Comput Sci Res Dev
DOI 10.1007/s00450-014-0268-6

SPECIAL ISSUE PAPER

Exploring energy-performance-quality tradeoffs for scientific workflows with in-situ data analyses

Georgiana Haldeman · Ivan Rodero · Manish Parashar · Sabela Ramos · Eddy Z. Zhang · Ulrich Kremer

Abstract Power and energy are critical concerns for high performance computing systems from multiple perspectives, including cost, reliability/resilience and sustainability. At the same time, data locality and the cost of data movement have become dominating concerns in scientific workflows. One potential solution for reducing data movement costs is to use a data analysis pipeline based on in-situ data analysis.However, the energy-performance-quality tradeoffs impact of current optimizations and their overheads can be very hard to assess and understand at the application level.In this paper, we focus on exploring performance and power/energy tradeoffs of different data movement strategies and how to balance these tradeoffs with quality of solution and data speculation. Our experimental evaluation provides an empirical evaluation of different system and application configurations that give insights into the energy-performance-quality tradeoffs space for in-situ data-intensive application workflows. The key contribution of this work is a better understanding of the interactions between different computation, data movement, energy, and quality-of-result optimizations from a power-performance perspective, and a basis for modeling and exploiting these interactions.

G. Haldeman · I. Rodero (✉) · M. Parashar
Rutgers Discovery Informatics Institute and NSF Cloud and Autonomic Computing (CAC) Center, Rutgers University, Piscataway, NJ, USA
e-mail: irodero@rutgers.edu

G. Haldeman
e-mail: haldeman@cac.rutgers.edu

M. Parashar
e-mail: parashar@rutgers.edu

S. Ramos
Computer Architecture Group, University of A Coruña, A Coruña, Spain
e-mail: sramos@udc.es

E. Z. Zhang · U. Kremer
Department of Computer Science, Rutgers University, Piscataway, NJ, USA
e-mail: eddy.zhengzhang@cs.rutgers.edu

U. Kremer
e-mail: uli@cs.rutgers.edu

Keywords Power/performance tradeoffs · In-situ data analysis · Data staging · Data speculation

1 Introduction

Recent technological advances and trends are significantly changing scientific computing along multiple dimensions. System architectures are composed of multicore processors with increasing core counts, closely coupled accelerators and/or co-processors, and deeper memory hierarchies. Furthermore, power and energy are becoming important concerns from multiple perspectives, including cost, reliability/resilience and sustainability.

As we approach the limits of current technologies, there will be even more severe constraints on energy and power at all levels, and tradeoffs between performance, power/energy, resilience, etc., will be essential. For example, building an exascale system with a budget of 20MW means a budget of 2pJ per operation. This 2pJ budget per operation includes getting the data for the operation, completing the operation, and storing the data. We believe that these performance and energy efficiency targets can only be achieved using a combination of optimizations along the dimensions of computation, data movement, desired result quality, and power/energy. Furthermore, data locality and data movement will be important aspects in satisfying these extreme efficiency constraints,

Published online: 05 July 2014

and it is essential to understand related costs and tradeoffs as part of the data analysis pipeline.

Current research efforts that address these challenges are largely disjoint and are mostly designed to optimize performance and utilization, and can negatively impact power/energy behaviors. For example, optimizations that include some form of speculative execution can introduce a significant work overhead along the non-critical path of a computation with only limited performance benefits. Finding the best energy-performance tradeoff, e.g., the right level of speculation, is therefore a crucial challenge. The same holds for adjusting the quality of specific application characteristics, for instance in terms of a selected spatiotemporal resolution granularity, or frequency of a data analytics feedback steps in a scientific workflow. We believe that the application should be involved in making energy-performance-quality tradeoff decisions. However, the energy, performance and quality impact of current optimizations and their overheads can be very hard to assess and understand at the application level.

In this paper, we target data-intensive application workflows that generate/process large amounts of raw data at runtime and analyze it in-situ (i.e., where it is generated). Specifically, we focus on a synthetic workflow that reproduces the behavior of a combustion simulation workflow with an in-situ data analysis pipeline. Such a workflow needs to process this data as often as possible to facilitate, for example, more accurate or faster scientific discovery. The quality of solution depends on different factors such as the frequency of analysis of the produced data, the accuracy of the analytics algorithm used (e.g., single precision vs. double precision) or number of cores used for performing the analysis.

We study performance and power/energy tradeoffs of different data processing configurations and data movement strategies, and how to balance these tradeoffs with the quality of solution. We analyze these tradeoffs in detail for a complete deep memory hierarchy using a canonical in-situ data analysis workflow. We specifically focus on performance and power/energy tradeoffs of different strategies for data movement, and study how to balance these tradeoffs with the quality of solution (i.e., frequency of analysis and number of resources/cores for performing the analysis) for the targeted type of workflow. We also propose and study data speculation techniques (also known as "prefetching") for transferring data across levels of the deep memory hierarchy leveraging the iterative and predictive behavior of scientific applications. Such strategies can significantly reduce I/O costs and optimize energy consumption if the extra power cost is acceptable and does not result in memory contention.

We also present an empirical evaluation of different system and application configurations that gives insights into the energy-performance-quality tradeoffs space for in-situ data-intensive application workflows. This work is crucial for better understanding the interactions between different computation, data movement, energy, and quality-of-result optimizations, and provides the foundations for modeling and exploiting these interactions. The main contributions of this paper are: (1) a comprehensive study of performance, power and energy behaviors of in-situ data analysis pipelines using different resource configurations and data paths in a (deep) memory hierarchy, and (2) a study of the tradeoffs between power/performance and quality of the solution, and speculation-based techniques. The tradeoffs observed in this study can be used to define an autonomic runtime that can dynamically select the most appropriate configurations, data placement, data paths and data movement optimizations.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the key data-related challenges in executing typical data-intensive scientific workflows and optimization strategies, such as speculation. Section 4 provides an experimental evaluation of power/performance behaviors and tradeoffs of different data-centric strategies. Finally, Sect. 5 concludes the paper and outlines directions for future work.

2 Related work

In this section, we first present existing work related to the cost of data movement. We then discuss prior studies on data speculation and, finally, we describe the related approaches characterizing power/performance tradeoffs for scientific workflows in systems with deep memory hierarchies.

Data movement in complex memory hierarchies imposes challenges in both energy efficiency and performance. Extensive prior work focus on algorithm optimization for data movement minimization [2,30]. Perrone et al. [25] present optimizations based on domain partitioning to reduce data movement on BlueGene. The contemporary heterogeneous accelerator-based system structure complicates the data movement between slower large storage, faster memories and on-chip caches [4], which motivates the development of tools like the one discussed in [9], that implement code-analysis components to enable efficient data movement in heterogeneous systems. The power consumption due to the data movement is non-trivial, and Kestor et al. [17] present a characterization of energy cost from data movement in scientific applications.

Speculative execution has been largely used to enhance parallelism, from instruction level parallelism such as branch prediction in a microprocessor, to task level parallelism in a heterogeneous system, such as [10], in which Diamos et al. explore a dynamic optimization technique for speculative execution of GPU kernels. In order to enable parallel execution of sequential codes, Hammond et al. [16] implement thread-level speculation, and Balakrishnan et al. [3]

use speculative inputs to launch methods on other processors for optimization of sequential programs. There is a large body of literature on speculation for modern multi-core systems, including [29] which provides profiled-based speculative parallelization, and many of them are focused on I/O data speculation. For instance, Nightingale et al. [24] propose Linux kernel support for multiprocess speculative execution, which improves the throughput of distributed file systems by masking I/O latency.

An extensive number of publications have already addressed energy efficiency focusing on the memory hierarchy and storage-class memory with solutions such as μblades [22], FAWN [1] and Gordon [6]. Some prior work [8] exploit the use of low-power (thin or wimpy) cores that consume less energy, typically with the help of micro-benchmarks [26]. Dong et al. [11] propose 3D stacked magnetic memory (MRAM) caches for better power efficiency. Yoon et al. [31] study the potential impact of NVRAM on the performance of deep memory hierarchies and Li et al. [21] study the potential impact of hybrid DRAM and byte-addressable NVRAM on both performance and energy perspectives of scientific applications. Our previous work [15] also addresses energy efficiency from an application-aware and data-centric perspective, focusing on optimizing data placement/movement and scheduling computations as part of end-to-end simulation workflows.

Energy-efficiency is becoming a major concern in large-scale scientific applications. Shalf et al. [12] explore energy efficiency for extreme-scale scientific applications and address software-architecture co-design by comparing different architectural alternatives such as multi-cores, GPUs and many-cores [18]. Rountree et al. [28] developed a system called Adagio to collect statistical data on task execution slacks for applying dynamic voltage and frequency scaling (DVFS) techniques. Rodero et al. [27] studied application-centric aggressive power management for HPC workloads considering power management mechanisms and controls available at different levels and for different subsystems. Li et al. [19] focus on the hybrid MPI/OpenMP programming model and used DVFS to reduce the energy requirements of hybrid application codes for several benchmarks in HPC systems. They also developed a framework to predict the performance effect of task aggregation in both computation and communication phases and its impact in terms of execution time and energy of MPI programs [20]. Lively et al. [23] investigate energy and performance characteristics of different parallel implementations of scientific applications on a multicore cluster system, and explore interactions between power consumption and performance. Durillo et al. [13] study the potential benefits of using a Pareto-based workflow scheduling algorithm using energy consumption and performance models for task executions. Gamell et al. [14] explored data-related energy-performance tradeoffs and co-design choices on current and ongoing high-end computing platforms.

Other existing work characterized power-performance behaviors and tradeoffs of scientific applications; however, at the best of our knowledge, this is the first work that explores energy-performance-quality tradeoffs for scientific workflows with in-situ data analyses with speculative data movement.

3 Scientific workflows with in-situ data analysis

In this section we provide a description of the specific class of scientific workflows that we target in this work, and the data management challenges addressed.

3.1 Targeted workflow

The synthetic workflow used in this paper reproduces the behavior of a class of scientific workflows with an in-situ analysis typically used in production simulations. Our ultimate goal is understanding data-related issues for an in-situ analysis workflow integrated with S3D [7], a massively parallel turbulent combustion code. S3D performs first-principles-based direct numerical simulations of turbulent combustion in which both turbulence and chemical kinetics associated with burning gas-phase hydrocarbon fuels introduce spatial and temporal scales spanning typically at least five decades. We use a matrix multiplication kernel in order to simplify the exploration space (S3D requires n^3 MPI ranks) and focus on the interaction between components. However, the selected matrix multiplication kernel and S3D have similar power signatures as shown in Fig. 1.

3.2 In-situ data analytics

In-situ analysis is performed where the data is located, and it typically shares memory resources with the primary scientific simulation. The main advantage of in-situ analysis is that it avoids the data movement. However, constraints on the acceptable impact on the simulation place significant restrictions on type and frequency of in-situ analysis. In contrast, in-transit pipelines analyze the data while it is being staged on separate resources, either local or remote. The decision on whether to perform in-situ and/or in-transit data analysis is based on data location and performance/energy constraints. For example, one may use NVRAM as extended DRAM to analyze data in-situ rather than transfer data over the network or offload it to disk, in order to save the time and energy required to transfer the data. In this paper, we focus on in-situ data analysis and on using a deep memory hierarchy to support data staging for the analysis, as illustrated in Fig. 2. Although multiple memory levels could be used simultane-

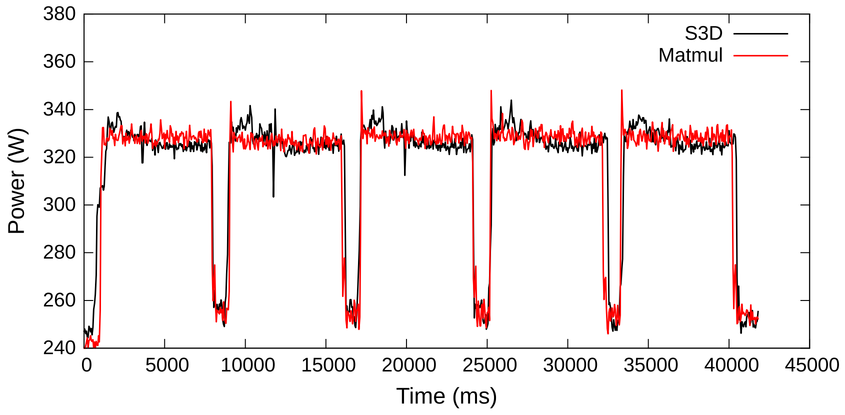


Fig. 1 Power signature of S3D and the matrix multiplication kernel. In both cases 8 cores were used and the generated data was written to hard disk

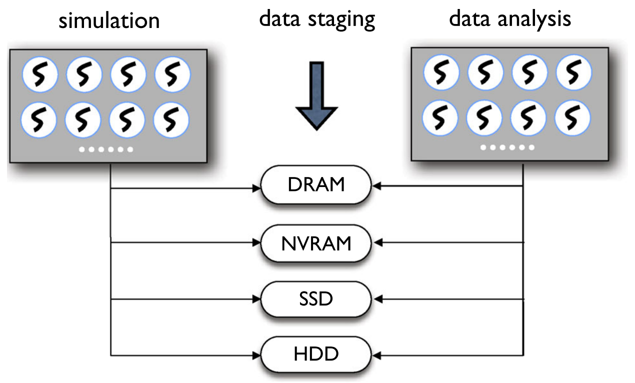


Fig. 2 Proposed data staging architecture. Simulation processes store data in one level of the memory hierarchy (e.g., NVRAM) and analysis components load the data from this memory level

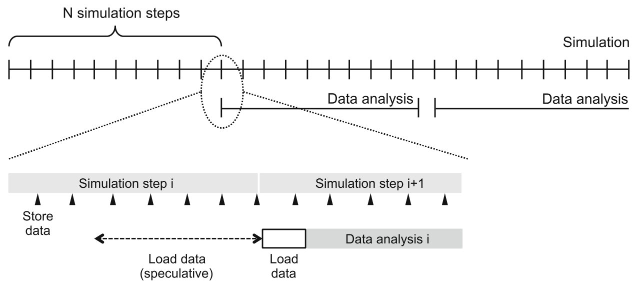


Fig. 3 Targeted data analysis pipeline illustrating execution stages for N simulation steps—analysis of data from step i can be overlapped with simulation step $i+1$; the simulation of step $i+2$ cannot start until analysis of step i data is completed

ously, in this paper we study the use of each memory level individually for data staging.

We target scientific and engineering workflows with coupled application components that typically generate large amounts of raw data at runtime. In general, we propose to perform the data analysis as often as possible to facilitate more accurate scientific discovery. This model of accompanying scientific workflows with a data analysis pipeline is illustrated in Fig. 3. In the data analysis pipeline, an arbitrary number of coupled parallel application components (e.g., simulations) generate data (typically in DRAM) at arbitrary rates, and data analytics components process the data where it is generated. However, as the execution of the scientific application progresses, DRAM becomes insufficient for storing the raw data, and a data staging area in other memory levels (e.g., disk or remote memory) is required, which introduces associated I/O overhead. Clearly, the workflows can utilize the resources available on emerging architectures more effectively if the workload can be mapped and scheduled to exploit the data locality as well as the communication patterns between workflow components. NVRAM is one of the most important elements of emerging storage architectures due to its power efficiency. We propose to use NVRAM as staging area for the data movement between simulation and data analysis components, the exploration of which will be elaborated in Sect. 4.

3.3 Quality of the solution

Simulation-based application workflows are typically iterative (e.g., consisting of multiple time steps as illustrated in Fig. 3). Each simulation step usually generates data, and, although this data is not typically processed and analyzed at every step, it must be analyzed as often as possible to improve the quality of the solution. For example, turbulent combustion direct numerical simulation currently resolve intermittent phenomena that occur on the order of 10 simulation steps; however, in order to maintain I/O overheads at a reasonable level, typically only every 400th step is saved to persistent storage for post-processing and, as a result, the data pertaining to these intermittent phenomena is lost [5]. We use the frequency of data analysis as metric for the quality of solution in this paper.

3.4 Data speculation

Speculative execution has been extensively used as an optimization technique in different areas such as branch prediction, pipelined processors and optimistic concurrency control. It provides more concurrency if extra resources are available. Speculative data movement can potentially improve the performance of application workflows; however, the possible power/energy costs associated with data movement may be large. In this paper we target a deep memory hierarchy composed of multiple levels such as DRAM, NVRAM and SSD/disk storage. Efficiently speculating about data transfers within this deep memory hierarchy requires understanding its possible impact on performance as well as on energy/power for each of the levels of the memory hierarchy.

In this paper, we study data speculation in the context of an in-situ scientific workflow that processes data iteratively. Hence, speculation consists on loading the data that is expected to be processed in the subsequent iteration while the current data is being processed. As data speculation may not always be accurate, it can result in larger energy consumption. Thus, it is important to identify when speculation is beneficial (e.g., to minimize energy consumption or capping power) and which levels of the memory hierarchy can be used. Furthermore, using speculation under memory contention may result in significant performance degradation. Such situations can be determined at runtime, which is one of the goals of this paper.

4 Experimental evaluation

4.1 Evaluation methodology

Hardware testbed The evaluation has been conducted on the NSF-funded research instrument "Computational and dAta Platform for Energy efficiency Research" (CAPER). This is an eight-node cluster based on SuperMicro SYS-4027GR-TRT system, which is capable of housing concurrently, in one node up to eight general-purpose graphical processing units (GPGPU), or eight Intel many-integrated-core (MIC) coprocessors—or any eight-card combination of the two; and up to 48 hard disk drives (HDD), or solid-state drives (SSD). Its nominal configuration features servers with two Intel Xeon Ivy Bridge E5-2650v2 (16 cores/node), 128 GB of DRAM, 1TB of Flash-based NVRAM (i.e., Fusion-io IoDrive-2), 2TB of SSD and 4TB of hard disk, one Intel Xeon Phi 7120P, and Infiniband FDR network connectivity. This platform also mirrors key architectural characteristics of high-end system, such as XSEDE's Stampede system at TACC, and provides several unique features to support our research goals. Furthermore, CAPER is instrumented with both coarse- and fine-grained power metering at server level—an instrumented Raritan PDU provides power measurements at 1 Hz, and a Yokogawa DL850E ScopeCorder data acquisition recorder provides power measurements at up to 1 kHz (from 1 Ms/s current and voltage modules). CAPER provides us with a platform to validate our models and investigate key aspects of data-centric and energy efficiency research. Our experimental evaluation was conducted using the fine-grained instrumentation system at server level with power readings at 50 Hz. The system was configured with turbo mode enabled and default OS-level DVFS capabilities.

Software framework We have built a multi-threaded framework which reproduces the behavior of the workflow described in Sect. 3. It can be used to evaluate the different configurations in terms of execution time and energy/power. The framework divides the execution into stages. It uses a synthetic workload, which (iteratively) performs the following tasks:

1. simulation stages execute matrix multiplication kernels (i.e., CPU-intensive computation),
2. the results of the matrix multiplications are stored to a predefined memory level (i.e., HDD, SDD, etc.),
3. loads the data into memory, and
4. performs the analysis using a word finding kernel.

The amount of data, the rate of the simulation and the analysis can also be configured. In the evaluation presented in this paper, stages are composed of six steps and the matrix multiplication kernel generates 1 GB of data per step (i.e., 6 GB per stage). The data is written to the devices using regular write operations.

The software framework uses a configuration file to define all parameters. After setup, a monitor thread is spawned. The monitor is in charge of spawning the necessary worker threads to execute different tasks and to ensure the flow of the application. Given that the framework is used to assess performance of the application, we want to limit the overhead of the monitor, so that all the computing power is used by the application. To do so, upon creation the workers are grouped based on the type of job they are executing (i.e., simulation, store data, analysis, load data) and each group is assigned a queue of tasks. This solution not only ensures limited overhead from the monitor's perspective, but it also enforces some control on the flow of the execution. As a result the program needs minimal synchronization which is another requirement because enforcing synchronization can result in some or all threads being idle. It is undesirable to have idle time since the server consumes power even when it's idle and there are two situation in which this can happen: (1) between stages of the execution (which is achieved by the use of limited synchronization tool alone), and (2) between threads from the same group, i.e., the threads finish their tasks at different times (which is achieved by dividing the workload in smaller tasks and making sure the number of tasks it is divisible by the number of threads). To ensure fair share of the CPU, the number of working threads at any time does not exceed the number of physical cores available on the machine, unless the overloading thread is signaled.

The framework's view of the memory is that it is divided into four levels: DRAM, NVRAM, SSD and HDD. So far we have explored fixed data paths, i.e., the memory level where the data will be stored exclusively needs to be specified in the configuration file. In this paper we do not use buffering-based optimizations (e.g., double- or triple-buffering) to retrieve data from different memory levels. This serves as a baseline for future experiments in which we plan to combine memory levels, optimizations and implement autonomic algorithms for the dynamic placement of the data.

The following subsections provide a characterization of the performance and energy/power behaviors of in-situ data analytics using different technologies for data staging and analyzes tradeoffs between the quality of the solution and speculative techniques for data staging.

4.2 Data staging over the (deep) memory hierarchy

Figure 4 displays the execution time (top), energy consumption (center) and average power (bottom) of the workflow's execution using different configurations and paths (i.e., devices) for data staging. It shows that when the number of cores used for simulation increases, the simulation time decreases while the analysis time increases. In general, the execution time with HDD is much longer than with the other devices; however, the difference in energy is lower due to the limited power requirements of the HDD. The I/O cost (in terms of execution time) is around 25 % larger with HDD compared to SSD, on average. However, the I/O cost difference between flash-based devices (i.e., NVRAM and SSD) is lower than 5 %, on average. The I/O costs are dominated by store operations as the workflow performs load operations at once while stores are interleaved within the simulation steps. Further, the workload execution time deviation is small for DRAM, NVRAM and SSD, but it is large for HDD.

Figure 4 (center) also shows that for 8s_8a configuration (i.e., 8 cores used for simulation and 8 for analysis) DRAM, NVRAM and SSD energy consumptions are close, for 10s_6a NVRAM and SSD energy consumptions are grouped together and at a equal difference between DRAM and NVRAM energy consumption, and for the other two configurations, the NVRAM and SSD energy consumptions get close to HDD. This is because with four or less cores the workload execution is dominated by the analysis time and it results in significant idle time in the cores running simulations. However, energy consumption and execution time are quite correlated.

We can also observe some tradeoffs, for example NVRAM is faster than SSD but it does consume more energy and requires higher power as seen in 10_6 and 12_4 configurations, because the average power—see Fig. 4 (bottom)—is higher in NVRAM compared to SSD and HDD.

4.3 Tradeoffs with quality of the solution

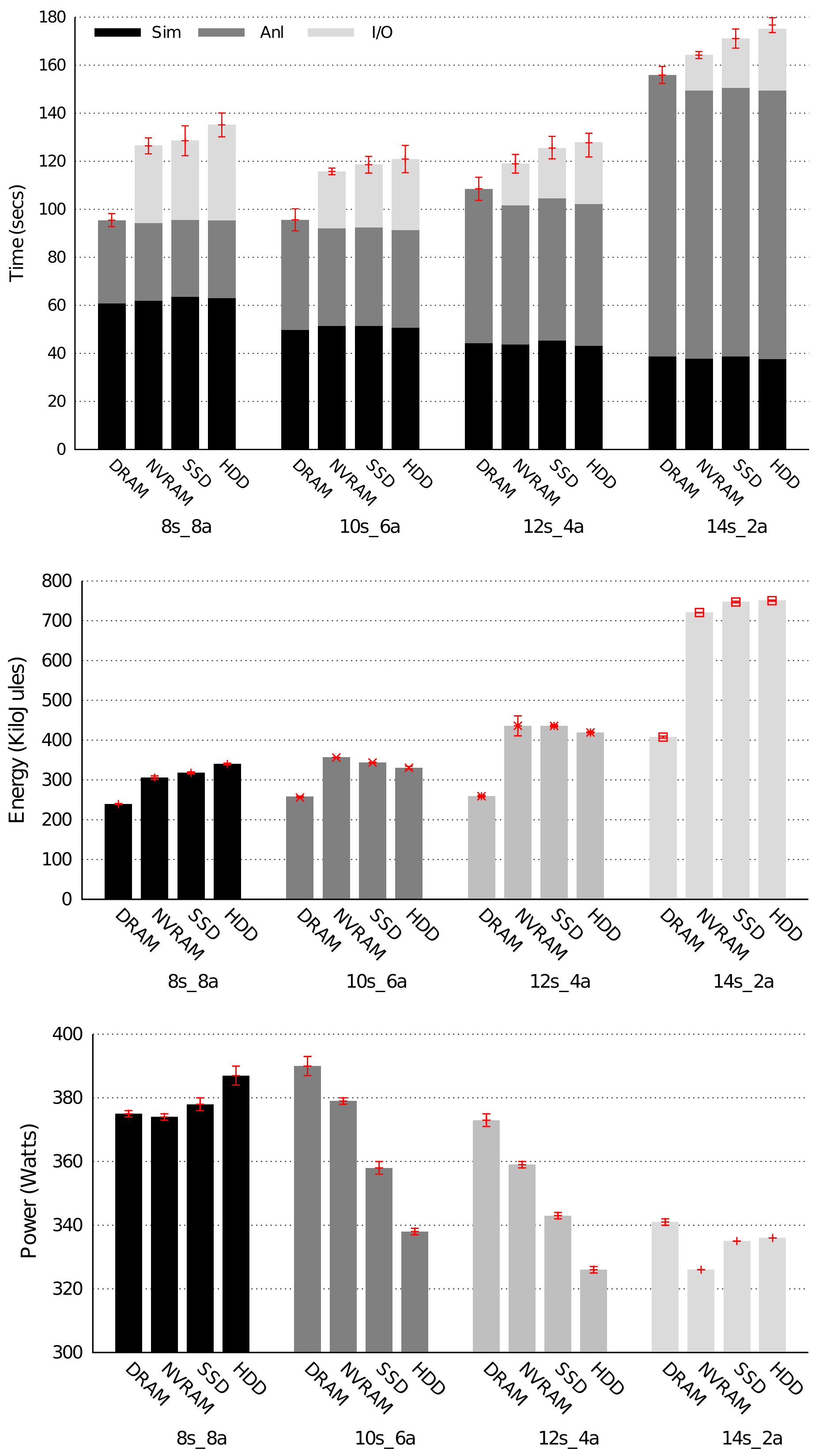
Figure 5 displays the execution time (top), energy consumption (center) and average power (bottom) of the workflow's execution for different frequencies of analysis. It shows that both execution time and energy decrease as the number of simulation steps between data analyses increases (i.e., frequency of analysis decreases). Execution time and energy consumption drastically decrease from $foa = 1$ to $foa = 4$; however, they do decrease moderately with $foa \geq 6$. It means that, for this specific configuration (i.e., 12 cores for simulation and 4 for analysis), it is not worth to perform the data analysis with less frequently than every six simulation steps, as the savings in execution time and energy are low while the impact on the scientific discovery (e.g., to be able to visualize phenomena that are changing rapidly) associated to the workflow's execution may be large. Thus, the best tradeoff between performance/energy and quality of solution is found when data is analyzed every 4–6 simulation steps. Figure 5 (bottom) also shows that power increases as the number of simulation steps between data analyses increases. The reason is that the execution time is shorter and, consequently, the CPU cores run simulation kernels (which are more power demanding than data analysis) for a larger percentage of time.

4.4 Data speculation

Since there are different dimensions associated to the speculation such as its starting point, and its accuracy, we have fixed the configuration to 12a_4a (i.e., 12 cores for simulation and 4 for analysis) and the speculation start at 50 % of the simulation stage. This is an approximation because data speculation can be performed only when the data to be moved is already available (i.e., existing data from previous simulation stages or data already produced in the previous simulation steps of the current stage). Figure 6 displays the execution time (top), energy consumption (center) and average power (bottom) of the workflow's execution using different devices for data staging and different speculation accuracy levels. It shows that as the accuracy increases the execution time and energy consumption decreases (between 10–20 %, on average depending on the device). When the speculation is accurate, the difference in energy of NVRAM, SSD, etc. is lower as speculation can be completed before the simulation stage finishes. Figure 6 (bottom) shows that average power is lower without speculation, especially with HDD.

To understand better the performance/energy/power costs and tradeoffs with speculation, Fig. 7 displays the outcomes

Fig. 4 Execution time (*top*), energy consumption (*center*) and average power (*bottom*) of the workflow's execution using different configurations and devices for data staging. Each group of columns represents one configuration (number of cores for simulation/analysis) and each column (*top*) provides the time breakdown for the different phases of the workflow. *Error boxes* show average and standard deviation of five runs



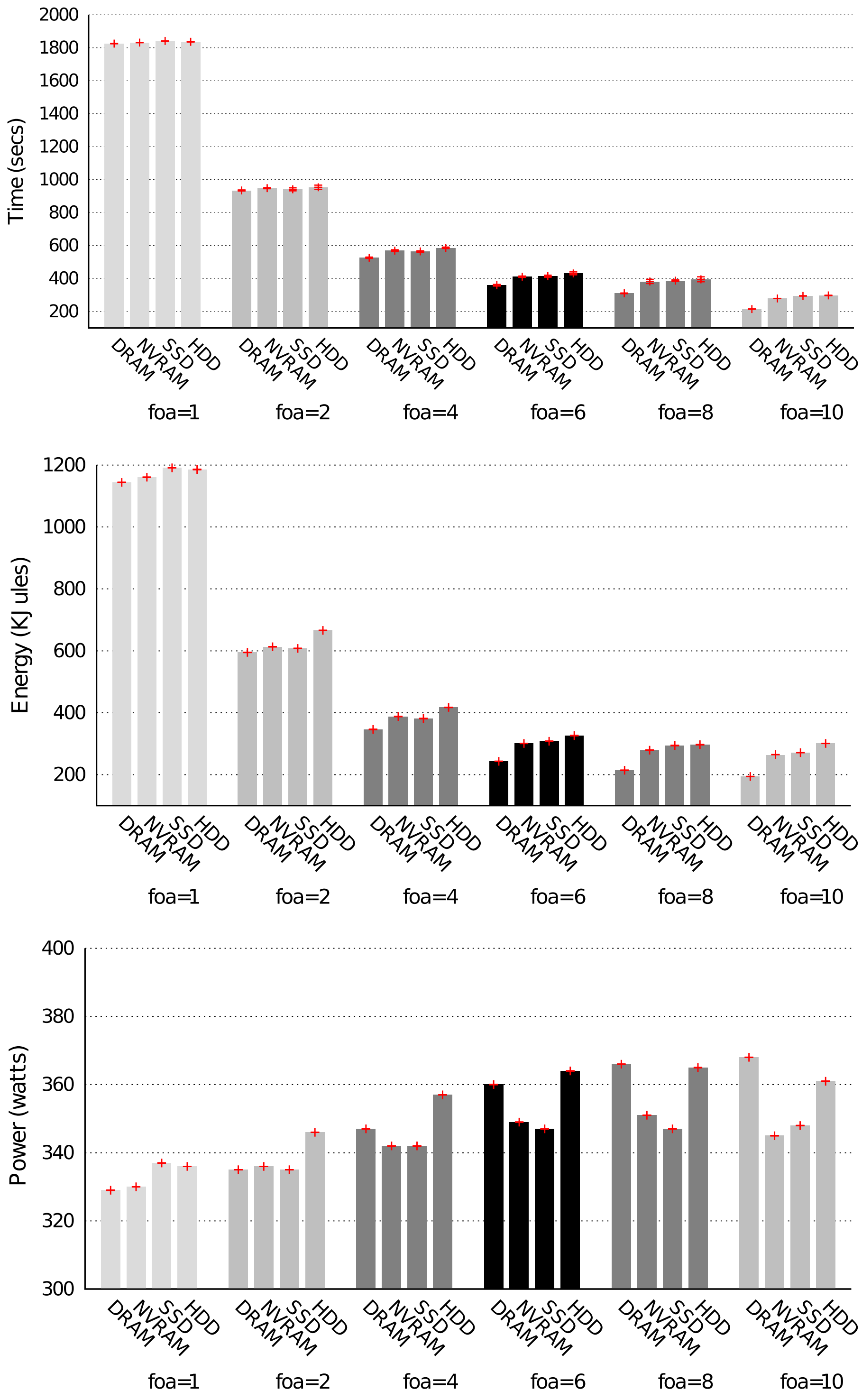


Fig. 5 Execution time (*top*), energy consumption (*center*) and average power (*bottom*) of the workflow's execution for different frequency of analysis ("foa") and different devices for data staging. Frequency of analysis $foa = k$ means that the data analysis is performed every k simulation steps. *Error boxes* show average and standard deviation of five runs

of the workflow's execution with some relevant configurations. The top three outcomes show the workload's stages over time using different devices in the data path. The execution time is shorter using DRAM and NVRAM compared to HDD, as expected. The bottom three outcomes show how speculation can reduce the execution time when the speculation is accurate. When the speculation is not accurate (last outcome in Fig. 7) there is no improvement in the execution time but there is a significant cost of energy and power. Thus, the key aspect is balancing the use of speculation and its associated costs depending on the confidence of the speculation, which can be realized using an autonomic runtime and/or prediction techniques. Figure 7 also shows relevant tradeoffs. For example, the use of NVRAM and HDD with

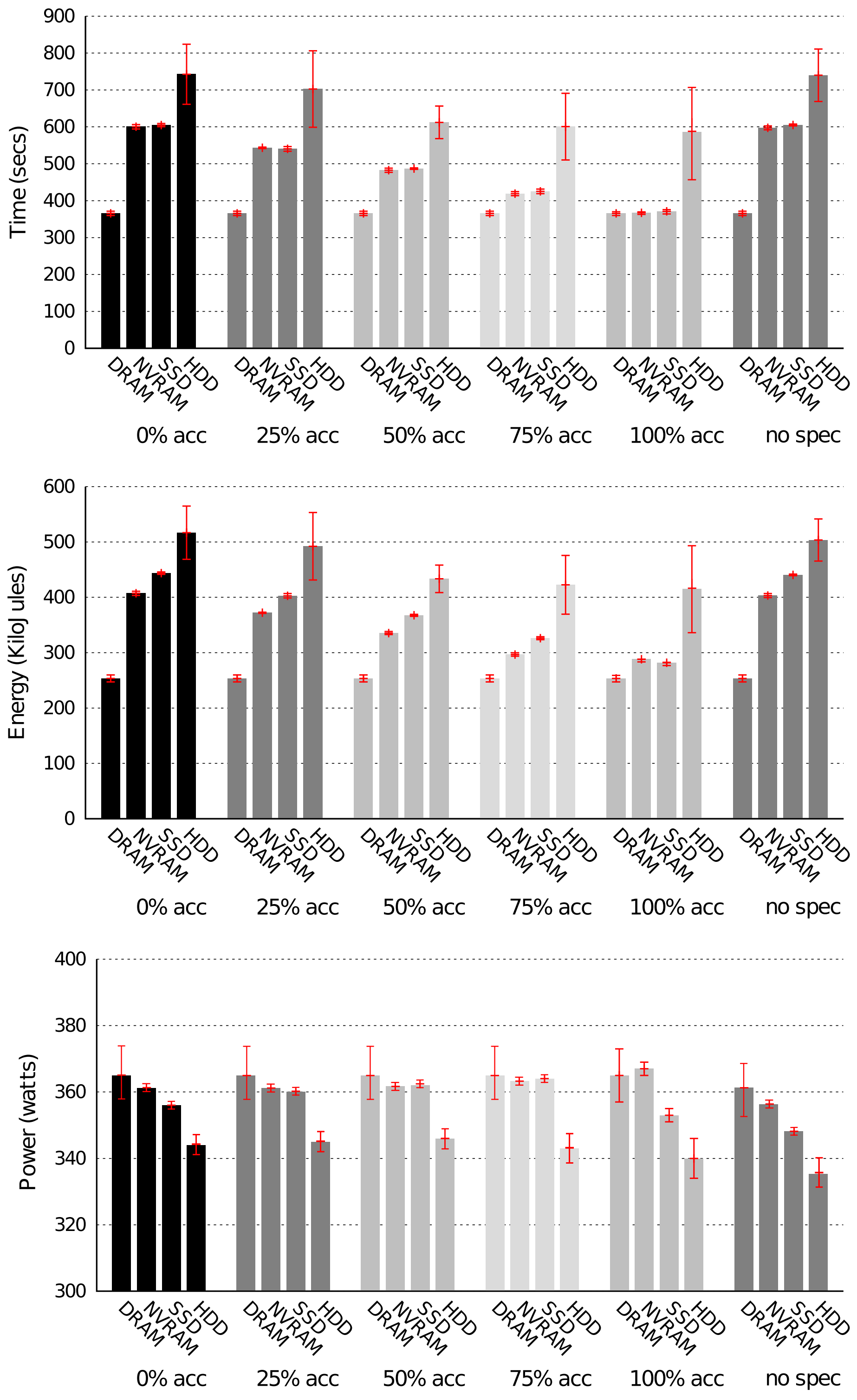


Fig. 6 Execution time (*top*), energy consumption (*center*) and average power (*bottom*) of the workflow's execution using different devices for data staging and different accuracy levels in the data speculation. *Error boxes* show average and standard deviation of five runs

accurate speculation provide similar execution times; however, the energy cost is lower when using speculation while the power cost is higher, as shown in Fig. 8. Thus, we can conclude that speculation can optimize execution time and energy, but it should be used only if there is enough power budget available.

5 Conclusion and future work

This paper provides an evaluation of data-intensive application workflows with in-situ data analysis in terms of tradeoffs between performance and power/energy behaviors. We analyze different configurations for both architecture and appli-

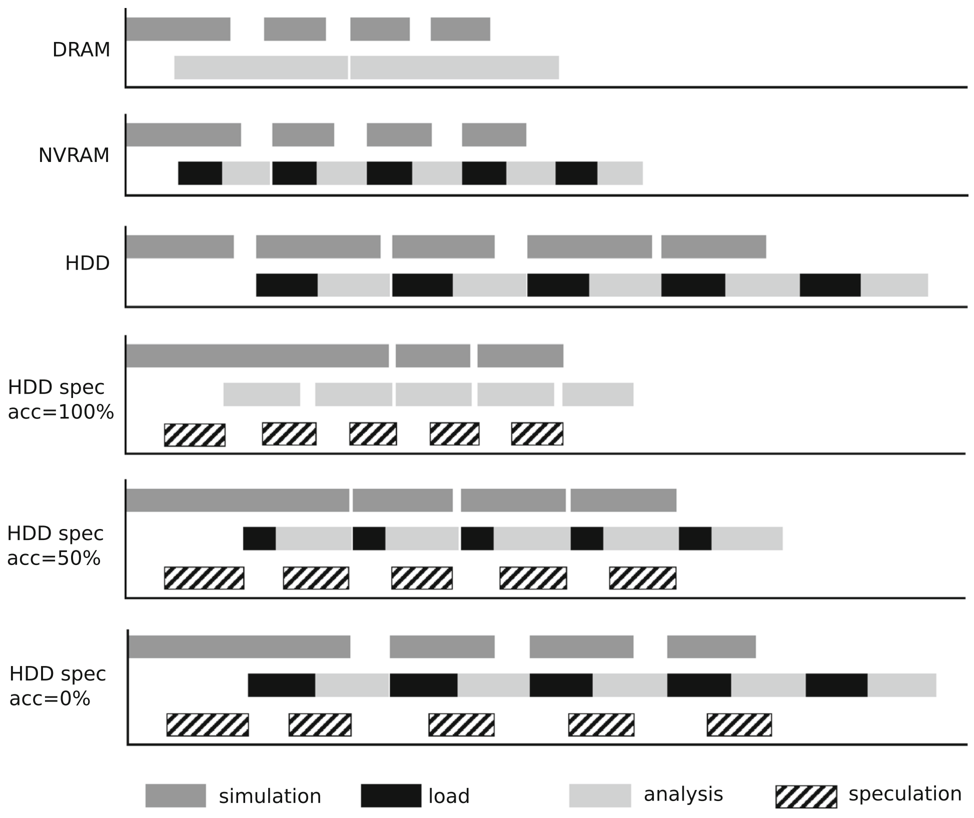


Fig. 7 Outcomes of the workload execution using different devices and configurations. In the outcomes that use HDD with speculation, the speculation starts at 50 % of the simulation stage. The *X* axis represents time

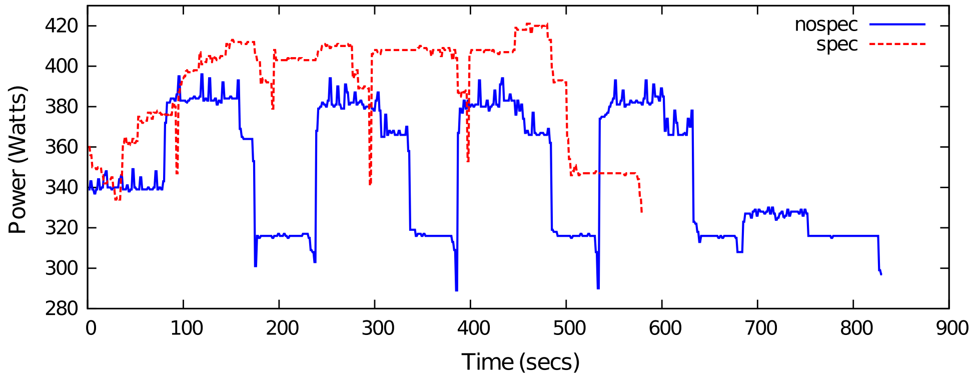


Fig. 8 Power dissipation over time using NVRAM (i.e., "nospec") and using HDD with speculation starting at 50 % with accuracy = 100 % (i.e., "spec"). The energy using NVRAM and HDD with speculation is 403 and 388 kJ, respectively

cation, considering deep memory hierarchies, data speculation and quality of the solution, among other parameters. The empirical evaluation and the analysis provided is key in setting the basis for modeling and optimizing data-intensive scientific workflows.

Our future work includes extending this characterization in order to analyze the impact of the use of co-processors along with the deep memory hierarchy (e.g., offloading the analytics component to Intel MIC co-processors). We also plan to validate these results with multiple nodes and using real applications (e.g., S3D) instead of a synthetic workflow. Furthermore, we will apply this work in order to develop a comprehensive model of data-intensive workflows and use it for developing an autonomic runtime that can balance dynamically the tradeoffs studied in this paper.

Acknowledgments The research presented in this work is supported in part by U.S. National Science Foundation (NSF) via grants numbers OCI-1339036, OCI-1310283 and CNS-1305375, and by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the US Department of Energy through the ExaCT Combustion Co-Design Center via subcontract number 4000110839 from UT Battelle. The research was conducted as part of the NSF Cloud and Autonomic Computing (CAC) Center at Rutgers University and the Rutgers Discovery Informatics Institute (RDI2) and partially supported by a Rutgers Research Council Grant. The research presented in this paper is also supported by the Galician Government (Projects CN2012/211 and GRC2013/055) and partially supported by FEDER funds of the

European Union. We also thank the anonymous reviewers for their comments and recommendations, which helped to considerably improve the quality of the manuscript.

References

1. Andersen DG, Franklin J, Kaminsky M, Phanishayee A, Tan L, Vasudevan V (2009) Fawn: a fast array of wimpy nodes. In: SIGOPS symposium on operating systems principles, pp 1–14
2. Avron H, Gupta A (2012) Managing data-movement for effective shared-memory parallelization of out-of-core sparse solvers. In: 2012 International conference for high performance computing, networking, storage and analysis (SC), pp 1–11
3. Balakrishnan S, Sohi GS (2006) Program demultiplexing: data-flow based speculative parallelization of methods in sequential programs. In: Proceedings of the 33rd annual international symposium on computer architecture, ISCA '06, pp 302–313
4. Baskaran MM, Bondhugula U, Krishnamoorthy S, Ramanujam J, Rountev A, Sadayappan P (2008) Automatic data movement and computation mapping for multi-level parallel architectures with explicitly managed memories. In: Proceedings of the 13th ACM SIGPLAN symposium on principles and practice of parallel programming, PPoPP '08, pp 1–10
5. Bennett JC, Abbasi H, Bremer PT, Grout R et al (2012) Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In: International conference on high performance computing, networking, storage and analysis (SC), pp 49:1–49:9
6. Caulfield AM, Grupp LM, Swanson S (2009). Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. In: International conference on architectural support for programming languages and operating systems, pp 217–228
7. Chen JH, Choudhary A, de Supinski B, DeVries M et al (2009) Terascale direct numerical simulations of turbulent combustion using s3d. Comput Sci Discov 2:1–31
8. Cockcroft AN (2007) Millicomputing: the coolest computers and the flashiest storage. In: International computer measurement group conference, pp 407–414
9. Dathathri R, Reddy C, Ramashekar T, Bondhugula U (2013) Generating efficient data movement code for heterogeneous architectures with distributed-memory. In: 2013 22nd International Conference on parallel architectures and compilation techniques (PACT), pp 375–386
10. Diamos G, Yalamanchili S (2010) Speculative execution on multi-GPU systems. In: 2010 IEEE international symposium on parallel distributed processing (IPDPS), pp 1–12
11. Dong X, Wu X, Xie Y, Chen Y, Li H (2011) Stacking MRAM atop microprocessors: an architecture-level evaluation. IET Comput Digit Tech 5(3):213–220
12. Donofrio D, Oliker L, Shalf J, Wehner MF, Rowen C, Krueger J, Kamil S, Mohiyuddin M (2009) Energy-efficient computing for extreme-scale science. Computer 42(11):62–71
13. Durillo J, Nae V, Prodan R (2013) Multi-objective workflow scheduling: an analysis of the energy efficiency and makespan tradeoff. In: 2013 13th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid), pp 203–210
14. Gamell M, Rodero I, Parashar M, Bennett J et al (2013) Exploring power behaviors and tradeoffs of in-situ data analytics. In: International conference on high performance computing networking, storage and analysis (SC). Denver, CO, pp 1–12
15. Gamell M, Rodero I, Parashar M, Poole S (2013) Exploring energy and performance behaviors of data-intensive scientific workflows on systems with deep memory hierarchies. In: Proceedings of the 20th international conference on high performance computing (HiPC), pp 1–10
16. Hammond L, Willey M, Olukotun K (1998) Data speculation support for a chip multiprocessor. In: Proceedings of the eighth international conference on architectural support for programming languages and operating systems, ASPLOS VIII, pp 58–69
17. Kestor G, Gioiosa R, Kerbyson D, Hoisie A (2013) Quantifying the energy cost of data movement in scientific applications. In: 2013 IEEE international symposium on workload characterization (IISWC), pp 56–65
18. Krueger J, Donofrio D, Shalf J, Mohiyuddin M, Williams S, Oliker L, Pfreund FJ (2011) Hardware/software co-design for energy-efficient seismic modeling. In: Proceedings of 2011 international conference for high performance computing, networking, storage and analysis (SC'11), pp 73:1–73:12
19. Li D, De Supinski B, Schulz M, Cameron K, Nikolopoulos D (2010) Hybrid MPI/openMP power-aware computing. In: 2010 IEEE International Symposium on parallel distributed processing (IPDPS), pp 1–12
20. Li D, Nikolopoulos D, Cameron K, De Supinski B, Schulz M (2010) Power-aware MPI task aggregation prediction for high-end computing systems. In: 2010 IEEE international symposium on parallel distributed processing (IPDPS), pp 1–12
21. Li D, Vetter JS, Marin G, McCurdy C, Cira C, Liu Z, Yu W (2012) Identifying opportunities for byte-addressable non-volatile memory in extreme-scale scientific applications. In: International parallel and distributed processing symposium, pp 945–956
22. Lim K, Ranganathan P, Chang J, Patel C, Mudge T, Reinhardt S (2008) Understanding and designing new server architectures for emerging warehouse-computing environments. In: Annual international symposium on computer architecture, pp 315–326
23. Lively C, Wu X, Taylor V, Moore S, Chang HC, Cameron K (2011) Energy and performance characteristics of different parallel implementations of scientific applications on multicore systems. Int J High Perform Comput Appl 25(3):342–350
24. Nightingale EB, Chen PM, Flinn J (2006) Speculative execution in a distributed file system. ACM Trans Comput Syst 24(4):361–392
25. Perrone M, Liu LK, Lu L, Magerlein K, Kim C, Fedulova I, Semenikhin A (2012) Reducing data movement costs: scalable seismic imaging on blue gene. In: 2012 IEEE 26th international parallel distributed processing symposium (IPDPS), pp 320–329
26. Rivoire S, Shah MA, Ranganathan P, Kozyrakis C (2007) Joule-sort: a balanced energy-efficiency benchmark. In: SIGMOD international conference on management of data, pp 365–376
27. Rodero I, Chandra S, Parashar M, Muralidhar R, Seshadri H, Poole S (2010) Investigating the potential of application-centric aggressive power management for HPC workloads. In: Proceedings of the IEEE international conference on high performance computing (HiPC). Goa, India, pp 1–10
28. Rountree B, Lownenthal DK, de Supinski BR, Schulz M, Freeh VW, Bletsch T (2009) Adagio: making DVS practical for complex hpc applications. In: International conference on supercomputing, pp 460–469
29. Tian C, Feng M, Nagarajan V, Gupta R (2008) Copy or discard execution model for speculative parallelization on multicores. In: Proceedings of the 41st annual IEEE/ACM international symposium on microarchitecture, MICRO 41, pp 330–341
30. Yan Y, Zhao J, Guo Y, Sarkar V (2010) Hierarchical place trees: a portable abstraction for task parallelism and data movement. In: Proceedings of the 22nd international conference on languages and compilers for parallel computing. LCPC'09. Springer, Berlin, pp 172–187
31. Yoon DH, Gonzalez T, Ranganathan P, Schreiber RS (2012) Exploring latency-power tradeoffs in deep nonvolatile memory hierarchies. In: Conference on computing frontiers, pp 95–102

Georgiana Haldeman received a B.S. in Computer Science from Rutgers University. She is currently pursuing her M.S. studies at Rutgers University. Her research interests are in the area of High-Performance Computing and heterogeneous systems performance, including modeling and optimization.

Ivan Rodero is an Assistant Research Professor of Electrical and Computer Engineering at Rutgers University and a member of the Discovery Informatics Institute and the US National Science Foundation (NSF) Cloud and Autonomic Computing Center. His research falls into the areas of parallel and distributed computing and includes High-Performance Computing, energy efficiency, autonomic computing, grid/cloud computing and big data management and analytics. His current research activities are in the intersection of green computing and big data with a focus on energy-efficient scientific data management and analytics. Dr. Rodero holds a M.S. and Ph.D. in computer science and engineering from Technical University of Catalonia, Spain, in 2004 and 2009, respectively. He is a senior member of IEEE, and member of ACM and AAAS.

Manish Parashar is Professor of Electrical and Computer Engineering at Rutgers University. He is the founding Director of the Rutgers Discovery Informatics Institute (RDI2) and of the NSF Cloud and Autonomic Computing Center (CAC), and is Associate Director of the Rutgers Center for Information Assurance (RUCIA). His research interests are in the broad areas of Parallel and Distributed Computing and Computational and Data-Enabled Science and Engineering. A key focus of his research is on addressing the complexity or large-scale systems and applications through programming abstractions and systems. Manish serves on the editorial boards and organizing committees of a large number of journals and international conferences and workshops, and has deployed several software systems that are widely used. He has also received numerous awards and is Fellow of AAAS, Fellow of IEEE/IEEE Computer Society and Senior Member of ACM. For more information please visit http://parashar.rutgers.edu/.

Sabela Ramos received the B.S. (2009), M.S. (2010) and Ph.D. (2013) degrees in Computer Science from the University of A Coruña, Spain. Currently she is a post-doc researcher at the Department of Electronics and Systems at the University of A Coruña. Her research interests are in the area of High Performance Computing (HPC), focused on message-passing communications on multicore architectures and cluster/cloud computing.

Eddy Z. Zhang is a new assistant professor in the Department of Computer Science at Rutgers University. She received her B.S. in electronics engineering from Shanghai Jiao Tong University. She obtained her M.S. and Ph.D. in computer science from the college of William and Mary. Her research interests are generally in the area of compilers and programming systems. She is a recipient of PPoPP'10 best paper award and QEST'08 best student paper award.

Ulrich Kremer is a Professor in the Department of Computer Science at Rutgers University. He graduated with a Diplom in Informatik from the University of Bonn, Germany. He went on to receive a M.S. and Ph.D. in Computer Science from Rice University in 1993 and 1995, respectively. His research interests include compilation techniques and interactive programming environments for parallel systems, compiler support for power and energy management, programming models and optimizations for dynamic networks of resource constrained devices (e.g.: smart phones), and novel programming architectures or autonomous robots such as autonomous underwater vehicles. Dr. Kremer received a NSF CAREER award for his work on compiler-directed low power and energy optimizations. He is a member of the ACM.