A Performance and Energy Analysis of I/O Management Approaches for Exascale Systems

Orcun Yildiz
Inria, Rennes Bretagne Atlantique Research Center
Rennes, France
orcun.yildiz@inria.fr

Matthieu Dorier
ENS Rennes, IRISA
Rennes, France
matthieu.dorier@irisa.fr

Shadi Ibrahim
Inria, Rennes Bretagne Atlantique Research Center
Rennes, France
shadi.ibrahim@inria.fr

Gabriel Antoniu
Inria, Rennes Bretagne Atlantique Research Center
Rennes, France
gabriel.antoniu@inria.fr

ABSTRACT

The advent of fast, unprecedentedly scalable, yet energy-hungry exascale supercomputers poses a major challenge consisting in sustaining a high performance per watt ratio. While much recent work has explored new approaches to I/O management, aiming to reduce the I/O performance bottleneck exhibited by HPC applications (and hence to improve application performance), there is comparatively little work investigating the impact of I/O management approaches on energy consumption.

In this work, we explore how much energy a supercomputer consumes while running scientific simulations when adopting various I/O management approaches. We closely examine three radically different I/O schemes including time partitioning, dedicated cores, and dedicated nodes. We implement the three approaches within the Damaris I/O middleware and perform extensive experiments with one of the target HPC applications of the Blue Waters sustained-petaflop/s supercomputer project: the CM1 atmospheric model. Our experimental results obtained on the French Grid'5000 platform highlight the differences between these three approaches and illustrate in which way various configurations of the application and of the system can impact performance and energy consumption.

Categories and Subject Descriptors

C.4 [**Performance of System**]: Performance
; D.4.8 [**Operating System**]: Performance—*Measurement*

General Terms

Measurement, Performance

Keywords

Exascale, I/O, Energy, Time partitioning, Dedicated Cores, Dedicated Nodes, Damaris

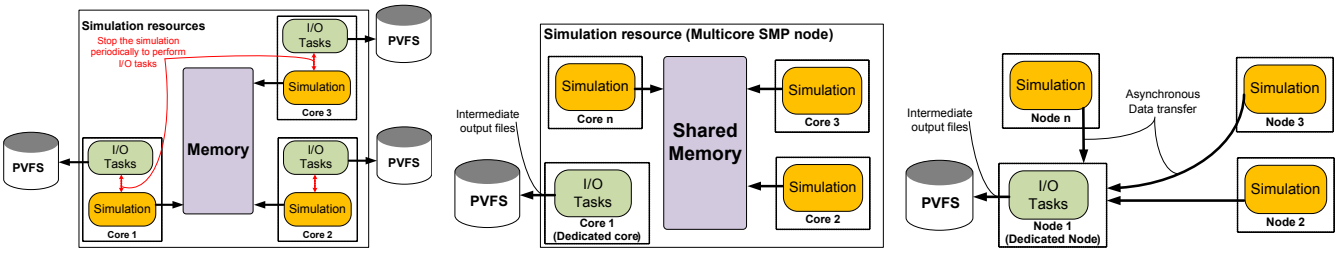
1. INTRODUCTION

Power has become an essential issue in the design of modern computing systems. Power bills become a substantial part of the total cost of ownership (TCO) of supercomputers: a typical supercomputer of thousands of cores consumes several megawatt of power [12] which in turn presents almost 40% of the total cost [1]. Performance has long been the major focus of the HPC community, today's supercomputers are therefore equipped with millions of processing cores that run parallel programs and consume a large amount of energy. For example, Tianhe-2, No.1 in the top 500 supercomputers list, is a 3,120,000 processor supercomputer with a Linpack performance of 33.8 petaflop/s,[1] but with a 17 megawatt of power consumption.[2] This amount of energy will even increase as we reach the era of exascale systems.

Scientific simulations running on these machines have been traditionally designed to write their data in the form of many files stored in a parallel file system. Yet, the increasing computational power of new supercomputers largely overcomes the performance of storage systems, which substantially impacts the performance of these simulations. The traditional approach of periodically checkpointing the simulation's data into files and processing it offline to retrieve scientific results does not scale anymore. Therefore, a number of new approaches to large-scale data management have been proposed that make use of dedicated resources such as dedicated cores [6], dedicated nodes [14], accelerators (such as GPUs) or SSDs to process data as it is being generated by the simulation. As energy is becoming an increasingly important concern for large-scale systems, a major challenge of exascale computing is how to sustain a high performance per watt ratio. While most studies have been focusing on profiling and characterizing power usage in supercomputers, modeling and exploring data-related energy/performance tradeoffs [7], and exploiting dynamic voltage frequency scaling (DVFS) techniques to reduce power consumption [9], there is comparatively little work on investigating the impact of I/O management on energy consumption (i.e., how much energy a supercomputer consumes while running a scientific simulation when adopting different data management approaches).

We implement three different approaches to I/O management (that is, (1) periodically stopping the simulation to write data, (2) using dedicated I/O cores and (3) using dedicated I/O nodes) within the Damaris middleware [6] and perform extensive experiments that involve one of the target HPC applications for the Blue Waters petascale supercomputer project –the CM1 atmospheric model– on Grid'5000 [2]. Our experimental results bring out the differences between these three approaches and show that they are only sub-optimal for different configurations of the application and of the system: the energy consumption under

[1] http://www.top500.org
[2] http://www.green500.org



(a) Time partitioning approach (b) Dedicated core(s) approach (c) Dedicated node(s) approach

Figure 1: Approaches to I/O management in HPC applications

different approaches significantly varies with the frequency of output and the system's architecture.

Specifically, we show evidence that a shorter execution time doesn't always imply better energy consumption, especially when comparing dedicated cores against dedicated nodes. Furthermore, there is a linearity between the output frequency and energy consumption regardless of which data management approach is employed.

The primary contributions of this paper are as follows:

1. It experimentally demonstrates that HPC applications experience variations in performance and power consumption under different I/O management approaches and provides a deep analysis to explain this variation and its cause.

2. It illustrates in practice how the different system's architecture influence the execution of HPC applications and how they shape the energy consumption of the entire system.

The proposed work aims at providing a clearer understanding of the interplay between current data management approaches in exascale supercomputers and energy consumption, and offers a preliminary insight into designing energy-efficient approaches for exascale supercomputers.

Paper Organization. The rest of this paper is organized as follows: Section 2 briefly presents the three I/O management approaches considered in our study: time partitioning, dedicated cores and dedicated nodes. Section 3 describes an overview of our methodology, followed by the experimental results in Section 4. Finally, we conclude the paper and propose our future work in Section 5.

2. APPROACHES TO I/O MANAGEMENT IN HPC APPLICATIONS

As we approach the exascale era in HPC, mismatch between computation and I/O performance have become a limiting factor for the performance and scalability of HPC applications. The traditional approach to I/O requires the simulation to periodically stop and output data for checkpointing, visualization or analysis purpose [8]. This results in a large number of processes competing for the access to the file system. Therefore, alternative approaches for performing I/O have been investigated recently by the HPC community [6, 14]. These approaches perform I/O while the simulation keeps running, in an asynchronous manner, taking advantage of dedicated resources such as cores or nodes. These approaches eliminate the additional machine time needed for performing I/O by overlapping I/O with the simulation, but also require more resources. In this section, we present three different data management approaches which can be used for performing I/O tasks in HPC applications.

2.1 Time Partitioning

Time partitioning is the traditional approach to I/O in HPC applications. In this approach, I/O operations are performed in a synchronous manner: the simulation stops periodically, performs I/O operations and continues its computation where it left it off. This I/O activity is presented in Figure 1(a). While this approach takes advantage of direct access to simulation data, it impacts the simulation run time since it stops the simulation periodically to perform I/O. Besides, the time partitioning approach also increases the run time variability [13, 10, 5]. In the following sections, we will present other approaches which use dedicated resources for performing I/O to minimize the I/O impact on the simulation.

2.2 Dedicated Cores

This approach takes advantage of the hierarchical nature of HPC systems and in particular the increasing use of multicore nodes, where many cores share a common memory. It dedicates one or multiple cores in each SMP nodes for performing I/O asynchronously while the simulation runs [6]. Since dedicated cores share the same resources as the cores which run the simulation, they can leverage the already available simulation data in the shared memory. By assigning asynchronous I/O tasks to dedicated cores, one can hide the I/O costs. Figure 1(b) presents this approach. In this approach, if I/O tasks do not scale with the simulation, they can start to impact the overall performance. What is more, memory space is limited and there is a need for buffer management coordination between simulation and dedicated cores.

2.3 Dedicated Nodes

With this approach, separate nodes are used for performing I/O operations while the simulation is running [14]. Data is transmitted from computing nodes to dedicated nodes through the network. Once the data transfer is completed, dedicated nodes interact with the file system to write the output files. This approach is demonstrated in Figure 1(c).

One of the challenges with this approach is the memory management on dedicated nodes. As we come closer to exascale, memory per core tends to shrink. When applying a dedicated nodes approach, one should be aware of the limited space in the dedicated nodes. While computing nodes send simulation data to dedicated nodes, they are not aware of the available space. Therefore, dedicated nodes should

handle the incoming transfers according to their memory availability. The ratio between dedicated nodes and computing nodes is thus an important factor. A mismatch between capacity of dedicated nodes and computing nodes can lead to inefficiency. Another drawback of this approach is the requirement for data transfer through the network, less efficient than a transfer through shared memory. However, asynchronous data transfers can have less impact on the running simulation if one can overlap the cost of the data movement with the simulation.

2.4 Discussion

These three approaches have inherent advantages and disadvantages. While the traditional time partitioning approach can still be viable for some non-data-intensive applications, scientists would not be able to tolerate its large impact on the simulation running time for applications that output massive amounts of data. On the other hand, the other two approaches minimize this impact by performing I/O in an asynchronous manner while they differ in the location where the actual I/O tasks are actually performed. Dedicated cores approach use the same resources as the simulation and thus may impact the simulation more than an approach based on dedicated nodes that opts for the cost of data network transfers rather than taking the risk of impacting the simulation.

3. METHODOLOGY OVERVIEW

We conducted a series of experiments in order to assess the impact of the three data management approaches on both energy consumption and application performance. We further describe the experimental environment: the platform, the HPC application, the tools used, and the deployment setup.

3.1 Platform

The experiments were carried out on the Grid'5000 [2] testbed, more specifically we employed nodes belonging to its Nancy and Rennes sites:
On the Nancy site: each node is a 4-core Intel 2.53 GHz CPU with 16 GB of RAM. Intra-cluster communication is done through a 1G Ethernet network.
On the Rennes site: each node is a 24-core AMD 1.7 GHz CPU with 48 GB of RAM. The nodes communicate through a 1G Ethernet network.
40 nodes of the Nancy site (the same on Rennes site) are equipped with power monitoring hardware consisting of 2 Power Distribution Units (PDUs) (4 PDUs on Rennes site), each hosting 20 outlets (10 outlets per PDU on Rennes site). Since each node is mapped to a specific outlet, we are able to acquire coarse and fine-grained power monitoring information using the Simple Network Management Protocol (SNMP). Grid'5000 allows us to create an isolated environment in order to have full control over the experiments and the obtained results.

3.2 Application use case

For our analysis, we chose one of the target HPC applications of the Blue Waters petascale supercomputer project [11]: the CM1 atmospheric model [3]. CM1 is used for atmospheric research to model small-scale atmosphere phenomena such as tornadoes. It alternates computation phases and I/O phases. The simulated domain is a fixed 3D array. The number of points along the x, y and z axes is given by the parameters n_x, n_y and n_z. Each point in this domain is characterized by a set of variables such as local *temperature* or *wind speed*. Parallelization is done using MPI, by splitting the 3D array along a 2D grid. Each process simulates a $n_{sx} \times n_{sy} \times n_z$ point subdomain. The I/O phase uses HDF5 to write one file per process at every output.

3.3 Metrics

In addition to the energy monitoring tools, described in section 3.1, we gathered information about the CPU usage, which is a crucial metric for many applications. These information are gathered using Dstat. Therefore, minimizing the effects of the monitoring system on our resource measurements. In each run we monitor the CPU usage for each node per second and we also measure the energy consumption with a resolution of one second.

3.4 Implementation of Data management approaches in Damaris

Damaris [6] is a middleware for I/O and data management (including data processing and in situ visualization). Initially developed to provide data management capabilities through dedicated I/O cores on multicore nodes, it has later been extended to support the more classical time partitioning approach. We further extended it to support dedicated nodes. Communications between the nodes running the simulation and dedicated nodes are performed using MPI (blocking *send* from clients, asynchronous *receive* posted by dedicated nodes to react to data management requests from clients). Although this preliminary implementation is potentially less efficient than other approaches using dedicated nodes along with RDMA technologies [14], it allows us to compare the three I/O approaches on a fair ground of a common implementation. In addition to dedicated nodes, we implemented in Damaris the capability to use more than one dedicated core per node.

3.5 Experimental deployment

We use 14 nodes (56 cores) to run CM1 on the Nancy site, 2 additional nodes are used by a PVFS [4] file system. These PVFS nodes are accessed by computation nodes through a 20G Infiniband network. On the Rennes site, we run CM1 on 12 nodes (288 cores) and 2 nodes are used by PVFS, also accessed from compute nodes through a 20G InfiniBand network. In addition, in both sites, we configure CM1 to complete 2400 time steps and vary the frequency of output, using 5, 10 or 20 time steps between outputs. Damaris is configured to run with CM1 in five different scenarios which cover the three I/O approaches: time partitioning (*i.e.*, TP), dedicating core(s) employing one core per node (*i.e.*, DC(ONE)) and employing two cores per node (*i.e.*, DC(TWO)), and dedicating node(s) employing one node (*i.e.*, DN(ONE)) and employing two nodes (*i.e.*, DN(TWO)).

In our first experiment, the I/O output frequency is performed at every 5 time steps. In order to understand the impact of the output (*i.e.*, checkpoints) frequency in the second set of experiments (Section 4.1), we vary the output frequency, using 5, 10 and 20 time steps between outputs. Finally, to illustrate the impacts of system's architectures, we complement the first set of experiments (*i.e.*, Nancy site) by running CM1 simulation on Rennes site. Here, we fixed the number of I/O output frequency to every 5 time steps.

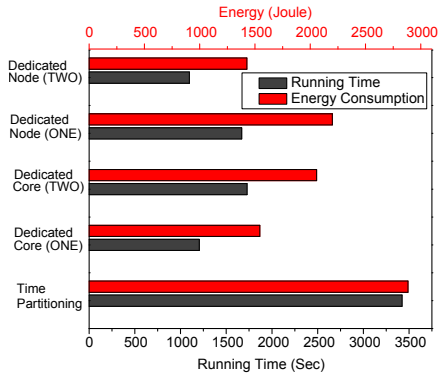


Figure 2: Energy vs completion time

4. KEY RESULTS

In this section, we provide an analysis of the experimental results we obtained. Our goal is to study the impact of various I/O approaches on the energy and completion time of CM1.

In terms of performance, Figure 2 shows that the time partitioning approach largely impacts the simulation. This behavior results from the fact that it performs I/O from all processes and by stopping the simulation periodically. On the other hand, approaches that perform I/O asynchronously and from a reduced number of writers obtained a much better run time. Among these approaches, dedicating two nodes outperforms the other configurations. We can argue that the approach based on dedicated cores can have a larger impact on the simulation because there are only four cores per node, i.e., dedicating even one core already removes 25% of the computation resources that could have been used by the simulation. Additionally, it is worthwhile to note the importance of the ratio between dedicated and compute nodes in the dedicated nodes approach. As we mentioned before, mismatch between capacity of dedicated and computing nodes can result in inefficient simulation runs. Dedicating only one node for thirteen compute nodes appeared not to be an optimal choice under high I/O frequency where large outputs are produced every 5 time steps.

In terms of energy consumption, Figure 2 shows a strong correlation between completion time and energy consumption. However, we can also see that although dedicating one node (i.e., DN(ONE)) finishes the simulation earlier than with two dedicated cores (i.e., DC(TWO)), the former consumes more energy than the latter. This implies that short execution times do not *always* lead to less energy consumption. Table 1 confirms our observation since in the dedicated nodes approach all cores only perform computation tasks whereas in the dedicated cores approach we employed one or two cores to perform I/O, which leads to less CPU usage on average. Additionally, we can see the decrease of average CPU usage when we dedicate two cores instead of one. Since the average CPU utilization affects the completion time, we observe longer execution times for lower CPU utilizations. However, this can lead to less energy consumption as we described for the DC(TWO) approach, at the expense of performance compared to DN(ONE). This energy/performance trade-off can be favored according to the users' priority and, of course, largely depends on the application considered.

4.1 Impact of the output frequency

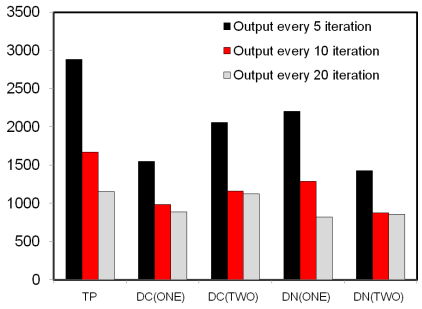


Figure 3: Measured energy Consumption under different approaches to I/O management in HPC applications with different output frequencies: *different number of iterations between each output*

Table 1: Average CPU Usage (percentage) for the different I/O management approaches

Iterations between outputs	TP	DC (ONE)	DC (TWO)	DN (ONE)	DN (TWO)
5	27.6	71.7	55.9	82.5	80.8
10	40	72	62	84	83
20	55.4	84.3	79.3	84.9	84.8

Figure 3 shows the completion time and resulting energy consumption of the different data management approaches under different output frequencies (i.e., different number of iterations between each output). These results show that there is a linear relation between the output frequency and the energy consumption regardless of which data management approach is employed. Decreasing the output frequency leads to lower energy consumption which stems from shorter completion times. For example, by dividing the output frequency by half the resulting energy consumption is reduced by almost 60% for each configuration.

4.2 Impact of the system architecture

Figure 4 shows the energy consumption of each I/O approach and Figure 5 depicts the throughput and average power usage during the simulation using different system architectures. The results indicate that the comparative behavior of the different approaches with respect to performance and energy efficiency depends on the system on which they run. With a larger number of cores per node, dedicating one core outperforms the other configurations while the dedicated nodes approach was the optimal choice with a smaller number of cores per node. The different behaviors in dedicated cores approaches stem from reducing the number of cores that could otherwise be used by the simulation. Since CM1 is computation-intensive, dedicating cores for performing I/O tasks reduces the computing resources dramatically in an architecture with a small number of cores per node. However, with a larger number of cores per node, dedicating cores removes only a small amount of computing resources (e.g., 4% when dedicating one core out of 24 cores) and therefore its impact on the simulation stays minimal. Additionally, the power usage of the dedicated cores approaches changes drastically with one or two cores. Dedicating one more core per node reduces the power usage by 10% with 4 cores per node, while in the latter system this reduction is 1%. This observation also complies with our reasoning that the impact of dedicating cores on a system with a larger number of cores per node is minimal compared to a system with a smaller number of cores per node.

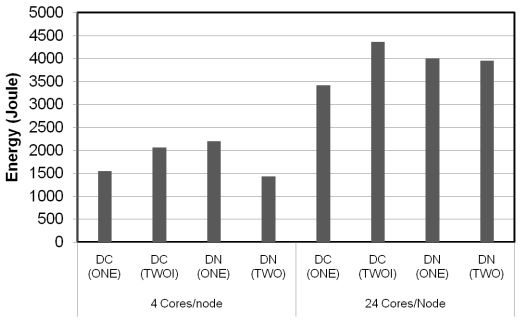


Figure 4: Energy consumption under different I/O management approaches and system architectures

These different behaviors highlight the need for an energy efficient framework which can select the best I/O approach in terms of energy efficiency by considering different parameters such as the system's architecture or the output size and frequency.

5. DISCUSSION AND FUTURE WORK

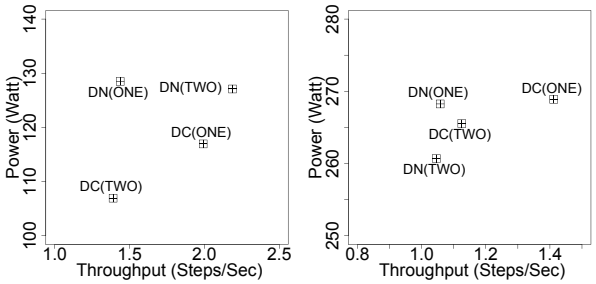
Power consumption has started to severely constrain the design and the way HPC systems are operated. As HPC systems and HPC application's data size rise, their power efficiency calls for empirical evaluations and technical innovations. In this study, we investigated the performance and energy efficiency of three I/O approaches in HPC systems: time partitioning, dedicated cores, and dedicated nodes. We implemented the three approaches within the Damaris I/O middleware and performed extensive experiments using the CM1 atmospheric model on the French Grid'5000 platform.

Our detailed study reveals a significant variation in the performance of CM1 and the energy consumption of the HPC system. Three factors at least contribute to such variations. First, the adopted I/O approach. Second, the output frequency (number of iterations between each output). Third, the architecture of the system on which we run the HPC application. As a future work, we plan to investigate an energy model based on the I/O management approach, HPC application's characteristics and the HPC system's architecture and use this model to pre-select an optimal I/O management approach prior to running the simulation, including the proper number of dedicated cores or nodes. In addition, within each I/O management approach, we plan to discuss the tradeoffs between reducing the CPU frequency of the resources that are dedicated to I/O (to reduce the overall energy consumption) and fully utilizing these dedicated resources (e.g., performing compression) and therefore reducing the I/O time and storage space. As a longer-term agenda, we are going to investigate the feasibility of building a hybrid I/O approach that not only incorporates all the aforementioned approaches but also adaptively selects the optimal approach at run time.

6. ACKNOWLEDGMENTS

This work is supported by the ANR MapReduce grant (ANR-10-SEGI-001), the Héméra INRIA Large Wingspan-Project (see http://www.grid5000.fr/mediawiki/index.php/Hemera) and the Data@Exascale Associate Team between the KerData team from INRIA Rennes - Bretagne Atlantique, Argonne National Laboratory (ANL) and the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana Champaign (UIUC).

The experiments presented in this paper were carried out using the Grid'5000/ALADDIN-G5K experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see `http://www.grid5000.fr/` for details).



(a) 4 Cores/Node (b) 24 Cores/Node

Figure 5: Average Power consumption and Throughput under different I/O management approaches and system architectures

7. REFERENCES

[1] James Hamilton, Cost of Power in Large-Scale Data Centers. `http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx`, November 2008.

[2] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, et al. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, 20(4):481, 2006.

[3] G. H. Bryan and J. M. Fritsch. A benchmark simulation for moist nonhydrostatic numerical models. *Monthly Weather Review*, 130(12):2917–2928, 2002.

[4] P. H. Carns, W. B. Ligon, III, R. B. Ross, and R. Thakur. PVFS: a parallel file system for linux clusters. In *Proceedings of the 4th annual Linux Showcase & Conference - Volume 4*, Berkeley, CA, USA, 2000. USENIX Association.

[5] M. Dorier, G. Antoniu, F. Cappello, M. Snir, and L. Orf. Damaris: Leveraging Multicore Parallelism to Mask I/O Jitter. Research report RR-7706, INRIA, Dec 2011.

[6] M. Dorier, G. Antoniu, F. Cappello, M. Snir, and L. Orf. Damaris: How to Efficiently Leverage Multicore Parallelism to Achieve Scalable, Jitter-free I/O. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*, pages 155 –163, Sept. 2012.

[7] M. Gamell, I. Rodero, M. Parashar, J. C. Bennett, H. Kolla, J. Chen, P.-T. Bremer, A. G. Landge, A. Gyulassy, P. McCormick, S. Pakin, V. Pascucci, and S. Klasky. Exploring power behaviors and trade-offs of in-situ data analytics. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 77:1–77:12, New York, NY, USA, 2013. ACM.

[8] A. Hoisie and V. Getov. Extreme-Scale Computing - Where 'Just More of the Same' Does Not Work. *Computer*, 42(11):24–26, Nov. 2009.

[9] J. H. Laros, III, K. T. Pedretti, S. M. Kelly, W. Shu, and C. T. Vaughan. Energy based performance tuning for large scale high performance computing systems. In *Proceedings of the 2012 Symposium on High Performance Computing*, HPC '12, pages 6:1–6:10, San Diego, CA, USA, 2012. Society for Computer Simulation International.

[10] J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, K. Schwan, and M. Wolf. Managing Variability in the IO Performance of Petascale Storage Systems. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, pages 1–12, Washington, DC, USA, 2010. IEEE Computer Society.

[11] NCSA. BlueWaters project, `http://www.ncsa.illinois.edu/BlueWaters/`.

[12] C. Patel, R. Sharma, C. Bash, and S. Graupner. Energy aware grid: Global workload placement based on energy efficiency. Hpl technical report, HPL Technical Report, Nov. 2002.

[13] D. Skinner and W. Kramer. Understanding the Causes of Performance Variability in HPC Workloads. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 137 – 149, Oct. 2005.

[14] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, Q. Liu, S. Klasky, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf. PreDatA - Preparatory Data Analytics on Peta-Scale Machines. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12, April 2010.