CSR:Large: VarSys: Managing variability in high-performance computing systems

1 Introduction

Exascale, high-performance computing (HPC) systems are required to meet the demands of many grand challenges for scientific computing and e-commerce in the 21st century [142,210]. Performance variability[1] in high-performance systems is growing intractably and leads to performance loss and energy waste [28,63, 80, 145]. Such variability is cited as a significant barrier to exascale computing [123, 170]. Unfortunately, variability is both ubiquitous and elusive as its causes pervade and obscure performance across the systems stack from hardware [8, 26, 98, 158, 161] to middleware [11, 73, 143, 172, 203] to applications [80] to large-scale systems [123, 140, 168, 170, 197].

We propose to build a software framework (VarSys) to identify, characterize, isolate and manage variability in high-performance parallel and distributed systems. Our overarching goal is to develop fundamental methodologies that exploit our improved understanding of performance variability in complex computing systems. We will use the VarSys framework to improve HPC experimental design; improve cloud efficiency and services; and improve malware detection. We plan community outreach activities that include a Jitter500 ranking of systems by variability (akin to the Top500) and a hackathon of student-team battles involving creation and elimination of system performance variability.

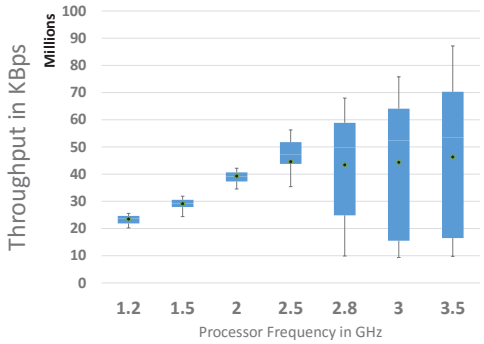


Figure 1. Box and whisker plot of I/O throughput vs. processor speed for a series of small, intensive write operations (64 threads of IOZone benchmark) from a lone guest Linux operating system (Ubuntu 14.04 LTS over XEN 4.0) with a dedicated 2TB HDD on an Intel (Haswell) platform.

1.1 Motivation and Approach

Variability is a persistent challenge in HPC experimental systems research that becomes intractable at exascale. Highly variable observations in extremely complex systems can make it difficult or impossible to fully optimize for performance. Additionally, as a number of recent reports attest [123, 170], performance variability at scale can significantly reduce performance and energy efficiency [28, 63, 80, 145]. For example [123], technologies that maximize processor speed within thermal envelopes can cause slowdowns due to additional load imbalance in a system. Similarly, resilience techniques triggered by hardware and software failures can cause variability and load imbalance that contributes to performance loss. In recent work, we have shown that default systems software parameter settings (e.g., I/O journaling) can significantly affect system performance variability [49, 50] and result in slowdowns of more than 50%.

To illustrate the impact of variability on system performance, Figure 1 shows an example of I/O performance variability in a virtualized environment. We obtained similar results for the same system without virtualization [49]. The figure shows a box and whisker plot of static processor frequencies (x-axis) versus achieved throughput in KB/s (y-axis). Notice that for higher frequency settings (e.g., 3.0 and 3.5 GHz), the variability is high and the standard deviation can exceed 50% of the mean. In contrast, lower frequency settings result in lower variability.

Figure 1 shows that system settings, such as processor scaling depicted along the x-axis, affect performance (e.g., mostly increasing from left to right in Figure 1). However, the figure also shows that lower frequencies result in more stability (i.e., less variability) in the achieved performance. The hypothesis of this proposal is that with a deeper understanding of the relationship between system configurations (e.g., processor frequency) and the resulting performance variability (e.g., expressed as a probability density function

[1] We use the term *performance variability* to describe the general spread or clustering of a measured performance metric such as execution time or throughput. Variability can be expressed in standard statistical terms such as standard deviation. Or in finer detail, we can describe variability as the probability density function (PDF) of the measured performance metric.

or PDF), we can manage the amount of variability on a system. The proposed Variability System (VarSys) Software Framework will enable *variability management*[2] in high-performance computing systems.

1.2 Research Overview

We believe that knowledge and management of variability across the system stack can be harnessed to address significant challenges facing exascale computing. Therefore, in this project we will demonstrate that the VarSys framework can improve the design and operational efficiencies of both HPC and cloud systems. Furthermore, to highlight the broader impact of VarSys beyond systems research, we will demonstrate use of variability management to improve malware detection.

HPC Systems Variability will significantly limit performance at exascale if it is not addressed [123]. The growing scale and complexity of commodity hardware and software contributes to variability and challenges our ability to rigorously examine and improve systems designs. Experimental HPC systems research uses sound methods generally, but often relies upon experiential techniques to address variability. Such techniques include disabling subsystems not under study, discounting outliers, using minimum average times, etc. These techniques may work well on one system and not hold true on another. They make work well initially but fail with increases in scale and complexity. And these techniques often fail when the system under study itself exhibits high variability. In many cases, such techniques have been handed down from researcher to apprentice over many years. This makes it often impossible to discern why the methods work or why they do not. Furthermore, the experimental systems researcher, intent on proving out their new technology, has little incentive to stop and question their methods deeply since many are seemingly intuitive. We propose to use variability management to improve our ability to conduct experimental systems research. Variability management promises not only to reduce our problem design space, but to explain how and why the design space can be reduced without compromising our experimental methods. The VarSys framework will additionally enable study of the use of variability management to improve the design of runtime systems that adapt resource usage to improve HPC efficiency.

Cloud services. Where exascale, high-performance systems focus on improving raw performance, large-scale clouds operated by Amazon and others seek to provide a guaranteed level of service to a client. This generally involves using past information and prediction of the probability distribution of requests to match resources that meet service-level agreements. In preliminary work for this proposal, we have observed tradeoffs between the number and type of virtual machine hosts on a system and the performance variation. The VarSys framework provides additional information about the performance variation expected from the available resources. For example, while a queuing system may predict arrival and service rates to determine worst-case service levels, VarSys can provide service rate variability information (Figure 1 provides an example data set) to recommend system settings that enable tighter bounds on service level estimates. This type of resource management provides new opportunities for tradeoffs between system stability and performance. This could lead to new cloud cost structures as tighter bounds on variability could be codified in cost structures bundled in operations or directly offered to clients.

Malware detection. The VarSys framework provides a platform that can dampen or increase variability at runtime. This is a potentially transformative technology for techniques that must operate effectively in high variability environments. For example, several types of malware detection, leverage the expected performance signature of a system [119]). Detecting attacks can involve matching known malware signatures to observed signatures [202]) or identifying deviations from normal system signatures [174]). The effectiveness (i.e., limited false positives and false negatives) of these types of techniques depends heavily on their ability to identify malware in systems with high variability. VarSys variability management can potentially increase the effectiveness of these types of malware detection techniques by creating conditions that reduce the amount of observed variability or through creation of techniques that adapt to current conditions.

[2]To be clear, we want to manage variability tradeoffs. Fig. 1 shows clear tradeoffs exist between raw, measured performance and performance variability. In our approach, we will create a framework that enables prediction of probability density functions and then use the framework to manage variability.

would reducing system variability not reduce the malware's variability also?

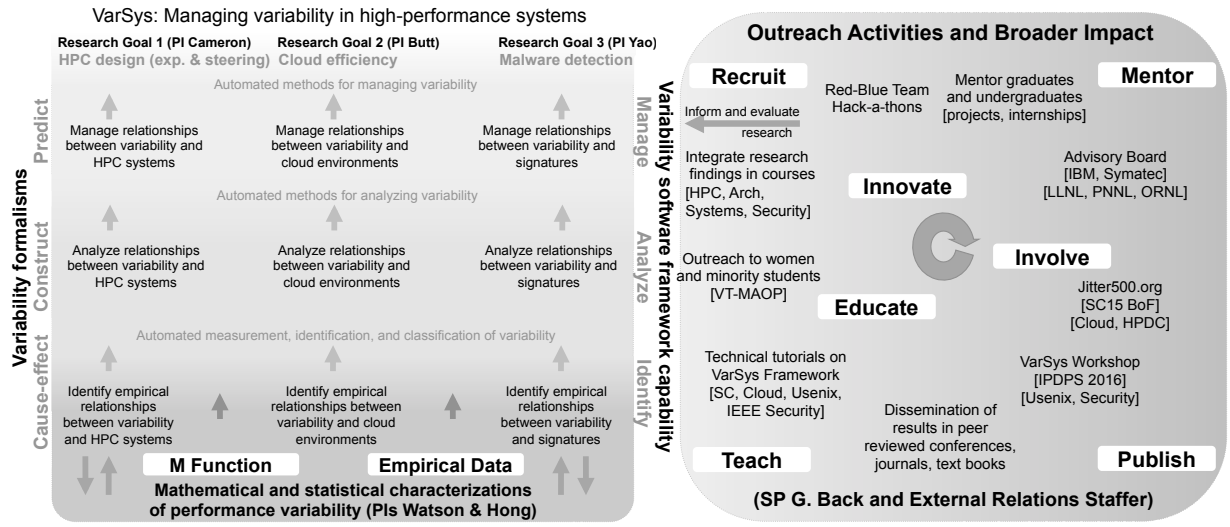


Figure 2. An integrated research, education and outreach plan.

1.3 Broader Impact of the Proposed Activity

Jitter500.org The variability challenge is real and growing. Despite the ambitious goals of the proposed work, variability permeates the entire systems stack and affects nearly every aspect of computing. Thus, addressing variability will ultimately take the concerted efforts of a community. To this end, we will work to improve awareness and track community progress with a Jitter500 List[3] (akin to the Top500) that ranks systems by their performance variability. The PI will leverage his experience creating the Green500 List and the proposed VarSys framework to establish methodologies for investigating HPC system variability.

Red-Blue Hackathons The variability challenge will require significant innovation and inspiration from the next generation of computer scientists. In this proposal, we hypothesize that performance variability can be managed; determining the manageability and the effects of management are primary contributions of the proposed work. To involve and excite students in this project and the variability challenges facing the community, we plan to create Red-Blue Team Hackathons. Student teams will compete in controlled environments that leverage the VarSys framework to create system variability (Red Teams) and to dampen system variability (Blue Teams). These efforts can be used to test many of the aforementioned research efforts as well as potentially reveal innovative techniques for adoption in the research effort. Most of all, the hackathons will be fun tools to engage and actively recruit students to the project.

1.4 Intellectual Merit of the Proposed Activity

Figure 2 shows ours plan of integrated research and educational outreach to build a Variability System (VarSys) Software framework that identifies, characterizes, isolates and manages variability in high-performance parallel and distributed systems. The intellectual challenges include establishing a rigorous methodology for characterization of performance variability in advanced computer systems. We will also determine the extent to which variability can be managed and the tradeoffs therein. We will demonstrate the use of variability management in two of the three CSR Highlighted Areas (Clouds and EDS) and as a technique to address a cybersecurity grand challenge [142, 210], malware detection.

2 Research Component

The proposed research was initially motivated by experiences designing next generation software to improve the efficiency of HPC systems. In recent work [47], the PIs showed that slowing processor speeds often led to faster performance. Somewhat unexpectedly, the PIs also observed that using dynamic runtime system parameterization to improve performance had the side effect of ~~X~~ altering the probability density function of the observed performance variability. Figure 1 shows just one example of this phenomenon.

[3] This name may change based upon community feedback. As a precaution, we have secured the domain rites to Jitter500.org , Var500.org, and VarSys.org.

A key supposition of our work is that system variability (for a given system specification) can be characterized as one of a range of possible probability density functions. We further hypothesize that if we can construct the mapping of parameter settings to changes in the variability probability density functions, this variability *mapping function* could form the basis of techniques that can manage variability in advanced computing systems.

Direct measurements described in an empirical variability mapping function are somewhat analogous to the sensor data used by engineers to inform the design decisions of planes and rockets. For example, in aerospace engineering [37], new planes or rocket designs are often premised on empirical measurements. Since the new design is not physically available to the engineers, they work closely with mathematicians and statisticians to use empirical measurements from previous designs to interpolate or extrapolate the expected performance of the new design.

In our work, we plan to use engineering principles to enable variability management of complex, high-performance computing systems. Leveraging empirical measurements for various system specifications (or input signatures), we will create *variability maps* to enable rudimentary variability management. We then will work closely with mathematicians and statisticians to use these maps from an existing system to interpolate or extrapolate (i.e., predict) the variability expectations for other parameter settings or new system designs.

If successful, this research will result in a novel, effective approach to managing variability in advanced computing systems. The resulting techniques will be fundamental and potentially transformative in nature. We will demonstrate the promise of our techniques for improving the efficiency of clouds and extensible distributed systems — two of the three CSR highlighted areas — in addition to HPC design and malware detection.

We begin with related research to identify the context and importance of the problems that we must solve. This includes motivating the need for variability management and identifying how current approaches to address variability fail to meet the needs of high-performance and cloud environments. We describe the proposed research, preliminary results, working theories, and open issues.

2.1 Background and Significance

2.1.1 Variability in HPC Systems

Given the origins of this work in HPC systems, we begin with an overview of techniques addressing variability primarily in HPC environments. HPC systems research is not immune to the current trend of criticisms of presentation of scientific results of rigor. Hoefler et al. [85] recently summarize the state of the practice for benchmarking in HPC and suggest ways to ensure repeatable results.

Despite the recent kerfuffle, HPC researchers have been examining variance for a long time, e.g., in the 1990s IBM observed variance in uniprocessors [141], and Kramer et al. explored variation in large distributed memory systems more than a decade ago [102]. Such well-cited studies establish the existence of variability, yet variability is not typically factored into modern HPC application design.

Much of the work emanating from the HPC community has been focused on OS jitter [145] or variations caused by the competition for resources between background processes and applications. Some have simulated these effects at scale [63] while others have proposed applications [80] or systems [28] that can account for jitter.

A number of tools and techniques have been proposed to measure and analyze jitter at scale [28, 62, 86, 140, 168]. OS performance behavior [172, 203] has been used to discern the causes of variations like jitter. Much of this work tends to be platform dependent [197] with the main goal of eliminating jitter due to the operating system [62].

Other research acknowledges that operating system processes that cause jitter are often unavoidable and potentially remedied with architectural support [46]. Some have taken to reducing the capabilities of the operating system kernels to create light weight versions that don't exhibit as much jitter [11, 73, 143]

Schedulers are often identified as causes of significant variability in HPC systems [182]. This includes introducing determinism in thread [149] and I/O management [120].

At the heart of any runtime system lies an analytical engine that uses some form of predictor to determine whether and how to adapt the systems use of resources. Performance prediction of HPC and distributed applications is a well-studied field and recent works have used analytical [60, 181, 186], profile-based [32, 166], and simulation-based [45, 148] approaches, or a combination of these [59, 212], to accurately predict overall performance.

In contrast, similarly detailed studies that result in models or predictors that consider variability are almost nonexistent. As discussed earlier, much of this work focuses on eliminating the variability due to a particular cause, e.g., [121] tries to manage variability in the I/O performance of petascale storage systems.

2.1.2 Reproducibility and Repeatability

Studies aimed at improving our ability to verify experimental results can directly or indirectly address variability. For example, recent studies have shown that minor aspects of experimental setups can have a significant impact on system performance. The high degree of sensitivity to experimental design can potentially invalidate conclusions. This is further complicated by lack of reproducibility, corroboration and repeatability, thereby, propagating invalid conclusions in academia [187]. Such complications delay the adoption of academic results, e.g., in industrial systems. There is a growing need to perform deep analyses to identify the root causes of observed results. Failure to reproduce a result can hinder identifying an issue if one exists [68].

Introducing determinism to achieve reproducibility has also been explored using environment categorization, [162], statistical modeling [92, 176], or variance-aware algorithm design [25]. Environment categorization considers the interactions and composition of hidden factors in the system such as the open benchmarking infrastructure for performance evaluation, DataMill [64]. Specifying high-level components [7] can further improve the accuracy of such approaches. However, the Collaborations Workshop for Software Engineering [180] stated recently that further understanding of the process of artifact evaluation (which includes variability) is required. Our focus on managing variability and the application to experimental design complements these approaches and may influence such best practices.

2.1.3 Variability in Real Time Operating Systems

Runtime predictability is a key concern in Real-time Operating System (RTOS) design, where mission critical application deadlines have to be met with high accuracy. As such, the proposed work shares the goal of achieving similar runtime performance and behavior guarantees with RTOSs. However, a key differentiater in this context is that RTOSs sacrifice throughput/performance for predictability [24, 164]. As high performance is the main concern in our work, the RTOS approach can not be simply used in our HPC application use cases because: (i) In RTOSs, only a limited number of tasks can run at the same time, hence limiting the achievable level of multitasking [9, 127]. HPC and cloud systems of scale routinely handle hundreds of thousands to millions of threads. (ii) By limiting the supported set of real-time threads and applications, such systems tend to have a higher ratio of resources to demands in order to meet deadlines, i.e., cores, memory and networking bandwidth are scaled to meet real-time design requirements. Applying these same principles to HPC, would likely be prohibitively expensive [9, 61, 127]. (iii) The algorithms and scheduling techniques at the heart of a RTOS are primarily designed to meet deadlines. Many general purpose applications common in HPC environments, do not benefit from the same types of optimizations. [9, 179]. (iv) RTOS interfaces are unfamiliar to most HPC researchers and would require significant retraining for the typical HPC researcher to use effectively [9, 100].

RTOSs also primarily focus on eliminating variability as opposed to managing it [27, 109]. In fact, different RTOSs are tuned to account for performance throughput, jitter, and the predictability of task performance [23, 34, 124]. Rather than react (or adjust for) variability, in the proposed work we will attempt to affect variability and ultimately to manage it meet users needs. Additionally, as our approach is fundamental, it is very likely that the mathematical and statistical formulations developed could be adopted for us in

real-time operating systems. Exploring the effectiveness of our techniques in this domain would fall outside the scope of the proposed work however.

2.1.4 Variability in Computer Architecture

A number of projects in computer architecture research have explored variability in studying and realizing new hardware, arising mainly from heterogeneity in the chip-level architecture. These projects are mainly focused on the consequences of the chips having a limited amount of power [26, 33, 98, 161, 201]. The resulting issues such as leakage current, cross-wire-signaling, and real-estate budgeting, result in expected performance to vary significantly. If these techniques affect software performance variability, then they can be captured and are orthogonal to our proposed work. Otherwise, such variations are likely to be small relative to the variations observed much higher in the systems software stack. As in the RTOS discussion, the fundamental nature of our approach could be applied to the programmatic design aspects of the chip such as the mapping of on-chip decisions to reduce power and the resulting affect on the probability density of the observed variations. Exploring the effectiveness of our techniques in this domain would fall outside the scope of the proposed work however.

Variability-aware computing Recent high-profile research has focused on the concept of variability-aware computing [8], wherein architecture-level variability is captured and controlled to guarantee hardware-level performance. This project considers variability from the point of memory, hardware, and storage systems. Variability-aware memory systems [65] introduce the concept of variability-aware management for nanoscale computing systems and show great benefits in making the software stacks variation-aware. In such systems, timing errors caused by manufacturing and environment variations are typically avoided by conservative design and circuit-level error correction. However, such approaches inflict performance and energy penalties [158]. Similarly, variance due to energy-efficient hardware [77] has also been studied for its impact on application performance.

2.1.5 Other Areas and Open Problems

The astute reader undoubtedly has considered areas not discussed in detail in this section. Studies of variability are nearly as old as computing [29]. We left out discussion of variability in accelerators [122, 159, 160], networking [165] and queuing theory [66] for example. Studies of variability tend to be area specific and we've tried to name those most closely aligned to our challenge of addressing variability at exascale. As our research progresses, we will undoubtedly look to some of these other areas to bring to bear their solutions on our problems.

Variability at exascale is clearly an open problem as evidenced by its inclusion in a 2014 list of Top 10 Exascale Research Challenges published by the DOE Office of Science [123]. Variability challenges in HPC pervade every aspect of the systems stack and require a holistic solution. The scale, complexity, and heterogeneity of large-scale systems require fundamental solutions that are applicable to a broad class of systems and applications.

2.2 Research Description

In our literature survey, we found no single solution that is holistic enough to solve variability issues across the system stack, fundamental enough to maintain platform independence, and scalable enough to withstand the challenges of exascale. Our approach begins with empirical, holistic measurement of system performance data of any type at any granularity. We then apply advanced statistical methods and construct mathematical formalisms using engineering approaches that have been proven effective at modeling extremely high order physical systems (e.g., rockets [37]). The ability to analyze and predict at high orders is uniquely suited to constructing performance variability probability density functions for systems of scale. We refine and test multiple prototypes of the VarSys software framework to manage variability with increasing levels of sophistication (i.e., fidelity and prediction accuracy) across three platform domains (HPC, cloud, and security). The remainder of this section describes the design considerations and challenges in the creation of the VarSys Framework.

is this w/ a ML technique?

2.2.1 Research Goal 1: HPC Systems Design

Experimental results with high variability are problematic. If the goal is to evaluate the performance of the system, increasing the number of experiments to achieve an acceptable statistical significance is required. Unfortunately, in computer systems research, the number of variables and experiments grows very large very quickly. Even after leveraging principal component analysis [94], in exascale HPC systems, the number of threads and components requires investigating thousands to millions [114] of scenarios that could take hundreds of thousands of CPU-hours to evaluate. This evaluation time to obtain statistical confidence in the results due to high variability could exceed the production life of the HPC systems under test. While there are ad hoc and rule-of-thumb methodologies for reducing the number of experiments needed, such methods tend to be task [200] or system [171] specific.

The variability management aspect of the VarSys Framework will provide the ability to cull the design space of HPC systems experiments. The mathematical and statistical foundations of our methods provide a description of the cause-effect relationships between system parameters and the resulting probability density functions. We plan to leverage our extensive work developing performance and power measurement tools [70] and the plethora of open source tools [36, 173] available to provide variability profiling software to this module and the HPC and security modules shown in the software architecture description of VarSys (see Figure 3). As our understanding of variability grows we will use the VarSys framework to reduce the problem space for HPC systems experimentalists, with an understanding of the consequences of experimental design choices.

We must work closely with our mathematical and statistical counterparts to develop protocols for packaging the two primary data structures needed: input signatures, and variability probability density functions. We will leverage techniques and lessons learned creating the PowerPack framework [70] that has been used by more than two dozen research groups worldwide. Our preliminary work for this proposal [50] provides us with a manual process for correlating system parameterizations with performance. We will extend these to capture variability automatically.

We will use the resulting Beta Phase I system shown in Figure 3 to create an empirical version of the memory map, which is a rudimentary approximation of the model of probability density functions that is the goal of the mathematical and statistical efforts. While creation of the measurement infrastructure is time consuming but straight forward, correlating input signature data to variability is without precedent. We will leverage what we learned tracing the cause of performance slowdowns [50] to develop systematic methods for creating meaningful input signatures that capture system specifics that correlate to significant changes in the probability density function. As the project progresses these efforts will get more difficult as we attempt to increase the fidelity and accuracy of the early versions of the set of probability density functions to exascale system variables. As things get more complicated, we will attempt to lean on engineering best practices [37] to determine research directions. → what are some of these?

In parallel to the development of Beta Phase I, we will refine our prototype to introduce automation in the system measurement and ideally in the identification of the input signatures. This will require developing microbenchmarks to aid in isolating the effects of variability. We will also need to establish methodologies for implementing the Jitter500 ranking of systems by variability. We plan to combine these microbenchmarking techniques with our experience creating the Green500 List [58] and SPECPower [110] methodologies to create a methodology for the Jitter500. Our current thinking is to create (or leverage existing [110]) frameworks that enable iterative benchmark measurements. These could be built upon slowly by extending the framework to include additional benchmark suites. This avoids the effort of maintaining and marketing a new benchmark allowing us to focus on capturing and analyzing the effects of variability. This effort will also become more challenging and potentially impactful with each successive generation of the VarSys Framework.

We also plan to integrate the variability map formalisms in runtime systems to improve performance and energy optimization methods. We have a good deal of experience optimizing with this topic [69] but our

previous methods (and all others we know of) failed to compensate for variability. We will use our previous work as a starting point for variability-conscious approaches to runtime adaptation. Should there be time and resources left in the project, we will consider investigating the affect variability management could have on thread scheduling [114] and similar topics.

2.2.2 Research Goal 2: Improving Cloud Efficiency

Measuring Variability in Cloud Environments Cloud computing environments comprise a complex array of compute, storage, networking, and I/O components. That coupled with the on-demand nature of cloud services, instantiation may result in unpredictable performance when utilizing cloud-based services. Recent studies [22, 167] have shown that performance unpredictability acts as one of the major obstacles for cloud computing. Cloud users expect consistent performance for their applications, independent of the underlying architecture or current workload of the cloud; this is important to predict the overall cost associated with providing a cloud based service. As more and more users build their customer-facing services using cloud-based backend components, the performance consistency requirements are becoming paramount.

Both the current load on a cloud as well as the underlying architecture used for launching a cloud instance play an important role in defining the performance of a cloud instance launched on that setup, e.g., [167] shows that MapReduce jobs perform better on EC2 when using a larger percentage of Xeon-based systems than Opteron-based systems. Similarly [101] identifies the problem of performance variability in their study, however solutions to address the problem or measure the variability have not been developed.

A couple of unique facets of the performance variability in the cloud is the multitenant use of the shared resources, and that different cloud service providers utilize different resource scheduling and allocation mechanisms. Thus, the impact on the performance of different workloads is different for different cloud providers. For example, in a multitenant environment, priority based scheduling [112, 113] chooses to offer more resources for workloads with higher priority, which may affect the performance of a certain workload. On the other hand, an environment equipped with load-balancing scheduling [78, 90, 185] may affect the same workload differently, e.g, loss of data locality. Such different scheduling choices and their impact on performance variability needs to be studied in detail to quantify the impact.

Similarly, cloud storage offers customers the key benefits of on-demand elastic scalability and usage-based pricing. However, variation in the performance of cloud storage services can lead to cloud applications violating their SLAs [195]. The main source of variability in cloud storage performance has been identified to be interference from co-located tenants. As the logical partitioning between different tenants data does not map to separate physical partitions, applications from different tenants could contend for the same disk resources, which result in lower overall IOPS for both of the co-located tenants [91, 175]. In addition, even if cloud storage services achieve even distribution of data across their deployed hardware, the skew in the demand of individual data objects will result in unfairness in the usage of storage devices [175]. We aim to study and quantify such impact and identify key factors affecting the performance. ~ of co-located hosts

Leveraging Variability for Cloud-based Service Design The cloud providers need to comply with their customers SLAs by scaling up their setup according to the load on the systems [21]. If we can accurately measure variability then this information can be useful for the cloud providers to offer tighter performance-based SLA guarantees to users, and hence scale their setup in a more cost-aware manner [101]. To this end, we plan to design several proof-of-concept cloud-based storage and instantiation services, wherein we will study how SLA and scaling can be achieved in such services.

A recent study [52] shows a clear trade-off between latency and cluster utilization for MapReduce applications in cloud environments. Such trade-off is caused by different and sometimes conflicting optimization goals of the application developers and cloud service providers. Different cloud environment variability yields new conflicts that might influence application performance in different ways. On the other hand, the diversity of MapReduce usage scenarios makes it hard to develop a single solution for all applications and environments. With the knowledge of variability measurement, an application could choose the cloud based

on its characteristics. For example, end-user services that are susceptible to performance variability, such as video streaming services, high-speed trading, etc., would choose more stable environments. At the same time, users on a tighter budget can make a more informed decision about whether to employ cheap services such as spot cloud instances, if the variability characteristics of such instances are known (or can be readily determined). We envision that our variability capturing models will enable cloud users in achieving such goals.

Similarly, to mitigate the impact of variability on cloud storage performance, tenants resort to employing redundant resources via replication and additional layers of load distribution to minimize the effect of this variability on the IOPS achieved by the applications [196]. For such tenants, the ability to measure the variability in IOPS as envisioned by the proposed work will provide for a powerful paradigm that users can leverage to determine the amount of redundancy in resources required to achieve a prespecified quality of service (QoS) for their applications. This, in turn will lead to better resource planning and stronger SLA guarantees. Good measures of variation can also help cloud providers with implementing better isolation and fairness mechanisms [175] as well. In the proposed work, we will investigate these aspects and explore the use of variability information in designing more robust cloud-based services.

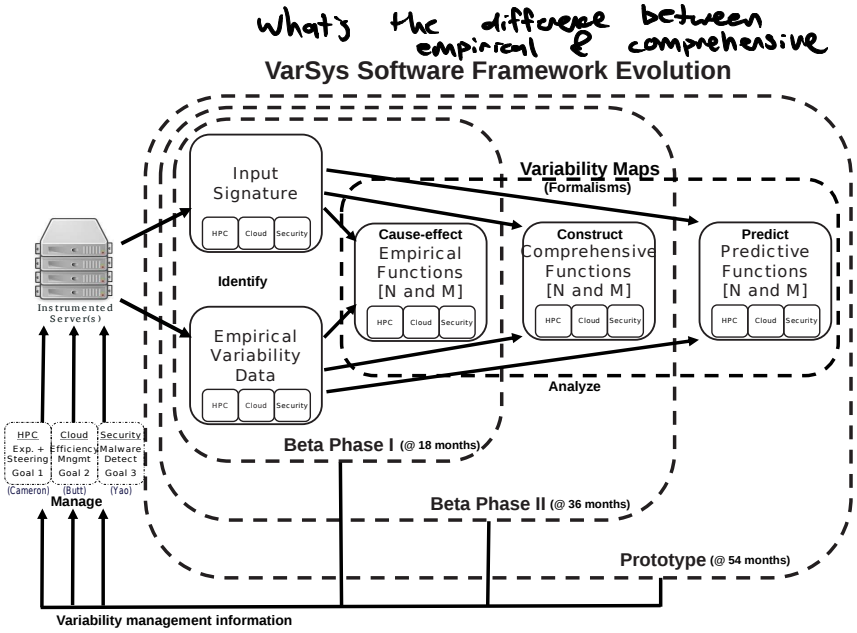


Figure 3. Overall architecture of VarSys.

Variability-proofing Cloud-based Services The variability in a cloud caused by differences in underlying architecture can be minimized by allowing cloud tenants to choose the underlying physical hardware configuration, e.g., network locality, processor, memory type, storage device, etc [101]. To help reduce the variability in multitenant or shared cloud environments, it is important to accurately predict the future resource usage by understanding the usage pattern of a cloud setup. Also, different tracking mechanisms can be used to keep tabs on performance variability so that performance variability can be studied and made more predictable.

A number of research projects [93, 125, 144, 204, 211] explore how to optimize MapReduce based on underlying cloud environments from variability perspectives. One of such optimizations can be adopted for a specific underlying environment to help reduce the performance variability if we know the dominant causes of the variability. For example, if the variability is mainly caused by heterogeneity, we might consider applying [211]; if the variability is mainly caused by data locality, [204] and [144] are worth consideration.

Such techniques can be used to build variability-proof applications atop cloud services. The motivation for such services is the Chaos Monkey toolkit [2, 87] used by Netflix Inc. to shield their services against vagaries of a cloud. At a high level, we propose to identify the variability in the system, and then introduce sufficient redundancy within the instantiated services so as to mitigate the expected loss in availability and performance. For instance, a compute instance can be enhanced with additional resources, a storage layer

can be given added copies, and I/O can be overprovisioned accordingly. We aim to study these aspects in the proposed work.

2.2.3 Research Goal 3: Improving Malware Detection

Variability-aware Anomaly Detection Variability management will enable design of *variability-aware* malware and anomaly detection solutions, allowing us to exploit variability system behavioral features previously unavailable. Low-level system behaviors have been reported for malware analysis purposes, e.g., [119, 169]. In VirusMeter [119], researchers detected Android malware through modeling runtime power consumption with techniques such as neural networks. They reported moderate false positive rates (>9.7%) with a slightly lower false negative rate (i.e., missed detection). Andromaly [169] used dynamic features, including memory page activity, CPU load, SMS message events, network usage, touch screen pressure, binder information, and battery information. Its detection accuracy is slightly lower than VirusMeter. The dynamic features evaluated by Amos et al. also include power and memory consumption [16]. However, their false positive and false negative rates are substantially higher (15% and 18%). In comparison, the proposed work will produce new variability-aware binary classification techniques and outlier-based detection techniques. Low-variability features will likely produce more distinguishable runtime system behaviors used by classification and improved malware detection rates.

We will design and evaluate program anomaly detection techniques and binary classification based malware detection methods with low-variability system behavioral features. We will take measurements from various aspects of a process, including CPU cycles, power consumption, and memory page activity and timing. In addition, we will also explore new dynamic system behavioral features, such as the timing of disk I/O, bus delay, I/O throughput. Some of these features are commonly known to have high variability and have not been reported for security classification purposes in the literature.

Anomaly detection (aka outlier detection) differs from binary classification based security approaches. It aims to recognize patterns in normal executions and detect deviations that are potentially due to operational errors, software/hardware design flaws, or malicious attacks. The advantage of anomaly detection is that it does not require the availability of labeled (training) data. However, anomaly detection with dynamic system features has not been successfully reported in the security literature, due to high variability in feature values. With low variability as a technical enabler, we will design dynamic behavioral feature-based novelty detection techniques. The main challenge is the diversity in normal execution, thus straightforward application of conventional outlier detection methods such as one class SVM would give low accuracy. We will focus on addressing this challenge of high diversity in normal system patterns. The key to our proposed solution is the *fine grained categorization* of normal system behaviors. We plan to first utilize data mining techniques (e.g., clustering) to categorize behavioral features. Then, we will perform novelty detection (e.g., one class SVM) within each cluster. Both training and testing will follow this new 2-step procedure. Binary classification for malware detection (e.g., two-class SVM, decision tree) will be more straightforward. We will correlate its accuracy with the degrees of variability. We will categorize several variability levels, e.g., representing high, medium, and low variability. Each behavioral feature may have a different variability level. For feature set $F = (f_1, \ldots, f_n)$, its corresponding variability set is also a vector $V = (v_{f_1}, \ldots, v_{f_n})$. Under each set V of variability conditions, we will collect the corresponding training and test data, i.e., each model is trained and tested with the dynamic features with the same variability degree. Detection accuracy values under different variability conditions will be compared and their correlation relations will be computed. We will detect programs in both conventional Linux environments as well as Android operating systems. We will evaluate against both real-world malware samples and exploits and synthetic malware and anomalies. The precision will be evaluated based on false positive (i.e., false alarm) and false negative (i.e., missed detection) rates. Our existing expertise in large-scale malware analysis and detection (e.g., [174, 206]) will enable us to swiftly engage in the proposed work.

Variability-driven Defense Against Side-Channel Attacks Variability management can also be used to address the challenging problem of timing-based side channel attacks. We aim to prevent and mitigate some

types of attacks by reducing the predictability of behavioral differences and dependences on sensitive inputs. Our approach will be *variability driven*, i.e., to realize this defense by proactively generating desirable system variability. Timing-based side-channel attacks (e.g., [30, 31, 35, 84, 163]) leverage statistical differences in execution time (e.g., CPU cycles, memory activity) for gaining sensitive information. For example, a recent attack against the Firefox browser utilized the timing variability in floating point operations [18]. Straightforward masquerades of computation (e.g., through inserting dummy computation) can be detected by advanced interactive attackers [84]. A deterministic time replay tool was recently developed for discovering timing based side channels in programs [51]. Increased variability in system behavioral data may alter attack complexities, affecting the number of observations needed by attackers. Our work will produce new variability-driven proactive side-channel defenses.

We will mitigate timing-based side-channel attacks with unpredictable variability management mechanisms and to evaluate the impact of low variability environments on existing side-channel defenses. The complexities and costs of timing-based side-channel attacks change with the signal variability. Existing successful side-channel attack demonstrations typically require the attackers to take large numbers of samples in order to achieve predictable and distinguishable data leaks (e.g., zero-one gap in CPU cycles of decryption) [35]. Our approach for thwarting side channels is to reduce the predictability of the system environments utilizing variability management capability. We will design randomized noisy system environments that make it substantially more costly for attackers to distinguish sensitive-data induced behavioral differences. Intuitively, our randomization approach is to inject artificial noise in system behavioral data that masks and flattens the side channel information. This manipulation will change (i.e., artificially boost or reduce) variability of system behaviors. Naive noisy environments that have predictable distributions (e.g., Gaussian) of noise are inadequate. Thus, the unpredictability requirement for the schedule, i.e., how variability changes with time. This subterfuge is necessary to prevent attackers from inferring variability changes in observable patterns and reverse-engineering the masks. The degree (e.g., order of magnitude) of artificial variability boost and reduction may vary and will be determined by the perceived threats and security requirement. Stronger defense will incur more costly variability changes, which may slow down performance. In addition, we will evaluate existing side-channel defenses (e.g., [53]) under the new low-variability environments for their robustness.

2.2.4 Statistics

Statistical Tools and Background The overarching goal of the statistical research is to create a collection of statistical tools that system computer scientists can use to visualize, characterize, and evaluate the performance variability.

Visualization Visualization of the performance variability will be an important step. For univariate cases, statistical plots such as boxplot, and error bar plot can be used. A wide spread in the boxplot shows (see 3 GHz data point in Figure 1) exhibits large variability. Let x be the variables in a system and $\mathrm{Var}(t(x))$ be the variance of the performance $t(x)$. If the design variable is multi-dimensional, the variance surface can be visualized in multiple pairwise plots. For example, a 3D perspective plot can be used to show the variance surface $\mathrm{Var}(t(x_1, x_2))$ of two variables x_1 and x_2. Smoothing techniques can be used to obtain a smooth surface. → does this have a chance of taking too long to compute?

Characterization The variance $\mathrm{Var}(t(x))$ provides a numerical characterization of the system variability. Large variability can result in low reproducibility. Consequently, one will need a larger number of samples if the interest is to estimate the mean performance metric (e.g., execution time) with desired precision. Large variance also suggests the need for better designs, experiments or configurations to reduce the variability. A complete characterization of variability is the statistical distribution of $t(x)$.

Comparison To evaluate the system variable, one can make comparisons among different systems. For example, we can make comparisons of the variance of two different CPUs, two different servers, two different clusters, or the QoS of cloud systems. Also, we can use the techniques to compare variance of cloud-based systems to bare-metal systems. We can have a two-sample test to compare, for example, the

mean execution times or variances between the two settings. Statistical hypothesis testing will examine performance with a certain confidence level. Most statistical tests assume normal distributions. We will investigate those assumptions. When the distributions are nonnormal, we can have nonparametric (without assuming a parametric distribution) tests or bootstrap methods to do the comparison.

Change Detection From the security side, we can use the variance signature to isolate and identify security breaches in a system by looking for change in the variance signature. This can create new techniques that better identify and isolate variance signatures in a system. The two-sample Kolmogorov-Smirnov test can be used to detect the change in the distribution of the variability. The test statistics is $D_{n_1,n_2} = \sup_x |F_{1,n_1}(x) - F_{2,n_2}(x)|$, where $F_{1,n_1}(x)$ and $F_{2,n_2}(x)$ are the ecdf based on samples from normal and suspicious conditions, respectively. If the interest is in comparing two joint distributions, the multivariate Kolmogorov-Smirnov goodness of fit test is also available [96].

Number of Experiment Runs To achieve a certain precision, the required number of experiment runs at a configuration is $n = z^2_{\alpha/2}\sigma^2/\mu^2$, where $(1-\alpha)$ is the confidence level, μ is the mean, and σ^2 is the variance. Here, z_α is the $100(1-\alpha)\%$ is percentile of the standard normal distribution. The variability has an important role in determining the number of runs. For our preliminary data (see Figure 1), when the hypervisor scheduler is NOOP, the VM scheduler is CFQ, and the number of threads is 64, the standard deviation (σ) for the 3.5 GHz setting is 24407247, which is 2.5 times of the setting at 2.5 GHz (24407247). Hence, if one has 100 runs at the 2.5 GHz setting, one will need $2.5^2 \times 100 = 625$ runs for the 3.5 GHz setting, to achieve the same amount of precision. Thus, our preliminary results indicate that it is possible to reduce the number of experiments needed for statistical confidence substantially by running at 2.5 GHz. When there are dozens or hundreds of variables, with the performance variability, we can control for multiple variables that have a comparable affect such as that observed in this example to reduce the number of runs needed.

Statistical Design of Experiments We propose to use the design of experiments (DOE) techniques in statistics to efficiently collect samples to construct the proposed variability map. The proposed use of DOE to sample in high dimensional (> 100) spaces to construct the variability map is novel in HPC and cloud systems to the best of our knowledge. These techniques are also designed for the scale of variables expected at the exascale.

The x we consider here is high dimensional. For example, suppose one wants to consider different processor frequencies (e.g., 15), the choice of I/O scheduler (e.g., 10), the operating system version (e.g., 20), the operating system type (e.g., 10), the system hardware configurations (e.g., 500). If each experiment point needs 50 runs and 5 seconds for each run. The total time for a full factorial design will be $15 \times 10 \times 20 \times 10 \times 500 \times 50 \times 5$ (more than 100 years) to identify the optimal case. Hence, innovation in the experimental design is sorely needed.

In this project, we propose to run a screening design first to learn the importance of the input signature impact on variability. The novel definitive screening developed in [95] will be used, which is a three level design with only $2m-1$ design points. Here m is the length of the signature vector x. The definitive screening will provide a subset of important variables and ranking, and reveal possible nonlinear relationships [146]. If the screening design can reduce the set to a relatively smaller set, the second stage is to run a space filling design. The advantage of a space filling design is that it provides good coverage of the design space, many levels for each variable, and good projection properties. Space filling designs with Latin hypercube and their variants [146] will be used in this proposed research. The existing statistical methods can provide a good starting point for the research. Due to the complexity of the research problem, the existence of both continuous and discrete variables, and the nonlinear constraints of design space (not all combinations are possible), the existing screening approach and space filling design needs to be transformed to address the particular challenges from the DOE in computer systems. Thus, the proposed project will also advance the state of the art for the statistical technology.

In scenarios where all variables are important, two strategies are proposed. We can still pick a subset

of variables according to the importance ranking, and do a space filling design under the affordable resources. Alternatively, one can use the incomplete orthogonal designs, in which case one needs to realize that important regions of space might be omitted.

2.2.5 Mathematics

Consider a computational task on a computer system, all characterized by some vector x of parameters, and the time $t(x)$ to completion for this task. The ultimate goal might be to adjust x, by changing both the implementation of the task and the configuration of the computing hardware, so as to minimize $f(x) = \text{Var}(t(x))$. The connection between x and $f(x)$, the variance $\text{Var}(t(x))$ in this case, over varying x, will be referred to as the *Variability Map*.

There are numerous situations where one wants to minimize $\text{Var}(t(x))$. Massively parallel computation of ensembles of runs are common in quantum chemistry, solid state physics, systems biology, vehicular traffic modeling, epidemiology, and conceptual aircraft design, to mention but a few areas of computational science and engineering. Often these runs are identical, or nearly so, analyses or simulations with different data, and ensemble results are periodically aggregated at synchronization points in time. If multiple executions of the same instruction stream all took the same time, massively parallel ensemble calculations would be efficient and straightforward. In practice, due to significant $\text{Var}(t(x))$, this is not the case, resulting in severe load imbalance (and concomitant wasted resources) and complicated algorithms to address the imbalance, further exacerbated by increasing parallelism. The proposed *Variability Map* directly addresses the problem of understanding, classifying, and reducing this variance $\text{Var}(t(x))$.

Suppose one knew that for a class of program and system parameters x, the execution time $t(x)$ had a gamma distribution $\text{Gam}(\alpha, \beta)$. One could then, from a relatively few observations, estimate α, β, and then predict future expected execution time $E[t(x)]$ with high confidence. The need for extensive experiments to directly estimate $E[t(x)]$ and $\text{Var}(t(x))$ would be eliminated if the *Variability Map* could provide the higher level information $\text{Gam}(\alpha, \beta)$.

Another direct application of the *Variability Map* concerns malware detection. If the parameter vector x were tuned so that $\text{Var}(t(x))$ is small, an increase in $\text{Var}(t(x))$ would indicate the presence of malware. More generally, a significant change in the distribution of $t(x)$, say from a uniform to a bimodal, indicates x has changed, meaning malware has been inserted somewhere. Further application of the *Variability Map* might indicate whether the change in x occurred in the program parameters or the system parameters.

Mathematical Background. A formal statement of the *Variability Map* is that it is a function

$$\mathcal{M} : (Z^{j_1} \times \mathbb{R}^{j_2}) \times (Z^{m_1} \times \mathbb{R}^{m_2}) \to \left\{ f \in L^1[0,\infty) \;\middle|\; f \geq 0,\ \int_0^\infty f(x)\,dx = 1 \right\},$$

mapping a job (task) signature $\zeta = (\zeta_1, \zeta_2) \in Z^{j_1} \times \mathbb{R}^{j_2}$ and computer system signature $\eta = (\eta_1, \eta_2) \in Z^{m_1} \times \mathbb{R}^{m_2}$ into the probability density function $\mathcal{M}(\zeta, \eta)$ describing the distribution of the time $t(\zeta, \eta)$ required for the completion of task ζ on system η. A much simpler, but still very useful, version of this statement is where $\mathcal{M}(\zeta, \eta)$ is some summary statistic(s) of the random variable $t(\zeta, \eta)$, e.g., the mean or variance of the time $t(\zeta, \eta)$, or generally a functional of the random variable $t(\zeta, \eta)$ and its probability density function. The signatures ζ and η are comprised of discrete integer and real continuous parameters, whose determination is a central goal of this work. The parameters will be both directly obtainable information such as cache size, communication latency, program memory footprint, program I/O, etc., and generalized coordinates obtained by, e.g., feature compression or dimension reduction algorithms. The PIs are experienced in modeling both physical systems (solids [188], fluids [199], composite materials [10, 76, 111], biology [12, 216], controls [72], circuits [128], aircraft [13, 103]) and computer systems (power [44], security [14], parallel [81–83, 177, 178, 198]), and are well positioned to create the *Variability Map*.

The iterative process for developing $\mathcal{M}(\zeta, \eta)$ is in Figure 4. An initial attempt at defining signatures (1) is followed by data acquisition (2) of triples $(\zeta_i, \eta_i, t(\zeta_i, \eta_i))$, which are then used with approximation theory and statistical methods to construct (3) a function $\mathcal{M}(\zeta, \eta)$. This function is used to design and predict the

outcome of new experiments (4), which informs refining the signatures (1). With the passage of time the amount of data (and its variety) only continues to grow, making the construction of $\mathcal{M}(\zeta, \eta)$ for fixed (ζ, η) better, and providing more different (ζ, η) pairs, all the while improving the choices for the components of the signatures (ζ, η).

The Challenges and Deliverables. Each of the four components in Figure 4 constitutes a technical challenge requiring new data, technologies, and algorithms. Defining the job and system signatures (ζ, η) is the bailiwick of the HPC systems PIs Butt, Cameron, and Yao, and creating, refining, and curating these signatures is a major deliverable. Given the likely high dimension of (ζ, η), data acquisition will require state-of-the-art statistical sampling, the bailiwick of PI Hong, and considerable supercomputer experimentation managed in close collaboration between the PIs Hong, Butt, and Cameron. The resulting signature-performance database is another major deliverable. The construction of the map $\mathcal{M}(\zeta, \eta)$, described in detail below, is a theoretical challenge, requiring state-of-the-art numerical analysis, optimization, statistics, and machine learning, the bailiwicks of Hong and Watson. The principal deliverable is the Variability Map $\mathcal{M}(\zeta, \eta)$. The final challenge is using prediction, both for applications (e.g., security) and feedback into the loop in Figure 4, which will involve all the PIs. Deliverables here are application specific uses (e.g., security, cloud management) of the Variability Map $\mathcal{M}(\zeta, \eta)$.

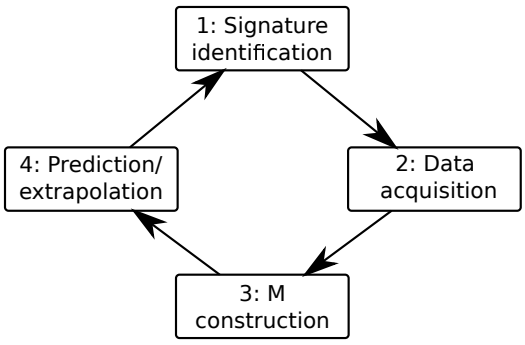


Figure 4. The iterative process for developing $\mathcal{M}(\zeta, \eta)$.

Construction of $\mathcal{M}(\zeta, \eta)$. To distinguish between the general case where $\mathcal{M}$ is a function whose value $\mathcal{M}(\zeta, \eta)$ is a function (the PDF of the time $t(\zeta, \eta)$) from the special case where the value $\mathcal{M}(\zeta, \eta)$ is a real number (e.g., the variance $\mathrm{Var}(t(\zeta, \eta))$), let $\mathcal{M}$ denote the *Variability Map* case with values

$$\mathcal{M}(\zeta, \eta) = f_{t(\zeta,\eta)}(x),$$

the probability density function of the random time to completion $t(\zeta, \eta)$ of job ζ on computing system η, and let $\mathcal{N}$ denote the *Variability Map* case with values

$$\mathcal{N}(\zeta, \eta) = \mathcal{F}(t(\zeta, \eta)),$$

a real-valued functional of $t(\zeta, \eta)$. Consider first the construction of an approximation to $\mathcal{N}$. For fixed (ζ, η), only samples $t_1, \ldots, t_N$ of $t(\zeta, \eta)$ are available, so the first issue is to construct a reasonable statistical approximation $\bar{\mathcal{F}}$ of $\mathcal{F}(t(\zeta, \eta))$ from these samples. e.g., if $\mathcal{F}$ is the expected value, then the approximation to $\mathcal{F}(t(\zeta, \eta))$ would be $\bar{\mathcal{F}} = (1/N) \sum_{i=1}^{N} t_i$. The data now consists of triples $(\zeta_i, \eta_i, \bar{\mathcal{F}}_i)$, and the construction of an approximation to $\mathcal{N}$ is proposed by decidedly nonclassical methods: modern statistical machine learning [131] and modern approximation theory [54]. Note that the dimension of the signatures (ζ, η) is expected to be high, the number of triples $(\zeta_i, \eta_i, \bar{\mathcal{F}}_i)$ will be large and growing with the passage of time, so classical statistical (e.g., general linear model regression) and numerical analysis (e.g., polynomial interpolation) techniques are unlikely to suffice. The construction of high dimensional approximations is a central problem in machine learning and aerospace engineering (where they are called "surrogates") now, and with which PI Watson has two decades of experience [13, 37].

Most of these new statistical, machine learning, and approximation theory methods for constructing an approximation to $\mathcal{N}$ involve minimizing something, and converging to a local minimum point is always a concern. Some very recent mathematical results in deterministic global optimization [17, 67] make it possible, with high end computing, to actually find globally optimal solutions. Thus using these new algorithms,

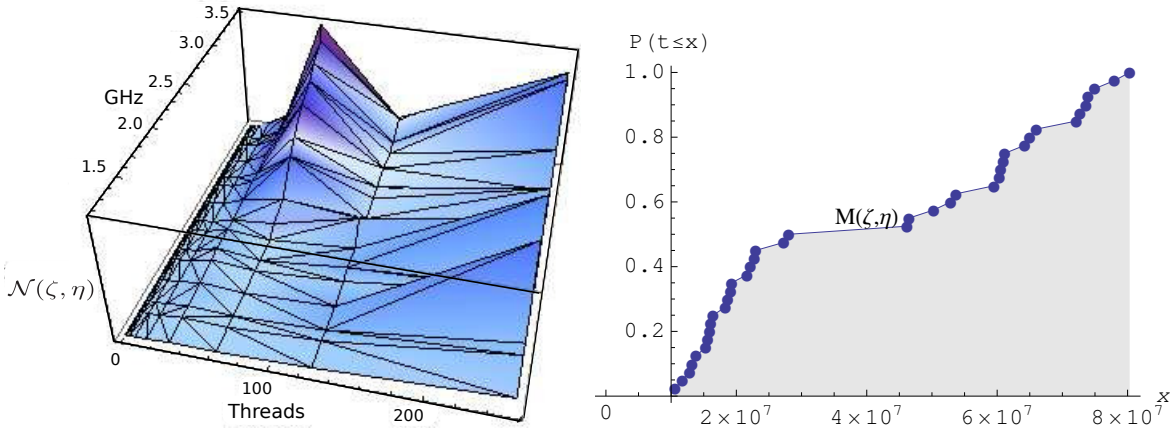


Figure 5. Illustration of the functional form $\mathcal{N}(\zeta,\eta)$ (left) and the general form $\mathcal{M}(\zeta,\eta)$ (right) of the Variability Map. Here (ζ,η) = (number of threads, clock frequency in GHz), $\mathcal{N}(\zeta,\eta) = \text{Var}(t(\zeta,\eta))$, and $\mathcal{M}(64,3.5)$ is the CDF shown at the right.

one can find the *best possible* approximation to $\mathcal{N}(\zeta,\eta)$ for a particular approximation technique, which heretofore was generally intractable.

Even if this research were unsuccessful in extending the real-valued *Variability Map* $\mathcal{N}(\zeta,\eta)$ to the PDFs of the general *Variability Map* $\mathcal{M}(\zeta,\eta)$, the general functional form of $\mathcal{N}(\zeta,\eta) = \mathcal{F}(t(\zeta,\eta))$ is still powerful and covers any statistic of $t(\zeta,\eta)$ relevant to system performance and design. Consider now the construction of $\mathcal{M}$ as a function of (ζ,η), given that the PDF $\mathcal{M}(\zeta_i,\eta_i)$ has already been approximated for a large number of signatures (ζ_i,η_i). Although most program and computer system parameters change in discrete increments (virtual memory pages, disk I/O blocks, buffer sizes, cache lines, etc.), it is reasonable to represent most (but not all, e.g., binary 0-1 parameters representing the presence of absence of a hardware component or program feature) of these parameters by real numbers. Referring back to the definition of $\mathcal{M}$, fix the first j_1 (discrete) components of ζ and the first m_1 (discrete) components of η, and construct the approximation of $\mathcal{M}(\zeta,\eta)$ with respect to the remaining (real) components of (ζ,η). Thus essentially a family of Variability Maps is built, one for each distinct discrete part of (ζ,η). It may be possible to combine some of these, but that is a later consideration.

Now, given that the discrete components of (ζ,η) are fixed, and a large number of probability density *functions* $\mathcal{M}(\zeta_i,\eta_i)$ are known (approximated from experiments), how is the full $\mathcal{M}(\zeta,\eta)$ built? This is where the advanced approximation theory [54] comes into play. Without delving deep into the details, essentially the idea is interpolation in a function space. The interpolant $\overline{\mathcal{M}}(\zeta,\eta)$ approximating $\mathcal{M}(\zeta,\eta)$ has the property that $\overline{\mathcal{M}}(\zeta_i,\eta_i) = \mathcal{M}((\zeta_i,\eta_i)$ for all i, and for $(\zeta,\eta) \neq (\zeta_i,\eta_i)$ for any i, $\overline{\mathcal{M}}(\zeta,\eta)$ is a blended linear combination of functions $\mathcal{M}((\zeta_i,\eta_i)$ at nearby points (ζ_i,η_i).

Figure 5 illustrates the ideas here for a simple 2-variable case where (ζ,η) = (number of threads, clock frequency in GHz). The surface plot at the left shows $\mathcal{N}(\zeta,\eta) = \text{Var}(t(\zeta,\eta))$ for some real performance data from Figure 1 [50], and the plot at the right shows the cumulative distribution function (CDF) from 40 samples for $t(\zeta,\eta)$ at $(\zeta,\eta) = (64,3.5)$. This CDF is the value of the Variability Map $\mathcal{M}(\zeta,\eta)$ evaluated at $(\zeta,\eta) = (64,3.5)$. There are two discrete scheduler variables in the actual data, both fixed for this illustration. To construct the entire map $\mathcal{M}$, an example of an interpolation scheme that generalizes to both high dimensions and function spaces is Shepard's algorithm [184].

Related Work. Why might all this work, and if it did, transform computer systems design? Answer: the design and performance characteristics of every single modern aircraft, airfoil, engine, and control system are in databases and analysis and design software for aircraft [13]. Every new aircraft is designed starting

with this "Aircraft Performance Map" and interpolating "inside the envelope" or extrapolating "outside the envelope". Certainly new analyses and simulations must still be done, but leveraging this existing performance map saves an enormous amount of time and materials. This is a proof of concept. What is fundamentally new in this proposal is the Variability Map $\mathcal{M}(\zeta, \eta)$ giving distributions, compared to the Aircraft Performance Map $\mathcal{N}(\zeta, \eta)$ giving a single (mean) performance value.

3 Outreach and Education Component

Our education and outreach plan is designed to match the ambition and potential impact of our research plan. We adopt a multi-pronged approach to this end, which is described next.

Red-Blue Team Hack-a-thons. By its very nature, variability management provides a spectrum of opportunities. For example, we may aim to reduce variability, e.g., for cloud service-level agreements, or increase variability, e.g., to protect against sophisticated profiling security threats. To consider such range of usecases, we will organize Red-Blue Team Hack-a-thons, where students compete to reduce (Red Team) or increase (Blue Team) the variability on a target system. To this end, SP Back will leverage his experiences coaching the Virginia Tech programming team to consecutive appearances at the ACM International Collegiate Programming Competition (ICPC) World Finals competition in 2014 and 2015, to recruit students for an inaugural Red-Blue event in the first year of the project.

In succeeding years, we plan to solicit sponsorship from our Computer Science Research Consortium (CSRC) [3] for prizes and expand the competition regionally. For recruitment, we will leverage our social media presence and existing relationships with nearby universities (UVA, George Mason, and UMD) as well as smaller schools (UNC-Charlotte, James Madison, Roanoke College) and historically black colleges (Norfolk State). We will also explore inclusion of local high school participants either as participants or perhaps as volunteers to help with organization.

We believe that this effort has the potential to transform our research efforts. First, we can use the competitions to test our research framework including our abilities to measure, identify and classify variability in open environments. Second, students may propose effective techniques that can be studied further in the project. Third, we can use the Red-Blue event to identify and recruit talented internal and external students to the project.

Advisory Board. Any funded team effort has limitations such as unintentional bias toward an expected outcome. Unbiased introspection is often difficult when one is deeply invested in research. We propose to create a Technical Advisory Board (TAB) that meets bi-annually to discuss and assess progress on the project. We have obtained TAB letters of commitment (see Supplementary Documents) from researchers at industrial laboratories (IBM, Symantec, Veritas) and national laboratories (LLNL, LBNL, ORNL, and PNNL). In addition to serving on the TAB in an advisory capacity, all TAB members (except IBM) have agreed to share variability data as permitted by their organizations. Members will also explore adoption of our technologies, participate in outreach activities such as the Jitter500, and actively recruit students from the project for internships and co-ops.

Jitter500.org. As discussed, variability can hinder performance in HPC systems. Notwithstanding efforts to achieve statistical confidence in benchmark results, we believe the software and formalisms of the VarSys framework provide the opportunity to establish methodologies for comparing the variability of systems. We will use VarSys to study the use of existing benchmark suites (NAS [5], SPEC [6], TPC [1]) to create a Jitter500 List of systems ranked by their variability. While the details of the Jitter500 methodology must be developed and studied, PI Cameron will leverage his experiences co-founding two successful benchmarking methodologies: the Green500 List [4] and the SPEC Power benchmark [6].

Recruiting and training diverse talent. PI Yao will organize and lead extensive efforts to recruit women to the project leveraging her accomplishments and experiences as well as Red-Blue Team events. For example, we will actively recruit underrepresented minorities via the National Consortium for Graduate Degrees for Minorities (GEM). We have also worked extensively with the Multicultural Academic Opportunities

Program (MAOP) to recruit undergraduates for summer research projects from historically black colleges and institutions of higher learning. We plan to continue to leverage all of these programs and additionally identify non-traditional CS students (e.g., with Math or Statistics backgrounds) with aptitude and interest in computer science. We will also leverage existing CS@VT relationships with minority-serving organizations such as the STARS alliance and the A4RC (Alliance for Advancement of African American Researchers in Computer Science) to recruit students at related events.

Identifying talent is only the first step to engagement in research and education. Students will have access to advanced computer systems and supercomputers locally and other systems provided during research internships at industry and national laboratories. Students will regularly attend a research reading group (run by PI Yao). The PIs have an exceptional record of job placement using these methods of mentoring. Recent Ph.D. graduates have joined government and research laboratories as research scientists and several have joined ranked universities as tenure-track faculty.

Other activities. The PIs will integrate research findings in courses such as parallel computing, computer architecture, operating systems, and secure software. By the end of the project, all code will be made available via open source licenses. We will give technical tutorials at major conferences describing the use of the VarSys framework for research. PI Cameron will leverage lessons and contacts from release of his PowerPack framework [71] that has been used by dozens of universities including UC Berkeley and Oxford. Lastly, we will seek to publish our research findings in top conferences and journals in the field.

4 Evaluation of Research and Outreach/Education Plans

The work plan (Figure 6) describes the overall goals, milestones and tasks of the proposed project. The plan was determined after careful consideration of the anticipated effort required, the role of each participant, and the inter-dependencies (see Collaboration Plan in Supplementary Documents) across the entire project. Thus, the work plan provides a conceptual model of the project, and identifies the key project outcomes as well as evaluation points.

Evaluating Research. All of the Research Goals (I–III) and the Mathematical and Statistical Characterizations (IV) can be evaluated experimentally. Namely, we will design and build software tools where the effects are demonstrable on real systems. For our three research goals for HPC, Cloud, and Security (see Figure 2), evaluation involves direct measurement before and after variability management for: 1) reduction of experiments and runtime steering versus best available methods in HPC environments; 2) tightening of service rates versus best-available methods in Cloud environments; and 3) reduction in the likelihood of false positives and negatives versus best-available malware detection methods. This method will be applied similarly to both tasks (i.e., software modules) and milestones (beta and prototype functionality in the project). The findings will provide indirect validation of the mathematical and statistical characterizations constructed over the course of the project and used in the resulting framework. We will present early work periodically to the full team and advisory board at our local seminars and bi-annual retreats. We will submit our findings to archived, peer-reviewed, highly-competitive workshops, conferences, and journals in the field.

The iterative design of nearly all aspects of our research plan means we have the opportunity to continually assess and adjust our research efforts. Prior to our bi-annual retreats, we will conduct surveys of participants using anonymous electronic means (https://survey.vt.edu or SurveyMonkey.com). Questions will focus primarily on identifying barriers to progress, effectiveness of communications, accuracy and usefulness of the current work plan, and soliciting suggestions to improve or adjust the project plan as necessary. We will discuss findings with key personnel and the technical advisory board to identify and implement solutions designed to improve the research effort. We will define evaluation criteria and then assess the effectiveness of plan adjustments using surveys at the next bi-annual meeting (or earlier as necessary). We will be open to the adoption of other assessment techniques as the project progresses as well.

Evaluating Education and Outreach. The goal of our education and outreach efforts (V) is to broaden

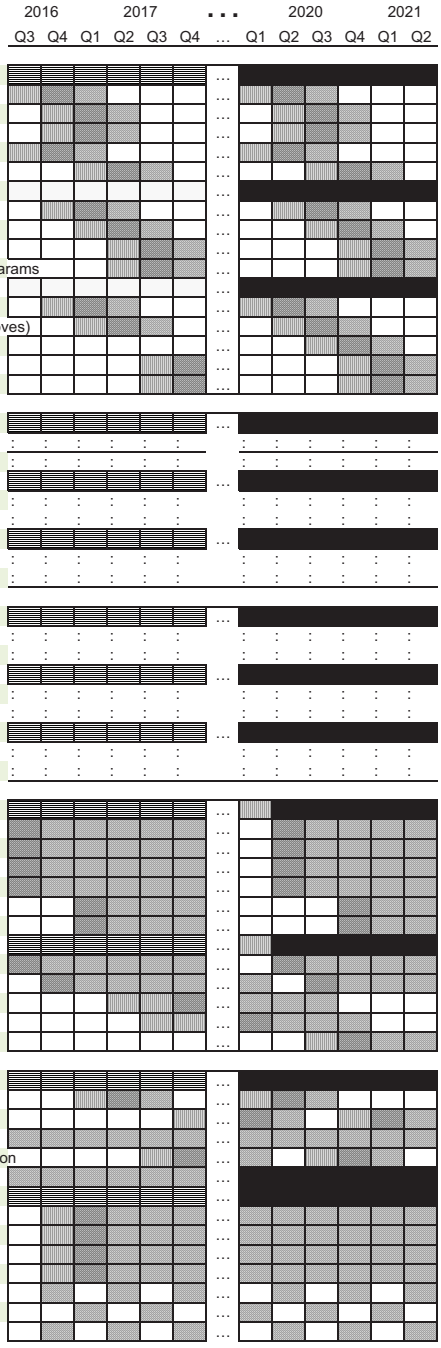
KEY:
Design/Hypothesize
Develop/Construct
Evaluate/Test

Beta Phase I
Beta Phase II
Prototype Complete

ABBREVIATIONS:
KC: Kirk W.Cameron; KC*=KC student
LW: Layne Watson; LW*=LW student
DY: Daphne Yao; DW*=DW student
AB: Ali Butt; AB*=AB student
YH: Yili Hong; YH*=YH student
GB: Godmar Back; GB*=GB student
ER: External relations (communications) staff
RP=all research personnel; AP=all personnel

Figure 6.Time and effort projected for all milestones and tasks in the project. Years 2018-19 (omitted) cover Beta Phase II and are similar in scope to years 2020-21.

	LEAD (SUPPORT)	Goals Milestones Tasks
I	KC(RP)	Research Goal 1: HPC experimental and steering design improvements
I.A	KC*(RP)	Identify empirical relationships between variability and HPC systems
I.A.1	KC*(RP)	Manually measure preliminary empirical variability data
I.A.2	KC*(RP)	Manually measure comprehensive empirical variability data
I.A.3	KC*(RP)	Automate measurement of empirical variability data
I.A.4	KC*(RP)	Manually classify empirical variability data
I.A.5	KC*(RP)	Automate classification of empirical variability data
I.B	KC*(RP)	Analyze relationships between variability and HPC systems
I.B.1	KC*(RP)	Manually analyze preliminary empirical variability data
I.B.2	KC*(RP)	Manually analyze comprehensive empirical variability data
I.B.3	KC*(RP)	Automate analysis of empirical variability data
I.B.4	KC*(RP)	Automate analytical classification of relationships between variability and sys params
I.C	KC*(RP)	Manage relationships between variability and HPC systems
I.C.1	KC*(RP)	Manually manage variability using N variability map (iterate as N improves)
I.C.2	KC*(RP)	Automate management of variability using M variability map (iterate as M improves)
I.C.3	KC*(RP)	Automate interpolation/extrapolation management(iterate)
I.C.4	KC*(RP)	Apply variability management to HPC experimental design
I.C.5	KC*(RP)	Apply variability management to HPC runtime system steering methods
II	AB(RP)	Research Goal 2: Cloud service efficiency improvements
II.A	AB*(RP)	Identify empirical relationships between variability and Cloud systems
II.A.1-5	AB*(RP)	<Similar to I.A.1-I.A.5 in content and work estimate, details omitted for space>
:	:	:
II.B	AB*(RP)	Analyze relationships between variability and Cloud systems
II.B.1-5	AB*(RP)	<Similar to I.B.1-I.B.5 in content and work estimate, details omitted for space>
:	:	:
II.C	AB*(RP)	Manage relationships between variability and Cloud systems
II.C.1-5	AB*(RP)	<Similar to I.C.1-I.C.5 in content and work estimate, details omitted for space>
:	:	:
III	DY(RP)	Research Goal 3: Malware detection technique improvements
III.A	DY*(RP)	Identify empirical relationships between variability and security signatures
III.A.1-5	DY*(RP)	<Similar to I.A.1-I.A.5 in content and work estimate, details omitted for space>
:	:	:
III.B	DY*(RP)	Analyze relationships between variability and security signatures
III.B.1-5	DY*(RP)	<Similar to I.B.1-I.B.5 in content and work estimate, details omitted for space>
:	:	:
III.C	DY*(RP)	Manage relationships between variability and security signatures
III.C.1-5	DY*(RP)	<Similar to I.C.1-I.C.5 in content and work estimate, details omitted for space>
:	:	:
IV	LW, YH (RP)	Mathematical and statistical characterizations of performance variability
IV.A	YH(RP)	Empirical data analyses
IV.A.1	YH*(RP)	Identify preliminary empirical variability input signature
IV.A.2	YH*(RP)	Identify comprehensive empirical variability input signature
IV.A.3	YH*(RP)	Develop classification schemes for variability based upon input signature
IV.A.4	YH*(RP)	Derive statistical analyses of variability (low dimensional)
IV.A.5	YH*(RP)	Derive statistical analyses of variability (high dimensional)
IV.A.6	YH*(RP)	Derive statistical signature for security application variability
IV.B	LW(RP)	Variability maps N and M
IV.B.1	LW*(RP)	Construct N function using empirical data
IV.B.2	LW*(RP)	Construct M function using empirical data
IV.B.3	LW*(RP)	Construct N function using comprehensive empirical data
IV.B.4	LW*(RP)	Construct M function using comprehensive empirical data
IV.B.5	LW*(RP)	Prediction via interpolation and extrapolation of N and M
V	GB, ER	Integrated, Impactful Education and Outreach Activities
V.A	GB(ER)	Education and recruitment efforts
V.A.1	RP	Integration of research findings in courses
V.A.2	RP	Technical tutorials on VarSys framework
V.A.3	RP	Mentor undergraduate and graduate students
V.A.4	ER(RP)	Red-Blue Team Hack-a-thons organization, recruitment, publicity, implementation
V.A.5	RP	Recruiting women and minority students
V.B	ER(GB)	Outreach and dissemination efforts
V.B.1	LW*(RP)	VarSys.org project website, collection/creation of social media content, etc.
V.B.2	LW*(RP)	Jitter500.org website, data collection, documentation, media relations, etc.
V.B.3	LW*(RP)	Organize local and remote retreats and workshops
V.B.4	LW*(RP)	Advisory board website, meeting management, board relations, etc.
V.B.5	RP	Bi-annual internal report data creation and sharing
V.B.6	LW*(RP)	Bi-annual internal reports,tracking assessment and evaluations, etc.
V.B.7	RP	Dissemination of research results in conferences, journals, textbooks, etc.



the impact of our research efforts. As in the research plan, the efforts are iterative and enable periodic adjustments to the plan to improve effectiveness. Since the evaluation methods are context sensitive, we list the evaluation methods for the individual tasks.

Education and recruitment tasks and evaluation methods:

- *Integration of research findings in courses:* We will track the number of courses that integrate research findings with a goal of one per year.
- *Technical tutorials on VarSys framework:* We will track the number of technical tutorials, webinars, or workshops on use of the VarSys framework and/or methodologies for the Jitter500. Our goal will be 3 technical tutorials over the course of the project.
- *Mentor undergraduate and graduate students:* We will track the number and type of students participating in the project. We typically mentor both funded and unfunded students, so our goal will be to maintain a steady state of six students after the first year.
- *Red-Blue Team Hack-a-thons organization, recruitment, publicity, implementation:* See Section 3 for details. We will track the quality of participation by polling participants after the hack-a-thons.
- *Recruiting women and minority students:* See Section 3 for details. We will aim for 1 female or minority student on average per year involved in the project.

Outreach and dissemination efforts:

- *VarSys.org, Jitter500.org, and project websites, content creation, media relations, reports, etc.* See Collaboration Plan and Section 3 for details. We will track traffic using Google, Twitter, and Facebook analytics. An overall goal will be to steadily increase content and inbound traffic. Reports and research findings must be delivered on time with expedience.
- *Organize retreats, workshops, board meetings.* We will track the quality of participation by polling participants after these events. On time scheduling and delivery of important documents including programs, internal reports and assessment data mainly measure success in this effort. The goal will be to effectively sustain about 4 events per year.

The goal of all of these tasks is to establish *Integrated, Impactful Education and Outreach Activities.* The collective data from the above efforts charted bi-annually will serve to assess our overall impact quantitatively. We will additionally track overall adoption of our methods by our advisory board members and the at-large community, and the adoption of variability as a marketable feature in future systems.

5 Management Plan

Overall roles and commitments. In addition to the 5 PIs and 1 Senior Personnel, the project will support 1 External Relations Staff and 6 graduate students, for a total of 13 key personnel involved in the project. We have assembled a world class team of researchers in the areas of high-performance computing (PI Cameron), cloud computing (PI Butt), computer security (PI Yao), operating systems (SP Back), parallel computation and mathematics (PI Watson), and statistics (PI Hong). We have also identified 7 external researchers to serve on our Technical Advisory Board (See Education and Outreach Plan). This committee will meet bi-annually and contribute data and expertise, hire interns, participate in the Jitter500 List, and consider adopting our techniques at their organizations (IBM, Symantec, Veritas, LLNL, LBNL, ORNL, and PNNL).

Work Plan. PI Cameron will be overall lead for the project, drawing upon his experience of being involved in Medium and Large NSF projects, as well as managing a startup company (from 2006-2014). We have created a detailed but agile (see next section) work plan, Figure 6, that describes the goals, milestones, tasks, and responsibilities for personnel necessary to complete the project. As Figure 6 illustrates, the work for each Research Goal includes a progression of iterative milestones (e.g., I.A-C) to identify, analyze, and manage variability in (I.) HPC, (II.) Cloud, and (III.) Security platforms, respectively. Each milestone requires a set of tasks (e.g., I.A.1-5) to be accomplished. The work plan highlights these.

Agile Approach. We believe more in the value of planning than the resulting plan. For a 5-year project, it is very likely we will have to adjust our plans along the way. For this reason, we follow an agile methodology if not in the pure sense, then in the philosophical sense. We apply agile methods in our software development through iterative design, development, test and integration. We will leverage a central code repository (e.g., github) since a number of the modules (e.g., variability measurement) have overlapping functionality.

All of the tasks and the resulting work plan were designed to foster an iterative process where the overarching software framework development goal is to create 3 functioning prototypes in 18-month intervals: a Beta Phase I, a Beta Phase II, and a comprehensive Prototype. The iterative creation of methods that progress from empirical to interpolating to extrapolating means that we can adjust if the research does not proceed as expected or the server community landscape changes abruptly perhaps due to a disruptive technology. We are absolutely convinced that variability research such as ours will be critical at scale no matter what new technologies are developed. The heterogeneity of resources and the slow adoption of new disruptive technologies significantly reduce the risk of the research overall.

6 Results of Prior NSF Support

Ali R. Butt ***Title:*** *Pythia: An Application Analysis and Online Modeling Based Prediction Framework for Scalable Resource Management* [CNS-1405697] (Butt (PI)); Amount: $750,000 10/01/14 to 09/30/17; ***Intellectual Merit:*** Develop a performance oracle for large cloud distributed software frameworks, and study the interactions between frequency scaling and I/O performance, respectively. Numerous publications and software artifacts [19, 20, 38–43, 55–57, 74, 75, 79, 97, 99, 104–108, 115–118, 129, 130, 132–139, 147, 150–157, 183, 189–194, 207, 214, 215]. ***Broader Impact:*** Mentoring of seven PhD thesis (2 females), four MS thesis (2 females) twelve REU students (4 females), CS6204 Cloud Computing course, exploration of deployment of the tools at ORNL, and outreach to industry, e.g., IBM and NetApp.

Kirk W. Cameron ***Title:*** *CSR:Small: Exploiting slowdowns for speedup in power-scalable HPC systems.* [CNS-1422788], Amount: $500,000, 08/01/14 to 07/31/17. ***Intellectual merit:*** Developed methods to reduce and eliminate slowdowns in high-performance advanced computing systems. Results have appeared in highly competitive conferences (e.g., [48], [50]). ***Broader impacts:*** Suuports 1 PhD graduate, Mentoring 2 PhD students (1 female), ongoing collaborations with LLNL and ORNL, integration of findings in coursework.

Yili Hong Co-PI Hong was funded by [CMMI 1068933] ***Title:*** *Reliability Prediction Based on Dynamic Data Collected with Modern Technology* (Hong (PI); Amount: $210,234, 07/01/11 to 06/30/14). ***Intellectual merit:*** The PI developed methods for predicting field failures based on degradation data and dynamic covariate information. By using the dynamic covariate information, one can obtain more accurate reliability predictions, which are useful in system health monitoring and maintenance scheduling. ***Broader impacts:*** The research outcomes have the potential to improve the competitive position of US manufacturers, and have been disseminated through industrial collaborations, conference presentations, posting preprints online and publications in refereed journals (e.g., Hong et al., 2015; Meeker and Hong, 2014; Hong and Meeker, 2013 [88, 89, 126]).

Danfeng (Daphne) Yao (PI) ***Title:*** *Human-behavior driven malware detection* [CNS-0953638], Amount: $530,000, 04/01/2010 to 03/31/2015. ***Intellectual merit:*** The project designs dependence-based analysis models and methods for detecting stealthy system and network anomalies. The techniques enforce normal dependence relations in system behaviors and achieve high detection accuracy. ***Broader impacts:*** Building proactive system defenses secures the cyberspace, and contributes to national security. Three new security courses and a new educational security software system were developed. Publications in top security conferences and journals (e.g., [15, 205, 213]), one patent approved [208] and one CIP patent pending [209].

References

[1] Active tpc benchmarks. http://www.tpc.org/information/benchmarks.asp. Accessed: 2015-09-09.

[2] ChaosMonkey application description. https://github.com/Netflix/SimianArmy/wiki/Chaos-Monkey. Accessed: 2015-09-09.

[3] Computer science resources consortium. https://www.cs.vt.edu/CSRC/Welcome. Accessed: 2015-09-09.

[4] The green500. http://www.green500.org/. Accessed: 2015-09-09.

[5] Nas parallel benchmarks. https://www.nas.nasa.gov/publications/npb.html. Accessed: 2015-09-09.

[6] Standard performance evaluation corporation. https://www.spec.org/benchmarks.html. Accessed: 2015-09-09.

[7] UCSC-SOE-15-07: Tackling the Reproducibility Problem in Systems Research with Declarative Experiment Specifications research description. https://www.soe.ucsc.edu/research/technical-reports/UCSC-SOE-15-07. Accessed: 2015-09-09.

[8] Variability expedition. http://variability.ucsd.edu/. Accessed: 2015-09-22.

[9] What are advantages and disadvantages of real time operating systems. http://www.itrelease.com/2014/07/advantages-disadvantages-real-time-operating-systems/. Accessed: 2015-09-09.

[10] David B Adams, Layne T Watson, Omprakash Seresta, and Zafer Gürdal. Global/local iteration for blended composite laminate panel structure optimization subproblems. *Mechanics of Advanced Materials and Structures*, 14(2):139–150, 2007.

[11] Hakan Akkan, Michael Lang, and Lorie M. Liebrock. Stepping towards noiseless linux environment. In *Proceedings of the 2Nd International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '12, pages 7:1–7:7, New York, NY, USA, 2012. ACM.

[12] Nicholas A Allen, Clifford A Shaffer, Naren Ramakrishnan, Marc T Vass, and Layne T Watson. Improving the development process for eukaryotic cell cycle models with a modeling support environment. *Simulation*, 79(12):674–688, 2003.

[13] Darcy L. Allison, Craig C. Morris, Joseph A. Schetz, Rakesh K. Kapania, Layne T. Watson, and Joshua D. Deaton. Development of a multidisciplinary design optimization framework for an efficient supersonic air vehicle. *Advances in aircraft and spacecraft science*, pages 17–44, 2015.

[14] Hussain Almohri, Danfeng Yao, Layne Watson, and Xinming Ou. Security optimization of dynamic networks with probabilistic graph modeling and linear programming. 2014.

[15] Hussain M.J. Almohri, Danfeng Yao, and Dennis Kafura. Process authentication for high system assurance. *IEEE Transactions on Dependable and Secure Computing*, 11(2):168–180, 2014.

[16] Brandon Amos, Hamilton Turner, and Jonathan White. Applying machine learning classifiers to dynamic android malware detection at scale. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 1666–1671. IEEE, 2013.

[17] Brandon D Amos, David R Easterling, Layne T Watson, William I Thacker, Brent S Castle, and Michael W Trosset. Algorithm xxx: Qnstopquasinewton algorithm for stochastic optimization. 2014.

[18] M. Andrysco, D. Kohlbrenner, K. Mowery, R. Jhala, S. Lerner, and H. Shacham. On subnormal floating point and abnormal timing. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 623–639. IEEE, May 2015.

[19] Ali Anwar, Andrzej Kochut Anca Sailer, Charles O. Schulz, Alla Segal, and Ali R. Butt. Scalable metering for an affordable it cloud service management. In *Proceedings of the IEEE International Conference on Cloud Engineering*, IC2E '15, Tempe, AZ, March 2015.

[20] Ali Anwar, K. R. Krish, and Ali R. Butt. On the Use of Microservers in Supporting Hadoop Applications. In *Cluster Computing (CLUSTER), 2014 IEEE International Conference on*, Madrid, Spain, Sep 2014.

[21] Ali Anwar, Anca Sailer, Andrzej Kochut, Charles O Schulz, Alla Segal, and Ali R Butt. Cost-aware cloud metering with scalable service management infrastructure. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 285–292. IEEE, 2015.

[22] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[23] Rafael V Aroca, Glauco Caurin, and Sao Carlos-SP-Brasil. A real time operating systems (rtos) comparison. In *XXIX Congresso da Sociedade Brasileira de Computação*, 2009.

[24] Siro Arthur, Carsten Emde, and Nicholas Mc Guire. Assessment of the realtime preemption patches (rt-preempt) and their impact on the general purpose performance of the system. In *Proceedings of the 9th Real-Time Linux Workshop*, 2007.

[25] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[26] Anys Bacha and Radu Teodorescu. Dynamic reduction of voltage margins by leveraging on-chip ecc in itanium ii processors. *ACM SIGARCH Computer Architecture News*, 41(3):297–307, 2013.

[27] Michael Barabanov and Victor Yodaiken. Real-time linux. *Linux journal*, 23, 1996.

[28] Pete Beckman, Kamil Iskra, Kazutomo Yoshii, Susan Coghlan, and Aroon Nataraj. Benchmarking the effects of operating system interference on extreme-scale parallel machines. *Cluster Computing*, 11(1):3–16, March 2008.

[29] Thomas Bell and Barry Boehm. *Computer Performance Evaluation: Report of the 1973 NBS/ACM Workshop*. U.S. Government Printing Office, Washington D.C., 9 1973. An optional note.

[30] Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against aes. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, pages 201–215. Springer, 2006.

[31] Andrew Bortz and Dan Boneh. Exposing private information by timing web applications. In *Proceedings of the 16th international conference on World Wide Web*, pages 621–628. ACM, 2007.

[32] Julien Bourgeois and François Spies. Performance prediction of an nas benchmark program with chronosmix environment. In *Euro-Par 2000 Parallel Processing*, pages 208–216. Springer, 2000.

[33] Keith Bowman, James W Tschanz, Shih-Lien L Lu, Paolo Aseron, Muhammad M Khellah, Arijit Raychowdhury, Bibiche M Geuskens, Carlos Tokunaga, Chris B Wilkerson, Tanay Karnik, et al. A 45 nm resilient microprocessor core for dynamic variation tolerance. *Solid-State Circuits, IEEE Journal of*, 46(1):194–208, 2011.

[34] Jeremy H Brown and Brad Martin. How fast is fast enough? choosing between xenomai and linux for real-time applications. In *proc. of the 12th Real-Time Linux Workshop (RTLWS12)*, pages 1–17, 2010.

[35] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[36] Holger Brunst, Dieter Kranzlmüller, Matthias S. Muller, and Wolfgang E. Nagel. Tools for scalable parallel program analysis: vampir ng, marmot, and dewiz. *Int. J. Comput. Sci. Eng.*, 4(3):149–161, July 2009.

[37] Susan Burgee, Anthony A Giunta, Vladimir Balabanov, Bernard Grossman, William H Mason, Robert Narducci, Raphael T Haftka, and Layne T Watson. A coarse-grained parallel variable-complexity multidisciplinary optimization paradigm. *International Journal of High Performance Computing Applications*, 10(4):269–299, 1996.

[38] Ali R. Butt, Xing Fang, Y. Charlie Hu, and Samuel Midkiff. Java, peer-to-peer, and accountability: Building blocks for distributed cycle sharing. In *Proc. 3rd USENIX VM*, pages 163–176, San Jose, CA, May 2004.

[39] Ali R. Butt, Xing Fang, Y. Charlie Hu, Samuel Midkiff, and Jan Vitek. An open peer-to-peer infrastructure for cycle-sharing. In *Work-in-Progress 19th ACM SOSP*, Bolton Landing, NY, Oct. 2003.

[40] Ali R. Butt, Chris Gniady, and Y. Charlie Hu. The performance impact of kernel prefetching on buffer cache replacement algorithms. *IEEE Transactions on Computers*, 56(7):889–908, 2007.

[41] Ali R. Butt, Troy A. Johnson, Yili Zheng, and Y. Charlie Hu. Kosha: A peer-to-peer enhancement for the network file system. *Journal of Grid Computing: Special issue on Global and Peer-to-Peer Computing*, 4(3):323–341, 2006.

[42] Ali R. Butt, Nipoon Malhotra, Sunil Patro, and Y. Charlie Hu. On the equivalence of forward and reverse query cache proxying in peer-to-peer overlay networks. In *Proc. 9th WCW*, pages 169–181, Beijing, China, Oct. 2004.

[43] Ali R. Butt, Rongmei Zhang, and Y. Charlie Hu. A self-organizing flock of Condors. *Journal of Parallel and Distributed Computing*, 66(1):145–161, 2006.

[44] Zhenwei Cao, David R Easterling, Layne T Watson, Dong Li, Kirk W Cameron, and Wu-Chun Feng. Power saving experiments for large-scale global optimisation. *International Journal of Parallel, Emergent and Distributed Systems*, 25(5):381–400, 2010.

[45] Henri Casanova, Arnaud Legrand, and Martin Quinson. Simgrid: A generic framework for large-scale distributed experiments. In *Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on*, pages 126–131. IEEE, 2008.

[46] Saravanan Chandran, Prathmesh Kallurkar, Puneet Gupta, and Smruti R Sarangi. Architectural support for handling jitterin shared memory based parallel applications. *Parallel and Distributed Systems, IEEE Transactions on*, 25(5):1166–1176, 2014.

[47] Hung-Ching Chang, Bo Li, G. Back, A.R. Butt, and K.W. Cameron. Luc: Limiting the unintended consequences of power scaling on parallel transaction-oriented workloads. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 324–333, May 2015.

[48] Hung-Ching Chang, Bo Li, Godmar Back, Ali R. Butt, and Kirk Cameron. Luc: Limiting the unintended consequences of power scaling on parallel transaction-oriented workloads. In *Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium*, IPDPS '15, Hyderabad, India, May 2015.

[49] Hung-Ching Chang, Bo Li, Godmar Back, Ali R Butt, and Kirk W Cameron. Luc: Limiting the unintended consequences of power scaling on parallel transaction-oriented workloads.

[50] Hung-Ching Chang, Bo Li, M. Grove, and K.W. Cameron. How processor speedups can slow down i/o performance. In *Modelling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on*, pages 395–404, Sept 2014.

[51] Ang Chen, W. Brad Moore, Hanjun Xiao, Andreas Haeberlen, Linh Thi Xuan Phan, Micah Sherr, and Wenchao Zhou. Detecting covert timing channels with time-deterministic replay. In *Proceedings of 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14)*, October 2014.

[52] Yanpei Chen, Archana Sulochana Ganapathi, Rean Griffith, and Randy H Katz. Towards understanding cloud performance tradeoffs using statistical workload analysis and replay. *University of California at Berkeley, Technical Report No. UCB/EECS-2010-81*, 2010.

[53] Zhimin Chen, Ambuj Sinha, and Patrick Schaumont. Using virtual secure circuit to protect embedded software from side-channel attacks. *Computers, IEEE Transactions on*, 62(1):124–136, 2013.

[54] Elliott Ward Cheney and William Allan Light. *A course in approximation theory*, volume 101. American Mathematical Soc., 2009.

[55] Yue Cheng, Aayush Gupta, and Butt Ali R. An in-memory object caching framework with adaptive load balancing. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 4:1–4:16, New York, NY, USA, 2015. ACM.

[56] Yue Cheng, M. Safdar Iqbal, Aayush Gupta, and Ali R. Butt. Cast: Tiering storage for data analytics in the cloud. In *Proceedings of the 24rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '15, New York, NY, USA, 2015. ACM.

[57] Yue Cheng, Iqbal M. Safdar, Aayush Gupta, and Ali R. Butt. Pricing games for hybrid object stores in the cloud: Provider vs. tenant. In *7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15)*, Santa Clara, CA, Jul 2015. USENIX Association.

[58] Wu chun Feng and Kirk Cameron. The green500 list: Encouraging sustainable supercomputing. *Computer*, 40(12):50–55, 2007.

[59] Bogdan Florin Cornea and Julien Bourgeois. Performance prediction of distributed applications using block benchmarking methods. In *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, pages 183–190. IEEE, 2011.

[60] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. *LogP: Towards a realistic model of parallel computation*, volume 28. ACM, 1993.

[61] Robert Davis, Nick Merriam, and Nigel Tracey. How embedded applications using an rtos can stay within on-chip memory limits. In *12th EuroMicro Conference on Real-Time Systems*, pages 71–77. Citeseer, 2000.

[62] Pradipta De, Ravi Kothari, and Vijay Mann. Identifying sources of operating system jitter through fine-grained kernel instrumentation. In *Cluster Computing, 2007 IEEE International Conference on*, pages 331–340. IEEE, 2007.

[63] Pradipta De and Vijay Mann. jitsim: A simulator for predicting scalability of parallel applications in presence of os jitter. In Pasqua DAmbra, Mario Guarracino, and Domenico Talia, editors, *Euro-Par 2010 - Parallel Processing*, volume 6271 of *Lecture Notes in Computer Science*, pages 117–130. Springer Berlin Heidelberg, 2010.

[64] Augusto Born de Oliveira, Jean-Christophe Petkovich, Thomas Reidemeister, and Sebastian Fischmeister. Datamill: Rigorous performance evaluation made easy. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, pages 137–148. ACM, 2013.

[65] Nikil Dutt, Puneet Gupta, Alex Nicolau, Luis Angel D Bathen, and Mark Gottscho. Variability-aware memory management for nanoscale computing. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 125–132. IEEE, 2013.

[66] Derek L. Eager, Daniel J. Sorin, and Mary K. Vernon. Amva techniques for high service time variability. In *Proceedings of the 2000 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '00, pages 217–228, New York, NY, USA, 2000. ACM.

[67] David R Easterling, Layne T Watson, Michael L Madigan, Brent S Castle, and Michael W Trosset. Parallel deterministic and stochastic global minimization of functions with very many minima. *Computational Optimization and Applications*, 57(2):469–492, 2014.

[68] Dror G Feitelson. From repeatability to reproducibility and corroboration. *ACM SIGOPS Operating Systems Review*, 49(1):3–11, 2015.

[69] Rong Ge, Xizhou Feng, Wu-chun Feng, and Kirk W Cameron. Cpu miser: A performance-directed, run-time system for power-aware clusters. In *Parallel Processing, 2007. ICPP 2007. International Conference on*, pages 18–18. IEEE, 2007.

[70] Rong Ge, Xizhou Feng, Shuaiwen Song, Hung-Ching Chang, Dong Li, and Kirk W. Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 21(5):658–671, 2010.

[71] Rong Ge, Xizhou Feng, Shuaiwen Song, Hung-Ching Chang, Dong Li, and Kirk W Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *Parallel and Distributed Systems, IEEE Transactions on*, 21(5):658–671, 2010.

[72] Yuzhen Ge, EG Collins Jr, Layne T Watson, and LD Davis. An input normal form homotopy for the l 2 optimal model order reduction problem. *Automatic Control, IEEE Transactions on*, 39(6):1302–1305, 1994.

[73] M. Giampapa, T. Gooding, T. Inglett, and R.W. Wisniewski. Experiences with a lightweight supercomputer kernel: Lessons learned from blue gene's cnk. In *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, pages 1–10, Nov 2010.

[74] Chris Gniady, Ali R. Butt, and Y. Charlie Hu. Program-counter-based pattern classification in buffer caching. In *Proc. 6th USENIX OSDI*, pages 395–408, San Francisco, CA, Dec. 2004.

[75] Chris Gniady, Ali R. Butt, Y. Charlie Hu, and Yung-Hsiang Lu. Program counter-based prediction techniques for dynamic power management. *IEEE Transactions on Computers*, 55(6):641–658, 2006.

[76] Amit Goel, Constantinos Phanouriou, Frederick A Kamke, Calvin J Ribbens, Clifford A Shaffer, and Layne T Watson. Wbcsim: a prototype problem solving environment for wood-based composites simulations. *Engineering with Computers*, 15(2):198–210, 1999.

[77] Mark Gottscho, Luis Bathen, Nikil Dutt, Alex Nicolau, Puneet Gupta, et al. Vipzone: Hardware power variability-aware virtual memory management for energy savings. *Computers, IEEE Transactions on*, 64(5):1483–1496, 2015.

[78] Jianhua Gu, Jinhua Hu, Tianhai Zhao, and Guofei Sun. A new resource scheduling strategy based on genetic algorithm in cloud computing environment. *Journal of Computers*, 7(1):42–52, 2012.

[79] Abdul Hafeez, Waseem Asghar, M. Mustafa Rafique, Samir M. Iqbal, and Ali R. Butt. GPU-based Real-time Detection and Analysis of Biological Targets using Solid-state Nanopores. *Medical and Biological Engineering and Computing*, 50(6):605–615, 2012.

[80] Adam Hammouda, Andrew R. Siegel, and Stephen F. Siegel. Noise-tolerant explicit stencil computations for nonuniform process execution rates. *ACM Trans. Parallel Comput.*, 2(1):7:1–7:33, April 2015.

[81] Jian He, Alex Verstak, Masha Sosonkina, and Layne T Watson. Performance modeling and analysis of a massively parallel directpart 2. *International Journal of High Performance Computing Applications*, 23(1):29–41, 2009.

[82] Jian He, Alex Verstak, Layne T Watson, and Masha Sosonkina. Performance modeling and analysis of a massively parallel directpart 1. *International Journal of High Performance Computing Applications*, 23(1):14–28, 2009.

[83] Jian He, Layne T Watson, and Masha Sosonkina. Algorithm 897: Vtdirect95: serial and parallel codes for the global optimization algorithm direct. *ACM Transactions on Mathematical Software (TOMS)*, 36(3):17, 2009.

[84] Grant Ho, Dan Boneh, Lucas Ballard, and Niels Provos. Tick tock: Building browser red pills from timing side channels. In *8th USENIX Workshop on Offensive Technologies, WOOT*, volume 14, 2014.

[85] Torsten Hoefler and Roberto Belli. Scientific benchmarking of parallel computing systems. In *Proceedings of the 2015 ACM/IEEE Conference on Supercomputing*, SC '15, New York, NY, USA, 2015. ACM.

[86] Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Characterizing the influence of system noise on large-scale applications by simulation. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–11, Washington, DC, USA, 2010. IEEE Computer Society.

[87] Todd Hoff. Netflix: Continually test by failing servers with chaos monkey, 2010.

[88] Yili Hong, Yuanyuan Duan, William Q Meeker, Deborah L Stanley, and Xiaohong Gu. Statistical methods for degradation data with dynamic covariates information and an application to outdoor weathering data. *Technometrics*, (just-accepted):00–00, 2014.

[89] Yili Hong and William Q Meeker. Field-failure predictions based on failure-time data with dynamic covariate information. *Technometrics*, 55(2):135–149, 2013.

[90] Jinhua Hu, Jianhua Gu, Guofei Sun, and Tianhai Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*, pages 89–96. IEEE, 2010.

[91] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. On the performance variability of production cloud services. In *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, pages 104–113. IEEE, 2011.

[92] Michiel JW Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117(1):35–43, 1999.

[93] Hui Jin, Xi Yang, Xian-He Sun, and Ioan Raicu. Adapt: Availability-aware mapreduce data placement for non-dedicated distributed computing. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 516–525. IEEE, 2012.

[94] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[95] Bradley Jones and Christopher J Nachtsheim. A class of three-level designs for definitive screening in the presence of second-order effects. *Journal of Quality Technology*, 43(1):1–15, 2011.

[96] Ana Justel, Daniel Peña, and Rubén Zamar. A multivariate kolmogorov-smirnov test of goodness of fit. *Statistics & Probability Letters*, 35(3):251–259, 1997.

[97] Aleksandr Khasymski, M. Mustafa Rafique, Ali R. Butt, Sudharshan S. Vazhkudai, and Dimitrios S. Nikolopoulos. On the Use of GPUs in Realizing Cost-Effective Distributed RAID. In *Proc. MASCOTS*. IEEE, 2012.

[98] Youngtaek Kim, Lizy Kurian John, Sanjay Pant, Srilatha Manne, Michael Schulte, W Lloyd Bircher, and Madhu Saravana Sibi Govindan. Audit: Stress testing the automatic way. In *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, pages 212–223. IEEE, 2012.

[99] Pavan Konanki and Ali R. Butt. Flexicache: A flexible interface for customizing Linux file system buffer cache replacement policies. In *Work-in-Progress, 5th USENIX FAST*, San Jose, CA, Feb. 2007.

[100] Gilad Koren and Dennis Shasha. Skip-over: Algorithms and complexity for overloaded systems that allow skips. In *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, pages 110–117. IEEE, 1995.

[101] Donald Kossmann, Tim Kraska, and Simon Loesing. An evaluation of alternative architectures for transaction processing in the cloud. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 579–590. ACM, 2010.

[102] William TC Kramer and Clint Ryan. *Performance variability of highly parallel architectures*. Springer, 2003.

[103] Denitza T Krasteva, Layne T Watson, Chuck A Baker, Bernard Grossman, William H Mason, and Raphael T Haftka. Distributed control parallelism in multidisciplinary aircraft design. 1998.

[104] K. R. Krish, Ali Anwar, and Ali R. Butt. hatS: A Heterogeneity-Aware Tiered Storage for Hadoop. In *The 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2014.

[105] K. R. Krish, Ali Anwar, and Ali R. Butt. Sched: A Heterogeneity-Aware Hadoop Workflow Scheduler. In *Proc. 22nd IEEE/ACM MASCOTS*, Paris, France, Sep. 2014.

[106] K. R. Krish, Aleksandr Khasymski, Ali R. Butt, Sameer Tiwari, and Milind Bhandarkar. Aptstore:dynamic storage management for hadoop. In *Proceedings of the IEEE International Conference on cloud computing Technology and science*, CloudCom '13, Bristol, UK, 2013.

[107] K. R. Krish, Aleksandr Khasymski, Guanying Wang, Ali R. Butt, and Gaurav Makkar. On the use of shared storage in shared-nothing environments. In *Proceedings of the IEEE International Conference on Big Data*, BigData '13, Santa Clara, CA, 2013.

[108] K. R. Krish, Guanying Wang, Puranjoy Bhattacharjee, Ali R. Butt, and Chris Gniady. On Reducing Energy Management Delays in Disks. *Journal of Parallel and Distributed Computing*, 73(6):823–835, June 2013.

[109] C Mani Krishna. *Real-Time Systems*. Wiley Online Library, 1999.

[110] Klaus-Dieter Lange. Identifying shades of green: The specpower benchmarks. *Computer*, 42(3):95–97, 2009.

[111] Jong N Lee, Frederick A Kamke, and Layne T Watson. Simulation of the hot-pressing of a multilayered wood strand composite. *Journal of composite materials*, 41(7):879–904, 2007.

[112] Zhongyuan Lee, Ying Wang, and Wen Zhou. A dynamic priority scheduling algorithm on service request scheduling in cloud computing. In *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, volume 9, pages 4665–4669. IEEE, 2011.

[113] Bing Li, A Meina Song, and Junde Song. A distributed qos-constraint task scheduling scheme in cloud computing environment: model and algorithm. *AISS: Advances in Information Sciences and Service Sciences*, 4(5):283–291, 2012.

[114] Dong Li, Bronis R De Supinski, Martin Schulz, Dimitrios S Nikolopoulos, and Kirk W Cameron. Strategies for energy-efficient resource management of hybrid programming models. *Parallel and Distributed Systems, IEEE Transactions on*, 24(1):144–157, 2013.

[115] Min Li, Shravan Gaonkar, Ali R. Butt, Deepak Kenchammana, and Kaladhar Voruganti. Cooperative Storage-Level De-Duplication for I/O Reduction in Virtualized Data Centers. In *Proc. MASCOTS*. IEEE, 2012.

[116] Min Li, Dinesh Subhraveti, Ali R. Butt, Aleksandr Khasymski, and Prasenjit Sarkar. CAM: A Topology Aware Minimum Cost Flow Based Resource Manager for MapReduce Applications in the Cloud. In *The 20th International Symposium on High Performance Distributed Computing*, pages 159–170, 2012.

[117] Min Li, Sudharshan S Vazhkudai, Ali R Butt, Fei Meng, Xiaosong Ma, Youngjae Kim, Christian Engelmann, and Galen Shipman. Functional Partitioning to Optimize End-to-End Performance on Many-core Architectures. In *Proc. 2010 Supercomputing Conference*, November 2010.

[118] Min Li, Liangzhao Zeng, Shicong Meng, Jian Tan, Li Zhang, Ali R. Butt, and Nicholas Fuller. MRONLINE: MapReduce Online Performance Tuning. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pages 165–176, New York, NY, USA, 2014.

[119] Lei Liu, Guanhua Yan, Xinwen Zhang, and Songqing Chen. Virusmeter: Preventing your cellphone from spies. In *Recent Advances in Intrusion Detection*, pages 244–264. Springer, 2009.

[120] Jay Lofstead, Fang Zheng, Qing Liu, Scott Klasky, Ron Oldfield, Todd Kordenbrock, Karsten Schwan, and Matthew Wolf. Managing variability in the io performance of petascale storage systems. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE Computer Society, 2010.

[121] Jay Lofstead, Fang Zheng, Qing Liu, Scott Klasky, Ron Oldfield, Todd Kordenbrock, Karsten Schwan, and Matthew Wolf. Managing variability in the io performance of petascale storage systems. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE Computer Society, 2010.

[122] Atieh Lotfi, Abbas Rahimi, Luca Benini, and Rajesh K Gupta. Aging-aware compilation for gp-gpus. *ACM Transactions on Architecture and Code Optimization (TACO)*, 12(2):24, 2015.

[123] Robert Lucas, James Ang, Shekhar Borkar, William Carlson, Laura Carrington, George Chiu, Robert Colwell, William Dally, Jack Dongarra, Al Geist, Gary Grider, Rud Haring, Jeffrey Hittinger, Adolfy Hoisie, Dean Klein, Peter Kogge, Richard Lethin, Vivek Sarkar, Robert Schreiber, John Shalf, Thomas Sterling, and Rick Stevens. Ascac subcommittee for the top ten exascale research challenges. 2014.

[124] Mastura Diana Marieska, Achmad Imam Kistijantoro, and Muhammad Subair. Analysis and benchmarking performance of real time patch linux and xenomai in serving a real time application. In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[125] Fabrizio Marozzo, Domenico Talia, and Paolo Trunfio. P2p-mapreduce: Parallel data processing in dynamic cloud environments. *Journal of Computer and System Sciences*, 78(5):1382–1402, 2012.

[126] William Q Meeker and Yili Hong. Reliability meets big data: Opportunities and challenges. *Quality Engineering*, 26(1):102–116, 2014.

[127] Michael Melkonian. Get by without an rtos. *Embedded Systems Programming*, 13(10), 2000.

[128] Robert Melville, Shahriar Moinian, Peter Feldmann, and Layne Watson. *Sframe: An efficient system for detailed dc simulation of bipolar analog integrated circuits using continuation methods*. Springer, 1993.

[129] Chreston Miller, Patrick Butler, Ankur Shah, and Ali R. Butt. PeerStripe: A P2P-based large-file storage for desktop grids. In *Proc. IEEE HPDC*, Monterey, CA, June 2007.

[130] Chreston Miller, Ali R. Butt, and Patrick Butler. On utilization of contributory storage in desktop grids. In *Proc. International Parallel and Distributed Processing Symposium (IPDPS'08)*, Miami, FL, Apr. 2008.

[131] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[132] Henry Monti, Ali R. But, and Sudharshan S. Vazhkudai. On Timely Staging of HPC Job Input Data. *IEEE Transactions on Parallel and Distributed Systems*, 2012.

[133] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. A result-data offloading service for hpc centers. In *Proc. ACM Petascale Data Storage Workshop*, Reno, NV, Nov. 2007.

[134] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. Just-in-time staging of large input data for supercomputing jobs. In *Proc. ACM Petascale Data Storage Workshop*, Austin, TX, Nov. 2008.

[135] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. Timely offloading of result-data in hpc centers. In *Proc. 22nd ACM International Conference on Supercomputing (ICS'08)*, Kos, Greece, Jun. 2008.

[136] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. /Scratch as a Cache: Rethinking HPC Center Scratch Storage. In *Proc. ACM ICS*, New York, NY, 2009.

[137] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. Reconciling Scratch Space Consumption, Exposure, and Volatility to Achieve Timely Staging of Job Input Data. In *Proc. IPDPS*, Atlanta, GA, 2010.

[138] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. CATCH: A Cloud-based Adaptive Data Transfer Service for HPC. In *Proc. IPDPS*, Anchorage, AK, 2011.

[139] Henry Monti, Ali R. Butt, and Sudharshan S. Vazhkudai. Timely result-data offloading for improved hpc center scratch provisioning and serviceability. *IEEE Transactions on Parallel and Distributed Systems*, 22(8):1307–1322, 2011.

[140] Alessandro Morari, Roberto Gioiosa, Robert W Wisniewski, Francisco J Cazorla, and Mateo Valero. A quantitative analysis of os noise. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 852–863. IEEE, 2011.

[141] Ronald Mraz. Reducing the variance of point to point transfers in the ibm 9076 parallel computer. In *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 620–629. IEEE Computer Society Press, 1994.

[142] J.T. Oden, O. Ghattas, J.L. King, B.I. Schneider, K. Bartschat, F. Darema, J. Drake, T. Dunning, D. Estep, S. Glotzer, M. Gurnis, C.R. Johnson, D.S. Katz, D. Keyes, S. Kiesler, S. Kim, J. Kinter, G. Klimeck, C.W. McCurdy, R. Moser, C. Ott, A. Patra, L. Petzold, T. Schlick, K. Schulten, V. Stodden, J. Tromp, M. Wheeler, S.J. Winter, C. Wu, and K. Yelick. Cyber science and engineering: A report of the national science foundation advisory committee for cyberinfrastructure task force on grand challenges. 2011.

[143] Jiannan Ouyang, Brian Kocoloski, John R. Lange, and Kevin Pedretti. Achieving performance isolation with lightweight co-kernels. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '15, pages 149–160, New York, NY, USA, 2015. ACM.

[144] Balaji Palanisamy, Aameek Singh, Ling Liu, and Bhushan Jain. Purlieus: locality-aware resource allocation for mapreduce in a cloud. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 58. ACM, 2011.

[145] Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asci q. In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing*, SC '03, pages 55–, New York, NY, USA, 2003. ACM.

[146] Martin Pifflˆ and Ernst Stadlober. The depth-design: An efficient generation of high dimensional computer experiments. *Journal of Statistical Planning and Inference*, 164:10–26, 2015.

[147] Ramya Prabhakar, Sudharshan S. Vazhkudai, Youngjae Kim, Ali R. Butt, Min Li, and Mahmut Kandemir. Provisioning a Multi-tiered Data Staging Area for Extreme-Scale Machines. In *Proc. ICDCS*, pages 1–12, 2011.

[148] Sundeep Prakash and Rajive L Bagrodia. Mpi-sim: using parallel simulation to evaluate mpi programs. In *Proceedings of the 30th conference on Winter simulation*, pages 467–474. IEEE Computer Society Press, 1998.

[149] Kishore Kumar Pusukuri, Rajiv Gupta, and Laxmi N Bhuyan. Thread tranquilizer: Dynamically reducing performance variation. *ACM Transactions on Architecture and Code Optimization (TACO)*, 8(4):46, 2012.

[150] M. Mustafa Rafique, Ali R. Butt, and Dimitrios S. Nikolopoulos. Dma-based prefetching for i/o-intensive workloads on the cell architecture. In *Proc. ACM International Conferene on Computing Frontiers (CF '08)*, Ischia, Italy, May 2008.

[151] M Mustafa Rafique, Ali R Butt, and Dimitrios S Nikolopoulos. Designing Accelerator-Based Distributed Systems for High Performance. In *The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2010.

[152] M Mustafa Rafique, Ali R Butt, and Dimitrios S Nikolopoulos. A Capabilities-aware Framework for using Computational Accelerators in Data-intensive Computing. *Journal of Parallel and Distributed Computing*, 71(2):185 – 197, 2011.

[153] M Mustafa Rafique, Ali R Butt, and Eli Tilevich. Reusable Software Components for Accelerator-based Clusters. *Journal of Systems and Software*, 84(7):1071–1081, 2011.

[154] M. Mustafa Rafique, Benjamin Rose, Ali R. Butt, and Dimitrios S. Nikolopoulos. CellMR: A Framework for Supporting MapReduce on Asymmetric Cell-Based Clusters. In *Proc. IEEE IPDPS*, Rome, Italy, 2009.

[155] M. Mustafa Rafique, Benjamin Rose, Ali R. Butt, and Dimitrios S. Nikolopoulos. Supporting MapReduce on Large-Scale Asymmetric Multi-core Clusters. *ACM Operating Systems Review*, 43(2), 2009.

[156] M.M. Rafique, S. Cadambi, K. Rao, A.R. Butt, and S. Chakradhar. Symphony: A Scheduler for Client-Server Applications on Coprocessor-Based Heterogeneous Clusters. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 353–362, Sep 2011.

[157] M.M. Rafique, N. Ravi, S. Cadambi, A.R. Butt, and S. Chakradhar. Power Management for Heterogeneous Clusters: An Experimental Study. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8, Jul 2011.

[158] A. Rahimi, D. Cesarini, A. Marongiu, R.K. Gupta, and L. Benini. Task scheduling strategies to mitigate hardware variability in embedded shared memory clusters. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6, June 2015.

[159] Abbas Rahimi, Luca Benini, and Rajesh K Gupta. Aging-aware compiler-directed vliw assignment for gpgpu architectures. In *Proceedings of the 50th Annual Design Automation Conference*, page 16. ACM, 2013.

[160] Azar Rahimi, Amirali Ghofrani, Miguel Angel Lastras-Montano, Kwang-Ting Cheng, Luca Benini, and Rajesh K Gupta. Energy-efficient gpgpu architectures via collaborative compilation and memristive memory-based computing. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6. IEEE, 2014.

[161] Vijay Janapa Reddi, Svilen Kanev, Wonyoung Kim, Simone Campanoni, Michael D Smith, Gu-Yeon Wei, and David Brooks. Voltage smoothing: Characterizing and mitigating voltage noise in production processors via software-guided thread scheduling. In *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, pages 77–88. IEEE, 2010.

[162] Robert Ricci, Gary Wong, Leigh Stoller, Kirk Webb, Jonathon Duerig, Keith Downie, and Mike Hibler. Apt: A platform for repeatable research in computer science. *ACM SIGOPS Operating Systems Review*, 49(1):100–107, 2015.

[163] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 199–212. ACM, 2009.

[164] Steven Rostedt and Darren V Hart. Internals of the rt patch. In *Proceedings of the Linux symposium*, volume 2, pages 161–172. Citeseer, 2007.

[165] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, pages 91–92, New York, NY, USA, 2002. ACM.

[166] Rafael H Saavedra and Alan J Smith. Analysis of benchmark characteristics and benchmark performance prediction. *ACM Transactions on Computer Systems (TOCS)*, 14(4):344–384, 1996.

[167] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment*, 3(1-2):460–471, 2010.

[168] B.W. Settlemyer, S.W. Hodson, J.A. Kuehn, and S.W. Poole. Confidence: Analyzing performance with empirical probabilities. In *Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), 2010 IEEE International Conference on*, pages 1–8, Sept 2010.

[169] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss. "andromaly": a behavioral malware detection framework for Android devices. *Journal of Intelligent Information Systems*, 38(1):161–190, 2012.

[170] John Shalf, Sudip Dosanjh, and John Morrison. Exascale computing technology challenges. In *Proceedings of the 9th International Conference on High Performance Computing for Computational Science*, VECPAR'10, pages 1–25, Berlin, Heidelberg, 2011. Springer-Verlag.

[171] Manu Shantharam, Padma Raghavan, and Mahmut Kandemir. Hybrid techniques for fast multicore simulation. In *Proceedings of the 15th International Euro-Par Conference on Parallel Processing*, Euro-Par '09, pages 122–134, Berlin, Heidelberg, 2009. Springer-Verlag.

[172] Kai Shen. Request behavior variations. *ACM SIGARCH Computer Architecture News*, 38(1):103–116, 2010.

[173] Sameer Shende and Allen D. Malony. Integration and applications of the tau performance system in parallel java environments. In *Proceedings of the 2001 Joint ACM-ISCOPE Conference on Java Grande*, JGI '01, pages 87–96, New York, NY, USA, 2001. ACM.

[174] Xiaokui Shu, Danfeng Yao, and Naren Ramakrishnan. Unearthing stealthy program attacks buried in extremely long execution paths. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, October 2015.

[175] David Shue, Michael J Freedman, and Anees Shaikh. Performance isolation and fairness for multi-tenant cloud storage. In *OSDI*, volume 12, pages 349–362, 2012.

[176] David Skinner and William Kramer. Understanding the causes of performance variability in hpc workloads. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 137–149. IEEE, 2005.

[177] Maria Sosonkina, Donald CS Allison, and Layne T Watson. Parallel adaptive gmres implementations for homotopy methods. *SIAM Journal on Optimization*, 9(4):1149–1158, 1999.

[178] MASHA SOSONKTNA, Donald CS Allison, and Layne T Watson. Scalability analysis of parallel gmres implementations. *Parallel Algorithms and Application*, 17(4):263–284, 2002.

[179] John A Stankovic, Marco Spuri, Krithi Ramamritham, and Giorgio C Buttazzo. *Deadline scheduling for real-time systems: EDF and related algorithms*, volume 460. Springer Science & Business Media, 2012.

[180] Shoaib Sufi, Neil Chue Hong, Simon Hettrick, Mario Antonioletti, Stephen Crouch, Alexander Hay, Devasena Inupakutika, Mike Jackson, Aleksandra Pawlik, Giacomo Peru, et al. Software in reproducible research: advice and best practice collected from experiences at the collaborations workshop. In *Proceedings of the 1st ACM SIGPLAN Workshop on Reproducible Research Methodologies and New Publication Models in Computer Engineering*, page 2. ACM, 2014.

[181] David Sundaram-Stukel and Mary K Vernon. Predictive analysis of a wavefront application using loggp. *ACM SIGPLAN Notices*, 34(8):141–150, 1999.

[182] Vahid Tabatabaee, Ananta Tiwari, and Jeffrey K Hollingsworth. Parallel parameter tuning for applications with performance variability. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 57. IEEE Computer Society, 2005.

[183] Ali R. Butt Tariq Kamal, Keith Bisset, and Madhav Marathe. Cost estimation of parallel constrained producer-consumer algorithms. In *Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed and Network-based Processing*, PDP '15, Turku, Finland, Feb. 2015.

[184] William I Thacker, Jingwei Zhang, Layne T Watson, Jeffrey B Birch, Manjula A Iyer, and Michael W Berry. Algorithm 905: Sheppack: Modified shepard algorithm for interpolation of scattered multivariate data. *ACM Transactions on Mathematical Software (TOMS)*, 37(3):34, 2010.

[185] Wenhong Tian, Yong Zhao, Yuanliang Zhong, Minxian Xu, and Chen Jing. A dynamic and integrated load-balancing scheduling algorithm for cloud datacenters. In *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pages 311–315. IEEE, 2011.

[186] Arjan JC Van Gemund. Symbolic performance modeling of parallel systems. *Parallel and Distributed Systems, IEEE Transactions on*, 14(2):154–165, 2003.

[187] Jan Vitek and Tomas Kalibera. Repeatability, reproducibility, and rigor in systems research. In *Proceedings of the ninth ACM international conference on Embedded software*, pages 33–38. ACM, 2011.

[188] C-Y Wang and LT Watson. Equilibrium of heavy elastic cylindrical shells. *Journal of Applied Mechanics*, 48(3):582–586, 1981.

[189] Guanying Wang, Ali R. Butt, Chris Gniady, and Puranjoy Bhattacharjee. A Light-weight Approach to Reducing Energy Management Delays in Disks. In *Proc. 1st IEEE International Green Computing Conference (IGCC)*, Chicago, IL, 2010.

[190] Guanying Wang, Ali R. Butt, Henry Monti, and Karan Gupta. Towards Synthesizing Realistic Workload Traces for Studying the Hadoop Ecosystem. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 400–408, July 2011.

[191] Guanying Wang, Ali R. Butt, Prashant Pandey, and Karan Gupta. A simulation approach to evaluating design decisions in mapreduce setups. In *Proc. 17th IEEE/ACM MASCOTS*, London, UK, Sep. 2009.

[192] Guanying Wang, Ali R. Butt, Prashant Pandey, and Karan Gupta. Using realistic simulation for performance analysis of mapreduce setups. In *Proc. 1st ACM LSAP '09*, New York, NY, USA, Jun. 2009.

[193] Guanying Wang, Aleksandr Khasymski, Ali R. Butt, and K. R. Krish. Towards improving mapreduce task scheduling using online simulation based predictions. In *Proceedings of the 19th IEEE International Conference on Parallel and Distributed Systems*, ICPADS '13, Seoul, Korea, 2013.

[194] Guanying Wang, Aleksandr Khasymski, Ali R. Butt, and K. R. Krish. Towards improving mapreduce task scheduling using online simulation based predictions. In *IEEE Mascots 2013*, San Francisco, CA, 2013.

[195] Jian-zong Wang, Peter Varman, and Chang-sheng Xie. Optimizing storage performance in public cloud platforms. *Journal of Zhejiang University SCIENCE C*, 12(12):951–964, 2011.

[196] Jianzong Wang, Peter Varman, and Changsheng Xie. Avoiding performance fluctuation in cloud storage. In *High Performance Computing (HiPC), 2010 International Conference on*, pages 1–9. IEEE, 2010.

[197] Sarp Oral Feiyi Wang, David A Dillow, Ross Miller, Galen M Shipman, Don Maxwell, and Dave Henseler Jeff Becklehimer Jeff Larkin. Reducing application runtime variability on jaguar xt5. 2010.

[198] Layne T Watson and Chuck A Baker. A fully-distributed parallel global search algorithm. *Engineering Computations*, 18(1/2):155–169, 2001.

[199] LT Watson, TY Li, and CY Wang. Fluid dynamics of the elliptic porous slider. *Journal of Applied Mechanics*, 45(2):435–436, 1978.

[200] R Clint Whaley, Antoine Petitet, and Jack J Dongarra. Automated empirical optimizations of software and the atlas project. *Parallel Computing*, 27(1):3–35, 2001.

[201] Paul N Whatmough, Shidhartha Das, Zacharias Hadjilambrou, and David M Bull. 14.6 an all-digital power-delivery monitor for analysis of a 28nm dual-core arm cortex-a57 cluster. In *Solid-State Circuits Conference-(ISSCC), 2015 IEEE International*, pages 1–3. IEEE, 2015.

[202] Britton Wolfe, Karim Elish, and Danfeng Yao. High precision screening for android malware with dimensionality reduction. In *Machine Learning and Applications (ICMLA), 2014 13th International Conference on*, pages 21–28. IEEE, 2014.

[203] Nicholas J Wright, Shava Smallen, Catherine Mills Olschanowsky, Jim Hayes, and Allan Snavely. Measuring and understanding variation in benchmark performance. In *DoD High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC), 2009*, pages 438–443. IEEE, 2009.

[204] Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–9. IEEE, 2010.

[205] Kui Xu, Huijun Xiong, Chehai Wu, Deian Stefan, and Danfeng Yao. Data-provenance verification for secure hosts. *IEEE Trans. Dependable Sec. Comput.*, 9(2):173–183, 2012.

[206] Kui Xu, Danfeng Daphne Yao, Barbara G. Ryder, and Ke Tian. Probabilistic program modeling for high-precision anomaly classification. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 497–511, July 2015.

[207] Luna Xu, Min Li, and Ali R. Butt. Gerbil: Mpi+yarn. In *Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGrid '15, Shenzhen, Guangdong, China, May 2015.

[208] Danfeng Yao, Deian Stefan, and Chehai Wu. Systems and methods for malware detection, 06 2014.

[209] Danfeng Yao and Hao Zhang. Detection of stealthy malware activities with transitional causality and scalable triggering relation discovery, 05 2014.

[210] Thomas Zacharia, Jim Kinter, Rob Pennington, Ron Cohen, Larry Davis, Tiziana Di Matteo, Bill Harrod, George Karniadakis, Rubin Landau, Rich Loft, Michael Macy, Dick McCombie, Dave Randall, Steve Scott, Horst Simon, Thomas Sterling, Theresa Windus, and Rob Pennington. Report of the high-performance computing task force. 2011.

[211] Matei Zaharia, Andy Konwinski, Anthony D Joseph, Randy H Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *OSDI*, volume 8, page 7, 2008.

[212] Jidong Zhai, Wenguang Chen, and Weimin Zheng. Phantom: predicting performance of parallel applications on large-scale parallel machines using a single node. In *ACM Sigplan Notices*, volume 45, pages 305–314. ACM, 2010.

[213] Hao Zhang, Danfeng (Daphne) Yao, and Naren Ramakrishnan. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In Shiho Moriai, Trent Jaeger, and Kouichi Sakurai, editors, *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 39–50. ACM, 2014.

[214] Rongmei Zhang, Ali R. Butt, and Y.Charlie Hu. Metastream: Topology-aware peer-to-peer on-demand streaming. In *Proc. IFIP Networking Conference*, pages 1–14, Waterloo, Canada, May 2005.

[215] Zhao Zhao, Guanying Wang, Ali R. Butt, Maleq Khan, V. S. Anil Kumar, and Madhav V. Marathe. SAHAD: Subgraph Analysis in Massive Networks using Hadoop. In *The 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Shanghai, China, May 2012.

[216] Jason W Zwolak, John J Tyson, and Layne T Watson. Parameter estimation for a mathematical model of the cell cycle in frog eggs. *Journal of computational biology*, 12(1):48–63, 2005.