SAND2013-6761C

Exploring Power Behaviors and Trade-offs of In-situ Data Analytics

Marc Gamell, Ivan Rodero, Manish Parashar
Rutgers University

Janine C. Bennett, Hemanth Kolla, Jacqueline Chen
Sandia National Laboratories

Peer-Timo Bremer
Lawrence Livermore National Laboratory

Aaditya G. Landge, Attila Gyulassy
University of Utah

Patrick McCormick, Scott Pakin
Los Alamos National Laboratory

Valerio Pascucci
University of Utah & Pacific Northwest National Laboratory

Scott Klasky
Oak Ridge National Laboratory

ABSTRACT

As scientific applications target exascale, challenges related to data and energy are becoming dominating concerns. For example, coupled simulation workflows are increasingly adopting in-situ data processing and analysis techniques to address costs and overheads due to data movement and I/O. However it is also critical to understand these overheads and associated trade-offs from an energy perspective. The goal of this paper is exploring data-related energy/performance trade-offs for end-to-end simulation workflows running at scale on current high-end computing systems. Specifically, this paper presents: (1) an analysis of the data-related behaviors of a combustion simulation workflow with an in-situ data analytics pipeline, running on the Titan system at ORNL; (2) a power model based on system power and data exchange patterns, which is empirically validated; and (3) the use of the model to characterize the energy behavior of the workflow and to explore energy/performance trade-offs on current as well as emerging systems.

1. INTRODUCTION

The path to exascale computing poses a number of significant challenges as researchers and potential exascale vendors attempt to deliver a hundred times performance improvement relative to today's fastest supercomputers. While simulations running at these extreme scales have the potential for achieving unprecedented levels of accuracy and providing dramatic insights into complex phenomena, they are also presenting new challenges. Key among these are the challenges related to the large data volumes and data rates, and the costs (power and latencies) associated with transporting, managing, and processing this data.

For example, the anticipated changes in system architectures and the anticipated tight power budgets are motivating a shift in scientific simulation workflows, away from a post-processing paradigm towards a framework in which simulation output is analyzed concurrently as it is generated. A variety of potential workflow options are being explored, including in-situ processing, in which data is analyzed on compute resources shared with the simulation, in-transit processing, in which secondary compute resources on the system are leveraged, and hybrid in-situ/in-transit processing that combines the approaches. Recent efforts [8] have explored trade-offs between these workflow options from a latency and performance perspective for a variety of algorithms on current architectures.

However, as we look to the future, it is imperative to consider energy-performance behaviors and trade-offs, determine how to optimize codes for future processor designs, and provide computer architects with a clear understanding of the characteristics, complexities, and power needs of the applications that will be deployed on these systems.

The goal of this paper is exploring data-related energy/performance trade-offs at extreme scales. Specifically, we analyze the behavior of a combustion simulation workflow with an in-situ data analytics pipeline, running on a current high-end computing platform (i.e., the Titan Cray-XK7 systems at Oak Ridge National Laboratory) and use machine-independent code characteristics (e.g., computational profiles, data access and exchange patterns, messaging profiles, etc.) to develop a power model, which is then empirically validated using an instrumented platform. We use this model to explore energy/performance trade-offs on current systems, to help answer system design questions, and to analyze the power requirements for emerging architectures. Our key contributions include the following:

SC13 November 17-21, 2013, Denver, CO, USA

http://dx.doi.org/10.1145/2503210.2503303.

- a power model based on machine-independent algorithm characteristics validated through empirical studies,
- a case study of the impacts of system architecture, algorithm design choices, and deployment options (e.g., node mapping) on data exchange patterns and overall energy-performance profiles, and
- a discussion of how our methodology can be extended to help answer questions related to the design of future architectures.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 presents background material including the specific use cases, as well as the scientific rationale for the paper. Section 4 presents our power model and Section 5 presents a validation of this model. Section 6 uses our model to investigate power/performance behaviors at scale, as explore power-related co-design issues. Section 7 concludes the paper and outlines directions for future work.

2. RELATED WORK

In-situ Analysis. Early in-situ analysis workflows focus largely on visualization for monitoring purposes [28, 47, 55, 62, 64, 69]. More recent efforts allow for the coupling of simulation codes with popular visualization and analysis toolkits, such as VisIt [11] and ParaView [25], exposing a broader suite of analytics tools to simulation scientists. As we look ahead to exascale, performance design issues and trade-offs associated with these workflows are becoming increasingly important, leading to the investigation of latency and run-time performance trade-offs between in-situ and in-transit [2,3,20–22,65,70] workflows as discussed by Bennett et al. [8]. In this paper we investigate the roles of system architecture, algorithm design choices, and deployment options on overall energy performance profiles for a particular system, and present a methodology that can be extended to answer co-design questions for emerging architectures.

Topological Analysis. Topology-based techniques have proven useful in the analysis of a wide variety of simulation data due to their efficient representation of the feature space of scalar functions [7, 12, 30, 40, 49, 50]. One class of algorithms, which includes Reeb graphs [59] and their variants, contour trees and merge trees [17], encode the level set behavior of a function. These structures are ideally suited for encoding threshold based feature definitions and have been used successfully in a number of large scale science applications including the analysis of extinction regions in non-premixed turbulent combustions [49], a study of lean pre-mixed hydrogen flames [12], and detection of bubbles in turbulent mixing [40]. Current techniques for computing these structures for large-scale data fall into two categories: 1) streaming out-of-core computation [14,57] and 2) divide-and-conquer parallel approaches [53,56] that rely on successive k-nary merging of regions of the domain.

Characterization Tools. The most common way to identify how an application is utilizing the underlying hardware is to monitor hardware performance counters. The PAPI library [15] abstracts hardware-specific counters into a consistent set of named counters. Unfortunately, the set of available counters are predefined and typically only a certain number may be active at any one time. Furthermore, the details of the behaviors and features can differ across processor generations and vendors and can also be inconsistent across runs of an application [66]. Static analysis using source-to-source translation tools such as ROSE [58] have the disadvantage that a substantial amount of application behavior (e.g., iterations needed for convergence) is not known at compile time. In addition, advanced compiler-optimization techniques can significantly impact code characteristics, and the details of code-generation for a specific target architecture are typically difficult to capture. Binary-code modification – patching an executable to include instrumentation code – as embodied in tools such as DynInst [16] and Pin [33, 46] is too far removed from application source code; it can be difficult to draw connections between the application behavior as understood by the developer and the architecture-specific behavior that such tools report. The approach we take in this work is to instrument code at compile time (using a view of the application close to the developer's) but gather data at run time (using run-time knowledge of what actually gets executed). The implementation is via a custom LLVM [1, 41] compiler pass that operates on the compiler's intermediate representation of the application. This enables us to maintain both programming language and hardware-architecture independence, while simultaneously providing a wealth of information that characterizes the application in a manner meaningful for energy analysis.

Power and Energy Modeling. There exists a large body of literature on power and energy modeling at different levels, ranging from the microprocessor to entire systems. Our models are built using a traditional coarse-grained system level energy/power formulation [10, 24, 36, 61, 63]. These models are simple, fast, have low overhead, and are accurate enough to characterize energy at the system level. Power models often rely on performance counters and power is usually correlated with these events or estimated using accurate linear power models [5, 19, 34, 35, 42, 44, 45, 48, 52]. In our work, we estimate subsystem activity based on application-centric metrics instead of hardware counters. Detailed subsystem power models have also considered the use of local events to represent power [29,31,32,37,43,60] as well as dynamic processor frequency scaling approaches [26, 38] and thermal considerations [6, 9]. Our work focuses on system-level power. Note that we do not use fine-grained power metering data in our extrapolations to Titan as such data is not typically available on leadership class computing facilities. However, our approach does provide us with a framework to explore relative behaviors and costs and associated tradeoffs. Our recent work has also focused on characterizing and quantifying energy/power behaviors of data-centric scientific workflows [27], but this is not at scale yet. Existing work by Shalf et al. [23] has also explored energy efficiency for extreme-scale scientific applications and addressed software/architecture co-design by comparing different architectural alternatives such as multi-cores, GPUs and many-cores [39]. However, there are still challenges on this front and, to the best of our knowledge, the performance/energy tradeoffs and co-design aspects that we address in this paper (i.e., combining software, runtime and architectural issues) have not been addressed at petascale (i.e., on Titan) and beyond.

3. BACKGROUND

This section describes the specific use case chosen for our

study, the different degrees of freedom inherent in the particular algorithm, as well as the scientific rationale behind it.

Workflow. In this paper we consider an in-situ analysis workflow integrated with S3D [18], 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. S3D is used to glean insights into fundamental turbulence-chemistry interactions and therefore, for production simulations, the time steps taken to advance the solution are smaller than the smallest time scales. In this paper we consider two different S3D simulations. The first is a turbulent auto-ignitive mixture of di-methyl-ether and air under typical homogeneous charge compression ignition (HCCI) conditions [4]. This simulation is aimed at understanding the ignition characteristics of typical bio-fuels for automotive applications and has a domain size of over 175 million grid points. The second simulation describes a lifted ethylene jet flame [68], involved in a reduced chemical mechanism for ethylene-air combustion, with a domain size of 1.3 billion grid points.

Of interest to scientists is the behavior of small, intermittent features within the simulations. Understanding the structures' shape and size distributions and tracking their interactions over time provides key insights into complex turbulence-chemistry interactions such as autoignition and extinction. Examples of such features include structures defined by the the scalar dissipation rate in the lifted ethylene jet simulation and structures defined by the reaction rate of OH in the HCCI simulation.

Topological analysis tools, such as merge trees have been used to define these features of interest, provide statistical summaries of their characteristics, and track them over time [7,13,14,51,67]. The advantages of using such topological analysis tools are twofold. First, features of interest are typically defined in terms a specific input parameter value (e.g., threshold or isovalue), and topological techniques provide a multi-resolution representation that captures these feature definitions over a *range* of input parameter values. Moreover, the results are stored in a compact fashion, allowing for dramatic data reductions while still maintaining complete flexibility in representing the features of interest.

Post-processing analysis workflows are not adequate as S3D currently saves only approximately every 500th timestep of the simulation to disk to mitigate I/O costs. However, due to their time scales, a frequency at least twice this is required to track such features of interest, motivating a shift to in-situ topological techniques. The in-situ merge tree analysis code in this work comprises three main stages: 1) a local compute stage in which families of features are identified locally on each processor and a local tree is generated to capture feature interactions over the range of input parameter values; 2) a merge stage in which the local trees from each processor are joined in a k-nary merge pattern to capture global relationships; and 3) a correction phase, in which the local features at each processor are updated to reflect the global relationships uncovered during the various merge stages. Figure 1 shows a corresponding flow graph for four local processors using a binary merge.

Here we explore two degrees of freedom in setting up the algorithm and study their impacts on the power profile of the overall in-situ workflow. First, each node in the diagram of Figure 1 represents an independent compute kernel which could be placed on an arbitrary MPI rank/compute core. In practice, both the local compute and the local corrections need access to the original data. To avoid excessive data movements these are therefore typically co-located with their corresponding sub-domain. However, there are exceptions: For example, since node-internal memory access is relatively fast and cheap one could gather the data of all cores onto fewer or even a single core for processing. This reduces the overall core count of the analysis code, which enables it to run in a more scalable regime. In this case the dataflow would gain an initial setup phase to collect the local data. However, the merge computation would be less dependent on data locality since all messages are typically small. There exist different strategies for the placement of the various compute kernels. For example, one can ensure that either the early merge phases or the late merge phases happen among cores on the same node, which would prevent MPI traffic in the corresponding portion of the algorithm.

The second major aspect of the algorithm's performance—and a controllable input parameter—is the degree of fan-in during the merge stage. Larger fan-ins produce a more shallow merging hierarchy with fewer dependencies and shorter chains of messages. However, larger fan-ins also reduce the number of active cores, which more quickly results in an unbalanced load, potentially introducing performance problems. Moreover, the fan-in can have effects on the overall amount of on- vs. off-node communication. For example, choosing a fan-in larger than the available core count on each node forces significantly more messages to travel on the network which slows down the algorithm and increases its power consumption. Below, we empirically investigate the impact these factors can have on the amount of on- vs. off-node communication required as well as their impact on total overall power efficiency.

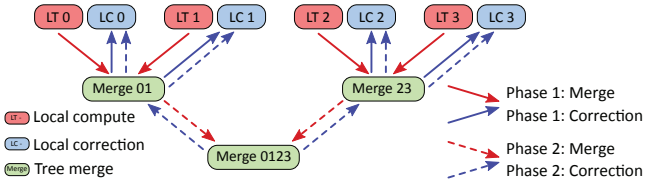


Figure 1: High level flow diagram of the parallel merge tree algorithm. First, local trees are computed (red) and subsequently merged hierarchically (green). Each merge results in a number of corrections which are passed upward to the corresponding leaves and used to iteratively integrate global corrections into the local trees (blue).

Profiling and Characterization. As detailed in Section 4, our power model accounts for both on-node and off-node communication characteristics of the analysis code. One of the primary goals is to develop a methodology that is not limited to the performance characteristics of a single machine, but rather is flexible enough to map to potential future architectures. Because envisioned exascale architectures are not yet available, we cannot use hardware performance counters or binary instrumentation techniques. Furthermore, architecture simulators are prohibitively slow for

characterizing complete applications at scale. To address this challenge, we use compiler-based application analysis using Byfl [54] to identify key data-centric operations in an architecture-independent way. These include counts of the total number of memory accesses as well as the total number of operations performed. To capture communication patterns, we have instrumented our code to generate trace information that encodes the size as well as the rank IDs of the source and destination of all messages sent during application execution.

4. MODEL FORMULATION

This section describes our analytical model inspired by traditional coarse-grain system level energy/power formulations typically based on performance counters or linear models. In contrast, we model energy/power based on architectural characteristics and architecture-independent information provided by Byfl and MPI messages (see Section 3) instead of architecture-dependent performance counters. Other analysis methods (e.g., using data staged on disk) can also be studied using our framework, by modeling the involved subsystems behavior and power cost. Note that the goal of this formulation is to use global, coarse-grained, machine-independent information so we can extrapolate performance/energy behavior and trade-offs at scale (see Section 6).

Energy modeling. Since system-level information provided by Byfl is global for the entire application execution (i.e., can be averaged) and data communication is available for each MPI message, we consider the system and data communication energy consumption separately:

$$E = E_{system} + E_{comm}. \tag{1}$$

Energy results from integrating power over time. We consider both static (or idle) and dynamic power, i.e.,

$$E_{system} = T \cdot (P^{static}_{system} + P^{dynamic}_{system}). \tag{2}$$

Static system power can be decomposed into the power of each contributing subsystem at idle state, which are processor (cpu), memory (mem), network interface (nic) and other components such as control circuitry, power distribution, etc. ($misc$). We consider P^{idle}_{cpu}, P^{idle}_{mem}, P^{idle}_{nic} and P^{idle}_{misc} as architecture constants that can be estimated using hardware specifications or measured empirically. Thus,

$$P^{static}_{system} = (P^{idle}_{cpu} + P^{idle}_{mem} + P^{idle}_{nic} + P^{idle}_{misc}). \tag{3}$$

Dynamic power only includes processor and memory contributions because the network interface's dynamic power is accounted for in E_{comm}. Dynamic power can be determined using dynamic power dissipation models. For example, processor dynamic power can be formulated from its capacitance (C), an activity factor or switching activity (α), and the operational voltage (V) and frequency (f), i.e.,

$$P^{active}_{cpu} \sim C \cdot V^2 \cdot \alpha \cdot f. \tag{4}$$

Analogous to α in Equation 4, we approximate dynamic processor and memory power dissipation using an activity factor extracted from the Byfl data:

$$P^{dynamic}_{system} = \alpha_{cpu} \cdot P^{active}_{cpu} + \alpha_{mem} \cdot P^{active}_{mem}. \tag{5}$$

These activity factors are computed from the number of operations per second ($mips$) and number of memory accesses per second (mem_{bw}) as well as a normalization factor that represents the maximum capacity.

$$\alpha_{cpu} = \frac{mips}{MAX_{mips}} \quad ; \quad \alpha_{mem} = \frac{mem_{bw}}{MAX_{mem_{bw}}}. \tag{6}$$

The energy consumption from data communication depends on the number of MPI messages (M). Each MPI message is composed of source MPI rank (src), destination MPI rank ($dest$) and data size of the message. Thus,

$$M = (\{src_1, dest_1, data_1\}, \ldots, \{src_m, dest_m, data_m\}) \tag{7}$$

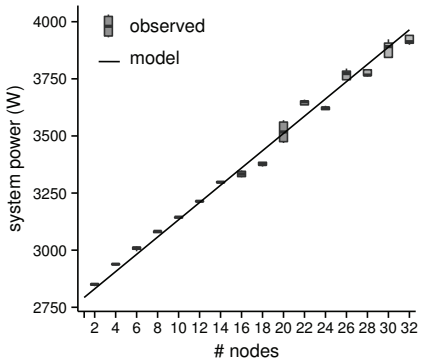
E_{comm} is defined in Equation 8 where $smp(i) = smp(j)$ indicates that the MPI ranks i and j are mapped to cores that share memory. BW_{net} and BW_{mem} are the network and memory bandwidth, respectively. We do not model network/memory contention since the driving application does not exhibit any contention. $P_{transfer}$ depends on the network characteristics. For example, $P_{transfer}$ depends mostly of P^{active}_{nic} in an InfiniBand network while it depends on the number of hops that each message requires to reach the destination MPI rank in a Gemini network (see Section 6.1). Furthermore, we model the energy for data communication within shared memory as processor and memory activity during the communication, as an approximation.

$$E_{comm} = \begin{cases} \sum_{i=1}^{M} \frac{data_i}{BW_{net}} \cdot P_{transfer} & \text{if } smp(src_i) \neq smp(dest_i) \\ \sum_{i=1}^{M} \frac{data_i}{BW_{mem}} \cdot (P^{active}_{cpu} + P^{active}_{mem}) & \text{if } smp(src_i) = smp(dest_i) \end{cases} \tag{8}$$

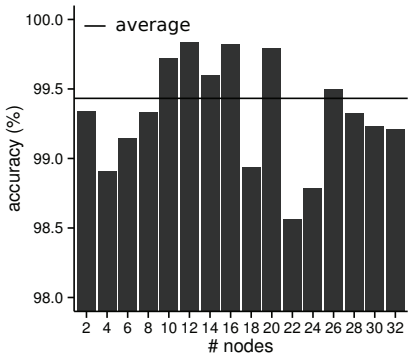
5. MODEL VALIDATION

In order to empirically validate our model, we have conducted a set of experiments using a cluster platform at Rutgers ("Dell cluster", hereafter). The cluster consists of two Dell PowerEdge M1000E blade enclosures maximally configured with sixteen PowerEdge M610 nodes (blades), each node having two Intel Xeon E5504 Nehalem processors at 2.4 GHz. Each node also has 6 GB RAM (6×1 GB DIMM Hynix DDR3-1333 ECC) and 73 GB of local disk storage. The network infrastructure comprises an integrated 16-port Mellanox M2401G InfiniBand switch within each blade chassis (running at 50W, 36.8W dissipated + 1.65W per port), each switch having eight uplink ports and linked via eight InfiniBand lanes to the uplink ports on the switch in the other chassis. All blades have Mellanox ConnectX MT26428 Quad-Data-Rate (QDR) InfiniBand interface cards. There is also an integrated (redundant) 1 Gigabit Ethernet within each chassis, with two pairs of 10 Gigabit uplink capabilities in each chassis. Each blade enclosure also includes metering functionality that can be queried on runtime using the Simple Network Management Protocol (SNMP).

Table 1 lists the system parameters that are used in our model. Note that P^{static}_{system}, BW_{net} and BW_{mem} were obtained empirically using standard benchmarking. The rest of parameters are theoretical from the vendors' specifications. We separately validate our energy model and the associated data communication (E_{comm}; see Equation 8) because data communication is an important aspect of our study.

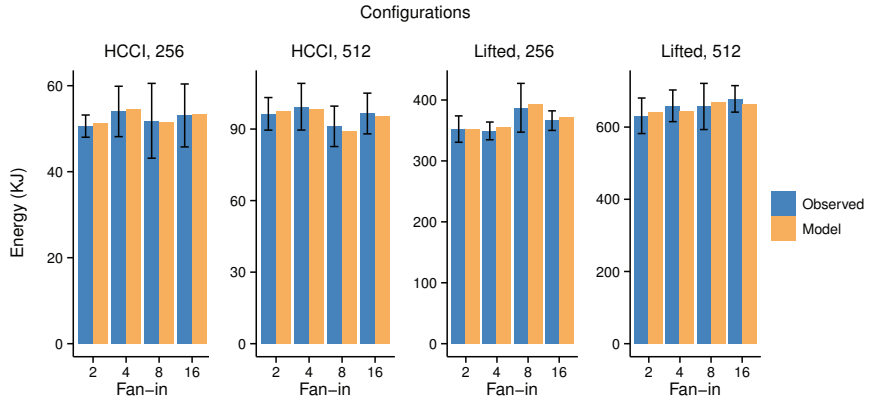


(a) Scalability tests

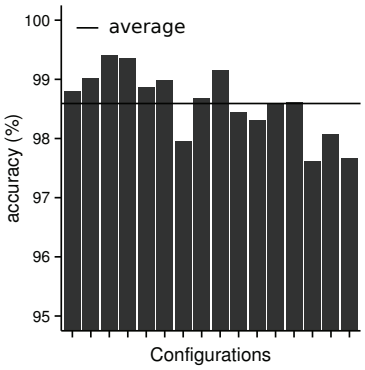


(b) Accuracy of the model

Figure 2: Results of ping-pong tests using 2 blade enclosures (32 nodes).



(a) Energy consumption measured (observed) and from our model with different configurations



(b) Accuracy of the model

Figure 3: Validation of the analysis code on the Dell cluster with 1 and 2 blade enclosures, i.e., 16 and 32 nodes, respectively. Note that x axis labels on subfigure (b) are not displayed for readability.

Parameter	Value
P_{system}^{static}	1,377W (enclosure)
P_{cpu}	80W (x2 per node)
P_{mem}	∼15W (node)
P_{nic}	9.69W
BW_{net}	1,183MB/s (empiric)
BW_{mem}	7,950MB/s (empiric)

Table 1: Dell cluster specifications.

Data communication model validation. We ran a MPI variant of ping-pong tests using $2, \ldots, 32$ nodes (i.e., by pairs) five times. The experiments were run with the whole cluster idle, and provided us with an empirical measurement of the network bandwidth (BW_{net} in Table 1). Figure 2(a) plots the average power dissipation of the system when running the ping-pong tests. We plot the average power and the variability in the experiments since power is independent of the data size. We also plot the power ($P_{transfer}$) obtained using our model (see Equation 8) for the Dell cluster, where P_{over} is the power from processor and memory associated with system overheads:

$$P_{transfer} = P_{nic}^{active} + P_{over} \quad (9)$$

Figure 2(b) shows the accuracy of our model. The figure plots variability as well as the average value (99.43%).

Model validation at small scale. We ran the in-situ topological analysis application using one and two blade enclosures of the dell Cluster (i.e., 256 and 512 MPI processes, respectively) with the HCCI and Lifted data sets and fan-in={2, 4, 8, 16}. We instrumented the application as described in Section 3 in order to collect data about architecture-independent operations and MPI communications. However, we also ran the same set of experiments ten times to measure real execution times and collect power readings.

Figure 3(a) displays the energy consumption and variability with the set of experiments described above, from empirical measurements (observed) and from our model. The error bars correspond to the standard deviation. Figure 3(b) shows the accuracy of the model for the different experiments, which is 98.59% on average.

6. EXPLORING POWER/PERFORMANCE TRADE-OFFS AT SCALE

This section describes how we extrapolate our power model to Titan, and presents result of experiments in which we have used our model to explore the impacts of system architecture, algorithm design choices, and deployment options on data exchange patterns and overall energy performance.

6.1 Power Model Extrapolation

System power model. We extrapolate our system power model to Titan based on its known power requirements and hardware vendor specifications with the following reasoning. Titan drains 8,209kW at full capacity (source: `http://www.top500.org`), which results on 439W per node (from a total of 18,668 nodes). Each node has one Opteron 6274 processor (P_{cpu}), 32GB of RAM in four 8GB DIMMs (P_{mem}), one NVIDIA K20x GPU (P_{gpu}), and each two nodes share a Cray Gemini interconnect ASIC (power information not publicly available). Consequently, the remaining 89W of power (i.e., $P_{system}^{dynamic} - P_{cpu} - P_{mem} - P_{gpu}$) comes from the Gemini interconnect and other components (e.g., control circuitry). We have fixed the amount of power associated to the network interface to 95% of this remaining power (i.e., $P_{nic} = 85$W) as a rough estimation. However, we address this problem from a co-design perspective as discussed in Section 6.2.

Parameter	Value
$P_{system}^{dynamic}$	439W
P_{cpu}	115W
P_{mem}	∼10W (node)
P_{gpu}	225W

Table 2: Titan specification parameters.

Gemini interconnect model. Titan contains 9,344 Gemini SoC ASIC controllers, each of them attached to two compute nodes via its two NICs (see Figure 4). The Gemini controllers are arranged following a 25×32×24 3D torus topology with 6 links per controller to its 6 surrounding neighbors. In contrast to switched networks such as InfiniBand, Gemini is a direct network, which means that the processors are integrated directly into the network fabric. As a result, the energy required to transfer data depends on locality. Our communication energy model ($P_{transfer}$ in Equation 8) therefore takes into account the number of Gemini controllers that are traversed to send a message from a source MPI rank to a destination MPI rank (i.e., number of hops).

We have implemented a simulator that, given a set of MPI messages M (see Equation 7) and MPI rank-core mapping policy, returns the number of hops required by each message using the steps listed below:

1. Creates the given $\{N_X, N_Y, N_Z\}$ 3D torus network topology ($\{25\times32\times34\}$ in our case),
2. Maps MPI ranks onto the 3D torus (e.g., onto consecutive cores to improve memory locality).
3. Finds the shortest path (in number of hops) from the source to the destination MPI ranks for each message.

We obtain the shortest path between two MPI ranks by dividing the three-dimensional space associated to the 3D torus into three simpler uni-dimensional spaces. Specifically,

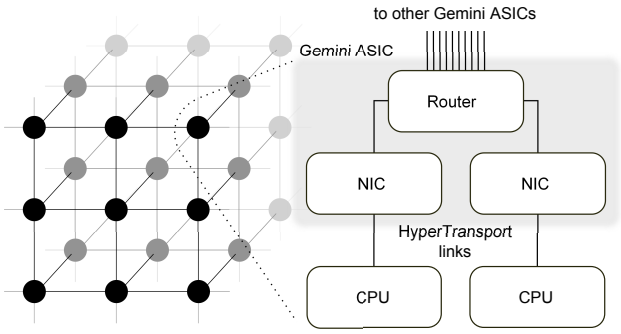


Figure 4: Block diagram of the Gemini network interconnect.

to find the shortest path from a MPI rank mapped onto $\{X_S, Y_S, Z_S\}$ to a MPI rank mapped onto $\{X_D, Y_D, Z_D\}$, we follow two main steps as below.

1. Find shortest path for each dimension, individually:
   - P_X, i.e., from $\{X_S, 0, 0\}$ to $\{X_D, 0, 0\}$
   - P_Y, i.e., from $\{0, Y_S, 0\}$ to $\{0, Y_D, 0\}$
   - P_Z, i.e., from $\{0, 0, Z_S\}$ to $\{0, 0, Z_D\}$
2. Sum the distance of paths P_X, P_Y and P_Z.

Each dimension (X for example) contains N_X nodes connected linearly as a ring, (i.e, the first node is connected with the second node, the second to the third, etc., and the last (N_X) node is connect to the first node). Consequently, we need to consider the following two paths in order to find the shortest distance (D).

$$D_{1,X} = \max(X_S, X_D) - \min(X_S, X_D) \quad (10)$$

$$D_{2,X} = \min(X_S, X_D) + N_X - \max(X_S, X_D) \quad (11)$$

As a result, the shortest path D_X will be the minimum of Equations 10 and 11. The same reasoning can be applied to the Y and Z dimensions.

6.2 Experimental Results

We performed a number of experiments using different fan-in parameters, domain decompositions, and node layouts, and the results of our analysis are captured in Table 3 and Figures 5, 6, and 7.

Figure 5 summarizes the macro communication behavior of the topological analysis of the lifted ethylene jet data set. The information was obtained through a collection of experiments performed using algorithmic variations, including total number of cores, domain decomposition, and merge fan-in, as well as varying the node-mapping strategies. Each column in the graphic corresponds to a different node-mapping configuration, while each row in the graphic corresponds to variations in fan-in merge values. Each column of each subfigure displays a histogram of the number of hops that MPI messages require to reach the destination rank from the source rank. The x axis of each subfigure corresponds to the number of cores involved in the run, which in turn determined the domain decomposition used. The default mapping algorithm maps MPI ranks consecutively across allocated cores, while the three subfigures in the rightmost column use a random mapping algorithm (within a partition of the torus). The thickness of each point

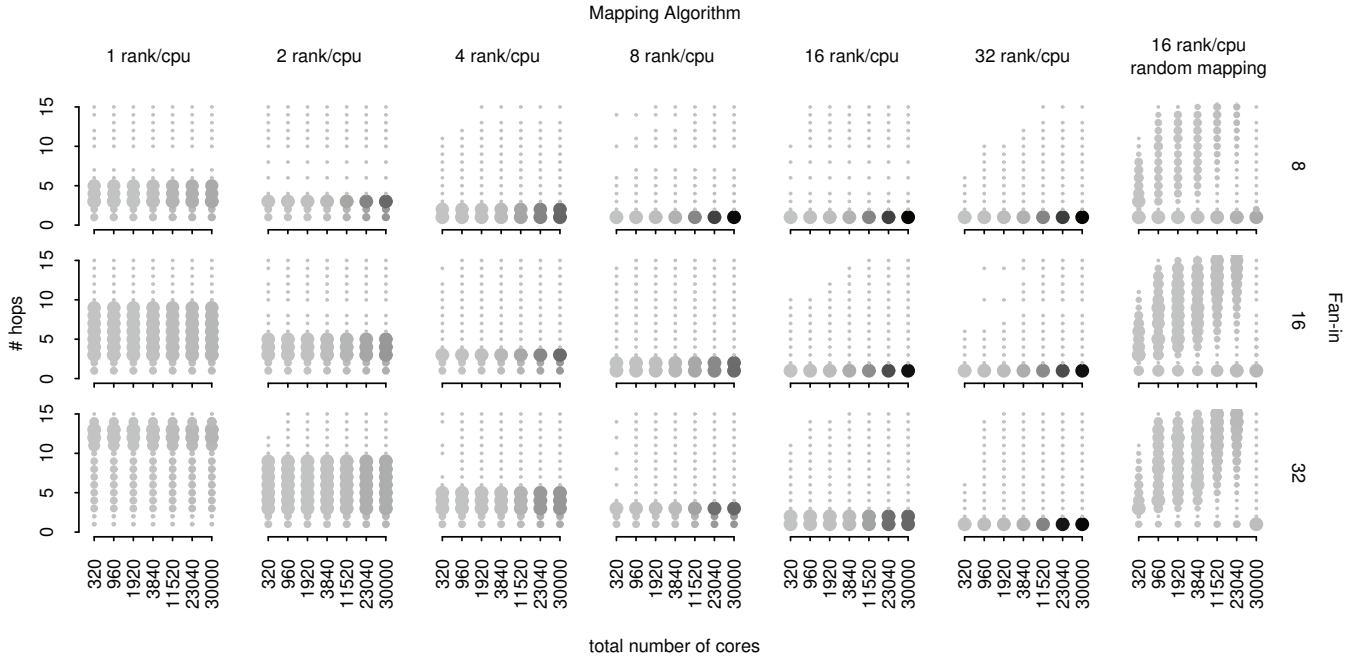


Figure 5: This graphic summarizes the macro communication behavior of the topological analysis of the lifted ethylene jet data set. The information was obtained through a collection of experiments performed using algorithmic variations (including total number of cores, domain decomposition, and merge fan-in) and node-mapping strategies. Each column in the graphic corresponds to a different node-mapping configuration, while each row in the graphic corresponds to variations in fan-in merge values. Each column of each subfigure displays a histogram of the number of hops that MPI messages require to reach the destination rank from the source rank. The x axis of each subfigure corresponds to the number of cores involved in the run, which in turn determined the domain decomposition used. The default mapping algorithm maps MPI ranks consecutively across allocated cores, while the three subfigures in the rightmost column use a random mapping algorithm (within a partition of the torus). The thickness of each point represents the number of messages that required the corresponding number of hops to be transferred, and the color of the point represents the relative number of messages with respect to the maximum number of messages for the set of experiments.

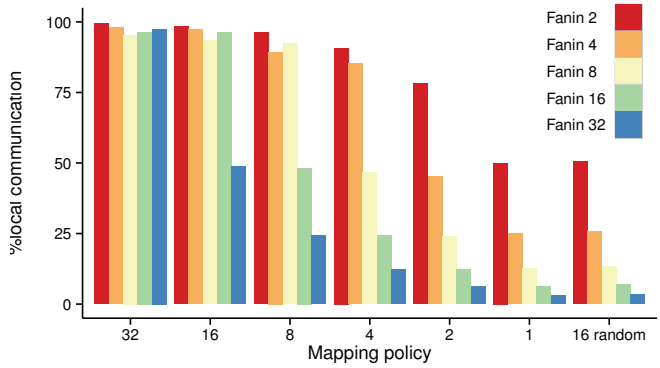


Figure 6: Percentage of local ("on chip") data communication.

represents the number of messages that required the corresponding number of hops to be transferred, and the color of the point represents the relative number of messages with respect to the maximum number of messages for the set of experiments.

Algorithmic effects. Figures 5 and 6 show the amount of local communication, which gives an estimation of the number of MPI messages (i.e., network data communication) for each configuration. On examination of the figures, we see several clear trends. First, total overall network data communication increases as the fan-in increases. Second, the total number of messages increases as the number of cores increases but the impact on the number of hops is not significant unless the random mapping algorithm is used. This suggests that the algorithm is highly scalable in terms of network communications, which is expected as data com-

# cores	Fan in	Time (s)	Energy (KJ)	$MIPS$ ($x10^3$)	MEM_{bw} (GB/s)	MPI MSGs
320	8	55.9	214.4	142	484	3,716
	16	57.7	221.3	138	473	4,752
	32	59.3	227.4	139	478	5,946
960	8	18.8	225.5	423	1,440	10,917
	16	19.0	228.7	420	1,434	11,116
	32	25.3	300.6	334	1,151	16,815
1,920	8	11.3	284.5	735	2,517	19,548
	16	12.4	310.0	700	2,417	24,958
	32	12.0	301.6	710	2,446	24,644
3,840	8	8.3	447.7	1183	4,135	57,424
	16	12.7	652.0	864	3,070	73,367
	32	8.2	440.5	1,172	4,085	50,902
11,520	8	6.9	1,227.1	1,869	6,717	194,043
	16	6.7	903.2	N/A	N/A	170,912
	32	9.9	1,668.2	1,526	5,593	194,596
23,040	8	15.7	5,095.2	1,215	4,508	417,360
	16	7.5	2,861.9	2,457	9,106	360,422
	32	18.0	6,060.9	1,477	5,641	508,704
30,000	8	8.0	4,104.4	2,777	10,399	600,072
	16	8.7	4,357.4	2,497	9,360	535,693
	32	12.3	5,828.7	2,047	7,778	556,098

Table 3: This table summarizes the results obtained from different variations (including total number of cores and merge fan-in) of the topological analysis of the lifted ethylene jet execution on Titan. The results include execution time, energy consumption, millions of integer operations per second, memory accesses per second and number of MPI messages. Note that the energy consumption is an estimation using our model.

munication is structured as a tree. The mapping algorithm can significantly impact the amount of local ("on-chip") communication, and therefore choosing the optimal fan-in will depend mainly of the number of MPI ranks that can be allocated per node and, consequently, the number of available cores. This has implications on architecture co-design as discussed at the end of this section.

Effects of runtime MPI rank mapping. As we can see from Figure 5, the mapping algorithm clearly impacts the number of hops required to send messages. Specifically, as the number of ranks per node increases, there is a corresponding decrease in the total data communication while the number of ranks per node is greater than the fan-in merge parameter. When the number of ranks/node is equal or larger than the fan-in merge parameter, most of the communication is local. As can be observed by looking at the diagonals of the graphic in Figure 5 from top-left to bottom-right, the mapping algorithm and fan-in are actually highly correlated (hops $\sim \frac{\text{fan-in}}{\text{ranks per node}}$). Since the communications are structured in a hierarchical merge pattern, when the fan-in is twice the number of ranks/node, this results in approximately half of the messages requiring off-node communication. For example, if we map 8 ranks per node, with a fan-in of 8, most of the communication will be local. However, with fan-ins of 16 and 32, around 50% and 25% of the communications respectively will be remote (see Figure 6). Moreover, as would be expected, the consecutive mapping algorithm presents better data locality than the random MPI mapping algorithm, regardless of the number of ranks per node. Random allocation results in a larger number of hops, which is not scalable from both a latency and energy perspectives as can be observed in Figure 8 (note the changes in scale along the Y axis across the various plots). This supports the argument that data locality is essential for energy efficiency at scale. It is also interesting to note that in Figure 5, it is clear that under a random distribution of MPI rankings, communications require more hops on average than with one MPI rank allocated per node, in spite of the fact that under the random distribution there are 16 ranks per node supported. However, upon examination of Figure 6, we note that the percentage of local communication is similar for these two configurations.

System architecture effects. Detailed power information is not always available for every component of the system. For example, we have little information about the power drawn by the Gemini NIC. Furthermore, it is often desirable to examine co-design questions such as, "What would be the change in full-system power if a future generation NIC were to consume less power than current-generation NICs?" Our power-modeling approach is robust to both these types of missing information. Specifically, we can examine ranges of power consumption to bound the impact of any component on the power consumed by the whole system. Figure 7 presents an example of such a study. The x axis represents the number of cores, and the y axis represents system power. Each curve represents a different value of NIC power (from 0% to 100%) as a percentage of the power remaining after subtracting the other subsystems' power consumption, as discussed in Section 6.1. As Figure 7 shows, the model indicates, for example, that if NIC power were reduced from 100% to 0%, the system power when running the in-situ topological analysis at 30,000 cores would drop by about 40%. This can be invaluable information when co-designing a system because it helps examine

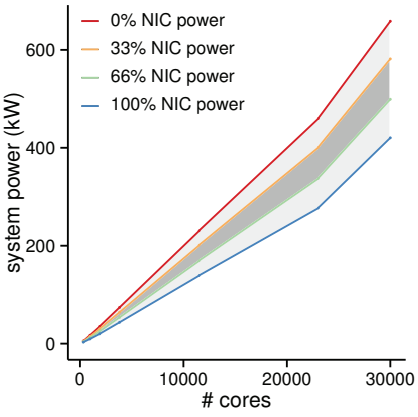
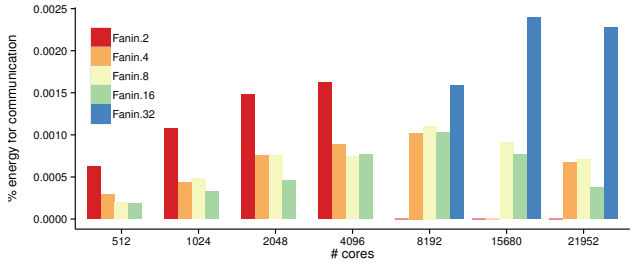


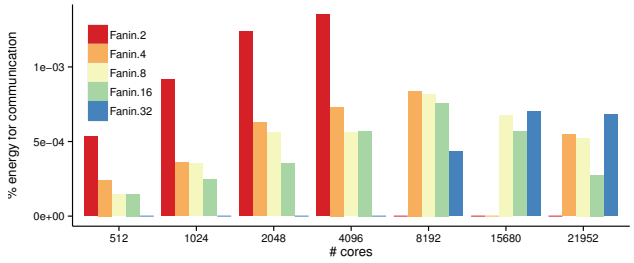
Figure 7: Power consumption range for the Gemini interface in Titan. The percentages are the fraction the amount of power remaining after subtracting the other subsystems' power consumption. 0% and 100% of power consumption serve as the lower andupper bounds even though they are not realistic in practice.

power/performance trade-offs in a system- and application-centric manner. Furthermore, we believe the current network interface operational power region to be within the dark shadowed area of Figure 7 (i.e., between 33% and 66% of the remaining node power is used by the NIC).

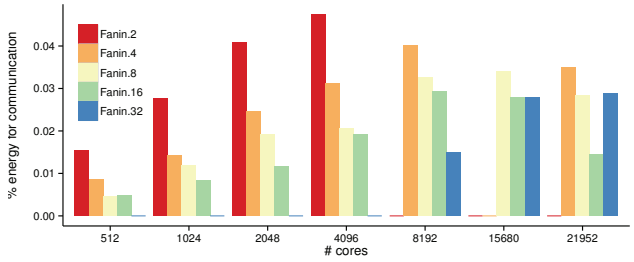
Architectural solutions could also mitigate algorithm and runtime rank mapping effects shown below. For example, Figure 8(a) shows that the percentage of energy consumed for data communication is high when the fan-in is larger than the number of ranks mapped per node. An algorithmic solution to this problem is to use a lower fan-in but would incur the associated penalties in execution time and energy. However, a potential architectural solution would be to consider using two shared memory multi-processors instead of four independent processors to mitigate the problem (see Figure 8(b)), which would also result in a reduction of the percentage of energy consumed for data communication.



(a) Consecutive allocation algorithm with 16 MPI processes (and cores) per node



(b) Consecutive allocation algorithm with 32 MPI processes (and cores) per node



(c) Random allocation algorithm with 16 MPI processes (and cores) per node

Figure 8: Percentage of energy accounted for communication.

7. CONCLUSION AND FUTURE WORK

As scientific applications target exascale, challenges related to data and energy are becoming dominating concerns. In this paper we explored data-related energy/performance trade-offs for end-to-end simulation workflows running at scale on current high-end computing systems. Specifically, we have 1) developed and validated a power model based on machine-independent algorithm characteristics, 2) used the model to explore the impacts of system architecture, algorithm design choices and deployment options on data exchange patterns and overall energy performance, and 3) discussed how to extend our model to help answer design questions for emerging architectures. For example, as Figure 7 demonstrates, it may be possible to utilize our model to co-design applications and networks for power efficiency. That is, given network power/performance trade-offs from a network vendor, we can model the overall system power consumption of in-situ topological analysis to jointly determine how to maximize the ratio of code performance to power consumption. While our current model considers macro system behaviors, we plan to extend it to account for deep-memory hierarchies, and to compare and contrast the behaviors of a larger set of use-case scenarios on potential architectures to further study the impact of various co-design trade-offs. Our ongoing work also includes the use of finer-grain modeling and system-specific parameters when they are available.

Acknowledgments

The research presented in this work is supported in part by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy through the Scientific Discovery through Advanced Computing (SciDAC) Institute of Scalable Data Management, Analysis and Visualization (SDAV) under ward number DE-SC0007455, the Advanced Scientific Computing Research and Fusion Energy Sciences Partnership for Edge Physics Simulations (EPSI) under award number DE-FG02-06ER54857, the ExaCT Combustion Co-Design Center via subcontract

number 4000110839 from UT Battelle, by the US National Science Foundation (NSF) via grants numbers ACI 1339036, ACI 1310283, DMS 1228203 and IIP 0758566, and by an IBM Faculty Award. The research at Rutgers was conducted as part of the NSF Cloud and Autonomic Computing (CAC) Center at Rutgers University and the Rutgers Discovery Informatics Institute (RDI2). The authors wish to thank the members of the the ExaCT Center for Exascale Simulation of Combustion in Turbulence for useful discussions and support. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. Los Alamos National Laboratory is operated by Los Alamos National Security LLC for the U.S. Department of Energy under contract DE-AC52-06NA25396.

8. REFERENCES

[1] The LLVM Compiler Infrastructure Project. `http://www.llvm.org`, Aug. 2013.

[2] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky. Just In Time: Adding Value to The IO Pipelines of High Performance Applications with JITStaging. In *Proc. of 20th International Symposium on High Performance Distributed Computing (HPDC'11)*, June 2011.

[3] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng. Datastager: scalable data staging services for petascale applications. In *Proc. of 18th International Symposium on High Performance Distributed Computing (HPDC'09)*, 2009.

[4] G. Bansal. *Computational Studies of Autoignition and Combustion in Low Temperature Combustion Engine Environments*. PhD thesis, University of Michigan, 2009.

[5] F. Bellosa. The benefits of event driven energy accounting in power-sensitive systems. In *Proc. of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, pages 37–42, Kolding, Denmark, 2000.

[6] F. Bellosa, A. Weißel, M. Waitz, and S. Kellner. Event-Driven Energy Accounting for Dynamic Thermal Management. In *Proc. of the Workshop on Compilers and Operating Systems for Low Power (COLP'03)*, Sept. 2003.

[7] J. Bennett, V. Krishnamoorthy, S. Liu, R. Grout, E. R. Hawkes, J. H. Chen, J. Shepherd, V. Pascucci, and P.-T. Bremer. Feature-based statistical analysis of combustion simulation data. *IEEE Trans. Vis. Comp. Graph.*, 17(12):1822–1831, 2011.

[8] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, P. Pebay, D. Thompson, H. Yu, F. Zhang, and J. Chen. Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In *Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC '12)*, pages 49:1–49:9, Salt Lake City, UT, USA, 2012.

[9] A. Bhattacharjee and M. Martonosi. Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors. In *Proc. of the 36th annual International Symposium on Computer Architecture (ISCA'09)*, pages 290–301, New York, NY, USA, 2009. ACM.

[10] W. L. Bircher and L. K. John. Complete system power estimation using processor performance events. *IEEE Trans. Comput.*, 61(4):563–577, Apr. 2012.

[11] J.-M. F. Brad Whitlock and J. S. Meredith. Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System. In *Proc. of 11th Eurographics Symposium on Parallel Graphics and Visualization (EGPGV'11)*, April 2011.

[12] P.-T. Bremer, G. H. Weber, V. Pascucci, M. S. Day, and J. B. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, 2010.

[13] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. Day, and J. B. Bell. A topological framework for the interactive exploration of large scale turbulent combustion. In *Proc. IEEE International Conference on e-Science*, pages 247–254. IEEE, December 2009.

[14] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 17:1307–1324, 2011.

[15] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci. A portable programming interface for performance evaluation on modern processors. *The International Journal of High Performance Computing Applications*, 14(3):189–204, 2000.

[16] B. Buck and J. K. Hollingsworth. An API for runtime code patching. *International Journal of High Performance Computing Applications*, 14(4):317–329, 2000.

[17] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Proc. of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 918–926, San Francisco, CA, 2000.

[18] J. H. Chen, A. Choudhary, B. de Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorski, R. Sankaran, S. Shende, and C. S. Yoo. Terascale direct numerical simulations of turbulent combustion using s3d. *Computational Science and Discovery*, 2:1–31, 2009.

[19] G. Contreras and M. Martonosi. Power prediction for intel xscale reg; processors using performance monitoring unit events. In *Proc. of the 2005 International Symposium on Low Power Electronics and Design (ISLPED'05)*, pages 221–226, 2005.

[20] C. Docan, M. Parashar, J. Cummings, and S. Klasky. Moving the Code to the Data - Dynamic Code Deployment Using ActiveSpaces. In *Proc. of 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS'11)*, May 2011.

[21] C. Docan, M. Parashar, and S. Klasky. DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows. In *Proc. of 19th International Symposium on High Performance and*

Distributed Computing (HPDC'10), June 2010.

[22] C. Docan, M. Parashar, and S. Klasky. Dataspaces: an interaction and coordination framework for coupled simulation workflows. *Cluster Computing*, 15(2):163–181, 2012.

[23] D. Donofrio, L. Oliker, J. Shalf, M. F. Wehner, C. Rowen, J. Krueger, S. Kamil, and M. Mohiyuddin. Energy-efficient computing for extreme-scale science. *Computer*, 42(11):62–71, Nov. 2009.

[24] D. Economou, S. Rivoire, and C. Kozyrakis. Full-system power analysis and modeling for server environments. In *In Workshop on Modeling Benchmarking and Simulation (MOBS*, 2006.

[25] N. Fabian, K. Moreland, D. Thompson, A. Bauer, P. Marion, B. Gevecik, M. Rasquin, and K. Jansen. The paraview coprocessing library: A scalable, general purpose in situ visualization library. In *Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 89 –96, October 2011.

[26] W. Felter, K. Rajamani, T. Keller, and C. Rusu. A performance-conserving approach for reducing peak power consumption in server systems. In *Proc. of the 19th annual International Conference on Supercomputing (ICS'05)*, pages 293–302, Cambridge, Massachusetts, 2005.

[27] M. Gamell, I. Rodero, M. Parashar, and S. Poole. Exploring Energy and Performance Behaviors of Data-Intensive Scientific Workflows on Systems with Deep Memory Hierarchies. In *Proc. of the 20th International Conference on High Performance Computing (HiPC)*, pages 1–10, Hyderabad, India, 2013.

[28] A. Globus. A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamics Results. Technical Report RNR 92-031, NAS Applied Research, NASA Ames Research Center, 1992.

[29] S. Gurumurthi, A. Sivasubramaniam, and V. K. Natarajan. Disk drive roadmap from the thermal perspective: A case for dynamic thermal management. In *Proc. of the 32nd annual International Symposium on Computer Architecture (ISCA'05)*, pages 38–49, 2005.

[30] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann. Topologically clean distance fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1432–1439, 2007.

[31] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and freon: temperature emulation and management for server systems. In *Proc. of the 12th international conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XII)*, pages 106–116, San Jose, California, USA, 2006.

[32] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., and R. Bianchini. Energy conservation in heterogeneous server clusters. In *Proc. of the 10th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP'05)*, pages 186–195, Chicago, IL, USA, 2005.

[33] K. Hoste and L. Eeckhout. Microarchitecture-independent workload characterization. *IEEE Micro*, 27(3):63–72, May–June 2007.

[34] C. Isci and M. Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proc. of the 36th annual IEEE/ACM International Symposium on Microarchitecture (MICRO 36)*, pages 93–, 2003.

[35] C. Isci and M. Martonosi. Phase characterization for power: evaluating control-flow-based and event-counter-based techniques. In *12th International Symposium on High-Performance Computer Architecture, 2006*, pages 121–132, 2006.

[36] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *Proceedings of the 2001 international symposium on Low power electronics and design*, ISLPED '01, pages 135–140, New York, NY, USA, 2001. ACM.

[37] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam. Understanding the performance-temperature interactions in disk i/o of server workloads. In *12th International Symposium onHigh-Performance Computer Architecture*, pages 176–186, 2006.

[38] R. Kotla, A. Devgan, S. Ghiasi, T. Keller, and F. Rawson. Characterizing the impact of different memory-intensity levels. In *IEEE International Workshop on Workload Characterization (WWC-7)*, pages 3–10, 2004.

[39] J. Krueger, D. Donofrio, J. Shalf, M. Mohiyuddin, S. Williams, L. Oliker, and F.-J. Pfreund. Hardware/software co-design for energy-efficient seismic modeling. In *Proc. of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*, pages 73:1–73:12, Seattle, Washington, 2011.

[40] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci. Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1053–1060, Sept. 2006.

[41] C. Lattner and V. Adve. LLVM: A compilation framework for lifelong program analysis transformation. In *2nd IEEE/ACM International Symposium on Code Generation and Optimization (CGO 2004)*, pages 75–86, San José, California, Mar. 20–24, 2004.

[42] K.-J. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. In *Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11*, pages 232.1–, 2005.

[43] A. Lewis, S. Ghosh, and N.-F. Tzeng. Run-time energy consumption estimation based on workload in server systems. In *Proc. of the 2008 conference on Power aware computing and systems (HotPower'08)*, pages 4–4, San Diego, California, 2008.

[44] A. W. Lewis, N.-F. Tzeng, and S. Ghosh. Runtime energy consumption estimation for server workloads based on chaotic time-series approximation. *ACM Trans. Archit. Code Optim.*, 9(3):15:1–15:26, 2012.

[45] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. In *Proc. of the 2003 ACM SIGMETRICS international*

conference on Measurement and modeling of computer systems (SIGMETRICS'03), pages 160–171, San Diego, CA, USA, 2003.

[46] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. Pin: Building customized program analysis tools with dynamic instrumentation. In *ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI '05)*, Chicago, Illinois, June 11–15, 2005.

[47] K.-L. Ma. Runtime Volume Visualization of Parallel CFD. In *Proceedings of Parallel CFD Conference*, pages 307–314, 1995.

[48] X. Ma, M. Dong, L. Zhong, and Z. Deng. Statistical Power Consumption Analysis and Modeling for GPU-based Computing. In *Workshop on Power Aware Computing and Systems (HotPower '09)*, 2009.

[49] A. Mascarenhas, R. W. Grout, P.-T. Bremer, E. R. Hawkes, V. Pascucci, and J. H. Chen. Topological feature extraction for comparison of terascale combustion simulation data. In V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, editors, *Topological Methods in Data Analysis and Visualization*, Mathematics and Visualization, pages 229–240. Springer Berlin Heidelberg, 2011.

[50] A. Mascarenhas and J. Snoeyink. Isocontour based visualization of time-varying scalar fields. In *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, Mathematics and Visualization, pages 41–68. Springer Berlin Heidelberg, 2009.

[51] W. McLendon, G. Bansal, P.-T. Bremer, H. Kolla, V. Pascucci, and J. Bennett. On the use of graph search techniques for the analysis of extreme-scale combustion simulation data. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 57–63, 2012.

[52] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. In *Proc. of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys'06)*, pages 403–414, Leuven, Belgium, 2006.

[53] D. Morozov and G. H. Weber. Distributed merge trees. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '13)*, pages 93–102, Shenzhen, China, 2013.

[54] S. Pakin and P. McCormick. Byfl: Compiler-based Application Analysis. `https://github.com/losalamos/Byfl`, Aug. 2013.

[55] S. G. Parker and C. R. Johnson. SCIRun: A Scientific Programming Environment for Computational Steering. In *Proceedings of ACM/IEEE Supercomputing Conference*, 1995.

[56] V. Pascucci and K. Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38:249–268, 2003.

[57] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas. Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. Graph.*, 26(3), July 2007.

[58] D. Quinlan. ROSE: Compiler support for object-oriented frameworks. *Parallel Processing Letters*, 10(2–3):215–226, 2000.

[59] G. Reeb. Sur les points singuliers d'une forme de pfaff completement intergrable ou d'une fonction numerique [on the singular points of a complete integral pfaff form or of a numerical function]. *Comptes Rendus Acad.Science Paris*, 222:847–849, 1946.

[60] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *Proc. of the 2008 conference on Power aware computing and systems (HotPower'08)*, pages 3–3, San Diego, California, 2008.

[61] I. Rodero, S. Chandra, M. Parashar, R. Muralidhar, H. Seshadri, and S. Poole. Investigating the Potential of Application-Centric Aggressive Power Management for HPC Workloads. In *Proc. of the 17th International Conference on High Performance Computing (HiPC)*, pages 1–10, Goa, India, Dec. 2010.

[62] J. Rowlan, E. Lent, N. Gokhale, and S. Bradshaw. A Distributed, Parallel, Interactive Volume Rendering Package. In *Proceedings of IEEE Visualization Conference*, pages 21–30, 1994.

[63] K. Singh, M. Bhadauria, and S. A. McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37(2):46–55, July 2009.

[64] T. Tu, H. Yu, L. Ramirez-Guzmanz, J. Bielak, O. Ghattas, K.-L. Ma, and D. R. O'Hallaron. From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing. In *Proceedings of ACM/IEEE Supercomputing Conference*, 2006.

[65] V. Vishwanath, M. Hereld, and M. Papka. Toward simulation-time data analysis and i/o acceleration on leadership-class systems. In *Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, October 2011.

[66] V. M. Weaver and J. Dongarra. Can hardware performance counters produce expected, deterministic results? In *Proc. of the 3rd Workshop on Functionality of Hardware Performance Monitoring, 43rd International Symposium on Microarchitecture*, Atlanta, Georgia, 2010.

[67] W. Widanagamaachchi, C. Christensen, P.-T. Bremer, and V. Pascucci. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pages 9–17, 2012.

[68] C. S. Yoo, E. S. Richardson, R. Sankaran, and J. H. Chen. A {DNS} study on the stabilization mechanism of a turbulent lifted ethylene jet flame in highly-heated coflow. *Proc. of the Combustion Institute*, 33(1):1619–1627, 2011.

[69] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma. In-situ visualization for large-scale combustion simulations. *IEEE Computer Graphics and Applications*, 30:45–57, 2010.

[70] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, S. Klasky, Q. Liu, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf. PreDatA - preparatory data analytics on peta-scale machines. In *Proc. of 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*, April 2010.