Special Session Paper: Data Analytics Enables Energy-Efficiency and Robustness: From Mobile to Manycores, Datacenters, and Networks

Sudeep Pasricha[*], Janardhan Rao Doppa[†], Krishnendu Chakrabarty[§], Saideep Tiku[*], Daniel Dauwe[*], Shi Jin[§], Partha Pratim Pande[†]
[*]Colorado State University, Fort Collins, CO
[†]Washington State University, Pullman, WA
[§]Duke University, Durham, NC
sudeep@colostate.edu, jana@eecs.wsu.edu, krish@duke.edu, saideep@rams.colostate.edu, ddauwe@rams.colostate.edu, sj137@ee.duke.edu, pande@eecs.wsu.edu

Abstract *– The amount of data generated and collected across computing platforms every day is not only enormous, but growing at an exponential rate. Advanced data analytics and machine-learning techniques have become increasingly essential to analyze and extract meaning from such "Big Data". These techniques can be very useful to detect patterns and trends to improve the operational behavior of computing platforms, but they also introduce a number of outstanding challenges: (1) How can we design and deploy data analytics and learning mechanisms to improve energy-efficiency in IoT and mobile devices, without introducing significant software overheads? (2) How to use machine learning and analytics techniques for effective design-space exploration during manycore chip design? (3) How can data analytics and learning improve the reliability and energy-efficiency of large-scale cloud datacenters, to cost-effectively support connected embedded and IoT platforms? (4) How can data analytics detect anomalies and increase robustness in the network backbone of emerging cloud datacenter networks? In this paper, we discuss these outstanding problems and describe far-reaching solutions applicable across the interconnected ecosystem of IoT and mobile devices, manycore chips, datacenters, and networks.*

Categories and Subject Descriptors: • Computing methodologies ~Machine learning • Computer systems organization~Embedded and cyber-physical systems • Computer systems organization~Dependable and fault-tolerant systems and networks • Hardware~Power and energy

Keywords: data analytics, mobile computing, manycore chips, datacenters, networks, energy-efficiency, robustness

1. INTRODUCTION

It is well known that we are in the midst of a Big Data revolution, with a rapidly expanding ecosystem of diverse sources of massive datasets. Recent conservative studies estimate that global datacenter traffic was 4.7 zettabytes (ZB) in 2015, and is expected to double every two years [1]. This increase in data over the past decade has been fueled by the proliferation of embedded Internet of Things (IoT) devices and mobile computing, which in turn have spurred the growth in cloud computing and the global data centers required to support the paradigm. The generated data can be structured (e.g., financial, electronic medical records), semi-structured (e.g., tweets, emails), unstructured (e.g., audio, video), or real-time (e.g., network traces, monitoring logs). All of these types of data have the potential to provide invaluable insights, if organized and analyzed appropriately. The analyses of such large and diverse datasets is fast emerging as an indispensable tool for innovations in various domains such as healthcare, business process optimization, and social-network-based recommendations.

Unfortunately, it is projected that data growth in the coming years will largely outpace foreseeable improvements in the cost, reliability, energy-efficiency, and performance of computing infrastructure such as manycore processing subsystems, mobile platforms, networks, and data centers. Thus, the design of future computing platforms requires significant innovations, to overcome increased energy costs and failures due to high computational power and increasing fault susceptibility, respectively, when processing and analyzing large datasets. Fortunately, system designers that are today grappling with the challenge of efficiently supporting large datasets and data analytics workloads can themselves benefit from data analytics techniques to architect better computing platforms.

In this paper, we argue that data analytics can help hardware designers and system architects in the design, control, and testing of emerging computing systems at different levels of abstraction to achieve energy-efficiency and robustness for the Big Data era. For instance, in the domain of mobile computing, massive amount of data is typically collected about user-device interactions and usage. How can this data be analyzed to improve energy efficiency and robustness? Similarly, in the Electronic Design Automation (EDA) domain, a large amount of data is available corresponding to simulation traces of chip-based designs at design-time, and sensing information from various in-situ sensors at run-time. How can we utilize this data to improve energy-efficiency and robustness of manycore processing? Datacenters and large-scale networks are at the forefront of handling massive volumes of data today. Can data analytics help improve their energy-efficiency and robustness?

In the rest of the paper, we discuss and explore the intriguing possibility of using data analytics as an enabler for energy-efficiency and robustness, across the interconnected domains of mobile computing (Section 2), manycore chip design (Section 3), datacenters (Section 4), and large-scale networks (Section 5).

CODES/ISSS '17 Companion, October 15–20, 2017, Seoul, Republic of Korea © 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5185-0/17/10…$15.00
https://doi.org/10.1145/3125502.3125560

2. DATA ANALYTICS IN MOBILE COMPUTING

There has been an increasing trend of people becoming dependent on mobile devices for communication and data access. According to a recent study [2], 99% of adults in the USA between 18-29 years of age own a smartphone. Another study from CISCO [3] in 2016 suggests that on an average 10.7 exabytes of mobile data traffic is offloaded each month from mobile phones to datacenters. The study further forecasts that this value is set to increase 8-fold by the year 2021. As the data processed and produced by mobile devices has been increasing, device battery lifetime has been steadily decreasing. Most portable devices today make use of lithium-ion polymer batteries that have been used in electronics since the mid 1990's. Although lithium-ion battery technology and capacity has improved over the years, it still cannot keep pace with the energy demands of today's mobile devices. Moreover, the market demand for thin and lightweight devices means that large batteries with high capacity cannot be integrated into most mobile devices.

As battery lifetime directly impacts the robustness and availability of mobile devices, which is critical in many scenarios (e.g., emergency communication between first responders after natural disasters), it has motivated researchers to focus on improving the energy-efficiency of mobile devices. A promising direction is to utilize analytics techniques on the data collected by mobile device users, to deliver insights into user behavior, and use that information for predictive data analytics to enhance energy-efficiency. The remainder of this section discusses several such techniques.

2.1. Techniques for Energy-Efficient Mobile Systems

To create energy-efficient mobile devices, it is important to first identify the components that are major contributors towards poor battery life. There are three major components of the smartphone that utilize most of its energy: the display, the processing (CPU/GPU) subsystem, and the various wireless radios (e.g., Wi-Fi, GPS, 4G/LTE). Any framework to optimize energy-efficiency must address the energy inefficiencies in these components.

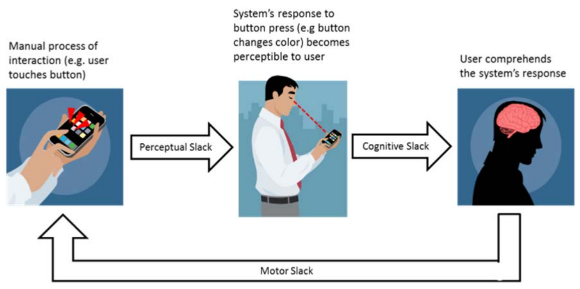
2.1.1. AURA framework for display and CPU management

Our AURA framework [4], [5] was one of the first to use data analytics to reduce energy costs for both the display and the processing subsystems. The framework involved an app-aware and user-aware energy optimization middleware that was based on applying analytics techniques to user-device interaction data. AURA integrates a runtime monitor that captures data to exploit user-specific and app-specific interaction slack to reduce processing energy costs. Interaction slack (Figure 1(a)) refers to the sum of the unused times between when a user first perceives a change on the display (perceptual slack) due to a previous interaction (e.g., a button on the screen changes color), then comprehends what the response "actually" represents (cognitive slack), and finally interacts with application again by touching the screen using his/her fingers (motor slack). By predicting this interaction slack interval on a per-app and per-user basis, it is possible to opportunistically reduce CPU voltage/frequency at the start of the interval and then increase the voltage/frequency just before the interval ends, to save energy without impacting user quality of service.

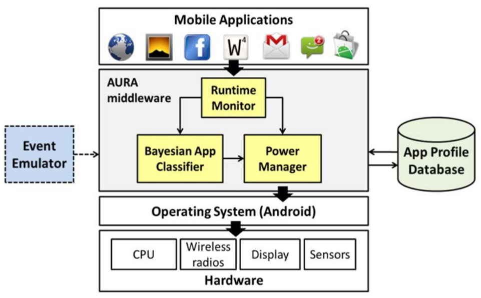
In a typical usage scenario with AURA, its runtime monitor (Figure 1(b)) checks if the current foreground app has an entry in an 'interaction database', and if so then the interaction data (standard deviation and mean of a user's observed slack values from previous interactions) in it can be used for slack prediction. If the database does not exist, the middleware starts collecting data about interaction pattern statistics. A Bayesian classifier is then used to classify the interaction profile for the app using the collected data [4]. Bayesian learning is a form of supervised machine learning that involves using evidence or observations along with prior outcome probabilities to calculate the probability of an outcome. Once the classification for the app is known, an MDP (Markov Decision Process) based power manager is used [4] to opportunistically decrease CPU voltage/frequency in between slack intervals. MDPs are discrete time stochastic control processes that are widely used as decision-making models for systems in which outcomes are partly random and partly controlled. In [5], this MDP was improved upon by a Q-Learning based power manager. Q-Learning is a reinforcement learning technique that does not require a model of the environment and has the advantage that the next state probability distributions that are used in MDPs are not required.

AURA also saves display energy by using the principle of change blindness, as revealed by researchers in human psychology and perception. Change blindness refers to the inability of humans to notice large changes in their environments, especially if changes occur in small increments. Many studies have confirmed this limitation of human perception– a majority of observers in one study failed to observe when a building in a photograph gradually disappeared over the course of 12 seconds; in another study, gradual color changes over time to an oil painting went undetected by a majority of subjects but disruptive changes such as the sudden addition of an object were easily detected. By gradually reducing screen brightness over time using user-device and app-specific interaction data, the power manager in AURA is able to reduce display energy consumption without causing user dissatisfaction.

Our experiments with AURA demonstrated 17% energy savings on average compared to the baseline Android device manager and approx. 2.5× more energy savings over the best known prior work [6] on mobile CPU and display energy optimization.


Manual process of interaction (e.g. user touches button)
System's response to button press (e.g button changes color) becomes perceptible to user
User comprehends the system's response
Perceptual Slack
Cognitive Slack
Motor Slack

(a)



(b)

Figure 1: (a) Interaction slack estimation; (b) AURA energy optimization middleware framework for mobile devices [5].

2.1.2. Context-aware wireless radio management

Smartphones often come packed with a myriad of wireless interfaces such as GPS, Wi-Fi and 3G/4G. Even when these interfaces are idle, it was shown in [7] that if they are enabled they account for more than 25% of the total power dissipation of the smartphone. Our framework in [7], [8] represents one of the first efforts for wireless interface energy management in mobile devices based on applying analytics on user and device context data.

Our middleware framework involves capturing contextual data attributes such as temporal use data (e.g., day of week and time), spatial environment data (e.g., ambient light, Wi-Fi RSSI, 3G/4G signal strength), and device state (e.g., battery status, CPU utilization). To prune the massive amount of data captured, we employed Principal Component Analysis (PCA), a form of dimensionality reduction, by projecting the captured data from various sources onto a fewer number of optimally selected eigenvectors, effectively reducing the attribute space to the number of eigenvectors.

We then explored the use of five different classes of machine learning algorithms, namely LDA (Linear Discriminant Analysis), LLR (Linear Logistic Regression), NN (Neural Networks), KNN (K-Nearest Neighbor) and SVM (Support Vector Machine), towards predicting user data/location usage requirements (e.g., is data transfer needed? is coarse-grained location needed? is fine-grained location needed?) based on the pruned spatiotemporal and device context data collected. The predictions are then used to adapt and turn the various data transfer and location estimation wireless interfaces off or on in an intelligent manner, to reduce their energy consumption. We demonstrated up to a 90% successful prediction using SVM, NN, and KNN algorithms, and accuracy improvements of approx. 50% over a self-organizing map prediction approach from [9]. Further, we achieved 24% higher energy savings on average than the state-of-the-art prior work [10].

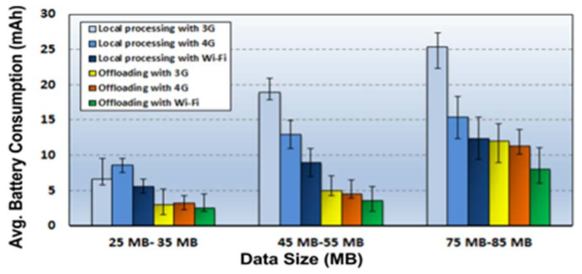


Figure 2: Average battery consumption on a mobile device for torrent file download operations; local processing is done on the device whereas offloaded processing is performed on Amazon EC2 cloud instances [11].

2.1.3. Mobile to cloud offloading

Data analytics can also be used to assist with the offloading of workloads from mobile devices to the cloud, to improve energy efficiency and robustness of computations. In [11], we proposed such a framework for mobile devices that utilized various sources of data such as the applications communication/computation intensiveness, type and status of available wireless networks, and the capabilities of cloud servers, to make decisions on when and how to offload an application from the mobile device to the cloud.

Figure 2 shows an example of how the Android based torrent app Flud can benefit from offloading, allowing for improved energy efficiency. Different network types, the state of the network, and data transfer sizes result in varying improvements in energy, as depicted by the different bars and error bars of the readings in the figure. We proposed a middleware framework to enable intelligent mobile to cloud offloading decisions. The framework was based on an unsupervised Q-learning technique that analyzed the data for the app type, network type, network conditions, and cloud capabilities to select the optimal network type and decide when and what to offload. Our experiments with real smartphones showed savings of up to 30% in battery life with up to 25% better response time when using our framework compared to a state-of-the-art fuzzy logic based offloading approach [12]. For certain applications, e.g., voice recognition, we found that offloading can also improve recognition robustness (accuracy) by approx. 10%.

2.2. Mobile Data Analytics: Indoor Navigation Case Study

The above discussion highlights the fact that data analytics can improve energy-efficiency and robustness of mobile devices during everyday usage scenarios. Moreover, data analytics also has the potential to improve performance for various applications of mobile computing. As an example, consider the domain of indoor navigation with mobile devices, where the goal is to use data captured through smartphone radio interfaces and sensors to estimate the location of the user indoors (inside of buildings, caves, etc.) in real time. This is a challenging problem because GPS signals are weak and ineffective in indoor environments, and the wireless signal-based infrastructure for indoor localization is diverse, prone to interference, and often entirely non-existent [13].

Our LearnLoc framework in [14] proposed using data analytics to improve indoor localization accuracy and energy efficiency in a variety of indoor environments. We capture WiFi "fingerprint" data (signal strength, signal type, access point ID) in a continually updated database for indoor locales together with inertial sensing readings (based on accelerometer, gyroscope, and magnetometer sensor data) on mobile devices. This data is then analyzed and parsed with machine learning techniques to estimate location in real time. Figure 3 shows a comparison of three variants of the Learnloc framework that use Linear Regression (LR), non-linear neural networks (NL-NN) and K-nearest neighbor (KNN) techniques to improve the robustness of indoor localization. The use of smart machine learning techniques helps LearnLoc significantly improve prediction accuracy and overcome noisy data readings compared to prior work that relies entirely on inertial navigation (Inertial Nav) or signal fingerprinting (Radar [15], PlaceLab [16]) without using such analytics/learning techniques.

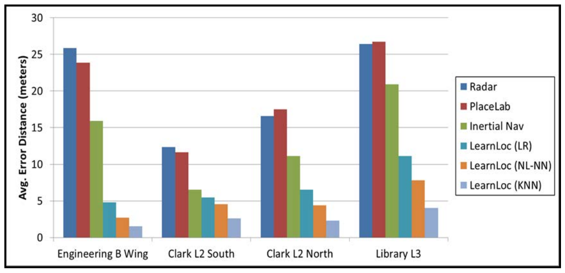


Figure 3: Comparison of indoor localization techniques [14].

3. DATA ANALYTICS IN MANYCORE DESIGN

In this section, we describe how data analytics can be leveraged for the design and optimization of manycore chip design.

3.1. Design-Time Optimization

Optimization processes that are carried out only once in the design process qualify as *static optimization* problems. One important example is the design of a Network-on-Chip (NoC) enabled manycore chip for a set of real-world applications [17]-[18].

Problem Description. The goal of an on-chip communication system design is to transmit data with low latencies and high throughput using the least possible power and resources. In this context, the design of a small-world (SW) network-based NoC architecture [17] is a notable example. It has been shown that either by inserting long-range shortcuts in a regular mesh architecture to induce a SW effect or by adopting a power-law based SW connectivity, it is possible to achieve significant performance gain and lower energy

consumption compared to traditional multi-hop mesh networks. However, the appropriate placement of the cores, routers, and links is crucial for maximizing the performance benefits. Hence, our goal is to optimize their placement to improve the network latency and power consumption per message for a given set of workloads.
Optimization Objective. We define an objective function O called communication cost (traffic weighted average hop count), which combines NoC performance metrics of network latency and energy consumption per message [17]. Optimizing communication cost ensures lower average hop count and improvement in network performance in terms of both latency and energy consumption.

Challenges. The space of physically feasible NoC-enabled manycore chip designs D is combinatorial in nature and our goal is to find the design $d \in D$ that minimizes O. In principle, one could employ popular algorithms such as hill-climbing with random restarts and simulated annealing. However, these algorithms do not scale for extremely large design spaces as in our case, and their anytime behavior (computation time vs. quality of solution) can be arbitrarily bad for practical time bounds [19]. Motivated by these observations, we leverage machine-learning techniques to explore the design space to achieve computational efficiency and better anytime behavior for design optimization [20]-[21].

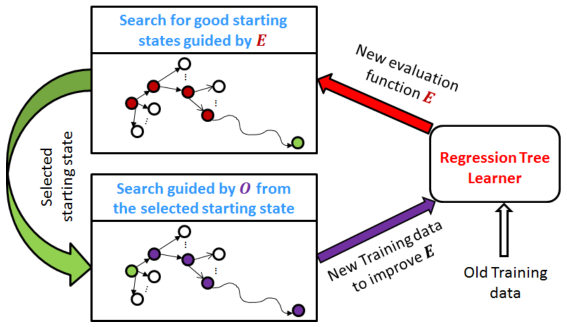

Search for good starting states guided by E
New evaluation function E
Regression Tree Learner
Selected starting state
Search guided by O from the selected starting state
New Training data to improve E
Old Training data

Figure 4: High-level overview of STAGE algorithm.

Design-Time Optimization using Machine Learning. We employ an online learning algorithm called STAGE [22], which was originally developed to improve the performance of local search algorithms (e.g., hill climbing) with random-restarts for combinatorial optimization problems. The high-level conceptual idea of the algorithm is shown in Figure 4. The key insight behind STAGE is to leverage some extra features $\phi(d) \in R^m$ (m is the number of features) of the optimization problem to learn an improved evaluation function E that can estimate the promise of a design d as a starting point for the local search procedure A. It employs E to intelligently select promising starting states that will guide A towards significantly better solutions. Past work in the search community concluded that many practical optimization problems exhibit a "globally convex" or "big valley" structure, where the set of local optima appear to be convex with one global optimum in the center [22]. The main advantage of STAGE over popular algorithms such as simulated annealing and Integer Linear Programming (ILP) is that it tries to learn the solution space structure, and uses this information in a clever way to improve both convergence time and the quality of the solution. This aspect of STAGE is very advantageous for large system sizes to improve the design-validate cycle before mass manufacturing and for dynamically adapting the designs for new application workload.

Overview of STAGE. The STAGE algorithm repeatedly alternates between two types of search as shown in Figure 4: 1) Base search, where A is run with the original objective O until it reaches a local optima and new training data is generated to improve E; and 2) Meta search, where it performs search with the learned evaluation function E to select good starting states to improve the performance of the local search procedure A. We want to learn E such that it can estimate the value of design d as a starting point for A guided by O. In the initial exploration phase, E may not lead to good solutions but as the iterations progress, E will improve with the training data generated from the search experience in the base search mode. In a recent work, we undertook a comparative performance analysis between STAGE, SA and GA in designing a TSV-enabled 3D SWNoC architecture [17]. Figure 5 shows the communication cost of the optimized network from the STAGE, SA, and GA algorithms as a function of time. We can see that STAGE uncovers high-quality designs very fast (within 5 minutes). SA and GA reach O_{best} more gradually compared to STAGE, and even after 50 minutes, their O_{best} does not reach the same solution as STAGE. It should be noted that we have to optimize the NoC design for various applications. Hence, this additional time needed by SA and GA will be a significant overhead when we have to optimize and reconfigure the SWNoC in the field. We conjecture that with the increase in the design space due to large system sizes and emerging technologies (e.g., monolithic 3D integration), STAGE will be even more efficient than SA and GA.

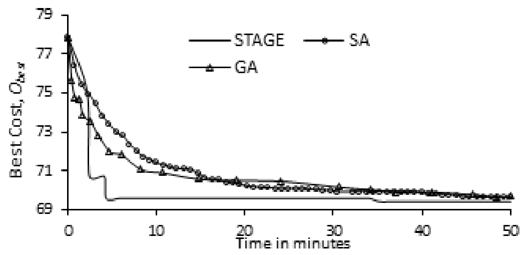


Figure 5: Performance comparison for the machine-learning-based algorithm (STAGE), Simulated Annealing (SA), and Genetic algorithm (GA).

3.2. Run-Time Optimization

Optimization processes that are carried out many times during the operation of manycore systems qualify as *dynamic optimization* problems. Some examples include dynamic power management [23]-[24] and adaptive routing [25] to optimize energy and thermal profiles. In these cases, machine learning techniques can be useful in learning the models and policies for dynamic manycore optimization by mapping the information from sensors at various levels to actionable decisions. Additionally, the policy can dynamically learn how its actions affect the system and readjust its model accordingly. We will discuss dynamic power management as a representative example of this problem space.

Problem Description. The design of high-performance manycore chips is dominated by power and thermal constraints. Voltage-Frequency Islands (VFI) have emerged as an efficient and scalable power management strategy [23]. In such designs, effective VFI clustering techniques allow cores and network elements (routers and links) that behave similarly to share the same Voltage/Frequency (V/F) values without significant performance penalties. Naturally, with time-varying workloads, we can dynamically fine-tune the V/F levels of VFIs to further reduce the energy dissipation with minimal performance degradation. For applications with highly varying workloads, machine learning methods are suitable to fine-tune the V/F levels within VFIs.

Optimization Objective. Consider a manycore system with n cores. Without loss of generality, let us assume that there exist k

VFIs. The DVFI control policy π, at each time interval or control epoch t (where N is the total number of epochs), takes the current system state s_t and generates the V/F allocation for all k VFIs, i.e.,

$$\pi: s_t \rightarrow \{V_1/F_1,\ V_2/F_2, \ldots, V_k/F_k\} \qquad (1)$$

Given a VFI-enabled manycore architecture, an application, and a maximum allowable performance penalty, our goal is to create a DVFI control policy π^* that minimizes energy dissipation within the user-specified maximum allowable performance penalty.

Reinforcement Learning. We can naturally formulate the problem of creating dynamic VFI (DVFI) control policies in the framework of *Reinforcement Learning* (RL). RL methods learn policies to efficiently control a system via a trial-and-error approach, i.e., interacting with the system and observing the resulting costs or reinforcements. Indeed, prior work has routinely employed RL methods to create fine-grained DVFS control policies [26]-[29]. However, three major drawbacks of this approach exist: i) RL approaches do *not* scale for large manycore systems: the computational complexity of learning high-quality control policies along with large hardware overheads to store the policy both increase with the system size; ii) RL ignores the rich structural dependencies between different parts of the system (e.g., between VFIs and their controllers); and iii) It can be difficult to design a cost function that leads RL to the desired behavior.

Imitation Learning. Imitation learning (IL) [30]-[31] is considered to be an exponentially better framework than RL for learning sequential decision-making policies, but assumes the availability of a good Oracle (or expert) policy to drive the learning process.

a) Oracle Construction. As the learning process is *offline*, we access future system states and perform a look-ahead search to find the best joint V/F allocation for all VFIs. This is accomplished by running the application with different V/F assignments to optimize global performance (i.e., EDP of the system). To overcome the computational challenge and closely approximate optimality, *our key insight is to perform local optimization followed by aggregation for global optimization*. First, we compute the optimal V/F (which minimizes EDP) for each VFI, at each control epoch, for *m* different execution time penalties (e.g., 0%, 5%, and 10% for $m = 3$). This gives us *m* different V/F assignments for each control epoch. Second, for every *n* control epochs, we compute the best V/F decisions by performing an exhaustive search over all possible combinations of local optima from the first step (m^n). Note that it is easy to find a small *m* that works well in practice, but both the quality and computation time of the Oracle depends on *n*.

b) DVFI Control Decision-Making. We formulate the problem of DVFI control decision-making as a *structured output prediction task* [20]. This is the task of mapping from an input structured object (a graph with features on the nodes and edges) to an output structured object (a graph with labels on the nodes and edges). In DVFI control, the input graph contains a node for each VFI with edges defined based on the inter-VFI traffic density. The label for each node in the output graph represents the V/F allocation of the corresponding VFI. The edges in the graph correspond to the structural dependencies between different input and output variables. To address the computational challenge of decision-making, we learn *pseudo-independent structured controllers* to achieve efficiency without losing accuracy. Specifically, we learn k controllers, one controller for each VFI.

c) DAgger Algorithm. Our goal is to learn a controller that closely follows the Oracle in terms of V/F allocation. We use an advanced imitation learning algorithm named DAgger [30] to learn robust policies (Figure 6). The key idea behind DAgger is to generate additional training data so that the learner can learn from mistakes.

In a recent work, for 5% performance penalty, IL reduces the energy dissipation over RL by up to 11.5% for SPLASH-2 and PARSEC benchmarks [32]. It should be noted that this gap widens if higher performance penalties are allowed. As an example, at 15% performance penalty, IL reduces the energy dissipation over RL by 22.5% for the FFT benchmark. Due to IL's consistent better performance, lower implementation and computational overheads, we conjecture that IL is an efficient, scalable, and robust methodology to learn DVFI policies for large-scale manycore systems.

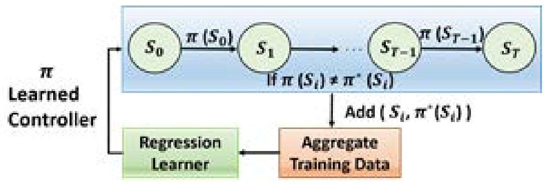


Figure 6: DVFI Controller learning via DAgger algorithm.

4. DATA ANALYTICS FOR DATACENTERS

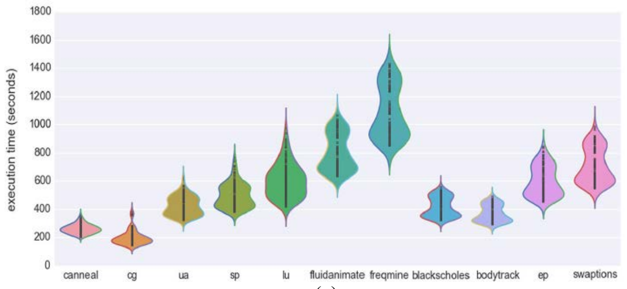
As the use of mobile computing devices and network-connected embedded devices becomes ubiquitous, there has been a corresponding increase in the utilization of cloud computing platforms to service the computational needs of these devices [2]. Sifting heavier computational workloads from smaller portable or embedded platforms to more powerful and potentially more efficient high performance computing (HPC) systems allows for the possibility of providing faster and more energy efficient services to groups of users. But this is only possible if the software and hardware stacks for HPC systems are designed in a manner that minimizes energy costs and maximizes the reliability of outputs. Data analytics plays a key role in allowing service providers the ability to design their systems with these goals in mind. This section discusses how utilizing data about application arrival and execution behavior, system resource usage, environmental conditions, and operational costs can provide opportunities for optimizing HPC systems.

4.1. Predicting Server Performance

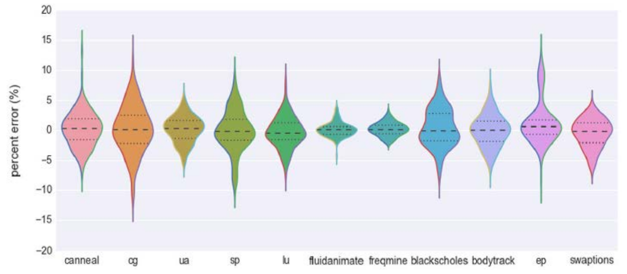
HPC systems such as datacenters are typically organized as a large collection (several thousands) of server nodes, with a node usually consisting of one to four sockets, each with a multicore processor. As threads of multiple parallel applications can co-exist on a multicore processor and share resources (e.g., the processor's last level cache), at the system level a prominent source of performance uncertainty arises for any executing workload: interference from other applications co-located on the same multicore processor. Such interference can 1) cause unpredictable execution time increases which can cause deadlines for mission critical tasks to be missed, 2) create load imbalance for large parallel applications that can reduce their throughput, 3) reduce the effectiveness of system resource managers that make decisions on where to map workloads in HPC systems and rely on execution time estimates to make these decisions, and 4) increase energy consumption that significantly drives up operating expenditures.

Our work in [33], [34] proposed a framework to make execution time predictions for applications in HPC systems through the use of data analytics and machine learning. We characterized the interference factors that impacted execution time increase on multiple real Intel Xeon servers, including the number of co-located applications, memory intensities of co-located applications, cache access frequencies, and power states (P-states) of cores. We trained models for execution time and energy use prediction after analyzing and processing data from the execution of several scientific workloads on servers. These models were based on linear regression as well as non-linear neural networks. Once the models were developed, we obtained values for the interference factors from performance counters at runtime, and the information was fed to the models to obtain fast real-time predictions.

Figure 7(a) shows the effects co-location can have on application execution times. Each object in the figure shows the distribution of execution times for eleven scientific workloads from the NAS and PARSEC scientific benchmark suites. For each distribution shown, the target application (indicated on the x-axis) is executed on a 6-core Intel Xeon E5649 server under a variety of co-location scenarios ranging from the application executing on the processor alone (no co-location, providing the best performance) to executing co-located with a variety of other applications with different execution characteristics. The dots inside the distributions indicate specific data points. The results in Figure 7(a) highlight the challenge faced by the machine learning techniques when attempting to predict application execution time. Note both the large (sometimes over 2×) increase in execution time that an application can experience as well as the large variation between distributions for application of different types.



(a)



(b)

Figure 7: (a) Distributions of each application's execution time. (b) The accuracy of the neural network model for predicting execution time [33].

A carefully chosen subset of data associated with all possible workload co-locations was used to train our neural network based prediction model. Figure 7(b) shows a detailed view of the performance of using this neural network based model on the execution time data shown in Figure 7(a). Objects in Figure 7(b) show distributions of the percent error between the model's execution time predictions and the actual execution times. Positive error values indicate that the model's prediction overestimated the actual execution time and negative error values indicate that the model underestimated the actual execution time. Lines across each distribution represent the median (dashed line) and upper and lower quartiles (dotted lines). These results demonstrate that the model's predictions are typically accurate (error is close to zero), that about half of the model's predictions are ±2% from the actual execution time values, and that nearly all of the predictions are within 5% of the actual execution time values.

4.2. Analytics for HPC System Resource Management

In an HPC environment, server nodes are assigned incoming workloads by a system resource manager. Data analytics and machine learning techniques can be very useful to predict incoming workload [35]-[37], as well as to schedule applications in the HPC system in a manner that improves system performance or energy efficiency, as discussed next.

4.2.1. Homogeneous HPC system scheduling

In a homogeneous HPC system, a resource manager attempts to distribute applications that arrive to the system among a set of identical server nodes in a way that optimizes some performance metric (e.g., minimizing execution time or energy use). Because co-locations of some application types can have a worse performance degradation effect than others, data analytic techniques for predicting application execution times such as those discussed in Section 4.1 can provide benefit to system resource management.

Figure 8 demonstrates that co-location aware predictions of node-level workload performance can be used to create a heuristic that improves system scheduling in a homogeneous system by reducing the total number of applications (referred to as *tasks* in the study) that miss their deadlines in an oversubscribed system, where some number of missed deadlines are inevitable. The figure compares the performance of a co-location aware heuristic, with access to the neural network performance prediction models described in Section 4.1, to a co-location naïve heuristic, as well as to a "perfect prediction" heuristic. The study is performed on a 500 node HPC system, with each server node comprised of a single socket with a 4-core Intel Xeon E3-1225v3 processor. The figure gives the percentage of tasks that either completed successfully (purple bars), are scheduled but miss their deadline due to performance degradation from co-location (brown bars), or are unable to be scheduled as the scheduling heuristic determines that there is not enough space available in the system (green bars). The data indicates that the co-location aware heuristic enables the system to successfully execute a higher percentage of tasks than the co-location naïve heuristic, providing the system greater robustness and value. Further studies in [33] showed how the same co-location aware scheduler can allow an undersubscribed system to better conserve energy by consolidating executing applications to fewer nodes.

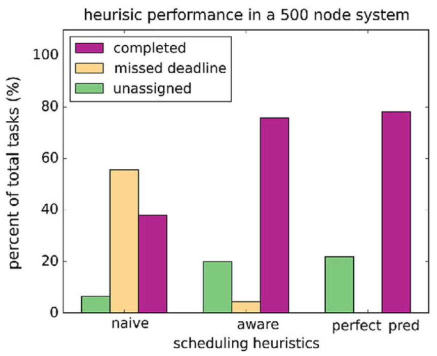


Figure 8: Performance comparison the co-location aware scheduling heuristic utilizing machine learning based execution time predictions [33].

4.2.2. Heterogeneous HPC system scheduling

As a heterogeneous system has multiple types of servers, it offers an even more complicated scheduling problem than the homogeneous system, but also allows for the possibility of faster or more energy efficient execution of the workload if the scheduling is performed well. Such heterogeneous systems can benefit from using data analytics to determine application performance on different types of hardware as well as to improve scheduling of the workload across diverse processing entities.

In [38]-[40], we proposed a design-time framework for resource management of heterogeneous datacenters with the goal of minimizing overall power consumption. The framework utilized our learning-based interference prediction models as well as new models for behavioral heterogeneity among different servers (offering different levels of performance and power consumption), the impact of dynamic voltage and frequency scaling (DVFS) in cores, and the interactions between temperatures of the server nodes and computer room air conditioning (CRAC) units. By distributing workloads, configuring DVFS, and setting CRAC thermostats in an intelligent manner, our approach was able to efficiently trade-off the compute performance with the power consumption and temperature profiles of the HPC facility.

In [41] we extended this work to create a runtime framework that utilized a novel real-time model for thermal profile estimation corresponding to different workload schedules, to minimize both computation and cooling power in heterogeneous datacenters, even for unpredictable workload arrival patterns. Figure 9 shows a high-level overview of the decision framework that utilizes design time (offline) computed thermal templates together with runtime (online) workload and sensor data to map the workload, adjust DVFS in cores, set the CRAC thermostat values, and adjust floor vent openings at runtime. Our proposed hybrid offline/online data analysis allows for a reduction in the complexity and volume of data that needs to be analyzed at runtime. Resource management in such environments can also be further improved with data analytics techniques based on statistical analysis of workload executions, and techniques for using historical information to reduce the solution space, as done in our prior work [42]-[43].

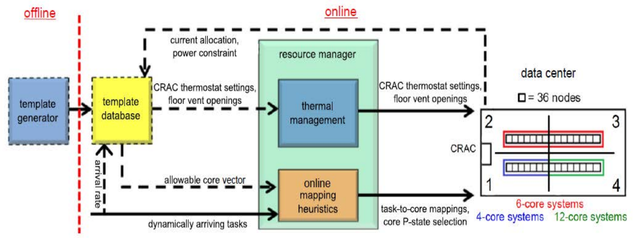

offline
online
current allocation, power constraint
resource manager
template generator
template database
CRAC thermostat settings, floor vent openings
thermal management
CRAC thermostat settings, floor vent openings
data center
□ = 36 nodes
2
3
CRAC
1
4
arrival rate
allowable core vector
online mapping heuristics
dynamically arriving tasks
task-to-core mappings, core P-state selection
6-core systems
4-core systems
12-core systems

Figure 9: Block diagram of the thermal-aware runtime resource management framework for heterogeneous HPC data centers [41].

4.2.3 Scheduling for geo-distributed HPC platforms

It is not uncommon for large service providers such as Amazon, and Google to operate multiple datacenters that are spread over several geographical locations, to reduce operating expenditures by exploiting time-of-use (TOU) electricity pricing [44]-[45]. By allocating more workload to datacenters with lower electricity cost at any given time, overall expenditures can be reduced. In [46] we proposed a resource manager for energy cost minimization in geographically distributed heterogeneous data centers. Building on our prior work for data analytics driven data center scheduling as discussed in prior subsections, we devised a global resource manager that additionally monitored TOU pricing data and workload arrival patterns to intelligently distribute workloads across various heterogeneous datacenters. Our framework reduced energy costs by 37% on average compared to the state-of-the-art [58].

5. DATA ANALYTICS IN NETWORKS

A three-layer hierarchical design is widely used in modern telecommunication systems to achieve high performance and reliability [47]. The three layers, namely *core*, *distribution*, and *access*, perform different roles for service fulfillment [47]. The core layer is also referred to as the network backbone, and it is responsible for the transfer of a large amount of traffic in a reliable and timely manner. The network devices (such as routers) used in the core layer are complex systems that contain both software and hardware (Figure 10), making them more vulnerable to hard-to-detect/hard-to-recover errors [48]. All these different types of faults can cause a core router to become incapacitated, necessitating the design and implementation of fault-tolerant mechanisms in the core layer.

Proactive fault tolerance is promising because it takes preventive action before a failure occurs [49]. The state of the system is monitored in a real-time manner. Anomalies are defined as abnormal patterns in the system's state. When anomalies are detected, proactive repair actions such as job migration are executed to avoid errors, thereby maintaining the non-stop utilization of the entire system [50]. The effectiveness of proactive fault-tolerance solutions depends on whether anomalies can be accurately pinpointed in a timely manner [51]. In this section, we show how data analytics can be used to detect anomalies in a high-performance and complex communication system [59].

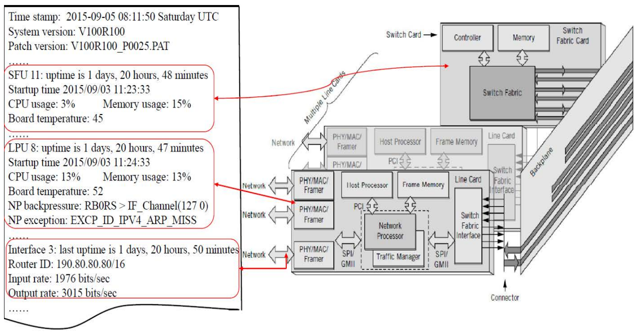

```
Time stamp:  2015-09-05 08:11:50 Saturday UTC
System version: V100R100
Patch version: V100R100_P0025.PAT
......
SFU 11: uptime is 1 days, 20 hours, 48 minutes
Startup time 2015/09/03 11:23:33
CPU usage: 3%          Memory usage: 15%
Board temperature: 45
......
LPU 8: uptime is 1 days, 20 hours, 47 minutes
Startup time 2015/09/03 11:24:33
CPU usage: 13%         Memory usage: 13%
Board temperature: 52
NP backpressure: RB0RS > IF_Channel(127 0)
NP exception: EXCP_ID_IPV4_ARP_MISS
......
Interface 3: last uptime is 1 days, 20 hours, 50 minutes
Router ID: 190.80.80.80/16
Input rate: 1976 bits/sec
Output rate: 3015 bits/sec
......
```

Figure 10. A multi-card chassis core router system and a snapshot of extracted (monitored) features [59].

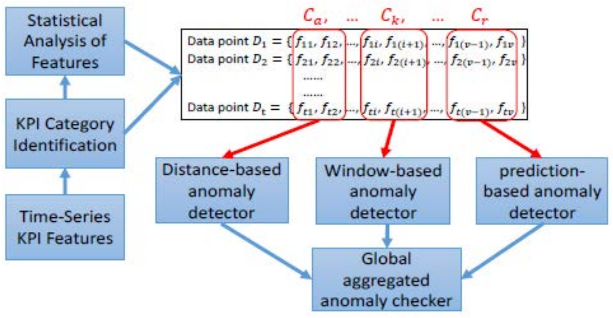
Problem Statement: The difficulty of developing an efficient anomaly detector for a complex communication system can be attributed to two reasons. The first reason is that features extracted from communication systems are far more complex than those from a general computing system. For example, a multi-card chassis core router system uses monitors to log a large number of features from different functional units. These features include performance metrics (e.g., events, bandwidth, throughput, latency, jitter, error rate), resource usage (e.g., CPU, memory, pool, thread, queue length), low-level hardware information (e.g., voltage, temperature, interrupts), configuration status of different network devices, and so on. Each of these features can have significantly different statistical characteristics, making it difficult for a single type of anomaly-detection technique to be effective. The second reason is that the monitored data in communication systems involves temporal measurements. Most existing anomaly-detection methods are not designed to address time-series data [52], hence they may not be able to detect time-series-specific anomalies such as the trend anomaly.

We therefore address the important practical problem of designing an anomaly detector that can be effectively applied to a commercial core router system. We use multiple anomaly- detection techniques to detect different types of anomalies. We describe a new feature-categorization-based hybrid method to improve the performance of our anomaly detector. A correlation analyzer is also developed to select the most important features and cluster correlated features.

5.1. Time-Series-Based Anomaly Detection

In complex communication systems such as a core router, data is collected in the form of time-series. A key challenge here is to detect anomalies in time-series data to determine whether the system is entering a degraded state or is likely to fail. Therefore, we have studied a range of techniques that may be used to detect anomalies in time-series data [53].

The first one is unsupervised distance-based anomaly detection, which utilizes a distance measure between a pair of time-series instances to represent the similarity between these two time-series [54]. Instances far away from others will be identified as being abnormal. The second one is window- based anomaly detection [52]. This method divides time-series instances into overlapping windows. Anomaly scores are first calculated per window, and then aggregated to be compared with a predefined threshold. Only when the overall anomaly score of a single time-series instance significantly exceeds a predefined threshold, will this instance be identified as being abnormal. The third method is supervised prediction-based anomaly detection [52]. First, a machine-learning-based predictive model is learned from historical logs. Next, predicted values are obtained by feeding test data to this predictive model. These predicted values are then compared with the actual measured data points. The accumulated difference between these predicted and the actual observations is defined as the anomaly score for each test time-series instance.



Statistical Analysis of Features
KPI Category Identification
Time-Series KPI Features
C_a, … C_k, … C_r
Data point $D_1 = \{f_{11}, f_{12}, \ldots, f_{1i}, f_{1(i+1)}, \ldots, f_{1(v-1)}, f_{1v}\}$
Data point $D_2 = \{f_{21}, f_{22}, \ldots, f_{2i}, f_{2(i+1)}, \ldots, f_{2(v-1)}, f_{2v}\}$
Data point $D_t = \{f_{t1}, f_{t2}, \ldots, f_{ti}, f_{t(i+1)}, \ldots, f_{t(v-1)}, f_{tv}\}$
Distance-based anomaly detector
Window-based anomaly detector
prediction-based anomaly detector
Global aggregated anomaly checker

Figure 11. Depiction of feature-categorization-based hybrid anomaly detection.

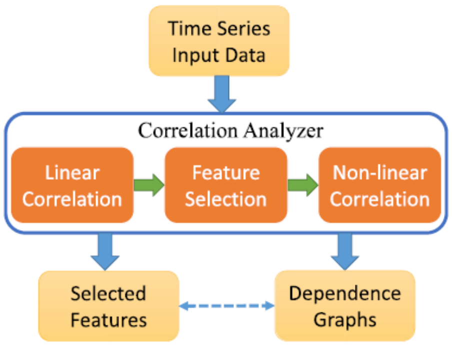
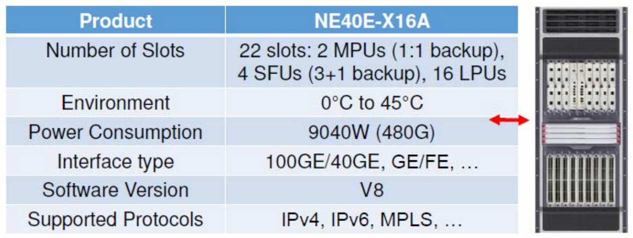


Figure 12. Architecture of the proposed correlation analyzer. Features within each group can be represented by a dependence graph $G = (V, E)$, where the set of vertices V represent feature candidates and the set of edges E represent dependent relationships between features. A dependence graph is generated for each group of features [59].

However, a single class of anomaly detection methods is effective for only a limited number of time-series types. Therefore, we have also developed a feature-categorization-based hybrid method (Figure 11). First, features that exhibit similar statistical characteristics are placed in the same category. Next, each group of features is fed to the anomaly detector that is most suitable for these types of features. Finally, the results provided by different anomaly detectors are aggregated so that an anomaly can be detected in terms of the entire feature space.



Product	NE40E-X16A
Number of Slots	22 slots: 2 MPUs (1:1 backup), 4 SFUs (3+1 backup), 16 LPUs
Environment	0°C to 45°C
Power Consumption	9040W (480G)
Interface type	100GE/40GE, GE/FE, …
Software Version	V8
Supported Protocols	IPv4, IPv6, MPLS, …

Figure 13. Description of the commercial core router used in our experiments.

5.2. Correlation-Based Feature Selection

The number of feature dimensions will increase from hundreds to tens of thousands when more new features are identified and extracted from raw log data, making it more difficult and time-consuming to detect anomalies [55]. Therefore, a correlation analyzer has been developed to remove irrelevant and redundant features. As shown in Figure 12, the correlation analyzer consists of three components: the linear correlation component, the feature selection component, and the non-linear correlation component. The linear correlation component is used to find linear-dependent feature pairs. The feature selection component is developed to select an effective, but reduced, set of non-linear-dependent feature groups [56]. The non-linear correlation component is implemented for fine-grained non-linear dependence analysis [57]. Finally, the correlation analyzer outputs a number of correlated feature groups. An effective feature subset can be generated by selecting most representative features from these correlated groups.

Table 1: Feature categories for the time-series data obtained from the core router.

Feature category	Number of features	Representative features	Potential anomalies
Gaussian Process	30	CPU/Memory Usage	Point anomaly
Monotonic Function	14	OSPF neighbor uptime	Point/Trend anomaly
Cyclic Process	12	ARP learnt count	Point/Trend anomaly
Bursty	48	Interface Input /Output Rate	Collective anomaly
Others	10	NP Exception /Interrupt	Point anomaly

Table 2: Features corresponding to various components.

Component	Number of Components	Number of features	Representative features
MPU	2 (1+1 backup)	112	CPU/Mem Usage Board Temperature
SFU	4 (3+1 backup)	196	Route Age Board Uptime
LPU	16	832	Lost Packet Count
Interface	93	1004	Input/Output Rate Utilization Ratio
Others	124	748	Exception/Assertion

5.3. Experiments and Results

The commercial core router system used in our experiments consists of a number of different functional units such as the main processing unit, line processing unit, switch fabric unit, etc (Figure 13). A total of 602 features were monitored and sampled every 30 minutes for 15 days of operation of the core router system, generating a set of multivariate time-series data consisting of 720 time points. Three types of anomalies, namely point anomaly, namely collective anomaly, and trend anomaly, were considered during our experiments [52]-[53].

Table I lists the feature categories and potential anomalies for the extracted time-series data. Table 2 shows the features corresponding to various components in the core router. The success ratio (SR) is the ratio of the number of correctly detected anomalies to the total number of anomalies in the testing set while the non-false-alarm ratio (NFAR) is defined as the ratio of the number of correctly detected anomalies to the total number of alarms flagged by the anomaly detector. We found that that for six anomaly detection methods, i.e., k-nearest neighbors (KNN), Window-based KNN, window-based support-vector machines, support-vector regression, auto-regression, and the feature-categorization-based hybrid method, the success ratios are 82.7%, 84.5%, 86.4%, 88.2%, 78.6% and 95.1%, respectively, and the non-false-alarm ratios are 73.1%, 76.3%, 80.7%, 88.1%, 71.6% and 92.1%. The reason that the feature-categorization-based hybrid method achieves much higher SR and NFAR than the other methods is that it can overcome the difficulty of adopting a single class of anomaly detection to features with significantly different statistical characteristics.

6. CONCLUSIONS

We are witnessing the rise of the data-driven science paradigm, in which massive amounts of data – much of it collected as a side-effect of ordinary human activity – can be analyzed to make sense of the data and to make useful predictions. In this paradigm, computing systems ranging from Internet of Things (IoT) and mobile devices to manycore chip platforms, High-Performance Computing (HPC) datacenters and networks play different roles, and they need to be optimized for appropriate objectives as per the needs of the applications. In this paper, we argue that data analytics can help hardware designers and computer architects during the design, control, and testing of computing systems at different levels of abstraction to achieve energy-efficiency and robustness.

ACKNOWLEDGMENTS

This work was supported by grants from Huawei Technologies, Hewlett Packard, and the National Science Foundation (ECCS-1646562, CCF-1252500). The authors thank Zhaobo Zhang and Xinli Gu from Huawei Technologies for contributing to the work on core routers.

REFERENCES

[1] Cisco Global Cloud Index: Forecast and Methodology, 2015–2020, White paper, [Accessed July 2017].

[2] 'Pew Research Center', 2017 [Online] Available: http://www.pewglobal.org/2016/02/22/smartphone-ownership-and-internet-usage-continues-to-climb-in-emerging-economies/ [Accessed: 25 June 2017]

[3] 'Cisco White Papers', 2017 [Online] Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, [Accessed: 26 June 2017]

[4] B. Donohoo, C. Ohlsen, S. Pasricha, "AURA: An Application and User Interaction Aware Middleware Framework for Energy Optimization in Mobile Devices", *IEEE International Conference on Computer Design (ICCD 2011), Oct. 2011.*

[5] B. Donohoo, C. Ohlsen, S. Pasricha, "A Middleware Framework for Application-aware and User-specific Energy Optimization in Smart Mobile Devices", *Journal of Pervasive and Mobile Computing, vol. 20, pp. 47-63, Jul 2015.*

[6] A. Shye, B. Scholbrock and G. Memik, "Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures," *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 168-178, 2009.*

[7] B. Donohoo, C. Ohlsen, S. Pasricha, C. Anderson, "Exploiting Spatiotemporal and Device Contexts for Energy-Efficient Mobile Embedded Systems", *IEEE/ACM Design Automation Conference (DAC 2012), Jul. 2012.*

[8] B. Donohoo, C. Ohlsen, S. Pasricha, C. Anderson, Y. Xiang, "Context-Aware Energy Enhancements for Smart Mobile Devices", *IEEE Transactions on Mobile Computing (TMC), Vol 13, No. 8, pp. 1720-1732, Aug 2014.*

[9] L. Batyuk, C. Scheel, S. A. Camtepe, S. Albayrak, "Context-aware device self-configuration using self-organizing maps" *OC, pp. 13-22, June 2011.*

[10] C. Lee, M. Lee, D. Han, "Energy efficient location logging for mobile device," *SAINT, pp. 84, Oct. 2010.*

[11] A. Khune, S. Pasricha, "Mobile Network-Aware Middleware Framework for Energy Efficient Cloud Offloading of Smartphone Applications", *to appear, IEEE Consumer Electronics, 2017.*

[12] H. R. Flores and S. Srirama, "Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning," *in Proc. ACM Mobisys, pp. 9-16, Jun. 2013.*

[13] C. Langlois, S. Tiku, S. Pasricha, "Indoor localization with smartphones", to appear, IEEE Consumer Electronics, 2017.

[14] S. Pasricha, V. Ugave, Q. Han and C. Anderson, "LearnLoc: A Framework for Smart Indoor Localization with Embedded Mobile Devices," *ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), Oct 2015.*

[15] P. Bahl, and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," *IEEE INFOCOM. 2000.*

[16] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit, "Place Lab: Device Positioning Using Radio Beacons in the Wild," *Proc. PERCOM, 2005, pp. 116-133*

[17] S. Das, J.R. Doppa, P. Pande, K. Chakrabarty, "Design-Space Exploration and Optimization of an Energy-Efficient and Reliable 3-D Small-World Network-on-Chip," *IEEE TCAD, vol 36, issue 5, pp 719-732*, 2017.

[18] W. Choi, K. Duraisamy, R. Kim, J.R. Doppa, P. Pande, R. Marculescu, D. Marculescu, "Hybrid Network-on-Chip Architectures for Accelerating Deep Learning Kernels on Heterogeneous Manycore Systems," in *Proc. of IEEE CASES Conference*, 2016.

[19] J.R. Doppa, A. Fern, P. Tadepalli, "Structured Prediction via Output Space Search," *Journal of Machine Learning Research, 15(Apr): 1317-1350*, 2014.

[20] J.R. Doppa, A. Fern, P. Tadepalli, "HC-Search: A Learning Framework for Search-based Structured Prediction," *Journal of Artificial Intelligence Research, vol 50, pp 369-407*, 2014.

[21] F.A.R. Chowdhury, C. Ma, M.R. Islam. M.O. Faruk, J.R. Doppa, "Select-and-Evaluate: A Learning Framework for Large-Scale Knowledge Graph Search," *Journal of Machine Learning Research, Proceedings Track, Vol 80*, 2017.

[22] J. A. Boyan, A.W. Moore, "Learning Evaluation Functions to Improve Optimization by Local Search," *Journal of Machine Learning Research, Vol 1, pp 77-112*, 2001.

[23] R. Kim, W. Choi, Z. Chen, J.R. Doppa, P. Pande, D. Marculescu, R. Marculescu, "Imitation Learning for Dynamic VFI Control in Large-Scale Manycore Systems," *IEEE TVLSI*, 2017.

[24] S. Das, D.J. Lee, W. Choi, J.R. Doppa, P. Pande, K. Chakrabarty, "VFI-based Power Management to Enhance the Lifetime of High-

Performance 3D NoCs," *ACM TODAES*, 2017.

[25] S. Das, J.R. Doppa, P. Pande, K. Chakrabarty, "Reliability and performance trade-offs for 3D NoC-enabled multicore chips," in *Proc. of DATE Conference*, 2017.

[26] H. Jung and M. Pedram, "Supervised Learning Based Power Management for Multicore Processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 29, no. 9, 2010.*

[27] G. Dhiman and T.S. Rosing, "Dynamic voltage frequency scaling for multi-tasking systems using online learning," *In Proc. of the ACM/IEEE International Symposium on Low Power Electronics and Design, pp. 207-212, 2007.*

[28] Z. Chen and D. Marculescu. "Distributed Reinforcement Learning for Power Limited Many-core System Performance Optimization," *In Proc. of the Design, Automation & Test in Europe Conference & Exhibition, pp.1521-1526, 2015.*

[29] A. Das, M.J. Walker, A. Hansson, B.M. Al-Hashimi, and G.V. Merrett, "Hardware-Software Interaction for Run-Time Power Optimization: A Case Study of Embedded Linux on Multicore Smartphones." *In Proc. of the IEEE/ACM International Symposium on Low Power Electronics and Design, pp.165-170, 2015.*

[30] S. Ross, G. Gordon, D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *Proc. of AISTATS Conference*, 2011.

[31] B. Minor, J.R. Doppa, D. Cook, "Data-Driven Activity Prediction: Algorithms, Evaluation Methodology, and Applications," in *Proc. of KDD Conference*, 2015.

[32] R. G. Kim; W. Choi; Z. Chen; J. R. Doppa; P. P. Pande; D. Marculescu; R. Marculescu, "Imitation Learning for Dynamic VFI Control in Large-Scale Manycore Systems," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , vol.PP, no.99, pp.1-14

[33] D. Dauwe, E. Jonardi, R. Friese, S. Pasricha, A. A. Maciejewski, D. Bader, H.J. Siegel, "HPC Node Performance and Energy Modeling Under the Uncertainty of Application Co-Location", *Journal of Supercomputing, Vol. 72, No. 12, pp. 4771-4809, Nov. 2016.*

[34] D. Dauwe, E. Jonardi, R. Friese, S. Pasricha, A. A. Maciejewski, D. Bader, H.J. Siegel, "A Methodology for Co-Location Aware Application Performance Modeling in Multicore Computing," *17th Workshop on Workshop on Advances in Parallel and Distributed Computational Models (APDCM), May 2015.*

[35] A. Mishra, et al., "Towards characterizing cloud backend workloads: insights from Google compute clusters," *SIGMETRICS Performance Evaluation, vol. 37, no. 4, pp. 34-41, 2010.*

[36] D. Gmach, et al., "Workload Analysis and Demand Prediction of Enterprise Data Center Applications," *International Symposium on Workload Characterization, pp. 171-180, 2007.*

[37] J. Prevost, et al., "Prediction of cloud data center networks loads using stochastic and neural models," *International Conference on System of Systems Engineering, pp. 276-281, 2011.*

[38] M. Oxley, E. Jonardi, S. Pasricha, H. J. Siegel, T. Maciejewski, P. J. Burns, and G. Koenig "Rate-based Thermal, Power, and Co-location Aware Resource Management for Heterogeneous Data Centers", *to appear, Journal of Parallel and Distributed Computing (JPDC), 2017.*

[39] M. Oxley, E. Jonardi, S. Pasricha, A. A. Maciejewski, G. Koenig and H. J. Siegel "Thermal, Power, and Co-location Aware Resource Allocation in Heterogeneous Computing Systems," *IEEE International Green Computing Conference (IGCC), 2014.*

[40] A. M. Al-Qawasmeh, S. Pasricha, A. M. Maciejewski, H. J. Siegel, "Power and Thermal-Aware Workload Allocation in Heterogeneous Data Centers", *IEEE Transactions on Computers, vol. 64, Iss 02, pp. 477-491, Feb 2015.*

[41] M. Oxley, S. Pasricha, T. Maciejewski, H.J. Siegel and P. Burns, "Online Resource Management in Thermal and Energy Constrained Heterogeneous High Performance Computing," *IEEE International Conference on Big Data Intelligence and Computing (DataCom), Aug 2016.*

[42] B. D. Young, et al., "Deadline and energy constrained dynamic resource allocation in a heterogeneous computing environment," *Journal of Supercomputing, vol. 63, no 2, pp. 326-347, 2013.*

[43] B. Khemka R. Friese, S. Pasricha, A. A. Maciejewski, H. J. Siegel, G. A. Koenig, S. Powers, M. Hilton, R. Rambharos, and S. Poole, "Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system," *Sustainable Computing: Informatics and Systems, Volume 5, pp. 14-30, 2015.*

[44] "Data center locations," [Online] Available: http://www.google.com/about/datacenters/inside/locations/index.html, [Accessed June 2017].

[45] "Global infrastructure," [Online] Available: http://aws.amazon.com/about-aws/global-infrastructure/, [Accessed June 2017].

[46] E. Jonardi, et al., "Energy cost optimization for geographically distributed heterogeneous data centers," *Green Computing Conference and Sustainable Computing Conference, 6 pp., 2015.*

[47] V. Antonenko and R. Smelyanskiy, "Global network modelling based on mininet approach.," *in Proc. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 145–146, 2013.*

[48] M. Me´dard and S. S. Lumetta, "Network reliability and fault tolerance," *Encyclopedia of Telecommunications, 2003.*

[49] P. K. Patra, H. Singh, and G. Singh, "Fault tolerance techniques and comparative implementation in cloud computing," *International Journal of Computer Applications, vol. 64, pp. 1–6, 2013.*

[50] C. Wang et al., "Proactive process-level live migration in hpc environments," *in Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, pp. 43:1–43:12, 2008.*

[51] A. Gainaru et al., "Fault prediction under the microscope: A closer look into hpc systems," *in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, pp. 77:1–77:11, 2012.*

[52] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks, vol. 51, pp. 3448–3470, 2007.*

[53] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR), vol. 41, pp. 15:1–15:58, 2008.*

[54] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security, vol. 21, pp. 439–448, 2002.*

[55] C.-K. Hsu et al., "Test data analytics–exploring spatial and test-item correlations in production test data," *in Proc. IEEE International Test Conference (ITC), pp. 1–10, 2013.*

[56] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy," *in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, pp. 1226–1238, 2005.*

[57] F. Ye, Z. Zhang, K. Chakrabarty, and X. Gu, "Board-level functional fault diagnosis using artificial neural networks, support-vector machines, and weighted-majority voting," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, pp. 723–736, 2013.*

[58] H. Goudarzi and M. Pedram, "Geographical load balancing for online service applications in distributed datacenters," *in CLOUD, pp. 351–358, 2013.*

[59] S, Jin, Z. Zhang, K. Chakrabarty, and X. Gu, "Accurate anomaly detection using correlation-based time-series analysis in a core router system", *Proc. IEEE International Test Conference, 2016.*