2017 IEEE International Parallel and Distributed Processing Symposium

Characterizing and Modeling Power and Energy for Extreme-Scale *In-situ* Visualization

Vignesh Adhinarayanan*, Wu-chun Feng*, David Rogers†, James Ahrens†, Scott Pakin†
*Department of Computer Science, Virginia Tech, Blacksburg, Virginia 24061
†Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, New Mexico 87545
{avignesh, wfeng}@vt.edu, {dhr, ahrens, pakin}@lanl.gov

***Abstract*—Plans for exascale computing have identified power and energy as looming problems for simulations running at that scale. In particular, writing to disk all the data generated by these simulations is becoming prohibitively expensive due to the energy consumption of the supercomputer while it idles waiting for data to be written to permanent storage. In addition, the power cost of data movement is also steadily increasing. A solution to this problem is to write only a small fraction of the data generated while still maintaining the cognitive fidelity of the visualization. With domain scientists increasingly amenable towards adopting an *in-situ* framework that can identify and extract valuable data from extremely large simulation results and write them to permanent storage as compact images, a large-scale simulation will commit to disk a reduced dataset of data extracts that will be much smaller than the raw results, resulting in a savings in both power and energy.**

The goal of this paper is two-fold: (i) to understand the role of *in-situ* techniques in combating power and energy issues of extreme-scale visualization and (ii) to create a model for performance, power, energy, and storage to facilitate what-if analysis. Our experiments on a specially instrumented, dedicated 150-node cluster show that while it is difficult to achieve power savings in practice using *in-situ* techniques, applications can achieve significant energy savings due to shorter write times for *in-situ* visualization. We present a characterization of power and energy for *in-situ* visualization; an application-aware, architecture-specific methodology for modeling and analysis of such *in-situ* workflows; and results that uncover indirect power savings in visualization workflows for high-performance computing (HPC).

***Keywords*-energy, energy efficiency, green computing, in-situ visualization, power, scientific visualization, scientific workflow**

I. Introduction

Future supercomputers are expected to be power-limited. For instance, the U.S. Department of Energy has capped the power consumption of the upcoming exascale supercomputer (projected to arrive in 2022–23) at 20 MW [1], [2]. For these power-limited supercomputers, it is important to utilize the allocated power effectively. However, an analysis of power usage of production supercomputers has revealed that these machines, on an average, use only 40–55% of their budgeted power [3]. A "trapped capacity" of more than 45% could adversely affect the ambitious performance goals of the exascale initiative. Like power, energy consumption has also emerged as an important issue. A typical estimate of *one million* dollars per megawatt means that over 40% of the acquisition cost of a supercomputer goes towards paying energy bills [1]. Thus, plans for exascale computing have identified power and energy as looming problems for simulations running at that scale.

This paper deals with addressing power and energy issues for an important class of applications: scientific data visualization and analytics. We explore using *in-situ* analysis and visualization methods to reduce energy consumption and increase power utilization. *In-situ* techniques identify and extract valuable data from extremely large simulation results and write them to permanent storage in a compact form. They therefore reduce the data movement to storage and I/O wait time. Past studies have shown that off-chip data movement is very expensive; its energy cost can be 100 times that of a floating-point operation [4]. Thus, by reducing off-chip data movement, *in-situ* techniques can be expected to reduce energy. In addition, reducing I/O wait times decreases the time that CPUs idle on I/O, which in turn, means that CPUs are busy doing more useful work, thus increasing CPU utilization. This increased utilization implies that the CPUs are consuming more power, thus harnessing trapped capacity. With *in-situ* analysis and visualization already being a core tool in scientific workflows at exascale—they help scientists find important features in data, extract valuable information, and investigate more manageable and compact results from large scientific simulations—we expect that existing *in-situ* frameworks can also help address power and energy challenges [2].

This paper studies the role of *in-situ* techniques in reducing power and energy for coupled simulation-visualization applications. While past work in this space used analytical power models to study *in-situ* pipelines [5], we use real measurements on supercomputers. To our knowledge, this is the first study of power and energy consumption that considers power consumed not only by a reasonably-sized compute cluster (150 nodes) but also by the associated storage subsystem when both are used concurrently in a scientific simulation with high I/O requirements. Our contributions in this paper can be summarized as follows:

- a characterization of power and energy of compute and storage subsystems for *in-situ* analysis and visualization,
- an application-aware, architecture-specific methodology for modeling and analysis of *in-situ* analysis and visualization, and
- results that reveal the nature of indirect energy savings in HPC visualization workflows.

1530-2075/17 $31.00 © 2017 IEEE
DOI 10.1109/IPDPS.2017.113

Our experimental results show that an *in-situ* pipeline runs 51% faster, consumes 50% less energy, and occupies 99.5% less disk space than a post-processing pipeline for an ocean simulation application. The power consumption, however, remains unaffected. While the results are along expected lines, ours is the first work that confirms (1) experimentally, (2) at a reasonable scale, (3) in a controlled environment, and (4) while simultaneously measuring both compute and storage subsystems.

The rest of the paper is organized as follows. We provide background on visualization techniques in Section II and discuss related work in Section III. Our characterization methodology is presented in Section IV with characterization results in Section V. The model developed based on the characterization results and the applications of the model are presented in Section VI and Section VII, respectively. Then, we discuss opportunities for power reduction in Section VIII. Finally, we draw conclusions from our experiments in Section IX.

II. Background

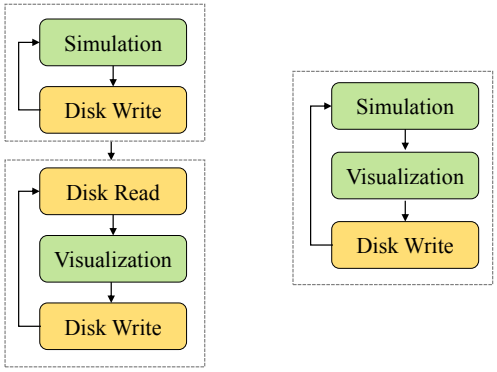
In this section, we provide some background on the two major types of visualization pipelines, namely post-processing and *in-situ* pipelines. We track their origins and traditional use cases. We bring attention to a modern trend, where scientists are forced to adopt *in-situ* pipelines due to the disparity between compute and storage performance in modern high-performance computing (HPC) systems. Finally, we formulate some hypotheses about *in-situ* techniques and their role in addressing emerging power and energy issues.

A. Terminology

The terms post-processing and *in-situ* visualization have been used in the literature to refer to a number of different techniques. We will first define these terms as they are used in this paper. In a traditional *post-processing* visualization, as shown in Figure 1(a), a simulation is performed; and at the end of each iteration of the simulation, raw data is written to the storage system. This storage system is typically a parallel file system in HPC environments. After the simulation is complete, the data is transferred to a rendering cluster where an image is rendered for each iteration. This is in contrast with *in-situ* visualization, as shown in Figure 1(b), where the simulation and visualization tasks are performed on the same machine *concurrently*. Instead of saving the raw data at the end of each iteration, an image corresponding to the data is rendered and saved to disk. The image thus produced is typically orders of magnitude smaller than the raw data.

B. Use Cases for In-situ Techniques

Earlier work on *in-situ* visualization viewed the technique as a way to monitor simulations. Ma [6] coupled a parallel computational fluid dynamics (CFD) solver with a volume renderer to create an *in-situ* pipeline. Such a coupling would, for example, enable scientists to quickly identify incorrect initial conditions in a simulation and abandon these incorrect simulations early on. This, in turn, could potentially save valuable supercomputing time.



(a) Post-processing pipeline (b) In-situ pipeline

Fig. 1: Types of visualization pipelines

A related use case is computational steering, in which the visualization is performed alongside the simulation in order to *guide* the simulation towards a better solution. Early examples include Haimes's work [7], in which computations in a CFD simulation can be inspected while the simulation is still running. Another example involves the use of *in-situ* techniques to guide convergence in protein-folding problems [8].

While *in-situ* techniques have their own advantages, they are not well-suited for exploratory research. That is, to make good use of *in-situ* pipelines, the analysis and visualization tasks need to be defined ahead of time. As such, post-processing visualization techniques became the *de facto* standard in high-performance computing. However, the balance has recently shifted to *in-situ* techniques due to storage performance and capacity constraints in HPC systems.

Until recently, it was practical to write the raw data for every iteration or timestep of a simulation. However, over time, HPC systems have grown in compute performance at a much faster rate than I/O performance. As a consequence, it is no longer practical to save a simulation's raw data every iteration. Scientists are forced to save their data only every few steps using a technique known as *temporal* sampling. However, understanding the simulation becomes difficult when the sampling frequency gets too low. Owing to this shift in bottleneck towards storage, *in-situ* techniques are becoming more popular. Several frameworks such as ParaView [9], VisIT [10], DataSpaces [11], and ADIOS [12] now support *in-situ* analytics and visualization. A number of applications are also making use of these frameworks to overcome storage bottlenecks [13]–[15].

While the advantage of *in-situ* techniques in terms of storage capacity is obvious, its impact on performance is not so straightforward to discern. On the one hand, when one adopts an *in-situ* pipeline, one can expect to reduce

execution time due to reduced I/O time (from the orders of magnitude smaller output). On the other hand, running a visualization task on an HPC system can consume additional supercomputing time. One can expect *in-situ* techniques to show better performance only if the additional time spent in performing visualization and analysis tasks on supercomputers is less than the saved I/O time. In practice, the vast majority of applications studied [13], [15]–[19] have observed better performance using *in-situ* pipelines.

C. Energy and Power Issues with In-Situ Techniques

The capacity and performance advantages of *in-situ* techniques are well documented. In addition to these, we also expect *in-situ* techniques to address power and energy issues.

Today's supercomputers do not use the entire power budget available to them as applications rarely sustain peak floating-point operations per second (FLOPs) for their entire execution. The reason for this ranges from limited parallelism in applications to memory and I/O stalls. When an application exhibits low power utilization, it leads to higher energy consumption. Higher energy consumption leads to more money spent on energy bills. *In-situ* techniques have the potential to increase power utilization while simultaneously reducing unnecessary power expenditure and reducing net energy consumption.

First, *in-situ* techniques reduce the amount of data that moves outside a node to the file system. Off-chip and off-node data movement consume *two orders of magnitude more power and energy* than on-chip data movement. This can be expected to save a significant amount of power and energy on the storage side, even on typical HPC systems where capacity is relatively small. The savings can be expected to be even higher for *in-situ* techniques for big-data systems that have much larger storage. Second, reducing data movement also reduces I/O wait time. Because energy is the integral of instantaneous power over time, we also expect *in-situ* techniques to save overall energy (on both the storage and compute side). In addition, reducing I/O wait time keeps the compute units more utilized, which should result in increased power utilization (or reduced trapped capacity). In summary, based on first principles, we can expect the following hypotheses to hold true:

Hypothesis 1: *In-situ* techniques reduce the power (and consequently energy) consumption of the storage subsystem.

Hypothesis 2: *In-situ* techniques reduce the overall energy consumption (i.e., on both storage and compute).

Hypothesis 3: *In-situ* techniques increase the overall power consumption and thus power utilization. In other words, *in-situ* techniques can harness trapped capacity.

In this paper, we test these hypotheses from first principles.

III. Related Work

Our work is closely related to Gamell et al. who evaluated the performance and energy trade-offs in an *in-situ* combustion simulation on a large-scale system [5]. Our work differs from theirs in the following ways. First and foremost, their analysis of power and energy is based on analytical models while ours is based on measurements from a real power meter. Second, their study emphasizes the power consumed by the interconnect whereas ours focuses on the entire supercomputer along with the storage subsystem (i.e, the file system). Third, our tests are run on a dedicated supercomputer with no interference from other applications, while the same cannot be said for theirs. Fourth, they study topological analysis – a data analytics task. We study the visualization of eddy currents – a visualization task. Fifth, they study a combustion simulation application whereas we study a climate change application. Finally, even though we both studied the energy cost associated with data movement, we arrive at vastly different conclusions. These differences are summarized in Table I. Nevertheless, the biggest difference between the two works comes from using real power meters on multiple individual components of a dedicated supercomputer and the different conclusions we draw.

TABLE I: Comparison with related work [5]

Paper	Related Work [5]	Our Work
Power	Estimated	Measured
Component	Interconnect	Storage and Compute
Application	Combustion Simulation	Climate Simulation
Interference	Unknown	No
Task	Topological Analysis	Tracking Eddies

Other work in this area includes a multi-dimensional trade-off study on *in-situ* techniques, which consider quality of results in addition to energy and performance [20]. Another paper investigates using NVRAM-based deep memory hierarchy to speed up *in-situ* and in-transit data analytics [21]. They also study the power behavior of such an architecture. Rodero et al. explore yet another dimension in this problem space; they analyze how best to distribute the simulation and visualization tasks within a supercomputing cluster [22]. Adhinarayanan et al. compare power and energy of *in-situ* and post-processing workflows, but their comparison is limited to a single node, which is not representative of large-scale machines [23].

IV. Methodology

We test our initial hypothesis by creating and running post-processing and *in-situ* visualization pipelines on a supercomputer. We instrument the different components of the supercomputer and perform a multi-dimensional measurement study. The experimental details are elaborated in this section.

A. General Approach

We create *in-situ* and post-processing visualization pipelines of a real application (namely, climate simulation) by creating hooks from the simulation to an appropriate visualization framework such as ParaView. The coupled application is run on an instrumented supercomputer supported by a parallel file system. From the instrumentation, we can obtain power readings for the different components such as compute and

storage. For storage and performance results, we read the values directly from the supercomputer.

We run the application on the entire supercomputer so that the power readings are not affected by any other interfering application. We measure and compare the following metrics for *in-situ* and post-processing pipelines: performance, power, energy, and storage requirements. To make best use of supercomputer time, we ran many smaller problems at different application configurations. To quantify energy and storage savings for larger problems, we create empirical models from the initial measurements and extrapolate for realistic configurations used by climate scientists.

B. Experimental Setup

We now describe our HPC test system, our power-monitoring setup, and the scientific application used as a driver for our study.

HPC system For our characterization experiments, we used *Caddy*, a 150-node/2400-core cluster located at Los Alamos National Laboratory. Each node contains two sockets of 8-core Intel E5-2670 Sandy Bridge CPUs running at 2.6 GHz as well as 64 GB of DRAM. Nodes are interconnected using a QLogic InfiniBand QDR network.

A storage cluster running a Lustre file system is attached to this supercomputer. The storage cluster consists of five nodes, which are configured as follows: one node serves as the master node, two nodes are used for metadata servers (MDS), and two nodes are used as object storage servers (OSS). The storage cluster provides 7.7 TB of storage and over 160 MB/s of bandwidth for random reads and writes. Note that this storage cluster is private to *Caddy*, so interference from other clusters is eliminated. We also ran our test application on the entire cluster so that we are measuring only the power consumed by our application.

Power monitoring Here, we describe the setup used for monitoring the power consumption of the compute components and the storage components for the hardware setup used in this study.

The storage cluster was mounted on a Raritan intelligent rack, which is equipped with metered PDUs that are capable of measuring the power consumption at the power inlet. The frequency of data collection was set to one measurement per minute, which is the maximum possible for this type of power meter. Within this one-minute interval, multiple measurements are made and an average power value for that interval is reported. From this average power profile, we calculate other derived metrics such as energy.

Because the compute cluster that we used was not instrumented to collect power at the rack-level, we used the Appro power monitoring interface [24] to collect the power profile at the *cage level*, where a *cage* is a group of ten nodes. Like the rack-level power meter used for the storage cluster, this power meter is capable of providing an average power estimation every minute. We collected data from 15 such power monitors, covering all 150 nodes.

One uniqueness of our work is that we measure the power consumption of *both* the supercomputer and the storage system at the same time while running an application that stresses both compute and storage. This is one of the major improvements over previous work on supercomputer power measurements. While compute and storage power have been studied separately, they have not previously been evaluated simultaneously in a single, dedicated system of reasonable scale. In addition, due to storage system being a shared resource (unlike the compute system, which is only spatially shared), it is rarely ever studied on a per-application basis. As a point of comparison, only *3 out of 500* supercomputers report the power consumed by the storage system to Green500, which tracks the power consumed by the top supercomputers [25].

Application For our investigation, we use a climate-simulation application known as Modeling for Prediction Across Scales (MPAS) [26]. More specifically, we use the ocean component of MPAS (MPAS-O) to compare the different types of visualization pipelines (namely, *in-situ* and post-processing) with different system configurations. The visualization task here is to identify and track *eddies*, which are rotating bodies of fluid surrounded by shearing fluid [27].

To accomplish the tasks we perform the following steps. First, we solve an unstructured grid problem in order to simulate the ocean. From the raw dataset produced by the simulation, we derive a metric known as Okubo-Weiss, which is used to identify eddies. For the post-processing pipeline, the Okubo-Weiss metric is extracted at the end of each timestep of the simulation and written as a netCDF file. We use the PIO library, which in turn uses parallel netCDF so that the output can be written to the parallel file system faster. After the simulation is complete, the netCDF files from each timestep are read back and visualized in parallel using the ParaView framework. An example image produced from the simulation showing the eddies using the Okubo-Weiss metric is shown in Figure 2.

For the *in-situ* pipeline, the extracted Okubo-Weiss metric is not written directly as netCDF. Instead, it is passed to the *Paraview Cinema* framework [28], where it is visualized and then written to the disk. To accomplish this, we used Catalyst adaptors [29] that seamlessly copy simulation data structures to Paraview data structures. While this incurs additional memory operations, it also avoids large data transfers to the storage system.

For all the direct measurements reported in this paper, we use the following problem sizes. We use a grid size of 60 km for the ocean simulation. The simulation is run for a simulated period of six months. Each time step corresponds to a simulated period of half an hour. We evaluate the *in-situ* pipeline and the post-processing pipeline in three different configurations: the output products are written once in every (i) 8 simulated hours, (ii) 24 simulated hours, and (iii) 72

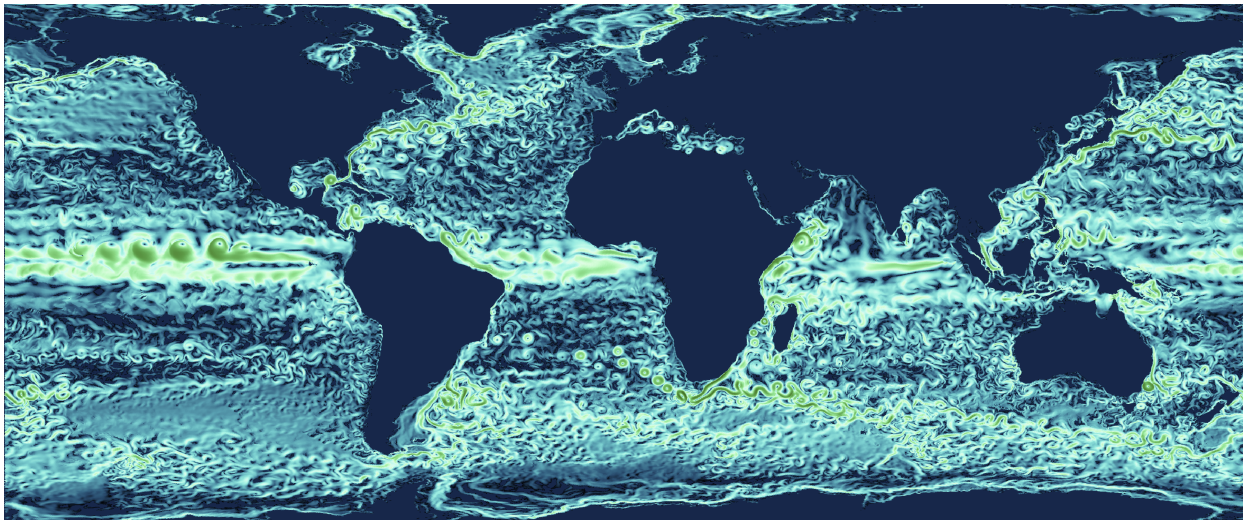


Fig. 2: **Visualization of Okubo-Weiss field across the earth.** The goal of the visualization task is to identify and track eddies which are rotating bodies of fluids surrounded by shearing fluids. Green regions represent rotation and blue regions represent shears. The visualization was generated with ParaView.

simulated hours. The output product can be either netCDF files or images depending on the type of pipeline being run.

V. Experimental Results

In this section, we present the characterization results for the various pipelines studied. The metrics reported include performance (i.e., execution time), power, energy, and storage.

Performance Figure 3 presents the execution time of *in-situ* and post-processing pipelines for the three different application configurations used in this study. The execution time reduced by 51% when going from a post-processing pipeline to *in-situ* pipeline and when the output products of the simulation were written every 8 simulated hours. The execution time improved by 38% and 19% when the frequency of writing the output was reduced to once every 24 hours and once every 72 hours, respectively. The reason for this improvement is that in the latter cases, the simulation phase (rather than the I/O and visualization phases) consumed a greater share of the overall execution time. Because this part of the pipeline is not affected by *in-situ* techniques, we see a diminishing benefit.

Power Next, we compare the power consumption of *in-situ* and post-processing pipelines. To perform this comparison, we first obtain the power profile for each pipeline from the rack-level power meters. An example profile obtained for a post-processing pipeline is shown in Figure 4. From this profile, the average power consumed by the pipeline for the compute and storage components are calculated and summed up. The resulting values for total average power are presented in Figure 5.

As can be seen from the figure, there is practically no difference in the power consumed by the various pipelines

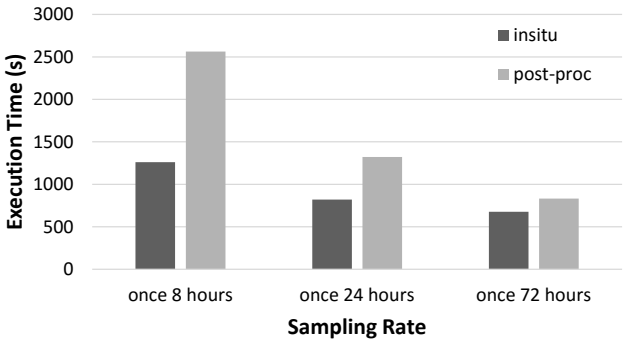


Fig. 3: Comparison of execution time of *in-situ* and post-processing pipelines at three different sampling rates

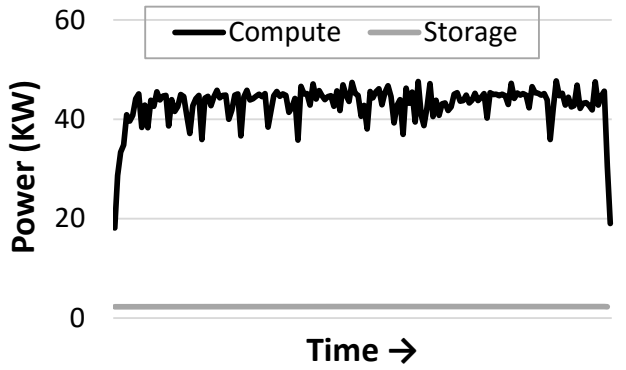


Fig. 4: An example power profile for the compute and storage cluster shown for a post-processing pipeline

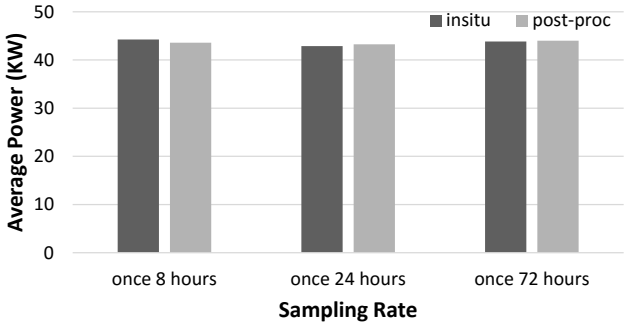


Fig. 5: Comparison of power consumption of *in-situ* and post-processing pipelines at three different sampling rates

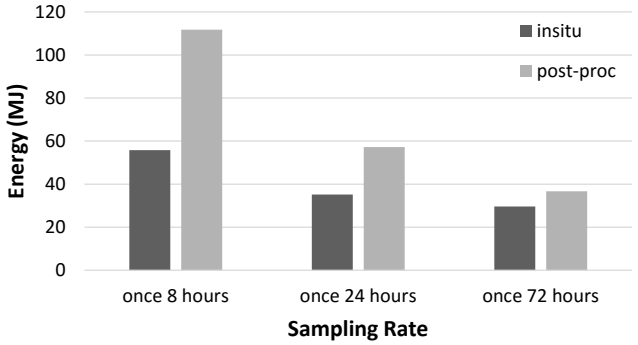


Fig. 6: Comparison of energy consumption of *in-situ* and post-processing pipelines at three different sampling rates

studied. The expectation among the visualization community was that *in-situ* techniques will help save a significant amount of power as data movement is expected to be a big power consumer [5], [30]. However, that popular hypothesis is in fact belied by our measurements.

To explain our observation, we present the power proportionality of the storage subsystem that we obtained by benchmarking the storage cluster. The power consumed by the storage system at idle is 2273 W, and the power consumed at full load (i.e., maximum I/O bandwidth) is 2302 W. That is, full-load power represents a negligible 1.3% increase over idle power. For comparison, the compute cluster consumed about 15 kW at idle and 44 kW when running our workload—a 193% increase. Clearly, the storage subsystem in a HPC data center is one of the least power-proportional components and needs improvement. Thus, even though the storage bandwidth decreased from over 150 MB/s to almost zero when we moved to an *in-situ* pipeline, we did not save any power. The lesson learned is that reducing storage bandwidth does not noticeably improve power consumption in today's HPC systems.

Looking at this from another perspective, while data movement is a significant source of power consumption, the part of the storage hierarchy affected by *in-situ* techniques moves very little data. In our cluster, the storage bandwidth (the part that gets affected) is at most 160 MB/s. In comparison, the data movement occurring within a single node is of the order of 1 TB/s. Therefore, unless the data reduction happens at the register or cache levels, we should not expect to see a decrease in overall power consumption.

Energy Figure 6 compares the energy consumption of the two visualization pipelines. Energy consumed was calculated as the product of average power and execution time for each pipeline. Because power was nearly constant, the energy consumed closely tracks execution time. The energy consumed by *in-situ* across the three sampling rates was 50%, 38% and 19% lower than by post-processing, respectively. This shows that *in-situ* can have a significant impact on the operating cost of the HPC data center as the energy cost of a data center over its lifetime is rapidly approaching its acquisition cost [31].

Storage Figure 7 compares the storage requirements of the two kinds of pipelines. With increases in storage capacity lagging behind a supercomputer's ability to generate data, it is important for HPC applications to consume no more data than is absolutely necessary. As Figure 7 indicates, *in-situ* techniques help achieve that goal by reducing the required storage from 230 GB to less than 1 GB when the sampling rate was set to record output once every 8 hours. For the other two configurations, the storage requirements decreased from 80 GB and 27 GB, respectively, to negligible amounts. In all these cases, we observed a data size reduction of over 99.5%.

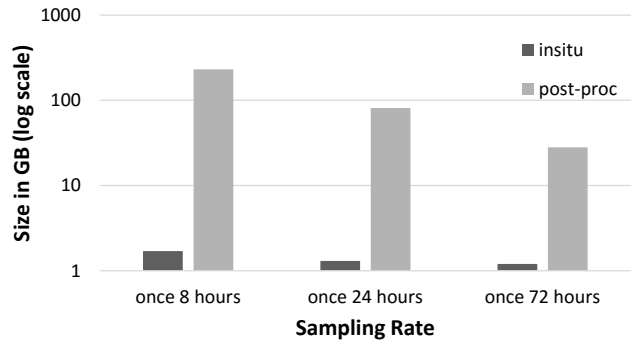


Fig. 7: Comparison of storage requirements for *in-situ* and post-processing pipelines at three different sampling rates

A. Summary of Findings

We summarize our major findings below.

- **Finding 1:** *In-situ* visualization can result in lower supercomputing time even though the supercomputer must *necessarily* run the additional task of visualization. This is due to the significant reduction in I/O wait time.
- **Finding 2:** *In-situ* techniques cannot be expected to lower the power used by the storage system or by data movement. While this may seem obvious for a HPC system with a limited storage subsystem, our finding also applies to big-data systems with larger storage. This

is due to the low dynamic power range of the storage subsystem.

- **Finding 3:** *In-situ* techniques cannot be expected to improve overall power utilization, and, in turn, harness trapped capacity.
- **Finding 4:** *In-situ* techniques can result in massive savings in energy bills for coupled scientific applications.
- **Finding 5:** *In-situ* techniques will continue to play a role in combating issues associated with limited storage in HPC systems.

In all, our findings have disproved two of our initial hypothesis about *in-situ* techniques. That is, (i) we cannot expect *in-situ* techniques to reduce the power consumption of the storage subsystem, and (ii) *in-situ* techinques cannot be relied upon to harness trapped capacity. The other hypothesis, however, holds true – *in-situ* techinques can reduce overall energy consumption.

VI. Modeling Approach

We model the performance, energy, and storage of the visualization pipelines in order to estimate those metrics at different sampling rates and application configurations. This will help in evaluating a number of scenarios and answering several what-if questions. Note that while the model derived here is architecture-specific and application-aware, the methodology itself is generic and can be applied to other computing systems and applications. The symbols used in the model are explained in Table II.

TABLE II: Summary of symbols used in our model

E	Total energy consumed by the visualization pipeline
P	Average power consumption for the visualization pipeline
t	Total execution time for a visualization pipeline
t_{sim}	Time taken by the simulation phase
$t_{i/o}$	Time taken by the I/O phase
t_{viz}	Time taken by the visualization phase
$S_{i/o}$	Size of output (in GB) produced by the simulation
N_{viz}	Number of images produced by the simulation
α	Time cost to write 1 GB of raw data; value estimated by linear solver
β	Time cost to generate one image corresponding to one timestep; value estimated by linear solver
$iter_{ref}$	Number of iterations or timesteps in the reference configuration
$iter_{any}$	Number of iterations or timesteps performed
$rate_{ref}$	Output sampling rate used in the reference configuration
$rate_{any}$	Output sampling rate for which performance metrics must be estimated
$t_{sim.ref}$	Total execution time of the reference configuration
$S_{i/o.ref}$	Size of output produced for the reference configuration
$N_{viz.ref}$	Number of images produced for the reference configuration
$S_{i/o.any}$	Estimated size of output produced by any given configuration
$N_{viz.any}$	Estimated number of images produced by any given configuration

The energy consumed by a visualization pipeline can be expressed as the product of its average power and total execution time.

$$E = P \cdot t \tag{1}$$

As observed in Figure 5, the average power across all sampling rates can be considered constant. We need to only model the execution time of the application, which can be expressed as the time taken for the simulation, I/O, and visualization phases:

$$t = t_{sim} + t_{i/o} + t_{viz} \tag{2}$$

Here, the time taken for the simulation phase, t_{sim}, is a constant for a given number of timesteps or iterations of the simulation. The time taken for I/O and visualization phases are dependent on the amount of data written and the number of images visualized, which in turn are dependent on the sampling rate. Mathematically, we can express the time taken for an application as

$$t = t_{sim} + \alpha S_{i/o} + \beta N_{viz} \tag{3}$$

in which α and β denote, respectively, the time taken to write 1 GB of output and to produce one set of images corresponding to one timestep. $S_{i/o}$ and N_{viz} are, respectively, the total output size and number of images. These values can be estimated easily for any sampling rate given a reference point as they are linearly dependent on the sampling rate.

We can express Equation 3 in a more generic form:

$$t = \frac{iter_{any}}{iter_{ref}} \times t_{sim.ref} + \alpha S_{i/o} + \beta N_{viz} \tag{4}$$

That is, the simulation time will scale with the number of iterations or timesteps in the simulation.

We use a linear solver to estimate the values of α and β. The data collected from three different configuration points, namely, (i) *in-situ*, once every 8 hours, (ii) *in-situ*, once every 72 hours, and (iii) post-processing, once every 24 hours, was used to solve for α and β. Alternatively, regression techniques may be used to solve these equations using the following system of equations:

$$\begin{aligned} t_{sim} + 0.1\alpha + 60\beta &= 676 \\ t_{sim} + 0.6\alpha + 540\beta &= 1261 \\ t_{sim} + 80\alpha + 180\beta &= 1322 \end{aligned} \tag{5}$$

Solving this system of equations gives the following values: t_{sim}=603 , α=1.2, and β=6.3. That is, it takes 603 s to perform the simulation (for six simulated months), 1.2 s to produce one image and 6.3 s to read/write 1 GB of data.

Model validation The measured execution time is plotted against the modeled execution time in Figure 8. We observe that our model predicts the execution time accurately in all cases. The absolute error rate achieved was less than 0.5%.

Given a sample rate, it is also possible to estimate the storage requirements accurately. The storage size scales linearly with the sampling rate. That is, for example, when one samples at twice the rate of a reference configuration, the storage requirements double correspondingly:

$$S_{i/o.any} = S_{i/o.ref} \times \frac{rate_{any}}{rate_{ref}} \tag{6}$$

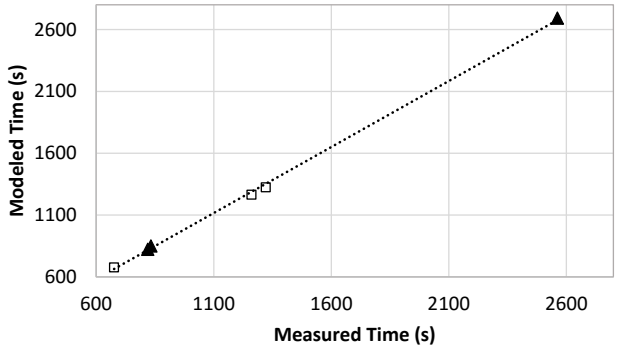


Fig. 8: Evaluation of our model for execution time. White squares denote the data points used for constructing the model. Black triangles denote the data points used in evaluation.

The data presented in Figure 7 is in agreement with Equation 6 for both the *in-situ* and post-processing cases, which validates our model for storage requirements. A similar equation is used to estimate the number of images produced during a simulation:

$$N_{viz.any} = N_{viz.ref} \times \frac{rate_{any}}{rate_{ref}} \quad (7)$$

Using our model, one could estimate the execution time, energy, and storage for any sampling rate $rate_{any}$ and timesteps $iter_{any}$ with data collected from one short run of the simulation.

VII. Application of our Model

So far, we have evaluated the different visualization pipelines at arbitrary points in the configuration space. With the model shown in Equation 4 we can evaluate many scenarios and help domain scientists optimize their application configurations given various constraints. We now present a selection of examples.

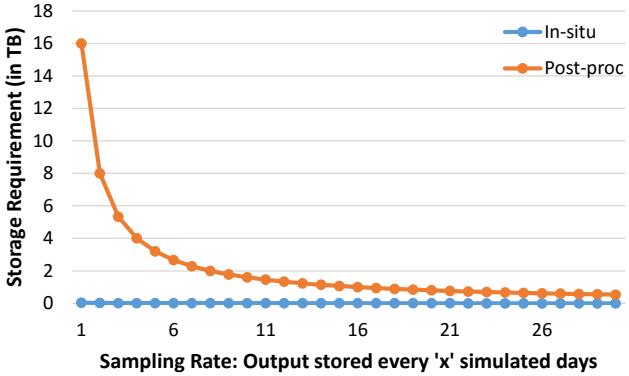


Fig. 9: **Storage vs. sampling rate.**The x-axis represents how often the output products are written to the storage system. The y-axis denotes the storage required to perform a simulation lasting 100 simulated years.

Storage vs. sampling rate Eddies in the ocean exist for hundreds of days while traveling hundreds of kilometers [27]. To effectively track their movement in the ocean, the output has to be written once per simulated day (or even hour). Climate scientists, however, are forced to make decisions on the sampling frequency due to the storage constraints they face instead of requirements imposed by the scientific phenomenon they study. Our model can be used to make these decisions for the different kinds of visualization pipelines.

Figure 9 shows how the sampling rate affects the storage required for the climate simulation application. The x axis represents how often the output products (i.e., raw data or images) are written to the storage system. The y axis denotes the storage required to perform a simulation lasting 100 simulated years. The simulated time used in this analysis is a typical time range for climate simulation. The storage size of our experimental setup is 7.7 TB. It is reasonable to expect that the storage reservation allocated for one user not exceed 2 TB. Our model shows that climate scientists can run a hundred-year simulation while storing images at a frequency of once per day or higher without any issues with respect to storage if they adopt *in-situ* visualization. However, when using a post-processing simulation, they would be forced to run the simulation at a frequency of only once every 8 days when faced with a 2 TB storage budget, as shown in Figure 9.

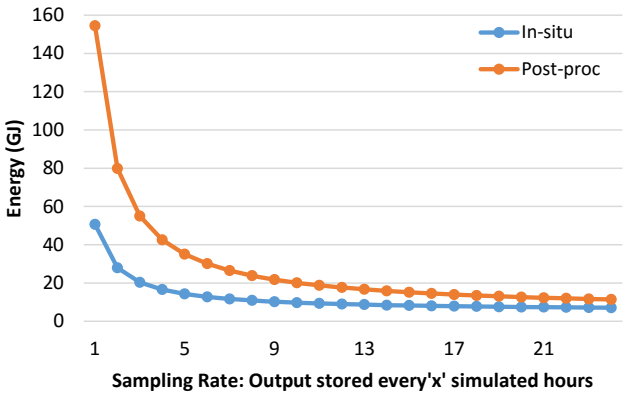


Fig. 10: **Energy vs. sampling rate.** The x axis represents how often the output products are written in terms of simulated hours. The y axis represents the energy that would be consumed while running a hundred-year simulation.

Energy vs. sampling rate Our model can evaluate the energy that is necessary to run a hundred-year simulation given a target sampling rate. Figure 10 shows the energy consumed by the two pipelines at different sampling frequencies. The x axis represents how often the output products are written in terms of simulated hours. The y axis represents the energy that would be consumed while running a hundred-year simulation. This graph can be used to evaluate the energy that can be saved from *in-situ* visualization under different sampling assumptions. Assume that the climate scientists need to track the eddies by the hour. In this case, using *in-situ* techniques

will help them save 67.2% of the energy needed for the workflow. If the required sampling rate is once every 12 hours, up to 49% energy can be saved using *in-situ* techniques. Similarly, 38% of workflow energy can be saved at a sampling rate of once per day. One could also evaluate the sampling rate possible for a given energy budget. Note that such constraints can also be specified in terms of time.

We envision our model being used in an automated framework to decide the sampling rate and the pipeline automatically depending on a given set of constraints.

VIII. Discussion

While our experiments have noted that significant energy savings are possible from *in-situ* visualization, we also identify a few scenarios in which power could be saved. Broadly speaking, we seek improvements in two areas:

1) Improving the energy proportionality of the storage subsystem.
2) Effectively managing the I/O wait states on the compute subsystem.

The storage subsystem consists of several components that are not directly related to storage (e.g., CPUs). These components consume a significant amount of power even when the subsystem is idle. Unlike the actual disks, power management techniques (e.g., DVFS) for these components are well-established. These techniques ought to be incorporated and/or enabled for the storage subsystem as well. The CPUs, for instance, should operate at the minimum frequency necessary to handle the various I/O requests from the client. Also, the design of the storage subsystem must also be rethought from a power perspective. The "brawny" CPUs on the storage side may be replaced with "wimpy" ones with little to no difference in the storage bandwidth offered by the storage subsystem.

Second, I/O-bound applications such as scientific visualization introduces a lot of I/O wait time during a simulation. These I/O wait times are typically of short duration, and several such intervals of I/O wait times occur during a simulation. Current idle period management techniques in HPC systems target only prolonged periods of idleness. With several techniques that operate at the millisecond level coming up from the computer-architecture community, it may be possible to manage idle periods during a simulation by putting the CPUs in a low-power state. Alternatively, techniques that utilize the idle periods by running a different job may be embraced. Research solutions for effectively utilizing idle periods already exist (in, for example, Legion [32]).

IX. Conclusions

This work represents, to our knowledge, the first study of power and energy consumption that considers power consumed not only by a reasonably sized compute cluster (150 nodes) but also by the corresponding storage subsystem when both are used by a scientific simulation with high I/O requirements. We measured and analyzed the performance, power, energy, and storage characteristics of two visualization pipelines—*in-situ* and post-processing—to test the conventional wisdom that *in-situ* analysis is superior on all fronts to a workflow based on data post-processing.

We draw the following two conclusions from the work presented in this paper. First, *in-situ* visualization can result in lower execution times even though a non-simulation task (i.e., visualization) is run concurrently. This is because of the significant reduction in I/O wait time resulting from an over 99.5% reduction is storage requirements. Second, despite common belief, *in-situ* techniques *cannot* be expected to reduce power consumption from reduced data movement. This is because (i) the portion of the storage hierarchy that observes reduced data movement moves only a small amount of data relative to the rest of the system, and (ii) the storage system is not power-proportional.

Our approach can inform trade-offs among energy, storage capacity, and execution time and guide the selection of a visualization pipeline—a capability becoming increasingly critical as HPC systems approach exascale.

Acknowledgment

We thank Dominic Manno for configuring our Lustre rack's power monitoring and setting up the logging system, and Mike Lang for getting us Caddy's power readings. We also thank HB Chen, Parks Fields, Jeff Kuehn, and Daryl Grunau for their help with setting up the power monitoring infrastructure.

This work was supported in part by a grant from the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) via DE-SC0012637. Los Alamos National Laboratory is operated by Los Alamos National Security LLC for the US Department of Energy under contract DE-AC52-06NA25396.

This document was approved for release under LA-UR-16-22435.

References

[1] N. Gholkar, F. Mueller, and B. Rountree, "Power Tuning HPC Jobs on Power-Constrained Systems," in *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation (PACT)*, 2016, pp. 179–191.

[2] S. Ahern, A. Shoshani, K.-L. Ma, A. Choudhary, T. Critchlow, S. Klasky, V. Pascucci, J. Ahrens, W. Bethel, H. Childs *et al.*, "Scientific Discovery at the Exascale: Report from the DOE ASCR 2011 Workshop on Exascale Data Management, Analysis, and Visualization," 2011.

[3] S. Pakin, C. Storlie, M. Lang, R. E. Fields, E. E. Romero, C. Idler, S. Michalak, H. Greenberg, J. Loncaric, R. Rheinheimer *et al.*, "Power Usage of Production Supercomputers and Production Workloads," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 2, pp. 274–290, 2016.

[4] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale Computing Technology Challenges," in *Proceedings of the International Confernce on High Performance Computing for Computational Science (VECPAR)*, 2011.

[5] 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 2013 International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2013, pp. 77:1–77:12.

[6] L. Kwan-Ma, "Runtime Volume Visualization for Parallel CFD," Tech. Rep., 1995.

[7] R. Haimes, "Concurrent Distributed Visualization and Solution Steering," *Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers*, vol. 5, 1995.

[8] S. Crivelli, O. Kreylos, B. Hamann, N. Max, and W. Bethel, "ProteinShop: a tool for interactive protein manipulation and steering," *Journal of Computer-Aided Molecular Design*, vol. 18, no. 4, pp. 271–285, 2004.

[9] N. Fabian, K. Moreland, D. Thompson, A. Bauer, P. Marion, B. Geveci, M. Rasquin, and K. Jansen, "The ParaView Coprocessing Library: A Scalable, General Purpose In Situ Visualization Library," in *Proceedings of the 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, Oct 2011, pp. 89–96.

[10] B. Whitlock, J. M. Favre, and J. S. Meredith, "Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System," in *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2011, pp. 101–109.

[11] M. Franklin, A. Halevy, and D. Maier, "From Databases to Dataspaces: A New Abstraction for Information Management," *ACM SIGMOD Record*, vol. 34, no. 4, pp. 27–33, 2005.

[12] J. F. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, and C. Jin, "Flexible IO and Integration for Scientific Codes Through the Adaptable IO System (ADIOS)," in *Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments*, 2008, pp. 15–24.

[13] 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 *Proceedings of the 2012 International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012, pp. 49:1–49:9.

[14] J. Biddiscombe, J. Soumagne, G. Oger, D. Guibert, and J.-G. Piccinali, "Parallel Computational Steering for HPC Applications Using HDF5 Files in Distributed Shared Memory," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 852–864, 2012.

[15] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen, "An Image-based Approach to Extreme Scale in Situ Visualization and Analysis," in *Proceedings of the 2014 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014, pp. 424–434.

[16] 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 Applications*, vol. 30, no. 3, pp. 45–57, May 2010.

[17] H. Karimabadi, B. Loring, P. O'Leary, A. Majumdar, M. Tatineni, and B. Geveci, "In-situ Visualization for Global Hybrid Simulations," in *Proceedings of the 2013 Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery (XSEDE)*, 2013, pp. 57:1–57:8.

[18] V. Vishwanath, M. Hereld, and M. Papka, "Toward Simulation-time Data Analysis and I/O Acceleration on Leadership-class Systems," in *Proceedings of the 2011 IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, Oct 2011, pp. 9–14.

[19] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling In-situ Execution of Coupled Scientific Workflow on Multi-core Platform," in *Proceedings of the 26th IEEE International Parallel Distributed Processing Symposium (IPDPS)*, May 2012, pp. 1352–1363.

[20] G. Haldeman, I. Rodero, M. Parashar, S. Ramos, E. Z. Zhang, and U. Kremer, "Exploring Energy-Performance-Quality Tradeoffs for Scientific Workflows with In-situ Data Analyses," *Computer Science-Research and Development*, pp. 1–12, 2014.

[21] 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 *Proceedings of the 20th International Conference on High Performance Computing (HiPC)*, Dec 2013, pp. 226–235.

[22] I. Rodero, M. Parashar, A. G. Landge, S. Kumar, V. Pascucci, and P.-T. Bremer, "Evaluation of In-Situ Analysis Strategies at Scale for Power Efficiency and Scalability," in *Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 2016, pp. 156–164.

[23] V. Adhinarayanan, W.-c. Feng, J. Woodring, D. Rogers, and J. Ahrens, "On the Greenness of In-Situ and Post-Processing Visualization Pipelines," in *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*. IEEE, 2015, pp. 880–887.

[24] *Appro GreenBlade 2 SR5110–GB512X User Manual*, Appro International, Milpitas, California, USA, Nov. 2011, version 1.0.

[25] T. R. Scogland, C. P. Steffen, T. Wilde, F. Parent, S. Coghlan, N. Bates, W.-c. Feng, and E. Strohmaier, "A Power-measurement Methodology for Large-scale, High-performance Computing," in *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE)*. ACM, 2014, pp. 149–159.

[26] T. Ringler, M. Petersen, R. L. Higdon, D. Jacobsen, P. W. Jones, and M. Maltrud, "A Multi-Resolution Approach to Global Ocean Modeling," *Ocean Modelling*, vol. 69, no. 0, pp. 211 – 232, 2013.

[27] J. Woodring, M. Petersen, A. Schmeiber, J. Patchett, J. Ahrens, and H. Hagen, "In Situ Eddy Analysis in a High-Resolution Ocean Climate Model," *IEEE Transactions on Visualization Computer Graphics*, vol. 22, no. 1, pp. 857–866, 2015.

[28] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. Rogers, and M. Petersen, "An Image-based Approach to Extreme Scale in Situ Visualization and Analysis," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014, pp. 424–434.

[29] A. C. Bauer, B. Geveci, and W. Schroeder, "The paraview catalyst users guide," 2013.

[30] J. Ahrens, "Increasing scientific data insights about exascale class simulations under power and storage constraints," *IEEE Computer Graphics and Applications*, vol. 35, no. 2, pp. 8–11, Mar 2015.

[31] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *2013 IEEE international symposium on workload characterization (IISWC)*, 2013, pp. 56–65.

[32] M. Bauer, S. Treichler, E. Slaughter, and A. Aiken, "Legion: Expressing locality and independence with logical regions," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012, pp. 66:1–66:11.