2018 IEEE 38th International Conference on Distributed Computing Systems

Cognified Distributed Computing

Ozalp Babaoglu
Department of Computer Science and Engineering
University of Bologna
Italy
Email: ozalp.babaoglu@unibo.it

Alina Sîrbu
Department of Computer Science
University of Pisa
Italy
Email: alina.sirbu@unipi.it

***Abstract*—*Cognification* — the act of transforming ordinary objects or processes into their intelligent counterparts through Data Science and Artificial Intelligence — is a disruptive technology that has been revolutionizing disparate fields ranging from corporate law to medical diagnosis. Easy access to massive data sets, data analytics tools and High-Performance Computing (HPC) have been fueling this revolution. In many ways, cognification is similar to the *electrification* revolution that took place more than a century ago when electricity became a ubiquitous commodity that could be accessed with ease from anywhere in order to transform mechanical processes into their electrical counterparts. In this paper, we consider two particular forms of distributed computing — Data Centers and HPC systems — and argue that they are ripe for cognification of their entire ecosystem, ranging from top-level applications down to low-level resource and power management services. We present our vision for what *Cognified Distributed Computing* might look like and outline some of the challenges that need to be addressed and new technologies that need to be developed in order to make it a reality. In particular, we examine the role cognification can play in tackling power consumption, resiliency and management problems in these systems. We describe intelligent software-based solutions to these problems powered by on-line predictive models built from streamed real-time data. While we cast the problem and our solutions in the context of large Data Centers and HPC systems, we believe our approach to be applicable to distributed computing in general. We believe that the traditional systems research agenda has much to gain by crossing discipline boundaries to include ideas and techniques from Data Science, Machine Learning and Artificial Intelligence.**

***Index Terms*—High-Performance Computing, Data Centers, energy efficiency, resiliency, Data Science, Machine Learning, Artificial Intelligence.**

I. Introduction

Motivation: Data Centers and High-Performance Computing (HPC) systems have become indispensable for economic growth and scientific progress in our modern society. Data centers are the engines of the Internet that run e-commerce sites, cloud-based services and social networks utilized by billions of users each day. HPC systems, on the other hand, have become fundamental "instruments" for driving scientific discovery and industrial competitiveness — much like the microscopes, telescopes and steam engines of the previous century. Continued desire to achieve higher-fidelity simulations, build better models and analyze greater quantities of data put increasing demands for higher performance from these systems. As the performance of HPC systems increases, the value of the results they produce increases, enabling improved techniques, policy decisions and manufacturing processes in areas such as agriculture, engineering, transportation, materials, energy, health care, security and the environment. Today, HPC systems are also essential for achieving groundbreaking results in basic sciences ranging from particle physics to cosmology while touching genomics, pharmacology, neuroscience, geology, material science, meteorology and climate change in between [1]. Scaling current systems to meet these challenges is being limited by considerations for power consumption, heat dissipation, resiliency and management. Brute-force scaling of current technologies to the required performance levels would result in systems that consume as much power as a small-size city, that fail every several minutes and that are unmanageable.

Contributions: We believe that the traditional distributed systems research agenda can benefit greatly by crossing discipline boundaries to include ideas and techniques from Data Science, Machine Learning and Artificial Intelligence (AI). In particular, we argue that Data Centers and High-Performance Computing systems, which we collectively refer to as *High-Performance Distributed Computing* (HPDC) systems, have much to gain by incorporating data-driven, predictive and proactive software technologies into various phases of their operation. In support of our claim, we outline how these technologies can be exploited to dramatically reduce power consumption and significantly improve resiliency of future HPDC systems. Furthermore, we show how the data-driven predictive models built for energy efficiency and resiliency can also become the building blocks of an innovative system management platform built from open-source software packages. The biggest takeaway from our work is a perspective on *where* and *how* data analytics should be positioned to have the greatest impact in the general systems arena, and specifically in HPDC systems.

Organization: The rest of the paper is organized as follows. In Section II we identify power consumption, resiliency and manageability as the main obstacles for achieving ever-increasing performance in HPDC systems through ever-increasing parallelism. In Section III we recall how availability of massive data sets has led to significant advances in AI by abandoning traditional rule-based techniques in favor of data-driven techniques. We also introduce *cognification* as the transformation of ordinary services into intelligent ones by accessing the required data analytics and intelligence func-

2575-8411/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDCS.2018.00118

tions as commodities [2]. In Sections III-A and III-B we review some of the more important data-driven predictive AI technologies, such as *Machine Learning*, and show how they can be used to build powerful predictive models for power consumption and failures. In Section IV we present our vision for a HPDC system that has been cognified through data-driven dispatcher for increased energy efficiency (Section IV-A), just-in-time check-pointing mechanism for increased resiliency (Section IV-B) and predictive software tools for improved system management (Section IV-C). Section V discusses the remaining challenges that need to be solved in order to make our vision a reality. Section VI concludes the paper.

II. Challenges of High-Performance Distributed Computing

Future HPDC systems will achieve higher performance through a combination of faster processors and massive parallelism. With Moore's Law having essentially reached its limit, it is expected that continued die shrinking will deliver only a small additional increase in performance. The rest of the increase has to come from abandoning increased transistor density and switching to increased core count, which implies a substantial increase in the sockets count [3].

We argue that future HPDC systems have to be *sustainable* so that they are able to provide high-performance computing on a continual basis without interruptions. Achieving high performance by increasing the number of cores (and consequently, the number of sockets) presents two primary obstacles for sustainability: power consumption and resiliency. Considerations for the cost of power generation, power delivery and chiller/cooler infrastructures put 30MW as a practical upper limit for the power consumption of future HPDC systems [4]. In other words, energy efficiency of current systems has to increase by more than one order of magnitude to stay within this limit when they are scaled to extreme performance levels [5].

With everything else being equal, the failure rate of a system is directly proportional to the number of sockets used in its construction [6]. But everything is not equal: future HPDC systems not only will have many more sockets, they will also use advanced low-voltage technologies that are much more prone to aging effects [7] together with system-level performance and power modulation techniques, such as *Dynamic Voltage Frequency Scaling* (DVFS), all of which tend to increase failure rates [8]. Economic forces that push for building systems out of commodity components aimed at mass markets only add to the likelihood of more frequent unmasked hardware failures. Finally, complex system software, often built using open-source components to deal with more complex and heterogeneous hardware, failure masking and energy management, coupled with legacy applications will significantly increase the potential for software errors [9].

It is estimated that complex applications may fail as frequently as once every 30 minutes on future HPDC platforms [10]. At these rates, failures will prevent applications from making progress. Consequently, high performance, even when achieved nominally, cannot be sustained for the duration of most applications that are long running. Future HPDC systems must include a combination of hardware and software technologies that are capable of handling failures at accelerated rates from a broad set of sources [8]. Whether their origin is software, hardware or environmental, we limit our attention to those failures that result in computations simply stopping. In other words, we assume that failures resulting in *silent data corruptions* or computations that continue running but perform incorrect actions or produce incorrect results are extremely rare and can be ignored.

HPDC systems (especially Data Centers) evoke images of warehouse-size structures filled with racks housing tens of thousands of multi-core servers and storage devices, interconnected through a variety of networking technologies [11]. In HPDC systems, computing and networking are actually only a small part of their infrastructure that also has to guarantee electrical power, either from an external grid, from an on-site generation plant, or both. Furthermore, the infrastructure has to include subsystems for conditioning, storing and switching power as well as complex thermal cooling/chiller systems to dissipate the vast amount of heat that is produced. If in addition to this hardware infrastructure, we also consider the intricate tangles of advanced software that run their services and include external factors such as human operators, we end up with systems where interdependencies and interactions among a very large number of components result in nonlinear behaviors that are highly unpredictable and where small changes in one part often have large and unintended consequences in other distant parts. In short, attempts to manage HPDC systems through traditional human operator-based mechanisms become error prone at best, and impractical at worst. What is needed is an intelligent management system that eliminates reliance on human operators as much as possible for both routine maintenance and problem avoidance/resolution.

Amidst the wealth of challenges that future HPDC systems present, we limit our attention in this paper to the following three that are central to sustainability:

- **Energy efficiency**: Improve the energy efficiency of future HPDC systems so as to stay within the 30MW limit for power consumption when they are scaled up for high performance.
- **Resiliency**: Improve the resiliency of future HPDC systems by reducing the perceived failure rates to at most once-per-week levels when they are scaled up for high performance.
- **Manageability**: Improve management of future HPDC systems by limiting reliance on human operators and facilitating semi-autonomic control.

HPDC systems, in addition to being effective instruments for cognifying other fields, are ripe for benefiting from cognification themselves to solve many of their own challenges. In a way, this is akin to "turning the microscope onto itself to study the microscope". Cognification of HPDC systems can occur at any level of their programming workflow, all the way

from top-level applications down to low-level system software. In this paper, we examine how the system software of HPDC systems can be cognified so as to render them sustainable. We sketch solutions to the above challenges based on intelligent and proactive software technologies for vastly reducing power consumption and dramatically improving resiliency. Manageability, on the other hand, requires developing intelligent and proactive mechanisms for monitoring, controlling and debugging systems and applications by both administrators and users.

A. Energy Efficiency

In HPDC systems, energy efficiency can be approached at many levels ranging from electronic circuits and packaging innovations all the way up to scheduling, allocation and compiling strategies. Energy efficiency at the system software level has been typically delegated to the dispatcher component that is concerned with which jobs to run next (scheduling) and where to run them (allocation). An energy-aware dispatcher can resolve decisions based on numerous techniques, including *consolidation*, *energy-aware scheduling*, *power capping* and DVFS, either separately or in combinations. The basic idea behind consolidation is to gather many active jobs/threads on a few physical nodes/cores so that idle nodes/cores can be switched to low-power mode or powered off completely [12], [13]. In making consolidation decisions, the dispatcher can take into consideration not only the computational demands of jobs but also their communication needs [14]. Furthermore, power consumed by HPDC applications is often multidimensional, nonlinear and has large dynamic range [12]. In other words, power-aware allocation schemes have to consider multiple measures for workloads (e.g., memory size in addition to CPU utilization) and take into account the effects of co-locating jobs on the same node. In large Data Centers, consolidation has been facilitated to a large extent by the availability of virtualization [15] and container technologies such as Docker [16] and Kubernetes [17]. The fact that container technologies are not as widespread in current HPC systems makes consolidation less common as an energy efficiency mechanism for them. Power capping is a technique where an energy-aware scheduler selects the set of jobs to run such that their cumulative power needs do not exceed a certain threshold [18]. It can be implemented using a variety of techniques such as DVFS [14], [19], *Machine Learning* or hybrid optimization techniques including *Constraint Programming* [20].

B. Resiliency

Like energy efficiency, resiliency in HPDC systems can be confronted at many levels ranging from error correcting codes at the electronics level up to fault-tolerant algorithms at the application level. Process-level or hardware-level replication is an often-employed technique to increase resilience in Data Centers. It is less common in HPC systems for several reasons. First, failure independence, which is the foundation for replication is difficult to satisfy in HPC systems which tend to be more tightly coupled. Second, replication often incurs high overhead in order to guarantee replica equivalence despite non-determinism in applications. Finally, hardware-level replication contributes to increasing socket counts and power consumption, which are already at elevated levels in HPC systems. Among the many software-based resiliency mechanism that exist, *check-point/restart* is by far the most widely available (being included in popular systems like Charm++ and AMPI) and widely used in current HPC systems [6]. Check-pointing consists of taking a snapshot of the application in execution and saving it on nonvolatile media (usually a parallel file system). When a failure occurs, the application is restarted from the last check-point found on nonvolatile media (after moving it to main memory) and the application continues until the next check-point.

A number of challenges need to be resolved to make check-point/restart a viable technique for increased resiliency in future HPDC systems. Check-pointing and restarting can be made automatic and transparent to applications by initiating them pro-actively through a system software layer as in BLCR [21]. While this removes a big burden from users, it comes at the cost of increasing overhead since the state that is saved and restored to/from nonvolatile memory cannot exploit application semantics (to reduce its size) and has to include the entire application state [9], [10]. The challenge is to maintain the convenience of system-initiated check-pointing at a cost comparable to user-initiated check-pointing. Too frequent check-pointing with high overhead can slow down applications to a crawl and can also be detrimental for energy efficiency. "Optimal" values for check-point intervals can be computed based on averages for inter-failure times and check-pointing costs [22]. In future HPDC systems with high failure rates and large check-point/restart times, there is a real risk that the mechanism degenerate into a "pure overhead" scheme performing only check-points/restarts and no useful computation [23]. Under these conditions, failure masking through replication becomes a viable option as a resiliency mechanism. The challenge is to devise dynamic and adaptive algorithms for adjusting the check-point interval and for switching between check-point/restart and replication as the appropriate resiliency mechanism.

C. Manageability

From the point of view of system administrators, current HPDC management tools are limited essentially to resource allocation, scheduling and monitoring tasks. Although a number of efforts for building more-innovative monitoring and management platforms exist [24], [25], they lack predictive or social capabilities, do not provide control actions and are based on rather simple interaction models. Thus, many of the difficult management tasks rest on the shoulders of operators. From an end-user's point of view, systems such as *Compute Manager* from PBS Works[1] for job submission recreate the "batch processing" experience of early computing with no interactive control and debugging capabilities.

[1] https://www.altair.com/compute-manager/

What is needed is a management software platform that can facilitate semi-automatic control of HPDC systems with little or no human intervention. The platform should make use of the proactive and predictive software technologies developed for energy efficiency and resilience while at the same time should be open, extensible and social so that it can be used by both HPDC system administrators and end-users.

III. The Power of Data

In a growing number of areas where experimentation is either impossible, dangerous or costly, computing is often the only viable alternative towards confirming existing theories or devising new ones. The resulting *computational scientific discovery* is typically model-based: as a first step, a mathematical model of whatever it is that is being studied is built and then the model is solved numerically or through simulations. This approach, which remains one of the main pillars of scientific discovery, is conditioned on our ability to construct mathematical models relating the dependent variables (outputs) to the independent variables (inputs) of the process under study. Recent years have witnessed an emerging trend that abandons this model-based computational approach and replaces it with a *data-driven* approach seeking correlations that may exist among huge volumes of data collected from observing actual inputs and outputs of processes [26]. The big advantage of the data-driven approach is its ability to uncover interesting insights and properties of processes without having to construct cause-effect mathematical models, which typically require a complete understanding of their inner workings.

The data-driven approach outlined above can be used to build an intelligence component in the form of a *predictive computational model*, to be integrated into the system or service to be cognified. In addition to uncovering hidden correlations among historical archived data and gaining knowledge about the past, the predictive model allows reasoning about future or unseen states. Moreover, building the predictive models in an on-line manner driven by streamed data (rather than the traditional off-line approach driven by archived data) opens up the possibility of cognified services that can *enact control actions* on the system in real-time so as to keep their executions along desirable trajectories. In the following, we outline some of the AI technologies that are relevant for cognification and discuss their use in building predictive models for power consumption and failures in HPDC systems.

A. Machine Learning, Deep Learning and Other Predictive Technologies

AI research, which had been stagnant for several decades has been rejuvenated recently mainly due to the shift from a rule-based approach to a data-driven approach powered by *Machine Learning* technologies [27]. Early demonstration of the enormous gains possible with data were *recommender systems* [28]. These systems, which are now pervasive on the Internet, are based on classifier algorithms that can infer preferences of humans directly from data without actually modeling human behavior. Advanced classifiers trained with huge amounts of data have made enormous progress in many other fields including image recognition [29], medical diagnosis [30], mortgage appraisals [31], speech recognition [32], archeology [33] and machine translation [34] without any knowledge of the inner workings of the processes being considered.

One technology that has had an important role in these success stories is *Deep Learning* (DL). DL is based on *Deep Neural Networks* (DNNs) that are *Artificial Neural Networks* with a large number of hidden layers and that are trained with a Back-propagation algorithm adjusted for the large network size. Some of the network types that have been considered include Convolutional, Recurrent and Autoencoder Neural Networks. The power of DNNs rest in their ability to automatically build features from the data by mapping the input into larger dimensions at the various network layers. These features are then employed to solve the classification task at hand. This technology, although not new, has benefited enormously from big-data tools that have made large volumes of data easily accessible and easy to process. Another factor contributing to the enormous success of DL is widespread access to high-performance parallel processing in the form of multi-core CPUs and special-purpose GPUs [35], [36].

B. Predictive Models for Power Consumption, Workloads and Failures

A first step in the cognification process is building predictive models at different levels of an HPDC system. Among the possible behaviors to be modeled, we consider power consumption, workload and failures. Below we review some of the recent work in building predictive models for these behaviors based on data collected from real systems.

1) Power Consumption: Power monitoring, modeling and optimization have been areas of intense research activity in recent years. Modern computing units embed advanced control mechanisms such as *Dynamic Voltage Frequency Scaling* that seek to optimize performance and can affect power levels, making modeling problematic even for a single computational unit [37]. Several models trying to explain the relation between frequency, load, hardware counters and power for single units have been introduced for multicore CPUs [38], [39] and GPUs [40]. The success of modeling power varies widely depending on the workload, with errors between 3.65% and 14.4% for CPUs, and between 1.7% and 27.7% for GPUs. These errors are only expected to grow when multiple units have to be combined, as will be the case for future HPDC systems.

Some recent work has focused on modeling job or application power consumption. Performance counters are used to model application power on HPC platforms platforms by [41] and power used by CUDA kernels in [42]. These methods require instrumenting the applications to extract signatures and performance counters.

The models cited above are able to compute power consumption by reading various performance counters during an execution, however they are not able to predict power

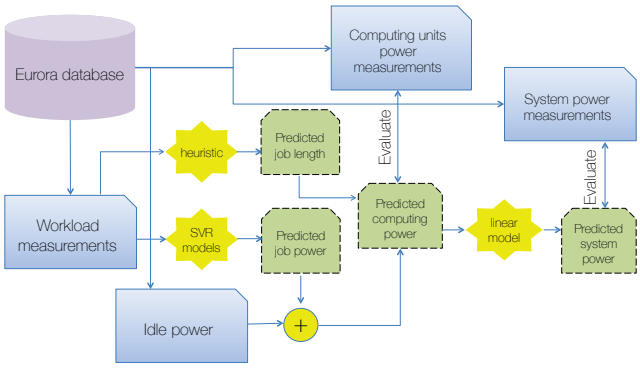

Fig. 1. Modeling system level power by integrating three different models, represented as stars in the figure: prediction of job power consumption using SVR models, prediction of job length through a data-driven heuristic and a linear model mapping computing unit power to system level power.

consumption prior to an execution. For the purposes of cognification, advance prediction is essential. This can be achieved by using only workload measures to predict power, as is done in [43], [44], [45]. Recently, we introduced a model [46] to predict job power consumption for a hybrid HPC system, Eurora [5]. This HPC system employs CPUs, GPUs and MIC technology to achieve low power consumption. Such heterogeneity in resources is a typical feature of modern HPDC systems, and makes power modeling more challenging. The predictive model we presented is fully data driven — no assumptions about the model structure nor additional instrumentation of application code are required. The only application-aware feature is the job name, making our method easily applicable to any system even when application code is not available. The power of our prediction derives from user history rather than from application counters, and our results show that when enough data is available, excellent predictive behavior can be achieved. We employ a multiple-*Support Vector Regression* (SVR) model to estimate job power as a function of time — we predict power *profiles* rather than an *averages* for the power consumed per job. When comparing the multiple-SVR approach to an *Enhanced Average Model* (EAM) where power depends only on the number of components used, SVR outperforms EAM for most users, obtaining good prediction (error under 20% or $R^2 \leq 0.5$) for 80% of the users analyzed. For the rest of the users, indications are that modeling power is affected by noise. An important aspect of our model is the fact that we take into account the effect of colocating jobs on the same node. This means that we can predict how different resource allocation schemes affect job power, in order to optimize allocation.

Our approach to predicting job power is intended to be used in *real-time*, where predictions are made as new jobs arrive at the scheduler. On-line use consists of training the model for each user, then applying it to real-time data. Periodically, the model is updated by incorporating recent data into the training dataset. We expect monthly model updates to be sufficient in order to capture changes in job structure.

While individual job power consumption is important, power consumption at the system level is also of interest. Recent work at Google [47] describes the use of Artificial Neural Networks to model *Power Usage Effectiveness* using a mixture of workload and cooling features. We have recently introduced a model of total system power for the Eurora HPC system [48]. Our predictive model consists of three components, displayed as stars in Figure 1. First, power consumption of jobs is predicted from workload data using the aforementioned SVR approach. Second, we introduce a simple heuristic that enables data-driven prediction of job duration (see section III-B2), which allows us to estimate which jobs will run in the system at a future time. The two predictions are then combined to estimate the power used by computing units. Third, we develop a relation between power used by computing units and the total system power, including networking, IO system and other elements, using a linear model. The 3-component model takes as input workload parameters, namely job names and resources allocated to each job. The approach achieves very good results in predicting system level power with errors under 5%. Figure 2 shows the predicted and real power consumption of the system during a one-week period. The methodology can be easily applied to other systems since the data types used (workload measures only) are generally available in most HPDC systems.

2) Workloads: Workload prediction is another important concern for cognification of HPDC systems. One aspect of workloads that can be predicted and that can have an impact on management of HPC systems is job duration. This can be achieved by monitoring jobs once they start, recording performance counters and estimating remaining duration based on them [49], [50]. A different approach is to build a predictive algorithm based on past job history only. We recently developed such a data-driven heuristic to predict job length [51] that uses only general job information that is available just before a job is started, hence it is much more flexible. We predict job durations in Eurora, the same HPDC system as above. The heuristic exploits time locality of job durations for individual users that was observed in the Eurora workload. Specifically, it has been observed that consecutive jobs by the same user tend to have similar durations, especially when they have the

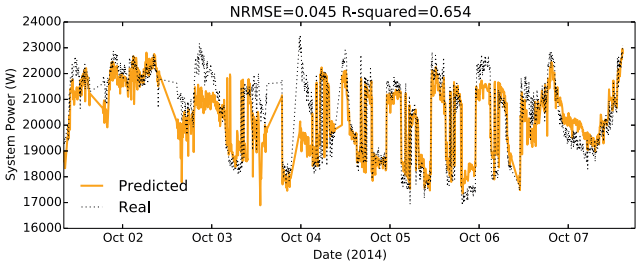

Fig. 2. Power predictions obtained by the model described in Figure 1. The model was applied to log data from Eurora, a hybrid HPC system. The plot includes measured and predicted power consumption during a one-week period [48]. Prediction results are very good, with mean error under 5%.

same profile (job name, resources requested, queue, etc). From time to time, a switch to a different duration is observed, which could happen, for example, when the user changes input datasets or the algorithm itself. Using this observation, we devised a set of rules to apply in order to predict job durations. We record job profiles for users, and their last durations. When a new job arrives in the system, we look for a job with the same or similar profile, and consider its duration to be also the duration of the new job. If no past profile is similar enough, the predicted duration is the wall-time of the job. In case a match is found, the wall-time is used as an upper bound for the predicted duration. This heuristic was shown to give very good results, with an average prediction error of 40 minutes over all jobs in the system.

3) Failures: Failure prediction is fundamental for system management and resiliency in HPDC systems [52]. It has been an active research area [53] and numerous failure prediction methods for single machines, clusters, application servers, file systems, hard drives, email servers and clients, etc. have been developed over the years. More recent studies concentrate on larger-scale distributed systems such as HPC and Data Centers for cloud computing.

Of particular interest for cognification are *job failure* predictions that can lead to economizing resources, as well as prediction of *component failures* (such as computing nodes). Job failures in a cloud setting have been studied by various authors [54], [55], using techniques such as the *naive Bayes Classifier* and *Principal Component Analysis*. The methods achieve good results on jobs from different application settings, with true positive rates above 80% and false positive rates under 30%. For node failures, a recent study of the Blue Waters HPC installation at Argonne National Laboratory achieved predictions with 60% TPR and 85% precision [56].

Recently we have presented a study of node failures for Data Centers based on log data from a Google cluster [57]. The dataset contains workload and scheduler events emitted by the Borg cluster management system [25] in a cluster of over 12,000 nodes during a one-month period [58], [59]. We employed BigQuery [60], a big data tool from the Google Cloud Platform that allows SQL-like queries to be run on massive data, to perform an exploratory feature analysis. This step generated a large number of features at various levels of aggregation suitable for use in a Machine Learning classifier. The use of BigQuery has allowed us to complete the analysis for large amounts of data(table sizes up to 12TB containing over 100 billion rows) in reasonable amounts of time.

The failure prediction problem was formulated as a classification task: decide whether a machine in the cluster will fail or not in the next 24 hours. We employed an ensemble of *Random Forest* (RF) classifiers to solve the problem. RF were employed due to their proven suitability in situations were the number of features is large [61] and the classes are "unbalanced" [62] such that one of the classes consists mainly of "rare events" that occur with very low frequency. Although individual RF were better than other classifiers that were considered in our initial tests, they still exhibited limited predictive power, which prompted us to pursue an *ensemble approach*. While individual trees in RF are based on subsets of features, we used a combination of *bagging* and data *subsampling* to build the RF ensemble and tailor the methodology to this particular dataset.

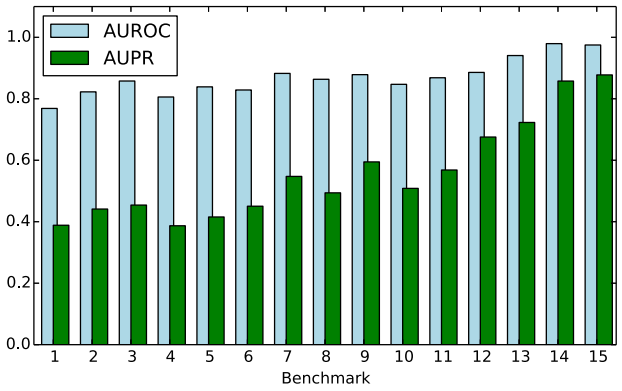

Fig. 3. Predictive power for classification of node failures in a Google cluster trace [57]. The log data released by Google was divided into 15 benchmarks (ten days of training data, one day of test data), and the plot shows *Area-Under-the-ROC* (AUROC) and *Area-Under-the-Precision-Recall* (AUPR) curves for each benchmark. Predictive power varies for different benchmarks, however AUROC values always stay above 0.76 and AUPR stay above 0.38.

To test our classifier in different settings, we split the available data into 15 benchmarks, each benchmark including 10 day of training data and one day of test data (with one day in between, so that train and test data do not overlap at all, given that we are predicting failures in windows of 24 hours). Figure 3 summarizes the results obtained, evaluating the model on each benchmark in terms of *Area-Under-the-ROC* (AUROC) curve and *Area-Under-the-Precision-Recall* (AUPR) curve. These show that we had very good predictive power on some days, and moderate on others, with AUROC values varying between 0.76 and 0.97 and AUPR values between 0.38 and 0.87. This corresponded to true positive rates in the range 27%-88% and precision values between 50% and 72% at a false positive rate of 5%. In other words, this means that in the worst case, we were able to identify 27% of failures, while if a data point was classified as a failure, we could have 50% confidence that we were looking at a real failure. For the best case, we were able to identify almost 90% of failures and 72% of instances classified as failures corresponded to real failures. All this, at the cost of having a 5% false alarm rate.

Again, the method we presented is very suitable for on-line use. A new model can be trained every day using the last 12 days of logs. This is the scenario we simulated when we created the 15 test benchmarks. The model would be *trained* with 10 days of data and *tested* on the next non-overlapping day, exactly like in the benchmarks. Then, it would be *applied* for one day to predict future failures. The next day a new model would be obtained from new data. Each time, only the last 12 days of data would be used rather than increasing the amount of training data. This would account for the fact that the system itself and the workload can change over time, so

old data may not be representative of current system behavior. This would ensure that the model is up-to-date with the current system state. Testing on one non-overlapping day is required for live use for two reasons. First, part of the test data is used to build the ensemble (prediction-weighted voting). Secondly, the true positive rates and precision values on test data can help system administrators make decisions on the *criticality of the predicted failure* when the model is applied.

Predicting failures in advance can help reduce resource wastage by modifying the resource allocation decisions dynamically. This is in itself an extensive research area, but a simple technique could be to *quarantine* nodes that are predicted to fail by not submitting any new jobs to them for some period of time. If consecutive failure alarms continue to appear, the quarantine period is extended until either the alarms stop or the node fails. We simulated such an approach. For better precision, we quarantine a node only if at two consecutive time points the node is predicted to fail. While a node is in quarantine, all tasks that would have otherwise run on that node need to be *redirected*. Among redirected tasks, some would have finished before the node failure, others would have been *interrupted*. We call the latter *recovered* tasks, since their interruption was avoided by proactively redirecting them. The aim of our proactive approach is to *maximize* the number of *recovered* tasks (the gain) while *minimizing* the number of *redirected* tasks (the cost). The two objectives are contradictory: the number of *recovered* tasks grows as the number of *redirected* tasks increases, so there is a tradeoff between the cost and the gain. With the predictions obtained by our RF approach, we showed that we can recover close to 1000 tasks using 800 CPU hours during the 15 days of prediction, which is about 50% of the resources that would be saved with a perfect prediction.

IV. Cognified High-Performance Distributed Computing

In this Section we sketch the architecture of a HPDC system that has been cognified to include the data-driven predictive software technologies described above in its dispatcher, check-point/restart and management functions. In what follows, we briefly discuss each one of these functions in light of the advantages offered by prediction.

A. Data-Driven Dispatching

Energy efficiency at the system software level in HPDC systems is typically delegated to the *dispatcher* component that is concerned with which jobs to run next (scheduling) and where to run them (allocation). Predictive models discussed earlier can be included in the dispatcher to achieve power and failure awareness as well as optimizing overall system performance. Specifically, the scheduler component can ensure power awareness through power capping and power reduction. Power capping can be achieved by including a constraint within the CP model specifying the threshold for total system power that cannot be exceeded. Being able to predict future system power accurately, the power capping constraint can precisely model the total system power. Power reduction, on the other hand, can be achieved by including total system power as part of the objective function, which then has to be minimized. The same job can consume different amounts of power depending on when it starts, what resources it uses and what resources it shares with other jobs, which will all be taken into account by both the scheduler and the allocator components of the dispatcher. To allow this, it is important that the predictive models for power include features related to shared resources.

Workload predictions, in particular job durations, are important for allocation since they provide a way to estimate which jobs will be in the system in the near future. Job duration predictions are useful also when trying to estimate future power consumption by summing the power consumption of all running jobs. CPU and memory size predictions can be used to optimize resource usage and minimize power by selecting jobs with vastly different profiles for co-location. For example, we can expect lower power consumption and better throughput when memory-intensive jobs are co-located with CPU-intensive jobs.

Failure awareness of the dispatcher can be achieved through several allocation decisions. First, resources can be ranked based on their predicted failure probability so that the next allocated resource is always the one that is considered least likely to fail. In this way, the allocator can avoid assigning new jobs to nodes that are likely to fail in the near future. Second, when replication is in use for resiliency, the dispatcher can try to locate replicas of a job on nodes that exhibit the greatest failure independence with the original node, as measured by predicted failure correlations between them.

B. Just-in-Time Check-Pointing

Check-pointing algorithms periodically save the state of a job to non volatile memory so that the job can be restarted from the last valid state in case of a failure. Prediction of failures can be integrated into this strategy by adjusting the time intervals between successive check-points. Cognified HPDC systems can make use of failure predictors that provide a probability of failure within a future time window, rather than a simple yes/no answer. The failure probability can be estimated using Bayesian classifiers, or artificial neural networks with a Softmax activation function at the output layer, or by transforming the yes/no classification problem into a regression problem. In the *just-in-time* check pointing scheme we propose, the failure probability is used as a weight in calculating the next time-to-check-point for the job. If the failure prediction accuracy is sufficiently high and if we can obtain good estimates for the time required to complete a check-point, we can time proactive check-points to complete shortly before failures occur, justifying the name *just-in-time*. To guard against underestimating failure probabilities, inter-check-point times can be bounded by forcing check-points at a fixed (low) rate even when very low failure probabilities are predicted.

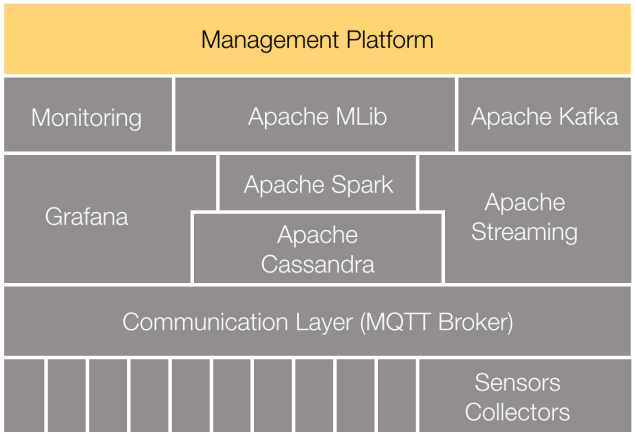

Fig. 4. Streamed data monitoring framework based on open-source software packages including *Apache Spark* for cluster computing, *Apache Cassandra* for database storage, *Apache MLib* for machine learning libraries, *Grafana* for data analytics and visualization, *Apache Streaming* and *Message Queuing Telemetry Transport* (MQTT) for communicating with sensors. The external interface is through *Apache Kafka* Publish/Subscribe service.

As the mean inter-failure time approaches the time required for check-point/restarts, the system begins to spend most of its time check-pointing and restarting instead of performing useful computation. Under these conditions, we must abandon check-point/restart and switch to failure masking through replication as the resiliency mechanism. Current HPDC systems are rich in inherent hardware redundancy in the form of multiple cores, CPUs, MICs and GPUs that can be exploited to achieve replication of jobs without increasing socket counts or costs. The challenge is guaranteeing failure independence among the replicas through intelligent allocation decisions. What are needed are adaptive, hybrid mechanisms that select automatically between just-in-time check-pointing and replication while dynamically adjusting the check-point interval and the number of activated replicas.

Yet another important piece of information that can increase system resiliency is the predicted job duration. For jobs that started recently and that are expected to take a long time to complete, a failure prediction even with modest probability may provoke a migration decision immediately to a safer node. Conversely, for a job that is expected to finish soon, the system may decide to continue its execution on the current node (without any action) even if a significant failure probability is predicted for the node.

C. Predictive System Management

Cognification of HPDC system software is not only essential for sustainability, it is also enables a radically new form of HPDC management. We envision a management platform that is open and extensible with modern social features, driven by predictive models built from streamed data. In the following, we briefly outline our vision for the architecture of such a management platform and describe some interesting usage scenarios.

The platform can be built on top of a *streamed data* monitoring framework, such as the one described in [24]. The framework, illustrated in Figure 4, is based on *Apache Spark* cluster computing and *Apache Cassandra* database open-source software platforms and streams data derived from a collection of sensors at various system levels using *Apache Spark Streaming*. Using the framework involves instrumenting the monitored system with sensors and installing the client layer of the *Message Queuing Telemetry Transport* (MQTT) Publish/Subscribe protocol, while on a service node installing the MQTT broker and *Apache Spark* software packages. The monitoring framework advertises to the world a *Publish* service to which clients can *Subscribe* to receive data pertaining to the sensors they are interested in.

The architecture of the management software platform we envision is illustrated in Figure 5. At the lowest layer it includes a *Publish/Subscribe* interface to be implemented using *Apache Kafka* for communicating with the monitoring framework. The lowest layer also includes a *Streamed Data Filtering* module that implements *Sensor Fusion* functions on data being received from the monitoring framework. The platform uses predictions to project the system into the future and to devise preventive actions based on control mechanisms in case of possible future anomalies. Detecting anomalous behavior is viewed as a big data classification problem and is tackled using DL techniques for recognizing outliers among cluster patterns arising in streamed data from a multitude of origins. Responding to anomalies is also driven by DL technologies inspired by *move generators* in two-person games. A fundamental piece of the architecture is a graphical user interface to implement a *route planner* metaphor not unlike

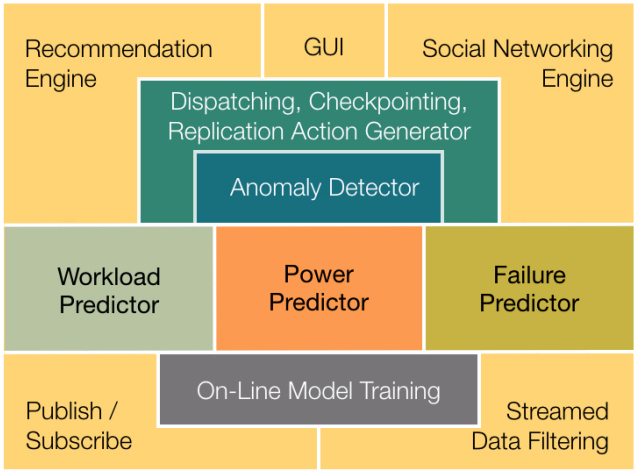

Fig. 5. Predictive system management platform. At the heart of the platform are predictive models for workload, power and failures that are built and trained using on-line streamed data. The *Anomaly Detector* component that sits on top of the predictive models is built using *Databricks Deep Learning Pipelines*, which is an implementation of *Google TensorFlow* technology on *Apache Spark ML Pipelines*. The *Action Generator* component, which is also driven by DL technology, interfaces with the cognified dispatcher and check-pointing systems.

Google Maps or its crowd-sourced counterpart, *Waze*.

We envision two different usage scenarios for this platform. In the first case, a system administrator is presented a global view of the system indicating the actual loads at various system resources (analogs of vehicle traffic on a transportation map) along with other measures such as power consumption, temperatures, queue lengths and job delays. The platform also signals possible anomalous situations in the current system state and suggests preventive actions that can be initiated by the administrator (e.g., check-pointing, replication, migration, consolidation). An innovative aspect of the system is the ability for the administrator to switch to a *crystal ball* view where the system metrics no longer correspond to *now* but reflect some time in the future. In presenting this view, the platform invokes its predictive capabilities for anticipating not only future workloads but also possible failures, along with their likelihood quantified as probabilities. In the second usage scenario, the management service is invoked by an end-user (with appropriate privileges) to guide the execution of her jobs. In this case, the service displays different options for executing a job that she is submitting (analogs of alternate routes for traveling from point A to point B on a map). The options are computed based on predictions of the job's demands along with predictions for future system states (including failures). For each option, she is given estimates for various metrics such as time-to-completion, cost and energy consumed. The time-to-completion estimate takes into consideration probability of failures during the job execution and anticipated system actions such as check-point/restart. Depending on her privileges, the user may be given the option to select an alternative execution path for her job or alternative system responses to potential anomalies, sorted by their "popularity". A recommendation engine powered by DL technology is included to incorporate innovative "social" features and learn from the choices made by other users so as to empower wisdom-of-the-crowd.

V. Remaining Challenges

To realize the vision presented in this paper, further progress in a number of technical areas is necessary. We discuss some of them briefly in what follows.

Prediction: The trend in prediction is towards on-line techniques based on streamed data since they allow not only generating predictions, but more importantly, allow acting on them in real-time [63]. Being able to generate accurate predictions in real-time based on high-volume and high-velocity data that is being streamed is challenging. It requires developing novel methods for on-line training and advanced data management techniques for minimizing the delay when making real-time predictions in live systems. It also requires devising adaptive mechanisms for throttling the volume and velocity of streamed data utilizing feedback from *Machine Learning* algorithms to limit storage needs. Developing an on-line prediction framework and using it to build predictive models with recall over 80% and precision over 90% for classification tasks and error under 5% for regression tasks in HPDC systems remains an ambitious objective.

Resiliency, energy efficiency: Existing software-based techniques for resiliency and energy efficiency in HPDC systems act in isolation as if the two properties are independent, when in fact they are often intertwined [64]. What is lacking is a holistic approach to these problems through a cognified dispatcher that can make intelligent and proactive decisions driven by a suite of predictive models. Only in this manner can we hope to navigate the complex trade-off space between resiliency and energy efficiency. For example, powering nodes on-and-off frequently to save energy can be detrimental for resiliency, while frequent check-pointing for increased resiliency can be detrimental for energy efficiency. How to effectively utilize information provided by different predictive models in making dispatching decisions with multi-dimensional objective functions remains a challenge.

Deep Learning: The vision presented uses DL technologies for predicting HPDC system behavior, for detecting anomalies in HPDC operations and for formulating corrective actions as part of a management platform. Unlike typical DL applications such as image recognition or machine translation that can rely on huge corpuses of data for training, anomaly detection in HPDC systems typically has to contend with smaller training sets since anomalies in HPDC systems, by definition, are rare and generate strong class imbalance problems. An additional issue is the structure of the feature space for sensor data, where many features can have zero values (e.g., resource usage for hybrid HPDC systems). While use of DL for detecting anomalies is relatively straightforward since it can be cast as a classification problem, use of DL for building predictive models requires solving regression problems. Thus, we need to progress beyond the state-of-the-art in DL technologies to develop new techniques by designing training algorithms that do not require huge data sets and that are suitable for applying DL to regression problems. One possibility is to seamlessly apply traditional regression methods on features extracted by under-complete stacked de-noising auto encoders or convolutional deep networks from time series data. Deep networks provide a concise description of the time series, but still enable regression methods to work on more informative, non-linear correlations extracted from raw data.

System management: Current system management tools are limited essentially to resource allocation, scheduling and monitoring. They lack predictive or social capabilities, do not provide control actions and are based on simple interaction models. The vision we presented in this paper goes well beyond the state-of-the-art and takes HPDC management systems to a whole new level by incorporating advanced predictive powers, social features empowering wisdom-of-the-crowd and unprecedented control, all delivered within a software tool featuring a modern user interface and advanced graphical capabilities.

VI. Conclusions

We have presented our vision for cognifying distributed systems in order to render their future development sustainable in terms of scale and computing power. Achieving this vision requires embracing new fields such as data science, machine learning and artificial intelligence. Specifically, data-driven predictive models of system behavior need to be developed for building intelligent systems to operate and maintain future distributed systems. We have identified where and how data analytics needs to be positioned for cognification to have maximal impact on three areas: energy efficiency, resiliency and manageability. For each area, we defined predictive models that can be built and integrated at various levels of the distributed system. These include power, workload and failure predictions that can help cognify job dispatching, fault tolerance mechanisms such as check-pointing, and management tools in general. In the case of management, we sketched the architecture of an intelligent platform that integrates all three components.

We have underlined research directions along with the corresponding challenges that need to be pursued to make our vision a reality. For on-line construction of data-driven predictive models, the challenges are in processing huge amounts of data at high resolution, without imposing significant overhead on the monitored system. Current machine learning methods need to be optimized for on-line use and the scope of novel techniques such as Deep Learning needs to be extended to this application domain. As for building intelligent software for management of distributed systems, what is missing is an integration step where various components are made to act in unison based on models built from data originating at different sources.

We believe that solutions to other well-known problems in distributed computing, such as coordination, consensus and replicated data management, can also benefit from being cognified. Consistent replication of data spanning multiple data centers is a common technique for satisfying the extreme scale and reliability requirements of data managed by the likes of Google [65] and Amazon [66]. Consensus is an important abstraction for guaranteeing replica consistency [67]. It is well known that consensus cannot be solved in asynchronous distributed systems without a form of *failure detector* [68]. Failure detection is also critical for determining the performance of the resulting solutions for consensus [69]. An interesting open question is the relation between our predictive models for failures and failure detectors that have been proposed in the context of consensus. We believe that being able to *anticipate* failures in advance, and not just *detect* them when they occur, can prove to be very useful in devising better consensus protocols.

ACKNOWLEDGMENT

A. Sîrbu acknowledges support from the European Union project *SoBigData Research Infrastructure — Big Data and Social Mining Ecosystem* under the INFRAIA-H2020 Program (Grant Agreement 654024). BigQuery analysis was carried out through a generous Cloud Credits grant from Google.

References

[1] S. Ashby, P. Beckman, J. Chen, P. Colella, B. Collins, D. Crawford, J. Dongarra, D. Kothe, R. Lusk, P. Messina *et al.*, "The opportunities and challenges of exascale computing," *Summary Report of the Advanced Scientific Computing Advisory Committee (ASCAC) Subcommittee*, pp. 1–77, 2010.

[2] K. Kelly, *The inevitable: understanding the 12 technological forces that will shape our future*. Penguin, 2017.

[3] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *International Conference on High Performance Computing for Computational Science*. Springer, 2010, pp. 1–25.

[4] O. Villa, D. R. Johnson, M. Oconnor, E. Bolotin, D. Nellans, J. Luitjens, N. Sakharnykh, P. Wang, P. Micikevicius, A. Scudiero *et al.*, "Scaling the power wall: a path to exascale," in *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*. IEEE, 2014, pp. 830–841.

[5] A. Bartolini, M. Cacciari, C. Cavazzoni, G. Tecchiolli, and L. Benini, "Unveiling eurora—thermal and power characterization of the most energy-efficient supercomputer in the world," in *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*. IEEE, 2014, pp. 1–6.

[6] F. Cappello, A. Geist, W. Gropp, S. Kale, B. Kramer, and M. Snir, "Toward exascale resilience: 2014 update," *Supercomputing frontiers and innovations*, vol. 1, no. 1, pp. 5–28, 2014.

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

[8] S. Hukerikar and C. Engelmann, "Resilience design patterns: A structured approach to resilience at extreme scale," *arXiv preprint arXiv:1708.07422*, 2017.

[9] W. M. Jones, J. T. Daly, and N. DeBardeleben, "Application monitoring and checkpointing in hpc: looking towards exascale systems," in *Proceedings of the 50th Annual Southeast Regional Conference*. ACM, 2012, pp. 262–267.

[10] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson *et al.*, "Addressing failures in exascale computing," *The International Journal of High Performance Computing Applications*, vol. 28, no. 2, pp. 129–173, 2014.

[11] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[12] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of hpc applications," in *Proceedings of the 22nd annual international conference on Supercomputing*. ACM, 2008, pp. 175–184.

[13] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in *Proceedings of the 1st International Conference on energy-Efficient Computing and Networking*. ACM, 2010, pp. 215–224.

[14] M. Shojafar, C. Canali, R. Lancellotti, and S. Abolfazli, "An energy-aware scheduling algorithm in dvfs-enabled networked data centers." in *CLOSER (2)*, 2016, pp. 387–397.

[15] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *ACM SIGOPS operating systems review*, vol. 37, no. 5. ACM, 2003, pp. 164–177.

[16] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[17] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.

[18] A. Borghesi, F. Collina, M. Lombardi, M. Milano, and L. Benini, "Power capping in high performance computing systems," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2015, pp. 524–540.

[19] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: adaptive dvfs and thread packing under power caps," in *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*. ACM, 2011, pp. 175–185.

[20] P. Baptiste, C. Le Pape, and W. Nuijten, *Constraint-based scheduling: applying constraint programming to scheduling problems*. Springer Science & Business Media, 2012, vol. 39.

[21] P. H. Hargrove and J. C. Duell, "Berkeley lab checkpoint/restart (blcr) for linux clusters," in *Journal of Physics: Conference Series*, vol. 46, no. 1. IOP Publishing, 2006, p. 494.

[22] G. Bosilca, A. Bouteiller, E. Brunet, F. Cappello, J. Dongarra, A. Guermouche, T. Herault, Y. Robert, F. Vivien, and D. Zaidouni, "Unified model for assessing checkpointing protocols at extreme-scale," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 17, pp. 2772–2791, 2014.

[23] F. Cappello, H. Casanova, and Y. Robert, "Checkpointing vs. migration for post-petascale supercomputers," in *Parallel Processing (ICPP), 2010 39th International Conference on*. IEEE, 2010, pp. 168–177.

[24] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools," in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 1038–1043.

[25] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015, p. 18.

[26] T. Hey, S. Tansley, K. M. Tolle *et al.*, *The fourth paradigm: data-intensive scientific discovery*. Microsoft research Redmond, WA, 2009, vol. 1.

[27] N. Cristianini, "Intelligence reinvented," *New Scientist*, vol. 232, no. 3097, pp. 37–41, 2016.

[28] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[29] S. S. Bucak, R. Jin, and A. K. Jain, "Multiple kernel learning for visual object recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1354–1369, 2014.

[30] R. Fakoor, F. Ladhak, A. Nazi, and M. Huber, "Using deep learning to enhance cancer diagnosis and classification," in *Proceedings of the International Conference on Machine Learning*, 2013.

[31] H. R. Varian, "Big data: New tricks for econometrics," *The Journal of Economic Perspectives*, vol. 28, no. 2, pp. 3–27, 2014.

[32] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8599–8603.

[33] L. van der Maaten, P. Boon, G. Lange, H. Paijmans, and E. Postma, "Computer vision and machine learning for archaeology," *Proceedings of Computer Applications and Quantitative Methods in Archaeology*, pp. 112–130, 2006.

[34] J. Hutchins, "Example-based machine translation: a review and commentary," *Machine Translation*, vol. 19, no. 3, pp. 197–211, 2005.

[35] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, "Deep learning with cots hpc systems," in *International Conference on Machine Learning*, 2013, pp. 1337–1345.

[36] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[37] J.C. McCullough et al., "Evaluating the effectiveness of model-based power characterization," in *USENIX ATC'11*, vol. 20, 2011.

[38] W. Dargie, "A stochastic model for estimating the power consumption of a processor," *IEEE Trans Comput.*, vol. 64, no. 5, pp. 1311–1322, 2015.

[39] P. Gschwandtner et al., "Modeling CPU energy consumption of HPC applications on the IBM Power7," in *PDP'14*, 2014, pp. 536–543.

[40] X. Ma et al., "Statistical power consumption analysis and modeling for GPU-based computing," in *ACM SOSP HotPower'09*, 2009.

[41] M. Witkowski et al., "Practical power consumption estimation for real life HPC applications," *Future Gener Comput Syst.*, vol. 29, no. 1, pp. 208–217, 2013.

[42] H. Nagasaka et al., "Statistical power modeling of GPU kernels using performance counters," in *IGCC'10*, 2010, pp. 115–122.

[43] H. Shoukourian and T. Wilde, "Predicting the Energy and Power Consumption of Strong and Weak Scaling HPC Applications," *Supercomp Front Innov.*, vol. 1.2, 2014.

[44] C. Storlie, Curtis et al., "Modeling and predicting power consumption of high performance computing jobs," *arXiv preprint arXiv: 14125247*, 2014.

[45] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Predictive modeling for job power consumption in hpc systems," in *International Conference on High Performance Computing*. Springer, 2016, pp. 181–199.

[46] A. Sîrbu and O. Babaoglu, "Power consumption modeling and prediction in a hybrid cpu-gpu-mic supercomputer," in *European Conference on Parallel Processing*. Springer, 2016, pp. 117–130.

[47] J. Gao and R. Jamidar, "Machine learning applications for data center optimization," *Google White Paper*, 2014.

[48] A. Sîrbu and O. Babaoglu, "A data-driven approach to modeling power consumption for a hybrid supercomputer," *Concurrency and Computation: Practice and Experience*, 2018.

[49] J. Ejarque, A. Micsik, R. Sirvent, P. Pallinger, L. Kovács, and R. M. Badia, "Semantic resource allocation with historical data based predictions," in *Cloud computing 2010. The First international conference on cloud computing, GRIDs, and virtualization*. Lisbon: IARIA, November 2010, pp. 104–109.

[50] X. Chen, C.-D. Lu, and K. Pattabiraman, "Predicting job completion times using system logs in supercomputing clusters," in *Dependable Systems and Networks Workshop (DSN-W), 2013 43rd Annual IEEE/IFIP Conference on*. IEEE, 2013, pp. 1–8.

[51] C. Galleguillos, A. Sîrbu, Z. Kiziltan, O. Babaoglu, A. Borghesi, and T. Bridi, "Data-driven job dispatching in hpc systems," in *International Workshop on Machine Learning, Optimization, and Big Data*. Springer, 2017, pp. 449–461.

[52] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, "Failure prediction for hpc systems and applications: Current situation and open issues," *The International Journal of High Performance Computing Applications*, vol. 27, no. 3, pp. 273–282, 2013.

[53] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Computing Surveys (CSUR)*, vol. 42, no. 3, pp. 1–68, 2010.

[54] T. Samak, D. Gunter, M. Goode, E. Deelman, G. Juve, F. Silva, and K. Vahi, "Failure analysis of distributed scientific workflows executing in the cloud," in *Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualiztion management (svm)*. IEEE, 2012, pp. 46–54.

[55] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in *32nd IEEE Symposium on Reliable Distributed Systems (SRDS)*, Braga, Portugal, Sep. 2013, pp. 205–214.

[56] A. Gainaru, M. S. Bouguerra, F. Cappello, M. Snir, and W. Kramer, "Navigating the blue waters: Online failure prediction in the petascale era," *Argonne National Laboratory Technical Report, ANL/MCS-P5219-1014*, 2014.

[57] A. Sîrbu and O. Babaoglu, "Towards operator-less data centers through data-driven, predictive, proactive autonomics," *Cluster Computing*, vol. 19, no. 2, pp. 865–878, 2016.

[58] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, Posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[59] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Obfuscatory obscanturism: making workload traces of commercially-sensitive systems safe to release," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 1279–1286.

[60] J. Tigani and S. Naidu, *Google BigQuery Analytics*. John Wiley & Sons, 2014.

[61] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010.

[62] T. M. Khoshgoftaar, M. Golawala, and J. Van Hulse, "An empirical study of learning from imbalanced data using random forest," in *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, vol. 2. IEEE, 2007, pp. 310–317.

[63] P. V. Hentenryck and R. Bent, *Online stochastic combinatorial optimization*. The MIT Press, 2009.

[64] O. Sarood, E. Meneses, and L. V. Kale, "A 'cool' way of improving the reliability of hpc machines," in *High Performance Computing,*

Networking, Storage and Analysis (SC), 2013 International Conference for. IEEE, 2013, pp. 1–12.

[65] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.

[66] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *ACM SIGOPS operating systems review*, vol. 41, no. 6. ACM, 2007, pp. 205–220.

[67] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, 1998.

[68] T. D. Chandra, V. Hadzilacos, and S. Toueg, "The weakest failure detector for solving consensus," *Journal of the ACM (JACM)*, vol. 43, no. 4, pp. 685–722, 1996.

[69] N. Sergent, X. Défago, and A. Schiper, "Impact of a failure detection mechanism on the performance of consensus," in *Dependable Computing, 2001. Proceedings. 2001 Pacific Rim International Symposium on*. IEEE, 2001, pp. 137–145.