are separated by the underlying classifier (which leads to support vector machines).

Another issue with high-dimensional data is that there are a large number of variables that are observed that are difficult to handle using traditional methods such as MLE. Regularization approaches are often used in these cases. Examples of such methods include ridge regression and the Lasso method for least-squares fitting. Often, nonlinear prediction methods can achieve better performance than linear methods

and an important research topic in massive data analysis is to investigate nonlinear prediction models that can perform efficiently in high dimensions. In many cases, this requires primitive scalable methods for least-squares regression and low-rank matrix approximation. Developing improved algorithms for these problems is a continuing challenge and in recent years the use of randomization as a resource has led to qualitatively improved algorithms for regression and low-rank matrix approximation problems in very large-scale settings.

An important challenge is to see how these types of approaches can fruitfully be combined with online prediction methods (which can be regarded both as modeling for sequential prediction and as optimization over massive data) such as stochastic gradient methods. Online algorithms do not require all data to be stored in memory, since each time they are invoked, they look at one or a small batch of observations. One popular approach is stochastic gradient descent.

7.2 Current and Future Research Directions from Topic 3

Over the last few years, resilience has become a major issue for HPC systems, especially, in the context of DOE's vision for exascale computing [22, 36, 42]. State-of-the-practice methods (i.e., checkpoint-restart [43], event-correlations [52], replication [47, 40] and failure prediction [57]) have become operationally infeasible or do not scale for millions of cores; accommodate heterogeneity in architectures (CPU, GPU, etc.); account for different failure modes (hardware, software, application, etc.); and, different hierarchies (input-output, memory, disk, network, etc.) of possible failures. The outstanding challenge is finding new proactive methodologies that will reduce the instability of exascale systems while allowing users' applications to run without interruption.

Major challenges in applying machine learning to resilience and trust in supercomputing are the multiple types of heterogeneities in the log data (i.e., including hardware faults, network-faults, soft errors, etc.). This requires machine learning algorithms to include:

- A suite of methods to handle different aspects of variations (as in multi-task learning, science domain-specific adaptation);
- View-based learning (as in multi-view learning—different codes at different times), instance variations (i.e., multi-instance learning—same code run at different times);
- Label variations (i.e., multi-label learning—scaling to increasing types of faults); and,
- Oracle discrepancies (as in labeling method—manual, sensed, automatic), etc.

While novel methodologies, applications, and theories for effectively leveraging these heterogeneities are being developed, the work is still in nascent stages.

There are multiple challenges:

1. How can we effectively exploit the label/example structure to improve the classification performance?;
2. How can we handle the class imbalance problem when facing one or more types of heterogeneities?;
3. How can we improve the effectiveness and efficiency of existing learning techniques for large-scale problems, especially when both the data dimensionality and the number of labels/examples (different types of failures) are large?;
4. How can we jointly model multiple types of heterogeneities to maximally improve the classification performance?; and
5. How do the underlying assumptions associated with multiple types of heterogeneities affect the learning methods?

7.2.1 General Failure Model

Most existing resilience solutions are application-specific and difficult to adapt into other applications. Although few application blind solutions are reported, since they need to consider unnecessarily large amounts of application elements with no failure models being known, they inevitably incur large overhead. Machine learning-based general failure models will mitigate both limitations; they are based on abstract signatures, not by features specific to an application, and thus are easy to apply to other applications and lightweight in footprint.

Machine learning-based failure models for resilience are broadly categorized into two areas: failure detection and failure prediction. Although the latter is a more rigorous and proactive form than the other, they are both based on data analytics. In general, log data such as syslog or resilience, availability, and serviceability (RAS) outputs and synthetically injected failure data are considered essential for this end. However, data analytics on this data imposes challenges. Data is voluminous, complex, and

heterogeneous (i.e., structured or unstructured, textual description or numeric readings).

In order to enable applications to make decisions based on the trustworthiness of resources, multiple hybrid learning techniques will be applied in an ensemble learning approach using a metareasoner to select from or combine solutions provided by multiple learning modules. The approach for learning and reasoning about trust in a large network of nodes uses an iterative, multi-step hybrid reasoning algorithm consisting of causal models represented as Bayes nets, being fed by information retrieved from network probes or logs. The models capture causality among variables that represent precursors to trust outcomes, and produce data for use by other machine learning methods. These methods include knowledge-based approaches, including case-based learning and reasoning, and statistical approaches. There are many machine learning approaches that both separately and in combination could be applied to provide insight, both quantitative and qualitative, into the trustworthiness of entities. These approaches will be investigated.

7.2.2 Fault Characterization

Characterize HPC faults using existing and extended ML and deep learning techniques to analyze fault data that is produced by HPC systems.

Building Resilient Scientific Applications via Generalized Machine Learning Models

Resilience—coping with runtime faults—has been identified as a top challenge to achieving exascale. As new generations of microprocessors are created, soft-error rates increase as a consequence of technology scaling and the need to reduce energy consumption. In addition, a significantly larger number of software/hardware components are expected at exascale, which will further increase failure frequencies. Applications must survive in unreliable environments, thus we need to design efficient resilience mechanisms.

Figure 18: Workflow to protect applications by learned models.

We propose a novel, automatic approach to protect scientific applications from faults by leveraging application-independent ML models. Most existing resilience solutions are algorithm-specific (and only suitable for a subset of applications) [20, 38, 74]. While others are algorithm-independent, they incur high overhead by over-protecting the application (since application failure models are unknown, they tend to protect unnecessary application elements or state) [17, 30, 29, 53]. By building general algorithm-independent failure models via ML, our solution is the first to automatically protect specific application components to reduce fault detection and recovery overhead. Developers will no longer spend time adapting resilience techniques to their algorithms or face unnecessary slowdown due to fault detection code.

Application Resilience Data

Fault injection into applications is the most common approach to study resilience properties and a huge amount of fault data can be obtained in this way. To have a sense of this data space, a single serial kernel can be injected with faults in different machine instructions on the order of 500 billion (depending on code size and runtime). With a more complex parallel code, this number can easily reach up to quadrillions (10^{15}) of fault data points that need to be analyzed just for a single application.

Building General Failure Models

By analyzing fault data, we can build a failure model for an application. A failure model explains how and when faults make different components of an application fail. Our research questions are: Can we use the failure model of an application to develop protection mechanisms for another application?

Further, can we use a small set of application kernels to build failure models that generalize to a larger set of applications? If this is possible, we would spend less time protecting new applications (by avoiding long fault injection campaigns) while maintaining efficiency.

We argue that it is possible to generate application-independent failure models by using hardware- and system-specific features rather than algorithm-specific features. Examples of algorithm-specific features include the number of particles in a simulation or the size of a matrix. System- and hardware-specific features might include the type of executed instructions or the number of bytes in a memory region.

The reasons behind our argument are twofold:

1. An algorithm-specific feature from application A may not exist in application B; and
2. Faults originate from the hardware and propagate to the system, thus hardware- and system-specific features are likely to capture failure characteristics in a general way.

The next section casts the problem of generating a failure model as an ML problem and presents a case study using our approach to efficiently protect applications against silent errors.

We can express the generation of a failure model as a pattern classification problem in supervised or unsupervised learning as follows (see Figure 18). Let S denote the state of an application or a system. S can take one state from a set, for example {faulty, non-faulty} or, if we want to predict future states, {will-fail,will-not-fail}. Let \vec{v}. $\vec{V}$ denote a vector of values for n collected metrics $[m_0,...,m_n]$. These metrics can be any features of the system or application. A pattern classification problem would be to induce or learn a classifier function F mapping the universe of possible values for $\vec{V}$ to the range of states S. Given a vector $\vec{V}$, we can then use this function to predict if the application is (or will be) in a faulty state or not.

Case Study

Silent errors are some of the most catastrophic errors because they may affect numerical results—applications show no fault symptoms, although their final output may be incorrect. Our goal is to detect silent errors and to make them visible, thus users can take remedial actions promptly. A common approach to detect silent errors is to duplicate computations and to compare them; if they differ, an error is detected. A widely accepted approach is the one proposed by Chang, *et al* [25], which duplicates computation at the granularity of instructions. However, this method incurs a high overhead because almost all instructions are duplicated in all applications. We can improve the efficiency of Chang's method using our model-generalization approach as follows. We train an ML classifier (decision-tree) where feature vectors comprise characteristics of an instruction, such as its type and the instructions it influences (all hardware- and system-level features). To train the classifier, we use labeled data from instruction-level fault injection in a molecular dynamics code (CoMD) using LLVM. For each instruction, we obtain a feature vector $\vec{V}$, and, by observing whether the application crashed or not, we label each vector with non-silent or silent. Out of 365 injections, 233 (63.8 percent) result in non-silent errors and 132 (36.2 percent) result in silent errors. If our model is general enough (see Figure 19), we hypothesize it can be used to protect new applications, or different versions of the same application, using a simple algorithm:

1. Check every instruction in the code of the new application;
2. Ask the classifier if an error in this instruction would cause a silent error; and if so,
3. Protect that instruction.

With a perfect classifier, our approach would incur a slowdown of only 1.36 (since we would protect only instructions in which a silent error can occur), whereas the traditional approach [25] of naively protecting all instructions would have a slowdown of at least 2. Thus, our approach would be $^2/_{1.36}$ = 1.47 × faster than [25].

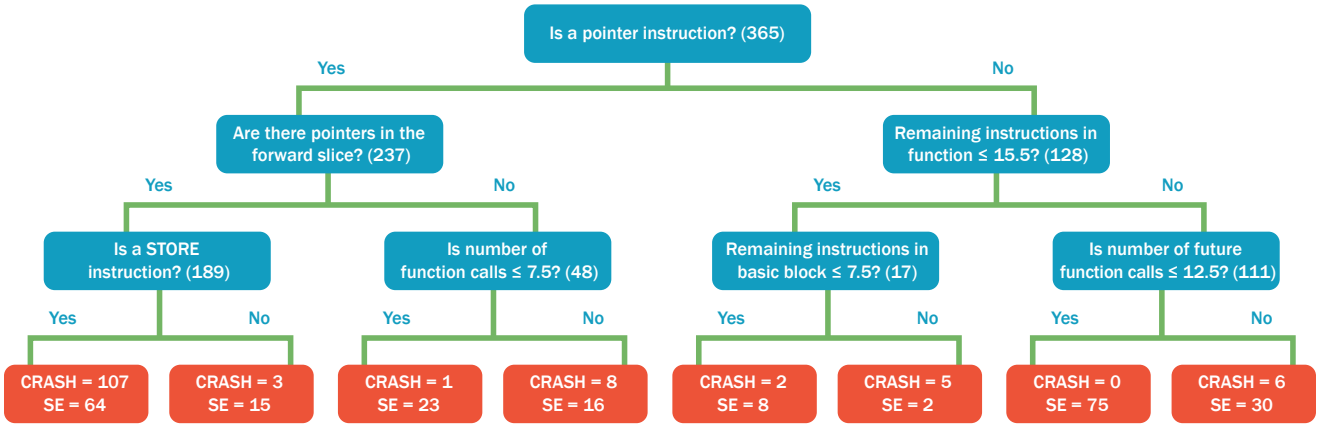


Figure 19: Decision tree of fault injection results in molecular dynamics code (SE = silent error, CRASH = application aborted)

Feature Selection

Hardware and systems provide a wide range of features to model a resilience problem. Selecting the right features (or filtering out those that do not provide much information) is critical. Key research questions are:

1. What features should be measured to model the type of faults that we want to detect? (some features may be fault specific); and
2. How to sample measurements effectively at runtime without incurring high overhead?

Model Adaptation

What if a model from application A is not general enough to be used in application B? Can model A be adapted (perhaps automatically) for B, and, if so, how? Can we form equivalence classes of similar failure behavior? What makes an application similar to another application in terms of resilience? Answering these valid research questions will require extensive experimentation, analyzing useful features, and developing practical algorithms to compare code structures and fault propagation properties.

Leveraging Proxy Applications

Proxy applications—used by DOE national labs in co-design efforts—are small, tractable codebases that represent some functionality of larger, more complex applications (their parents). Protecting a full application from faults requires spending a large amount of time in fault injections. In a large-scale application, it may be prohibitively expensive to collect sufficient training data. However, if a proxy application resembles its parent in terms of resilience, we can train models from that proxy application and can use it to protect its parent in less time.

We propose to improve the efficiency and applicability of resilience techniques using generalized failure models that can be built via ML. This work will spawn novel resilience solutions for exascale.

7.2.3 *In Situ* Fault Detection

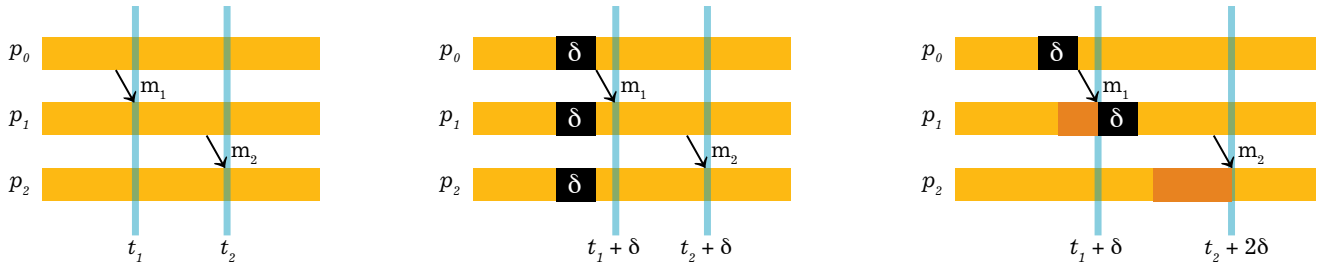
Use classifier models developed offline to detect faults and learn new patterns *in situ*.

Using Machine Learning to Optimize Uncoordinated Checkpointing Performance

In response to alarming projections of high failure rates due to increasing scale and complexity of HPC systems [19], many researchers have focused on methods and techniques for resilient extreme-scale HPC systems and applications. Considering non-algorithm-specific resilience approaches, researchers have studied both coordinated checkpoint/restart (cCR) and uncoordinated checkpoint/restart (uCR) protocols, with cCR having emerged as the *de facto* standard.

cCR protocols preempt all application processes to record a snapshot of the application's global state. cCR is attractive for several reasons. Its coordination protocol guarantees that the most recent global checkpoint captures a consistent global view, removing the need to store multiple checkpoints, sent messages, or other additional state information and thereby minimizing storage requirements. cCR also admits a relatively simple recovery procedure that does not suffer from rollback propagation, a scenario in which the most recent checkpoints from each application process do not comprise a consistent global state [41]. cCR does suffer from I/O contention issues since all processes checkpoint simultaneously, and with cCR protocols, upon a failure, even the surviving processes are perturbed as they must rollback to their most recent checkpoint. The rework executed by surviving processes also results in potentially unnecessary energy expenditures. uCR protocols, in which each process in an application makes independent decisions about when to checkpoint, can mitigate cCR's I/O contention problem since processes are not forced to take checkpoints simultaneously. Additionally, when uCR is coupled with message logging, when failures occur, surviving processes are not forced to rollback to their most recent checkpoint and therefore can run ahead in their execution—unless and until they depend on a message from a failed process.

Though uCR protocols show promise, recent results show that the communication of an application and its associated dependencies can significantly impact the performance of uCR [49]. This impact can be so great that, at certain scales, cCR to a shared parallel filesystem can outperform uCR to local non-volatile storage [49].



(a) Without checkpointing (b) Coordinated checkpointing (c) Uncoordinated checkpointing

Figure 20: Propagation of uncoordinated checkpointing delay through application communication dependencies. The processes p_0, p_1, and p_2 exchange two messages m_1 and m_2 in each of the three scenarios. The black regions denote coordinated (b) and uncoordinated (c) checkpoint delays marked with δ.

The possibility of uCR protocol activities inducing delays amongst processes, including processes that do not communicate directly with each other, is analogous to the manner in which operating system noise can affect HPC applications [48, 70]. Figure 20 illustrates this phenomenon. Figure 20 (a) shows a simple application running across three processes (p_0, p_1, and p_2). These three processes exchange two messages, m_1 and m_2. We assume here that these messages represent strict dependencies: any delay in the arrival of a message requires the recipient to stall until the message is received. Figure 20 (b) shows the impact of coordinated checkpoint/restart (cCR). Because all of the checkpointing activity is coordinated across processes, the relative progress of the processes is unperturbed and all of the dependencies are satisfied at the appropriate time. Figure 20 (c) illustrates the potential impact of relaxing the coordination requirement in uCR. If p_0 initiates a checkpoint at the instant before it would have otherwise sent m_1, then p_1 is forced to wait (the waiting period is shown in grey) until the message arrives. If p_1 subsequently initiates a checkpoint before sending m_2, then p_2 is forced to wait. Part of the time that p_2 spends waiting is due to a delay that was originated by p_0. The key point is that without coordination, checkpointing delays can propagate based on communication dependencies in the application.

Our position is that machine learning can be used to increase the performance of uCR by determining when local checkpoints can be taken such that they do not amplify the overheads as illustrated in Figure 20. Standard methods are incapable of effectively determining the proper time to take a checkpoint as a local cannot determine *a priori* if it is currently involved in a communication dependency chain. In addition, system- and platform-level algorithms can create dependencies with nodes the application does not directly communicate with.

Prompt Failure Detection through Integrating Disparate and Heterogeneous Offline and Online HPC Instrumentation Data

The Advanced Scientific Computing Research (ASCR) program and the HPC community have invested in a number of resilience technologies, including checkpoint/restart [66], containment domains [30], and resilient solvers [30, 68]. However, we still lack more fundamental capabilities that can assist proactive decisions in using these technologies: prompt detection of faults or abnormal status of the system. HPC systems are heavily instrumented for monitoring system health, producing voluminous data that can be used to disclose insight in this regard. However, no systematic approach to integrating diverse datasets exists today, which restricts primary use of the data to monitoring of mere occurrences of some known patterns in each data separately. The main challenge resides in discrepancy between different data sources. These data sources, collected at different locations and layers of the system, are inherently heterogeneous and disparate; some data includes numeric readings while other data includes textual content or both. Furthermore, despite their enormous practical importance, certain types of instrumentation data are never logged due to insurmountable overheads to collect and populate them into a designated database or repository. For example, real-time CPU status, memory utilization levels, or congestion status of the high speed interconnect between routers (or switches) are produced not only at a high-speed rate, but also at an embarrassingly large number of locations. Since compiling a full collection of these instrumentation data without inflicting the rest of the system is currently impossible, the data ia used either to capture a statistical synopsis or simply discarded. Therefore, while learning and correlating logged data sources is a large-scale offline data analytic problem by itself, incorporating such online data into the holistic learning framework opens up another set of algorithmic and architectural challenges.

For the offline analysis, in addition to heterogeneous nature of the data, the fact that very few failure cases are typically reported in logs should be considered. Since this constitutes a weakly supervised setting where data for the target concept is scarce but other related data ia abundant, relation extraction [10, 92] methods in conjunction with semi-supervised learning such as multi task learning [9] to disclose semantic associations among word terms from the textual logs are adequate to apply. Likewise, mappings between multi-dimensional spaces constructed from numerical instrumentation data can be learned using regression-based learning techniques, such as smoothing kernel-based regressors to learn non-linear relationships in the data or deep learning approaches. More importantly, mappings between groups of relations from textual and a set of features from numeric data should also be examined to establish comprehensive characterizations of a fault model. Such a model should include causal relations, occurrences of the identified relations and associated numerical readings over time. Statistical Markov models, such as Hidden Markov Models (HMMs) or Dynamic Bayesian Networks (DBNs), where different compositions of relations possibly constitute latent variables or different states, are good candidates to apply.

For the online analysis, a distributed online machine learning approach seems to be an excellent choice. In particular, we propose to create a distributed streamlined data cube that, at a conceptual level, follows a conventional Online Analytic Processing (OLAP) cube, i.e., (a) different instrumentation data readings constitute one set of dimensions, (b) locations or layers in a hierarchy another set of dimensions, and (c) time the third type of dimension. Each cell of the cube represents a reading measured from an instrumentation point at a particular timestamp. The challenge is then to apply analytic algorithms over the cube while keeping it at its minimal footprint. Whereas we can dynamically compress the time dimension by adopting a tilted time frame [60] that produces finer resolutions for recent data and coarser resolutions for distant data, reduction of the cube in the other dimensions without much compromising analytic quality is a bigger challenge. More specifically, the cube should be abstracted in a hierarchy where information minimally necessary for the analytic algorithms are represented as layers that aggregate data from lower layers which do not need to be stored. Since the cube will be inherently distributed over the underlying network, it is important to construct a topology-aware hierarchy. This will pose an interesting optimization problem with respect to an allowable error bound and different network topologies. Even preliminary results on this research will provide valuable feedback to vendors for their next data instrumentation design. In parallel, a set of distributed, streamlined algorithms that together construct failure models from the cube should be developed.

Online Data-driven Statistical Inference Tools for Fingerprinting and Improving Resilience of High-Performance Computing Application

HPC applications are undergoing a dramatic shift in their overall design and architecture as a consequence of the availability of heterogeneous computing resources. Instead of having to scale HPC applications exclusively based on CPUs, modern compute architectures are imposing on developers to scale their applications across CPUs, graphics processing units (GPUs) and other emerging architectures such as co-processors. The combination of heterogeneous resources within emerging exascale architectures promises to provide unprecedented compute capabilities for scientific applications. However, one of the pitfalls of heterogeneous architectures is that it is difficult to predict how and when different components will fail. Additionally, future exascale applications will have to make efficient use of inter- and intra-node parallelism (i.e., concurrency), memory footprint, data footprint and locality, as well as be reliable to node-level or component-level failures. HPC applications typically generate vast amounts of data about the processes they run on supercomputing systems in the form of "event logs." As heterogeneous compute resources become more prevalent, these event logs have become more complex and represent examples of big data, consisting of both structured and unstructured data. Apart from recording the status of current progress of jobs/processes, these event logs record rich information about the status of hardware and software, data, memory and network resource usage. This information represents multi-modal spatial and temporal data that has immediate value in not only profiling applications but also in learning to predict where applications may fail when resource failures occur. We propose a scalable framework that uses event logs and other performance information collected from other sources (e.g., file systems logs, performance management logs, network logs, etc.) to perform near-real time analysis of HPC applications to develop novel statistical fingerprinting of resource usage, predict failure modes, and provide novel capabilities to recover from hardware/application failures. The framework utilizes a novel multi-modal representation of event logs where by these datasets are modeled as three-dimensional tensors [79]. For example, tracking network logs can provide information on the time point (t), the source

(s) and destination (d) ports. The entry within each (t) can indicate how many packets were routed on these (s) to (d) ports. Similar tensor representations can be built for other types of event logs that capture the state of the application/HPC resource. However, these tensors can be represented as independent streams of data that need to be analyzed in tandem to fingerprint applications or resources.

Our approach to fingerprint HPC resources and applications is based on dynamic and streaming tensor analysis algorithms [122, 14, 107, 109] that specifically capture time-evolving patterns of the aforementioned tensors. Using extensions to target multiple data streams [99] simultaneously, we will develop linear and hybrid ML models [28,108] that can provide quantitative insights into the resource utilization profiles of HPC resources. Further tensor analysis models can track application resource use profiles across multiple data streams to identify anomalous behaviors and provide insights into failure modes of these applications. The tensor analysis models can be summarized using higher-order techniques to obtain generative profiles of how applications use HPC resources. These quantitative insights can be integrated with existing integrated performance management tools to provide a seamless end-user experience that can:

1. Aid system administrators to prioritize resource utilization for specific applications;
2. Inform end-users about potential failures based on past/previous runs; and
3. Aid optimization of applications for HPC deployment.

The failure modes detected from application runs (using dynamic and streaming tensor analysis algorithms) will also aid in improving resilience of HPC applications. In particular, based on temporal profiles generated from multiple data streams, we build resource utilization maps that forecast how applications perform with only CPUs, or with CPU-GPU hybrid environments [8]. The utilization profiles can then be tailored to specific application contexts and made available to both end-users and system administrators to better aid recovery from failures. We will discuss our framework in the context of our experience in running long time scale molecular dynamics (MD) simulations. As MD simulations take up nearly 35 percent of supercomputing time allocations, the ability to address resiliency and trust issues from application specific contexts can be critical for their success at exascale.

7.2.4 Fault Prediction

Predict faults before they actually occur so that the system can be repaired or faults avoided when running an application.

Machine Learning for Failure Prediction

Understanding the behavior of failures in HPC systems is very important in order to address their reliability problems. Event logs are often a source of information for analyzing the cause of failures in cluster systems. However, the size of these files has continued to increase. Current systems can generate up to tens of GB of data per day, making any manual analysis unrealistic. Research in this field has turned to data mining and ML techniques to characterize events generated by current HPC systems. Machine learning is a powerful tool that efficiently extracts patterns in high-dimensionality sets and can provide accurate correlations between defined behaviors. We combined ML with signal processing techniques that are best suited to characterize the behavior of events affecting the system, highlighting the differences between failures [54, 55, 56]. Different system components exhibit different types of syndromes, both during normal operation and as they approach failure. In general, errors are often predicted by changes in the frequency or regularity of various events. Moreover, ML techniques become much more efficient when applied to the derived markers rather than to the original signal. Specifically, we adapted the sequential GRITE algorithm to work with our signals. By merging it with a fast signal analysis module we were able to guide the extraction process toward the final result, thereby reducing the complexity of the original data-mining algorithm. The correlations extracted with the ML algorithm are used online to predict future occurrences of failures. This step uses a novel methodology for online failure prediction based on a pattern-extraction algorithm specifically designed for streams of data with multiple dimensions. The method updates the correlations live as new events are generated in the system, in parallel with triggering predictions. A modified PCA algorithm extracts the most specific multidimensional items, then an alternative online PrefixSpan algorithm mines sequences of events. We have made multiple experiments on different past and current production HPC systems, from BlueGene systems, to NCSA's Mercury and Blue Waters. For BlueGene/L the results show that we can accurately predict almost 50 percent of total failures with high precision (90 percent). The Blue Waters has one order of magnitude more events generated that contain complex correlations. However, we showed that when

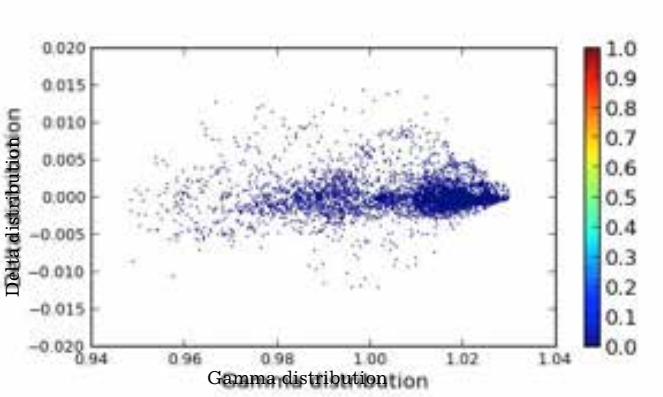
analyzing only a specific type of failures, the predictor can obtain over 60 percent recall and a precision of over 85 percent. These results are very promising. There are multiple directions to extend this research area. First, the prediction methodology can be easily extended to any other data gathered from the system, such as performance metrics and information about the application behavior. Second, instead of generalizing the existing predictors in order to capture all failure types and their varying behaviors, this direction focuses on developing specific predictors for each unpredicted failure type, starting with file system failures. Finally these techniques could be extended for performance degradation prediction.

7.2.5 Trusted Results

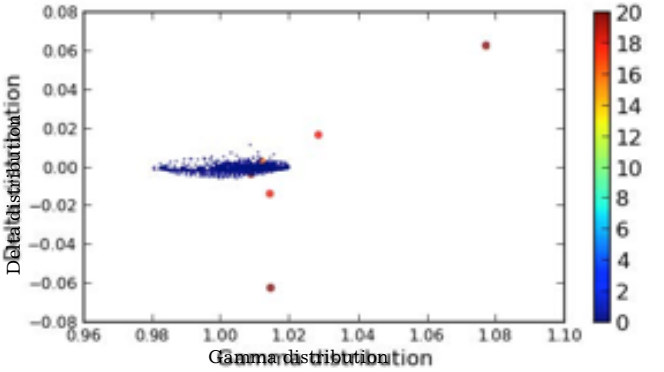
The first three areas above focus on creating an HPC environment that is resilient to failure, which enables a scientist to have trust in an applications results.

Machine Learning for SDC Detection

Machine learning provides a promising, effective, and efficient direction to detect SDC during runtime in scientific applications [62]. A strategy based on data analytics has many advantages over classic detection techniques. In particular, a lightweight data-monitoring technique can impose a low overhead on the application compared to expensive techniques such as replication. Also, data monitoring and outlier detection can be offered by the runtime transparently to the user. The principle is to analyze the application datasets and its evolution in order to detect outliers. Here, we take the example of a computational fluid dynamics (CFD) application simulating a turbulent flow and we analyze the velocity fields. In Figure 21(a) below, we show a multi-dimensional clustering of velocity field variable gradients. The same cluster in Figure 21(b) is plotted in the presence of data corruption, where the color denotes the magnitude (bit position) of the corruption.



(a) Fault free



(b) Data Corruption

Figure 21: A multi-dimensional clustering of velocity field variable gradients with (a) and without (b) data corruption.

From the results shown in the figure, we can observe how the corrupted data immediately stands out as clear outliers within the cluster. A point that should be taken into account while measuring the detection capabilities of these detectors is that not all the bits in the IEEE floating-point representation need to be protected. For instance, a corruption in the most significant bits is likely to generate a numerical instability, inducing the application to crash. Such soft errors might be silent to the hardware but not to the application. On the other hand, corruption happening in the least significant bits of the mantissa might be negligible because they produce deviations that are lower than the allowed error of the application. Coming back to the CFD example, if we neglected the four most significant bits (numerical instability) and the four least significant bits (negligible error), we could say that data analytics based detectors, provide a coverage of 83 percent in this example.

Using data analytics solutions for SDC detection opens a whole new field of research. The community is just at the beginning of this exploration. The first results are extremely encouraging in terms of detection performance and overhead on execution time. Advanced ML techniques combining multiple data analytics approaches need to be investigated to achieve a lightweight, transparent, accurate, and very portable SDC detection techniques.

Expanding on this, the trustworthiness of this type of output is not limited to exascale computing.

7.2.6 Representation of Trust: Supporting Resilient Decision-Making in the Context of Large-Scale Sensor Networks

Researchers are exploring the characterization, roles, and representation of trust in support of resilient decision-making in the context of a widely connected network of nodes such as the Internet of Things, networks of sensors, or data and sensor fusion in intelligence, surveillance, and reconnaissance applications. In this abstract we will explore concepts and approaches to develop the trust framework needed to support decision-making in a large-scale network of sensors, such as a smart power grid.

Our definition of trust should be framed in the context of the set of decisions that will be enabled by reasoning about the level of trust that the decision-making module has in individual nodes, sets of related nodes or types of nodes, providing information to the system or using or supplying services. The representation and reasoning approaches must be scalable, and this scalability is enabled by a distributed collaborative agent-based approach to machine learning and decision-making to evaluate the trust of nodes and to use that information for intelligent decisions.

A smart power grid will reason about resource management, power usage, resource needs and costs, while monitoring and controlling resources. Being able to reason about the level of trust in given nodes is important because decisions depend upon an accurate situation awareness (based on the knowledge aggregated and inferred from the sensor data) and on the reliability of appropriate resources assigned for services. The cognitive reasoning modules providing decisions will be able to query or probe nodes for features related to assessing a level of trust in the node. This probing can be used to increase confidence in the trust level being calculated by the system.

The features upon which the trust level of a node is based include:

- Uncertainty levels of measurements associated with the information coming from the node;
- Provenance—where does the information about this node come from?;
- Perishability—how current is the information that we have about this node and how quickly does the information change?; and
- Past performance and reliability.

Many of the features that we will use to represent the trust of a node will be predefined for us from the data existing in data logs. We will also have an opportunity to derive features from combinations of features. In addition, active learning techniques allow us to probe or query for pieces of information that would help to characterize the trust level of a node. These probes will be guided by a value-of-information (VOI) calculation produced by a Bayesian network used to model the causal relationships among trust-related variables in the network of nodes. A representation that characterizes different aspects of behaviors and behavior patterns for each node or node type will need to be developed so that behavioral features can be extracted and used as attributes in data analysis.

Nodes will inherit trust features based on their node type or characteristics enabling a calculation of trust for new nodes as they become part of the network. A confidence level will be associated with the trust value assigned to a node and a level of risk will be assigned to any decision involving a node. The level of risk is based on the consequences or cost of misjudging the trust level of a node. Trust is dynamic and dependent on the decision which will be based upon that trust.

We propose to use multiple hybrid ML techniques in an ensemble learning approach based on an approach used by the DARPA Integrated Learning Project (GTRI provided a case-based learning module) which used a metareasoner to select from or combine solutions provided by multiple learning modules. The approach for learning and reasoning about trust in a large network of nodes uses an iterative multi-step hybrid reasoning algorithm consisting of causal models of observable domain variables and their relationships to domain activities and outcomes, network structures, performance patterns and other domain concepts represented as Bayes nets being fed by information retrieved from network probes. The models capture causality among variables that enable precursors to trust outcomes. The information feeding the models comes from the massive data retrieved and analyzed

by the domain clustering and classification using multiple machine learning techniques that perform continually on streaming inputs and will be analyzed and exploited. These machine learning methods utilize dimension reduction, classification, and clustering as underlying methods for data analytics of high-dimensional data and will include both knowledge-based (case-based learning and reasoning) and statistical approaches. There are many machine learning approaches that both separately and in combination might provide insight (both quantitative and qualitative) into the trust indicators of individual nodes and of collections of nodes of a particular type or which share a common set of characteristics.

A similarity metric must be defined to represent the distance between any two instances in the trust feature space. The development of a similarity metric will include an analysis of the distance between sets of given values features with non-numeric values. There are a number of similarity approaches. The most common are based on variations of Euclidean distance (the square root of the sum of the squares of the differences in feature values). A similarity metric allows for the features to be weighted to represent their importance to the particular characteristic that we are interested in. These feature weightings can be learned (by regression analysis for example, or learned incrementally using the least mean squares algorithm) or weights can be given by a subject matter expert. A confidence calculation will be associated with each trust calculation to enable the decision-making algorithms to assess risk associated with trusting a particular node. The framework described here would support an evaluation of trust levels of nodes in a large network for the purpose of supporting intelligent resilient decision-making.

7.2.7 Extensions to Trust

Given the need to examine scientific output using machine learning methods, the natural extension would be to help the domain scientist understand more than just the trustworthiness of the results.

Application of Modern Machine Learning Methods for Self-Aware and Resilient Leadership High-Performance Computing

Over the last few years, resilience has become a major issue for HPC systems—especially in the context of DOE's vision for exascale computing [22, 36, 42]. State-of-the-practice methods such as checkpoint-restarting [43], event correlation [43], replication [40, 47] and failure prediction [57] become operationally infeasible or do not scale for millions of cores, accommodate heterogeneity in architectures (CPU, GPU, etc.) or account for different modes (hardware, software, application, etc.) and different hierarchies (input-output, memory, disk, network, etc.) of possible failures. The outstanding challenge is that of finding new proactive methodologies that will reduce the instability of exascale systems while allowing application runs of scientific user interest without interruption.

We posit that recent advances in scalable machine learning, when combined with the understanding and subject matter expertise from HPC operations and HPC event-log datasets, can enable the development of proactive failure management methods. Until recently, the application of machine learning algorithms on HPC event logs faced three major challenges in the iterative process of pattern discovery and pattern recognition to predict occurrence, coverage, and extent of failures at the leadership computing facilities. The three challenges were:

1. **The data science**—do we need another supercomputer to analyze the terabytes of event-log data generated by leadership computing infrastructure? Are data analysis algorithms going to be fast enough to crunch terabytes for HPC operations personnel to be proactive?;
2. **The science of data**—the understanding of the event logs (what is collected and how useful is it for predicting failures), understanding applications with respect to hardware interactions, the inter-application dependencies, etc.; and
3. **The scalable predictive functions**—the ability to construct, learn, infer from increasing data size, more data sources (logs, sensors, users, etc.), fewer examples of failures, and inter-dependent relationships hidden as hierarchies and cascades of events.

Today, based on experiments and benchmarks, we have learned that machine learning algorithms can be deployed at scale on terabytes of streaming data for real-time predictive analytics. One such benchmark study to understand algorithms and their dependencies to different scalable compute architectures (shared memory, shared storage, and shared nothing, etc.) was conducted at ORNL. The study showed that it is indeed possible to analyze event-log data (sizes as big as two years of archived data) fast enough to be proactive both with pattern discovery and recognition.

With the goal of addressing the science-of-data challenge and the design of predictive functions at scale, ORNL has invested in a scalable predictive analytics toolbox consisting of the following:

1. Deep learning algorithms [76] that automate the feature engineering process by learning to create and sift through data-driven feature representations;
2. Learning algorithms in associative-memory architectures [121] that can seamlessly adapt and include future data samples and data sources;
3. Faceted learning that can learn hierarchical structural relationships in the data; and
4. Multi-task learning frameworks that can learn several inter-related predictive functions in parallel [123].

We will showcase the opportunity to unleash these scalable machine learning tools and present the vision as a roadmap that includes:

1. Identification of data sources relevant to HPC resiliency;
2. Collection, integration, and staging of a variety of data sources (spatial arrangement of components, sensor information, event logs, etc.);
3. Automated and manual sifting of potential predictive indicators; and
4. Building, deploying, and validating those predictive models at scale for fault diagnosis and maintenance.

ML Techniques for Resiliency, Trust and Efficient Operating Systems

Consider large workflows with interrelated tasks that are common to numerous scientific computing applications of interest to ASCR. A key challenge of executing such workflows on extreme-scale systems with complex interconnection networks is mapping and scheduling the tasks to compute nodes in the most optimal manner possible. Attempting traditional optimization techniques on an extreme-scale system with 10^5 to 10^7 compute nodes would be computationally infeasible. Further, mapping may need to be done without explicitly building task graphs, and within certain time bounds on computation.

To address these challenges, novel methods need to be developed. Machine learning techniques based on dimensionality reduction and manifold learning are good candidates. For example, task graphs can potentially have complex interdependency structures. However, if there exists a low-dimensional manifold in this complex structure, then identifying such a structure will enable efficient mapping on the underlying system to minimize data movement. For example, two tasks (nodes in the task graph) can be further apart in the task graph, but can be closely interrelated and might be closer in a low-dimensional manifold of the task graph (illustrated in Figure 22). Thus, mapping them on two compute nodes closer on the network would minimize data movement. Further, identifying low-dimensional manifolds of communication networks is also important. The problem can then be formulated as alignment of two networks—task graph to computer network—to minimize data movement, and consequently energy consumption, costs [77].

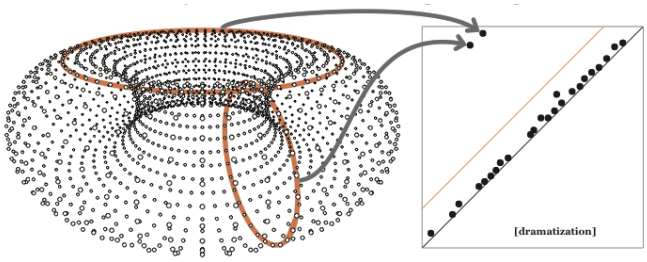


Figure 22: An illustration for manifold learning. Source: https://sites.google.com/site/nips2012topology/

There are several approaches to manifold learning, however they pose additional challenges at extreme scale. Graph-based techniques play an important role, but are not necessarily amenable to implementation on large-scale distributed architectures. Construction of k-nearest-neighbor graphs involve compute intensive $O(N^2)$ similarity comparisons for N elements and could result in imbalanced graphs. Further, (semi-) supervised learning algorithms on these graphs that are based on graph techniques such as random walks or eigenpair computation of the graph Laplacian are not ideal when applied to extreme-scale systems [63]. Potential research directions include development of scalable graph-based algorithms, randomized matrix algorithms that allow tradeoff between accuracy and performance, and novel distributed algorithms for computing eigenpairs of the graph Laplacian. Scalable techniques in the computation of k-nearest-neighbor graphs using approximate b-matching is also a potential research area with large impact on the quality of solutions [24, 77].

Identifying frequent patterns in extreme-scale computing can appear in several forms. For example, frequent task patterns (or kernels) can be highly optimized in the form of low-level implementations (assembly or hardware). Further, identification of code patterns can be used in auto-tuning of large-scale applications, especially running on heterogeneous architectures envisioned for extreme-scale systems.

Using machine learning techniques to identify frequent patterns in hardware or software failures will lead to improvements in the resilient design or configuration of a system. Given that domain experts can annotate part of the data, efficient supervised learning techniques would be especially informative. Existing ASCR HPC systems can be instrumented and studied to identify common failure points which can be used as design criteria for building trustworthy and resilient components in exascale systems. As stated earlier, several graph-based techniques can be employed, but they will need re-evaluation and re-implementation for extreme-scale architectures. In some cases, it is not always possible to introduce parallelism without a direct impact on the convergence properties of the algorithm. For example, it has been shown that the amount of parallelism is limited by the structure of the input expressed via its algebraic connectivity [114]. In this case, alternative scalable approaches need to be explored as well.

The cost and benefits of moving the data and processing modules around for purposes of computational locality and system resilience remain largely unexplored.

Classification and Optimization of Processing and Data Layouts for Resilience

The next generation of HPC infrastructures will evolve from being scientific data creation instruments to data-creating-and-analyzing instruments. This will lead to new system reliability profiles and constraints that applications and operating systems will need to operate within. A well-known recent example of a changing application usage and fault profile influencing the system software stack is the Google File System [59]—the use of extremely large-scale arrays of file storage devices and the accompanying proportional failure increase gave rise to the inherently redundant file-system design. Understanding failure characteristics and the predictive responses to fault events is a first discovery step. Analyzing system logs and failure characteristics with analytic techniques is a natural approach [39, 56, 97, 130] with several analytic methods discussed in the literature [128]. Here we discuss and explore the learning required to enable applications to dynamically employ combinations of redundancy-based techniques and algorithmic techniques to achieve trustworthy computing. While methods such as triple modular redundancy may be cost- and resource-prohibitive in general, the nature of the new data generating and analyzing paradigm can allow redundancy to be increasingly a complementary tool in reliability. The cost and benefits of moving the data and processing modules around for purposes of computational locality and system resilience remain largely unexplored.

The algorithmic approach for failure tolerance in combination with data and software replication can ensure resilience and trust in the results by using *a priori* execution plans together with dynamic run-time system performance predictions. These predictions would fuse information from machine hardware sensors and log records in order to predict both hard failures, as well as soft failures (e.g., resource bottlenecks) that may lead to irrecoverable failures or severe degrading of the system requiring different forms of rollbacks. The learning challenge we propose is to dynamically classify and compare the access and execution patterns with online, existing (or known) traces, and optimize their fit with the dynamically developed execution plans for the collection of active processes. The run-time optimization of the cost-benefits of replication will include a combination of access patterns prediction and classification on the fly, and co-scheduling the data and compute layouts.

8 Interdependencies with Other Efforts

8.1 Interdependencies with Other Efforts from Topic 2

As identified in Topic 1, machine learning can potentially enhance analytical methods with data-driven methods that learn in the operating environment to overcome noise while avoiding unnecessary detail in the performance, scheduling, and I/O models. Below we roughly map some of the specific exascale challenges that were called out to the challenges and advances in ML that are identified in this document.

Making More Effective Use of System Logs

System data is highly specialized, semi-structured and often nuanced by proprietary factors outside of the system owner's control. Even system experts often have a hard time determining why and where issues arise. Areas of machine learning that appear particularly relevant to this problem include:

- **Anomaly and change detection**—Recent ML advances have put these traditionally hard-to-define problems on firm mathematical footings and these advances could be used to identify low probability patterns in log data that may otherwise go unnoticed.
- **Interactive analysis tools**—While automated event detection and diagnosis for automated tuning is the eventual goal, machine learning may be able to provide productivity gains in the shorter term by simply making the system engineer's job easier through a combination of visualization and interactive anomaly and change detection tools.
- **Statistical relational learning**—ML methods that use relational representations may be able help anomaly and change detection methods better handle the semi-structured and highly specialized nature of log data.

ML for Control and Decision Problems

Topic 1 identified a number of scheduling and control problems that need to be automated, e.g., memory staging decisions, resource allocation and scheduling. Many of the challenging problems in reinforcement learning, decision making and control are being addressed by finding mappings (or reductions) to simpler problems (such as classification) where mature ML theory and methods can be applied.

9 Interdependencies with Other Efforts from Topic 3

The level of trust that a scientist has in simulation results depends on many potential sources of errors, vulnerabilities, and techniques to detect, potentially quantify, and mitigate these errors. On one hand, there are many sources of errors:

1. The mathematical model representing the physics;
2. The discretization, stochasticity, truncation, and iterative process involved in numerical methods;
3. Limited precision operators;
4. Bugs at any level between the application and the hardware;
5. Silent data corruptions from external uncoordinated causes;
6. Malicious potentially coordinated attacks; and
7. Input data.

On the other hand, there are domains, particularly in applied mathematics, developing research to mitigate some these errors: validation and verification, uncertainty quantification, and data assimilation. However, it is clear that existing efforts do not cover all sources of errors and this opens up wide research opportunities. In particular, we believe that machine learning has a role to play as a complement to existing applied mathematics approaches.

A machine learning resilience and trust program will need to work closely with efforts in exascale resilience, operating system resilience, deep learning, and scientific data analysis.

10 Common Themes, Findings, and Recommendations

10.1 Common Themes, Findings and Recommendations from Topic 2

- Topic 1 (Smart OS/R) and Topic 3 (Resilience and Trust) are important capabilities that will be critical to the execution of O(Billion)-way concurrent tasks on the exascale platform. Human control and intervention at that scale are impossible; we will need smart machine learning methods to enable a highly productive exascale environment.
- Scientific discovery in the DOE complex will be advanced by exascale simulations, as well as a range of scientific instruments. The scientist is increasingly becoming the bottleneck in the scientific discovery process. We believe that machine learning will be a central technology that will enable the scientist to overcome the big data deluge and further scientific progress.
- DOE needs to invest in a robust machine learning program that will make key advances in developing new capabilities, creating a production infrastructure for big data analytics and tackling complex modern scientific datasets. This program will enable dynamic, intelligent runtime systems to further productivity on exascale systems.

10.2 Common Themes, Findings, and Recommendations from Topic 3

The move to exascale computing will dramatically increase the number of components in the HPC system. The increase in computational power will enable new scientific discovery, but will also lead to shorter application mean time between failures as the number of hardware components increase. There will also be an unprecedented volume of system data available from these computers. In the past this data has been manually analyzed to improve the overall operation of the computer, which given the volume and complexity of exascale data, will no longer be feasible.

This data includes information about the facility where the computer is located, the various hardware components, the software systems, and the scientific applications. Data about the application development, the application users, and the data sources, to name a few, can contain highly relevant information to the operations of an exascale computer. The challenge comes in deriving valuable information from this massive collection of disparate data. Experts clearly understand the first-order performance effect on these computers, but this understanding does not account for all or even most of the faults observed. This data will be at a volume and complexity that makes it impossible to manually analyze—there needs to be a new way to analyze this data.

We believe that machine learning methods can help uncover second- and third-order effects, as well as previously unknown effects within this data. Not only does this provide a much greater insight into the operations of an exascale computer, but it also provides a fast and automated way to assess the reliability of the system and the trustworthiness of the scientific application results.

We believe these discoveries can lead to:

- Scientific applications that very rarely fail or perform poorly due to HPC system failures; and
- Scientific applications on HPC systems will have a quantifiable trustworthiness which will dramatically improve the scientific integrity of exascale computing.

There is clear evidence that such an approach will be successful. Machine learning techniques including clustering, feature extraction, and correlation analysis were applied successfully for failure prediction from system logs is some specific cases, like for the BlueGene/L systems and several of the LANL systems. However, failure prediction on recent systems like Blue Waters is more challenging due to the dramatic increases in both the system log size and system event categories. Blue Waters has about two orders of magnitude more event categories than systems where failure prediction was obtaining good results. These experiences suggest that more research is needed to apply machine learning techniques for failure prediction in current petascale systems and for even more complex future exascale systems.

These results point to the need for new and stronger methods for analyzing large, diverse, and complex data. Among the challenges are how to fuse, represent, and analyze environmental, hardware, software, and application data arriving at very high volumes, in different time scales, and often text-based. Various machine learning methods can be used to characterizing faults within this data, but new methods are needed to deal with the limited amount of labeled fault data.

Another valuable aspect of machine learning is the ability to work *in situ* on data streams so that data does not need to be stored and analyzed offline. This requires new methods to perform near-real time classification of potential faults, their locations, and systems affected with minimal impact to the application.

The ultimate goal is to be able to accurately predict and anticipate faults based on machine learning methods. This requires assessing faults based on time and location, and determining precursors for the faults. This will require a new understanding and machine learning experimentation on finding fault precursors, adaptive sampling, and innovative compression algorithms.

Exascale computing has the promise of spurring new scientific discovery, but runs the risk of jeopardizing scientific integrity if the mean time between failures is not maximized. Machine learning provides a way of understanding the system data in new deeper ways, and can provide dramatic improvements in resilience and the trust scientists put into application results. Therefore, we recommend that ASCR establish a machine learning program to ensure the resiliency of exascale systems and trust in the generated output, which ultimately determines the scientific integrity of such systems.

References

[1] Tarek M Taha, Raqibul Hasan, and Chris Yakopcic. Memristor crossbar based multicore neuromorphic processors. *2014 27th IEEE International System-on-Chip Conference (SOCC)*, 383–389, 2014, IEEE.

[2] Use cases from nist big data requirements wg v1.0. http://bigdatawg.nist.gov/_uploadfiles/M0180_v15_1305888391.docx, 2013.

[3] CESM CAM5. http://www.cesm.ucar.edu/working_groups/Atmosphere/development/, 2014.

[4] Coupled model intercomparison project phase 5. http://cmip-pcmdi.llnl.gov/cmip5/, 2014.

[5] Intergovernmental panel on climate change, fifth assessment report. http://www.ipcc.ch/report/ar5/, 2014.

[6] Supernovae detection. http://newscenter.lbl.gov/2014/05/21/confirmed-stellar-behemoth-self-destructs-in-a-type-iib-supernova/, 2014.

[7] WCRP coupled model intercomparison project phase 6. http://www.wcrp-climate.org/wgcm-cmip/wgcm-cmip6, 2014.

[8] P. K. Agarwal, S. S. Hampton, J. Poznanovic, A. Ramanathan, S. R. Alam, and P. Crozier. Performance modeling of microsecond scale biological molecular dynamics simulations onheterogeneous architectures. *Concurrency and Computation: Practice and Experience*, 25(13):1356–1375, 2013.

[9] R. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning and Research*, 6 (Nov):1817–1853, 2005.

[10] G. Angeli, J. Tibshirani, J.Y. Wu, and C. D. Manning. Combining distant and partial supervision for relation extraction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2014.

[11] J. Ansel, M. Pacula, Y. L.Wong, C. Chan, M. Olszewski, U.-M. O'Reilly, and S. Amarasinghe. Siblingrivalry: online autotuning through local competitions. In CASES, 2012.

[12] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina, T. Mezzacappa, P. Moin, M. Norman, R. Rosner, V. Sarkar, A. Siegel, F. Streitz, A. White, and M.Wright. The opportunities and challenges of exascale computing. *DOE ASCR Workshop Report*, 2010.

[13] A. G. Athanassiadis, M. Z. Miskin, P. Kaplan, N. Rodenberg, S. H. Lee, J. Merritt, E. Brown, J. Amend, H. Lipson, and H. M. Jaeger. Particle shape effects on the stress response of granular packings. *Soft Matter*, 10:48–59, 2014.

[14] K. Balasubramanian, J. Kim, A. A. Puretskiy, M. W. Berry, and H. Park. A fast algorithm for nonnegative tensor factorization using block coordinate descent and an active-set-type method. In Text Mining Workshop, Proceedings of the Tenth SIAM International Conference on Data Mining, pages 25–34, Columbus, OH, 2010. April 29–May 1.

[15] A. P. Bartok, M. C. Payne, R. Kondor, and G. Csanyi. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. Phys. Rev. Letters, 104:136403, Apr 2010.

[16] F. Bates and G. Fredrickson. Block Copolymers—Designer Soft Materials. *Physics Today*, 52(February):32, 1999.

[17] L. Bautista-Gomez, S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka. Fti: High performance fault tolerance interface for hybrid systems. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC '11, pages 32:1–32:32, New York, NY, USA, 2011. ACM.

[18] P. Beckman, R. Brightwell, B. R. de Supinski, M. Gokhale, S. Hofmeyr, S. Krishnamoorthy, M. Lang, B. Maccabe, J. Shalf, and M. Snir. Exascale operating systems and runtime software. *DOE ASCR Workshop Report*, 2012.

[19] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, and J. Hiller. Exascale computing study: Technology challenges in achieving exascale systems. Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Report, 15, 2008.

[20] G. Bronevetsky and B. de Supinski. Soft error vulnerability of iterative linear algebra methods. In Proceedings of the 22Nd Annual International Conference on Supercomputing, ICS '08, pages 155–164, New York, NY, USA, 2008. ACM.

[21] Caltech-JPL. The caltech-jpl summer school on big data analytics (coursera). https://www. mooc-list.com/course/caltech-jpl-summer-school-big-data-analytics-coursera, 2014.

[22] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir. Toward exascale resilience. *International Journal of High Performance Computing Applications*, 23(4):374–388, 2009.

[23] F. Cappello, A. Geist, W. D. Gropp, S. Kale, B. Kramer, and M. Snir. Toward exascale resilience: 2014 update. *Supercomputing Frontiers and Innovations*, 1:1–28, 2014.

[24] U.V. Catalyurek, F. Dobrian, A. Gebremedhin, M. Halappanavar, and A. Pothen. Distributed-memory parallel algorithms for matching and coloring. In Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, pages 1971–1980. IEEE, 2011.

[25] J. Chang, G. A. Reis, and D. I. August. Automatic instruction-level software-only recovery. *In Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 83–92. IEEE, 2006.

[26] J. Chen, L. K. John, and D. Kaseridis. Modeling program resource demand using inherent program characteristics. *SIGMETRICS Performance Evaluation Review,* 39(1):1–12, June 2011.

[27] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, and O. Temam. Benchnn: On the broad potential application scope of hardware neural network accelerators. In 2012 IEEE International Symposium on Workload Characterization (IISWC), pages 36–45. IEEE, 2012.

[28] C. S. Chennubhotla and A. Jepson. Hierarchical eigensolvers for transition matrices in spectral methods. *Advances in Neural Information Processing Systems (NIPS'05)*, 17:273–280, 2005.

[29] V. Chippa, D. Mohapatra, A. Raghunathan, K. Roy, and S. Chakradhar. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. *In Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 555–560, June 2010.

[30] J. Chung, I. Lee, M. Sullivan, J. H. Ryoo, D. W. Kim, D. H. Yoon, L. Kaplan, and M. Erez. Containment domains: A scalable, efficient, and flexible resilience scheme for exascale systems. *In High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pages 1–11, Nov 2012.

[31] Committee on the Analysis of Massive Data Committee on Applied and Theoretical Statistics Board on Mathematical Sciences and Their Applications and Division on Engineering and Physical Sciences *Frontiers in Massive Data Analysis. Frontiers in Massive Data Analysis*. The National Academies Press, 2013.

[32] D. J. Crandall, G. C. Fox, and J. D. Paden. Layer-finding in radar echograms using probabilistic graphical models. In Pattern Recognition (ICPR), 2012 21st International Conference on, pages 1530–1533. IEEE, 2012.

[33] S. Curtarolo, G. L. W. Hart, M. B. Nardelli, N. Mingo, S. Sanvito, and O. Levy. The highthroughput highway to computational materials design. *Nature Materials*, 12(3):191–201, Mar. 2013.

[34] P. F. Damasceno, M. Engel, and S. C. Glotzer. Predictive Self-Assembly of Polyhedra into Complex Structures. *Science*, 337(6093):453-457, July 2012.

[35] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[36] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, F. Cappello, B. Chapman, X. Chi, A. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, A. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, A. Hoisie, K. Hotta, Z. Jin, Y. Ishikawa, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, A. Lichnewsky, T. Lippert, B. Lucas, B. Maccabe, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. S. Mueller, W. E. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, A. Trefethen, M. Valero, A. Van Der Steen, J. Vetter, P. Williams, R. Wisniewski, and K. Yelick. The international exascale software project roadmap. *International Journal of High Performance Computing Applications*, 25(1):3–60, Feb. 2011.

[37] J. Dongarra, J. Hittinger, J. Bell, L. Chacon, R. Falgout, M. Heroux, P. Hovland, E. Ng, C. Webster, and S. Wild. Applied mathematics research for exascale computing. DOE ASCR Workshop Report, 2014.

[38] P. Du, A. Bouteiller, G. Bosilca, T. Herault, and J. Dongarra. Algorithm-based fault tolerance for dense matrix factorizations. SIGPLAN Not., 47(8):225–234, Feb. 2012.

[39] N. El-Sayed and B. Schroeder. Reading between the lines of failure logs: Understanding how hpc systems fail. In Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on, pages 1–12. IEEE, 2013.

[40] J. Elliott, K. Kharbas, D. Fiala, F. Mueller, K. Ferreira, and C. Engelmann. Combining partial redundancy and checkpointing for hpc. In Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on, pages 615–626, June 2012.

[41] E. N. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. ACM Computing Surveys (CSUR), 34(3):375–408, 2002.

[42] E. N. Elnozahy, R. Bianchini, T. El-Ghazawi, A. Fox, F. Godfrey, A. Hoisie, and J. Simons. System resilience at extreme scale. Technical report, Defense Advanced Research Project Agency (DARPA), Tech. Rep, 2008.

[43] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. ACM Comput. Surv., 34(3):375–408, Sept. 2002.

[44] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In 2011 38th Annual International Symposium on Computer Architecture(ISCA), pages 365–376. IEEE, 2011.

[45] A. L. Ferguson, A. Z. Panagiotopoulos, I. G. Kevrekidis, and P. G. Debenedetti. Nonlinear dimensionality reduction in molecular simulation: The diffusion map approach. *Chemical Physics Letters*, 509(1–3):1–11, 2011.

[46] E. Ferrara, M. JafariAsbagh, O. Varol, V. Qazvinian, F. Menczer, and A. Flammini. Clustering memes in social media. In Advances in Social Networks Analysis and Mining (ASONAM), 2013 IEEE/ACM International Conference on, pages 548–555. IEEE, 2013.

[47] K. Ferreira, R. Riesen, P. Bridges, D. Arnold, J. Stearley, J. H. L. III, R. Oldfield, K. Pedretti, and R. Brightwell. Evaluating the viability of process replication reliability for exascale systems. In Conference on High Performance Computing Networking, Storage and Analysis, SC 2011, Seattle, WA, USA, November 12–18, 2011, Nov. 2011.

[48] K. B. Ferreira, P. Bridges, and R. Brightwell. Characterizing application sensitivity to os interference using kernel-level noise injection. In Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, SC '08, pages 19:1–19:12, Piscataway, NJ, USA, 2008. IEEE Press.

[49] K. B. Ferreira, P. Widener, S. Levy, D. Arnold, and T. Hoefler. Understanding the effects of communication and coordination on checkpointing at scale. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pages 883–894. IEEE Press, 2014.

[50] G. Fox, S. Jha, Q. Judy, and L. Andre. Kaleidoscope of (apache) big data stack (abds) and hpc technologies. http://hpc-abds.org/kaleidoscope/, 2014.

[51] G. C. Fox, S. Jha, J. Qiu, and A. Luckow. Towards an understanding of facets and exemplars of big data applications. Proceedings of 20 Years of Beowulf Workshop, accecpted.

[52] S. Fu and C.-Z. Xu. Exploring event correlation for failure prediction in coalitions of clusters. In Supercomputing, 2007. SC '07. Proceedings of the 2007 ACM/IEEE Conference on, pages 1–12, Nov 2007.

[53] H. Fujuta, R. Schreiber, and A. Chien. It's time for new programming models for unreliable hardware. In Proceedings of the international conference on architectural support for programming languages and operating systems (ASPLOS), 2013.

[54] A. Gainaru, F. Cappello, J. Fullop, S. Trausan-Matu, and W. Kramer. Adaptive event prediction strategy with dynamic time window for large-scale hpc systems. In Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques, page 4. ACM, 2011.

[55] A. Gainaru, F. Cappello, and W. Kramer. Taming of the shrew: modeling the normal and faulty behaviour of large-scale hpc systems. In Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, pages 1168–1179. IEEE, 2012.

[56] A. Gainaru, F. Cappello, M. Snir, and W. Kramer. Fault prediction under the microscope: A closer look into hpc systems. In Proceedings of the International Conference on High 60 Performance Computing, Networking, Storage and Analysis, page 77. IEEE Computer Society Press, 2012.

[57] A. Gainaru, F. Cappello, M. Snir, and W. Kramer. Failure prediction for hpc systems and applications: Current situation and open issues. *International Journal of High Performance Computing Applications*, 27(3):273–282, Aug. 2013.

[58] X. Gao, E. Ferrara, and J. Qiu. Parallel clustering of high-dimensional social media data streams. 2014. *Technical Report*.

[59] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 29–43. ACM, 2013.

[60] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining,* 212:191–212, 2003.

[61] S. C. Glotzer and M. J. Solomon. Anisotropy of building blocks and their assembly into complex structures. *Nature Materials*, 6(7):557–562, Aug. 2007.

[62] L. B. Gomez and F. Cappello. Detecting silent data corruption through data dynamic monitoring for scientific applications. PPoP 2014, 2014.

[63] M. Halappanavar, J. Feo, O. Villa, A. Tumeo, and A. Pothen. Approximate weighted matching on emerging manycore and multithreaded architectures. International Journal of High Performance Computing Applications, Pages 413-430,Volume 26 Issue 4, November 2012.

[64] A. Y. Hannun, C. Case, J. Casper, B. C. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. CoRR, abs/1412.5567, 2014.

[65] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheer, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Mller. Assessment and validation of machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9(8):3404–3419, 2013.

[66] P. H. Hargrove and J. C. Duell. Berkeley lab checkpoint/restart (blcr) for linux clusters. In *Journal of Physics: Conference Series*, volume 46, page 494. IOP Publishing, 2006.

[67] R. Hasan and T. M. Taha. Enabling back propagation training of memristor crossbar neuromorphic processors. In 2014 International Joint Conference on Neural Networks (IJCNN), pages 21–28. IEEE, 2014.

[68] E. Heien, D. Kondo, A. Gainaru, D. LaPine, B. Kramer, and F. Cappello. Modeling and tolerating heterogeneous failures in large parallel systems. In High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for, pages 1–11. IEEE, 2011.

[69] A. Heller. Groundbreaking science with the worlds brightest x rays. https://str.llnl.gov/JanFeb11/hauriege.html, 2014.

[70] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the Influence of System Noise on Large-Scale Applications by Simulation. In International Conference for High Performance Computing, Networking, Storage and Analysis (SC'10), Nov. 2010.

[71] H. Hoffmann. Racing vs. pacing to idle: A comparison of heuristics for energy-aware resource allocation. In HotPower, 2013.

[72] A. Hoisie, D. Kerbyson, R. Lucas, A. Rodrigues, J. Shalf, and J. Vetter. Modeling and simulation of exascale systems and applications. *DOE ASCR Workshop Report*, 2012.

[73] G. B. Huang, M. Mattar, T. Berg, E. Learned-Miller, et al. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition, 2008.

[74] K.-H. Huang and J. Abraham. Algorithm-based fault tolerance for matrix operations. Computers, IEEE Transactions on, C-33(6):518–528, June 1984.

[75] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1), 2013.

[76] T. Karnowski. Deep machine learning with spatio-temporal inference. PhD thesis, University of Tennessee, 2012.

[77] A. Khan, M. Halappanavar, F. Manne, and A. Pothen. Computing approximate b-matchings in parallel. In CSC14: The Sixth SIAM Workshop on Combinatorial Scientific Computing, 2012.

[78] M. C. Kind and R. J. Brunner. Somz: photometric redshift pdfs with self-organizing maps and random atlas. Monthly Notices of the Royal Astronomical Society, 438(4):3409–3421, 2014.

[79] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[80] A. Krizhevsky and G. Hinton. Convolutional deep belief networks on cifar-10. Unpublished manuscript, 2010.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[82] A. G. Kusne, T. Gao, A. Mehta, L. Ke, M. C. Nguyen, K. Ho, V. Antropov, C. Wang, M. J. Kramer, C. Long, and I. Takeuchi. On-the-fly machine-learning for high-throughput experiments: search for rare-earth-free permanent magnets. *Scientific Reports*, 4, 2014.

[83] B. Lee, J. Collins, H. Wang, and D. Brooks. Cpr: Composable performance regression for scalable multiprocessor models. In MICRO, 2008.

[84] B. C. Lee and D. Brooks. Efficiency trends and limits from comprehensive microarchitectural adaptivity. In ASPLOS, 2008.

[85] B. C. Lee and D. M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In ASPLOS, 2006.
[86] J. Li and J. Martinez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In HPCA, 2006.

[87] A. Ma and A. R. Dinner. Automatic method for identifying reaction coordinates in complex systems. *The Journal of Physical Chemistry B*, 109(14):6769–6779, 2005. PMID: 16851762.

[88] R. J. Macfarlane, B. Lee, M. R. Jones, N. Harris, G. C. Schatz, and C. A. Mirkin. Nanoparticle superlattice engineering with dna. *Science*, 334(6053):204–208, 2011.

[89] M. W. Mahoney and P. Drineas. Cur matrix decompositions for improved data analysis. Proceedings of the National Academy of Sciences, 106(3):697–702, 2009.

[90] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch. Power management of online data-intensive services. *ISCA*, 2011.

[91] A. J. Meuler, M. A. Hillmyer, and F. S. Bates. Ordered network mesostructures in block polymer materials. *Macromolecules*, 42(19):7221–7250, 2009.

[92] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2–Volume 2, pages 1003–1011. Association for Computational Linguistics, 2009.

[93] N. Mishra, H. Zhang, J. D. Lafferty, and H. Hoffmann. A probabilistic graphical approach for optimizing energy under performance constraints. In ASPLOS, 2015.

[94] M. Z. Miskin and H. M. Jaeger. Adapting granular materials through artificial evolution. *Nature Materials*, 12(4):326–331, 2013.

[95] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. CoRR, abs/1312.5602, 2013.

[96] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, A. V. Lilienfeld, and K.-R. Müller. Learning invariant representations of molecules for atomization energy prediction. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 440–448. Curran Associates, Inc., 2012.

[97] N. Nakka, A. Agrawal, and A. Choudhary. Predicting node failure in high performance computing systems from failure and usage logs. In Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on, pages 1557–1566. IEEE, 2011.

[98] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, and A. Choudhary. Minebench: A benchmark suite for data mining workloads. In IISWC, 2006.

[99] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple timeseries. In Proceedings of the 31st International Conference on Very Large Data Bases, VLDB '05, pages 697–708. VLDB Endowment, 2005.

[100] P. Paschou, M. W. Mahoney, A. Javed, J. R. Kidd, A. J. Pakstis, S. Gu, K. K. Kidd, and P. Drineas. Intra- and interpopulation genotype reconstruction from tagging snps. Genome Research, 17(1):96–107, 2007.

[101] B. Peters, G. T. Beckham, and B. L. Trout. Extensions to the likelihood maximization approach for finding reaction coordinates. *The Journal of Chemical Physics*, 127(3):-, 2007.

[102] P. Petrica, A. M. Izraelevitz, D. H. Albonesi, and C. A. Shoemaker. Flicker: A dynamically adaptive architecture for power limited multicore systems. In ISCA, 2013.

[103] C. L. Phillips and G. A. Voth. Discovering crystals using shape matching and machine learning. Soft Matter, 9:8552–8568, 2013.

[104] D. Ponomarev, G. Kucuk, and K. Ghose. Reducing power requirements of instruction scheduling through dynamic allocation of multiple datapath resources. In MICRO, 2001.

[105] Prabhat, O. Rbel, S. Byna, K. Wu, F. Li, M. Wehner, and W. Bethel. Teca: A parallel toolkit for extreme climate analysis. Procedia Computer Science, 9(0):866–876, 2012. Proceedings of the International Conference on Computational Science, fICCSg 2012.

[106] J. Qiu, S. Jha, and G. C. Fox. Towards hpc-abds: An initial high-performance bigdata stack.

[107] A. Ramanathan, P. Agarwal, M. Kurnikova, and C. Langmead. Lecture Notes in Computer Science, volume 5541, pages 138–154. Springer Berlin Heidelberg, 2009.

[108] A. Ramanathan, A. J. Savol, P. K. Agarwal, and C. S. Chennubhotla. Event detection and sub-state discovery from biomolecular simulations using higher-order statistics: Application to enzyme adenylate kinase. *Proteins: Structure, Function, and Bioinformatics*, 80(11):2536–2551, 2012.

[109] A. Ramanathan, J. O. Yoo, and C. J. Langmead. On-the-fly identification of conformational substrates from molecular dynamics simulations. *Journal of Chemical Theory and Computation*, 7(3):778–789, 2015/01/27 2011.

[110] J. W. Reed, Y. Jiao, T. E. Potok, B. A. Klump, M. T. Elmore, and A. R. Hurson. Tf-icf: A new term weighting scheme for clustering dynamic data streams. In Machine Learning and Applications, 2006. ICMLA'06. 5th International Conference on, pages 258–263. IEEE, 2006.

[111] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Letters*, 108:058301, Jan 2012.

[112] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpaty, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. arXiv preprint arXiv:1409.0575, 2014.
[113] E. E. Santiso and B. L. Trout. A general set of order parameters for molecular crystals. *The Journal of Chemical Physics*, 134(6) 2011.

[114] C. Scherrer, A. Tewari, M. Halappanavar, and D. Haglin. Feature clustering for accelerating parallel coordinate descent. *In Advances in Neural Information Processing Systems*, pages 28–36, 2012.

[115] D. C. Snowdon, E. Le Sueur, S. M. Petters, and G. Heiser. Koala: A platform for os-level power management. In EuroSys, 2009.

[116] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke. Finding density functionals with machine learning. *Phys. Rev. Letters*, 108:253002, Jun 2012.

[117] S. Sridharan, G. Gupta, and G. S. Sohi. Holistic run-time parallelism management for time and energy efficiency. In ICS, 2013.

[118] R. St Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger. General-purpose code acceleration with limited-precision analog computation. In Proceeding of the 41st Annual International Symposium on Computer Architecuture, pages 505–516. IEEE Press, 2014.

[119] A. Stark. X-ray laser acts as tool to track lifes chemistry. https://www.llnl.gov/news/ x-ray-laser-acts-tool-track-lifes-chemistry, 2014.

[120] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams. The missing memristor found. *Nature*, 453(7191):80–83, 2008.

[121] S. Sukumar and K. Ainsworth. Pattern search in multi-structure data: a framework for the next-generation evidence-based medicine. Proceedings of SPIE, 9039, 2014.

[122] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 374–383, New York, NY, USA, 2006. ACM.

[123] C. Symons, S. Sukumar, B. Park, M. Shankar, R. Vatsavai, R. K. Archibald, A. Ganguly, and S. Prowell. A predictive analysis toolbox for ultrascale data exploration. Technical report, Oak Ridge National Laboratory, 2011.

[124] A. P. Thompson, L. P. Swiler, C. R. Trott, S. M. Foiles, and G. J. Tucker. A spectral analysis method for automated generation of quantum-accurate interatomic potentials, 2014. [125] N.Y. Times. Scientists see promise in deep-learning programs. http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-intelligence. html?pagewanted=all&_r=0&pagewanted=print, 2014.

[126] L. Vu. Confirmed: Stellar behemoth self-destructs in a type iib supernova. http://newscenter.lbl.gov/2014/05/21/confirmed-stellar-behemoth-self-destructs-in-a-type-iib-supernova/, 2014.

[127] S. Wang, J. Ward, S. Leyffer, S. M. Wild, C. Jacobsen, and S. Vogt. Unsupervised cell identification on multidimensional X-ray fluorescence datasets. *Journal of Synchrotron Radiation*, 21(3):568–579, May 2014.

[128] Z. Xue, X. Dong, S. Ma, and W. Dong. A survey on failure prediction of large-scale server clusters.

In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on, volume 2, pages 733–738. IEEE, 2007.

[129] J. J. Yi, D. J. Lilja, and D. M. Hawkins. A statistically rigorous approach for improving simulation methodology. In HPCA, 2003.

[130] Z. Zheng, Z. Lan, R. Gupta, S. Coghlan, and P. Beckman. A practical failure prediction with location and lead time for blue gene/p. In Dependable Systems and Networks Workshops (DSN-W), 2010 International Conference on, pages 15–22. IEEE, 2010.

Appendix: Workshop Participants and Other Contributors

Dorian C. Arnold	University of New Mexico
Prasanna Balaprakash	Argonne National Laboratory
Michael Berry	University of Tennessee, Knoxville
Ron Brightwell	Sandia National Laboratories
Franck Cappello	Argonne National Laboratory
Barry Chen	Lawrence Livermore National Laboratory
Stephen P. Crago	University of Southern California
Kurt B. Ferreira	Sandia National Laboratories
Todd Gamblin	Lawrence Livermore National Laboratory
Mahantesh Halappanavar	Pacific Northwest National Laboratory
Hank Hoffmann	University of Chicago
Steven Hofmeyr	Lawrence Berkeley National Laboratory/UC Berkeley
Tamara G. Kolda	Sandia National Laboratories
Ignacio Laguna	Lawrence Livermore National Laboratory
Seung-Hwan Lim	Oak Ridge National Laboratory
Robert Lucas	University of Southern California
Michael Mahoney	University of California, Berkeley
Una-May O'Reilly	Massachusetts Institute of Technology
Byung Park	Oak Ridge National Laboratory
Carolyn Philips	Argonne National Laboratory
Reid Porter	Los Alamos National Laboratory
Thomas E. Potok	Oak Ridge National Laboratory
Prabhat	Lawrence Berkeley National Laboratory/UC Berkeley
Judy Fox Qiu	Indiana University
Arvind Ramanathan	Oak Ridge National Laboratory
Abhinav Sarje	Lawrence Berkeley National Laboratory/UC Berkeley
Dylan Schmorrow	SoarTech
Justin Shumaker	U.S. Army Research Laboratory
Marc Snir	Argonne National Laboratory
Sreenivas R. Sukumar	Oak Ridge National Laboratory
Tarek Taha	University of Dayton
Raju Vatsavai	North Carolina State University
Abhinav Vishnu	Pacific Northwest National Laboratory
Elizabeth Whitaker	Georgia Tech Research Institute
Yan Yufik	VSR, Inc.