2015 IEEE International Parallel and Distributed Processing Symposium Workshop

On the Greenness of In-Situ and Post-Processing Visualization Pipelines

Vignesh Adhinarayanan*, Wu-chun Feng*, Jonathan Woodring†, David Rogers†, James Ahrens†
*Department of Computer Science, Virginia Tech, Blacksburg, Virginia 24060
†Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, New Mexico 87545
{avignesh, wfeng}@vt.edu, {woodring, dhr, ahrens}@lanl.gov

***Abstract*—Post-processing visualization pipelines are traditionally used to gain insight from simulation data. However, changes to the system architecture for high-performance computing (HPC), dictated by the exascale goal, have limited the applicability of post-processing visualization. As an alternative, in-situ pipelines are proposed in order to enhance the knowledge discovery process via "real-time" visualization. Quantitative studies have already shown how in-situ visualization can improve performance and reduce storage needs at the cost of scientific exploration capabilities. However, to fully understand the trade-off space, a head-to-head comparison of power and energy (between the two types of visualization pipelines) is necessary.**

Thus, in this work, we study the greenness (i.e., power, energy, and energy efficiency) of the in-situ and the post-processing visualization pipelines, using a proxy heat-transfer simulation as an example. For a realistic I/O load, the in-situ pipeline consumes 43% less energy than the post-processing pipeline. Contrary to expectations, our findings also show that only 9% of the total energy is saved by reducing off-chip data movement, while the rest of the savings comes from reducing the system idle time. This suggests an alternative set of optimization techniques for reducing the power consumption of the traditional post-processing pipeline.

I. Introduction

The computational demand of high-performance computing (HPC) applications has brought major changes to the HPC system architecture. As a result, it is now possible to run simulations faster and get more accurate results. But, the recent changes to the HPC systems solve just one part of the problem - getting high quality data from simulations. What is arguably more important, is the insight behind the data.

Scientists visualize the simulation data to understand a complex physical phenomenon and solve real-world problems. A typical visualization pipeline involves simulating a phenomenon, and writing the simulation data to a disk. After the simulation is complete, the data is sent to a rendering farm, where it is visualized and interpreted to draw meaningful conclusions. This kind of pipeline, known as the post-processing visualization pipeline, has been popular for a long time.

However, the changes dictated by the exascale goal has affected the traditional post-processing visualization pipelines. Faster processors have encouraged scientists to perform larger simulations, producing more simulation data, which cannot be handled by the slower I/O. Also, off-chip data movement is now estimated to consume nearly hundred times as much energy as on-chip movement [1]. This trend is concerning, particularly for exascale simulations, as the U.S. Department of Energy's (DOE) goal is to support exascale systems under a maximum power budget of 20 MW [2]. These trends, combined, has resulted in a paradigm shift, away from post-processing visualization.

To continue knowledge discovery via visualization, two major challenges must to be addressed. First, the I/O bottleneck, which affects the performance, should be overcome. Next, the energy consumption of visualization pipelines should be limited. In order to solve both these problems, researchers have advocated in-situ data analytics and visualization, where the data is processed/visualized alongside the simulation [3]. This reduces the total amount of off-chip transfers, thereby avoiding the I/O bottleneck and reducing the energy consumption. On the flip side, scientists lose their ability to perform exploratory analysis when they use in-situ techniques.

Researchers have studied several in-situ pipelines and have found them to be better than post-processing pipelines, in terms of storage requirements and performance. However, to the best of our knowledge, there exists no studies that quantitatively compares the power and energy consumption of the two types of visualization pipelines. Improvements in performance and reduction in storage requirements are assumed to automatically translate into energy and power savings. We seek to determine the magnitude of these savings, to fully understand the advantages and the disadvantages of both the pipelines. Our major contributions include the following:

- We provide a subsystem-level characterization of instantaneous power for in-situ and post-processing pipelines using a proxy heat transfer application configured for different I/O loads.
- We provide a direct comparison of the two pipelines, in terms of the following metrics: performance, power, energy consumption, and energy efficiency.
- We provide a breakdown of the energy savings from the in-situ approach. We estimate the energy saved by reducing data movement and by reducing idle time.
- We present a hypothetical case where an alternative set of techniques applied to the post-processing pipeline will consume nearly the same amount of energy as the in-situ pipeline.

Our major findings are presented below. Please note that our findings are based on the study of a single proxy application

978-1-4673-7684-6/15 $31.00 © 2015 IEEE
DOI 10.1109/IPDPSW.2015.132

in a limited setting. The application was run on a single node, using a traditional hard disk, local to the node.

- Energy saved by adopting in-situ visualization is as high as 43% for a proxy application configured for a realistic I/O load.
- As much as 91% of the energy is saved by minimizing the time spent idling, while only 9% of the energy is saved by actually avoiding off-chip accesses.
- Using *fio* benchmark, we show cases where the I/O bottleneck and the associated power overhead could be overcome even in the post-processing pipelines, using techniques such as software-directed data reorganization.

While the scope of this study is limited, it will be expanded to real applications running on state-of-the art HPC systems in the future.

The rest of the paper is organized as follows. Section II covers background on visualization pipelines, the target application, and energy measurement. Related work is presented in Section III. The experimental setup is described in Section IV. The results are discussed in Section V. We present our future work and conclude in Section VI.

II. Background

In this section, we explain our proxy application and the different pipelines studied. We also provide some background on the power monitoring capability of our target system.

A. Proxy Heat-Transfer Simulation

Our proxy application performs a heat transfer simulation based on the finite-element method, similar to the methods described by Reddy and Gartling [4]. This method solves a partial-differential equation by performing a series of stencil computations. The application operates on a three-dimensional grid. In each timestep, the temperature of every element in the grid is computed from its six face-centered cubic neighbors. At the end of each timestep, either the raw data or the image representation of the data can be stored on the disk. The image representation of the z-plane at timesteps 2000 and 10000 is shown in Figure 1. The color of each point in the grid represents its temperature.

The size of the grid (in MB), is given as input to the application. This size also represents the amount of data that is written at the end of each timestep. The application creates an NxNxN grid corresponding to the input size and simulates the transfer of heat within the grid. The user can also control the size of the chunks in which data is written to and read from the disk. To control the time spent in simulation and disk I/O, the user can also specify how often the simulation data is read and written.

The application supports the following modes

- **heat:** Performs a heat transfer simulation based on stencil computations.
- **nnread:** Performs disk read operations. The read requests go to sequential locations.
- **nnwrite:** Performs disk write operations. The write requests go to sequential locations.

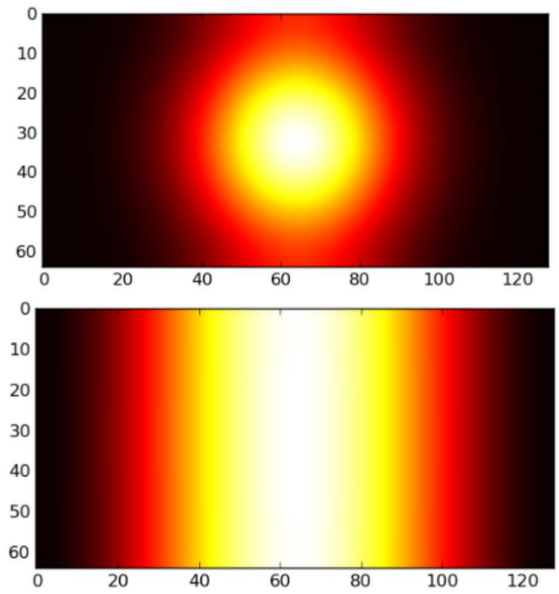


Fig. 1: Heatmap visualization of the z-plane at timesteps 2000 and 10000

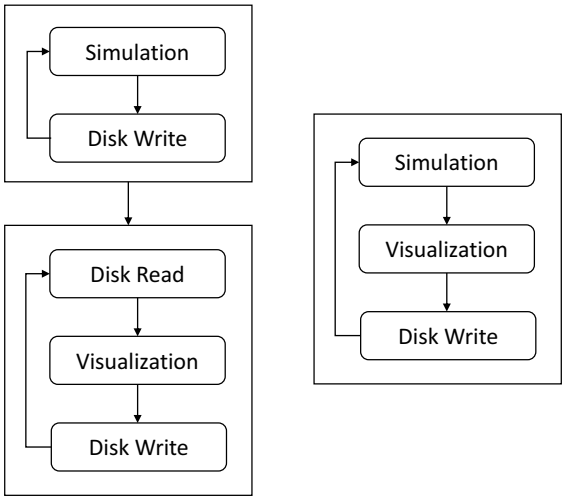
- **heat+nnwrite:** Performs *heat* and *nnwrite* in succession. This corresponds to the first phase of a post-processing pipeline.
- **nnread+image:** Performs *nnread* and visualization in succession. This corresponds to the second phase of a post-processing pipeline.
- **heat+image:** Performs *heat* and visualization in succession. This corresponds to an in-situ pipeline.

B. Visualization Pipelines

In an in-situ pipeline, the simulation and the visualization of the simulated data are co-located in the same machine, running side by side. In a post-processing pipeline, data is visualized after the simulation, either in the same machine or in a separate rendering machine. This involves one or more off-chip transfers, either to the disk, or to the network, or both. Within the context of this definition, several pipelines are possible. In this section, we explain what we mean by a post-processing and an in-situ pipeline.

Figure 2a shows the post-processing pipeline. We simulate heat transfer in a 3-D grid and write the raw data to the disk at the end of every iteration. After the simulation is complete, we read the entire data back from the disk, create image representations of the grid, and write the images back to the disk. The images may later be used to compose a movie clip.

Figure 2b shows the in-situ pipeline. We start with the simulation. At the end of each timestep, we generate an image which is a reduced representation of the raw data and write the generated image to the disk.



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

Fig. 2: Different types of visualization pipelines

C. Power Monitoring

In our experiments, we collect the power consumed by the sub-components of the CPU through Intel's Running Average Power Limit (RAPL) interface [5]. RAPL was introduced in Intel Sandy Bridge systems and provides power-limiting and energy-monitoring capabilities. We make use of the energy-monitoring feature to obtain a component's power profile. The underlying mechanism is described as follows. The RAPL interface reads values from model-specific registers (MSR) that are available in the hardware to monitor the system activities of three components, namely PP0 (core), package (processor), and DRAM (memory). Using a pre-validated model, RAPL estimates the energy consumed by the three components. The average power consumption for any time slice is then computed from the corresponding energy estimates. The estimated power values closely track true power consumption, with an average error rate of less than 1% [5]. Of particular interest to us, is the power consumed by the package and the DRAM.

III. Related Work

In-situ visualization pipelines have been explored for a long time. Originally conceived as a way to enable scientists to monitor their simulations [6], in-situ visualization is now being adopted to overcome performance bottlenecks associated with large I/O operations. Numerous in-situ algorithms [7], [8], applications [9]–[14], and frameworks [15], [16] have been developed.

Tu et al. couple the simulation and visualization components of a finite-element simulation of earthquake to overcome scalability bottlenecks of traditional approaches [17]. Yu et al. present an in-situ approach for jet-lifted combustion simulation [9]. Kariamadi et al. present an in-situ visualization pipeline for electron fluid and kinetic ions simulations in order to study the effect of solar wind on planetary bodies [11]. Ahrens et al. present an image-based approach for interactive in-situ visualization and apply it to MPAS-Ocean, an unstructured-mesh simulation of oceans [12]. Many other applications have their own in-situ implementations [13], [14].

With visualization-based approaches proving to be popular for knowledge discovery, a number of visualization frameworks such as ParaView [15], VisIT [16], DataSpaces [18], and ADIOS [19] have been developed to quickly build in-situ and post-processing visualization pipelines. Bennett et al. combine in-situ and in-transit techniques using DataSpaces and ADIOS frameworks to analyze data from S3D, a massively parallel turbulent combustion simulation [10]. Biddiscombe et al. use ParaView to build and evaluate their application [20]. All these studies have shown that in-situ approaches have better I/O characteristics and performance. Techniques such as *data sampling* [21], [22] and *data triage* [23] have been developed to further improve the performance.

The metrics of interest in all the above studies are performance/speed and storage size. Studies characterizing the power and energy behavior of in-situ pipelines are limited. Recently, Gamell et al. looked at the performance and energy trade-offs of an in-situ combustion simulation in a large-scale system [24]. Haldeman et al. explored the energy-performance-quality tradeoffs of different data movement strategies applicable to in-situ pipelines [25]. Gamell et al. evaluated the energy and performance behaviors of in-situ and in-transit pipelines on an NVRAM-based deep memory hierarchy systems [26]. While the above works evaluate power and energy consumption of in-situ pipelines, none of them provide a direct comparison with post-processing pipelines to help truly understand the energy, performance, and quality trade-offs. In this work, we provide a direct comparison of in-situ and post-processing pipelines in terms of performance, power, energy, and energy efficiency using a proxy heat-transfer simulation as an example.

IV. Experimental setup

In this section, we describe the hardware platform, the setup for power monitoring, and the different configurations of the application used in the study.

A. Hardware Platform

The system under test contains a dual-socket Intel Sandy Bridge, where each socket contains an 8-core Intel Xeon E5-2665 CPU (for a total of 16 cores in the node). It has 64 GB of DDR3 memory, and a Seagate 500GB 7200rpm HDD. This system runs Ubuntu 12.04 operating system with GNU/Linux 3.2.0-23 kernel. Other details of the hardware platform is given in Table I.

B. Setup for Power Monitoring

The power measurements come from two different sources, as shown in Figure 3. The system under test is connected to a Wattsup Pro power meter which is connected to a power outlet. The power meter provides system-wide power measurements

TABLE I: Hardware specification

H/W Type	H/W Detail
CPU	2x Intel Xeon E5-2665
CPU frequency	2.4 GHz
Last-level cache	20 MB
Memory	4x 16GB DDR3-1333
Memory size	64 GB
Hard disk	Seagate 7200rpm disk
Storage size	500GB
Disk bandwidth	6.0 Gbps

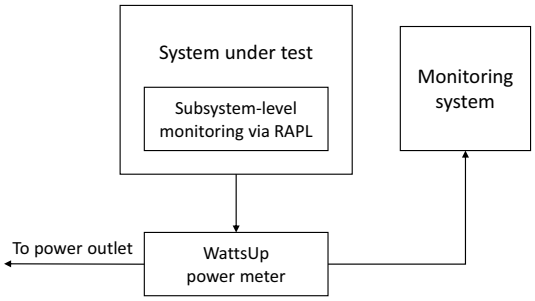


Fig. 3: Power monitoring setup

at a frequency of 1 Hz, i.e., one reading per second. A different monitoring system collects the power measurements through a USB interface and writes them to its local disk. This approach minimizes the error in the application's power profile as there is no interference from a monitoring process and additional data writes to the disk.

Simultaneously, the Intel Sandy Bridge CPU provides power measurements for the components within the CPU via the RAPL interface. This measurement cannot be monitored by a different machine. To reduce the interference, we set the monitoring resolution to 1 Hz even though RAPL provides measurements at a frequency of over 1 KHz. At this low resolution, the power consumption increases by 0.2 W on an average, which is negligible. We collect the processor's power consumption (package power) and DRAM power consumption using this interface. Power consumption of the rest of the system, which includes the hard disk, network, motherboard, and fans, is estimated by subtracting the processor power and the DRAM power from the full-system power obtained using the Wattsup Pro meter.

C. Application Configuration

Three different configurations of the proxy application were used in our study. In all three cases, the application is run for fifty iterations or timesteps. The grid size and the chunk size were fixed at 128 KB for all the cases. For case study #1, I/O operations and visualization is performed in every iteration. For case study #2, it is done every alternate iteration, and for case study #3, every eighth iteration. This experiment is done to show the impact of I/O time on energy savings. In all these cases, we perform a *sync* operation and drop the caches between phases. This ensures that the data does not get cached in memory and is actually written to the disk.

Fig. 4: Percentage of execution time spent in simulation, disk writes, disk reads, and visualization for the three cases

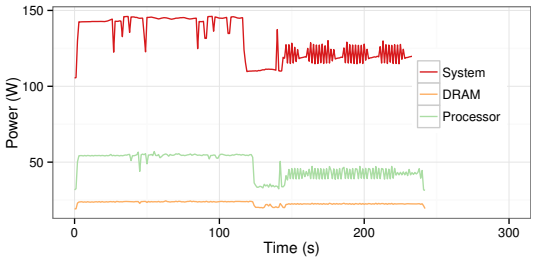
The break up of execution time for the three cases is shown in Figure 4. The percentage of execution time spent in simulation, write, read, visualization stages are 33%, 30%, 27%, and 10%, respectively, for case study #1. These values are not very different from some real applications where over 70% of the total time is spent in I/O operations [27]. The corresponding values for case study #2 are 50%, 22%, 21%, and 7%; For case study #3, the values are 80%, 9%, 8%, and 3%.

V. Results and Discussion

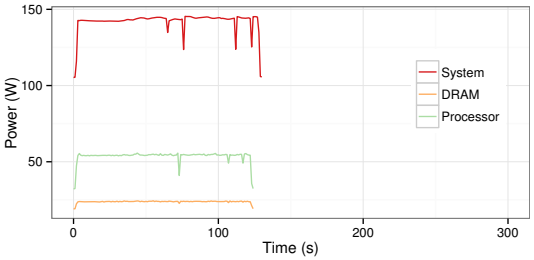
In this section, we first present the power profile of the three different instances of the in-situ and the post-processing pipelines for the proxy heat-transfer simulation. Then, we characterize and compare the two pipelines in terms of the following metrics: performance, power, energy, and energy efficiency. Next, we provide a breakdown of the energy savings and discuss its implications on the power optimization of visualization pipelines.

A. Power Profiles

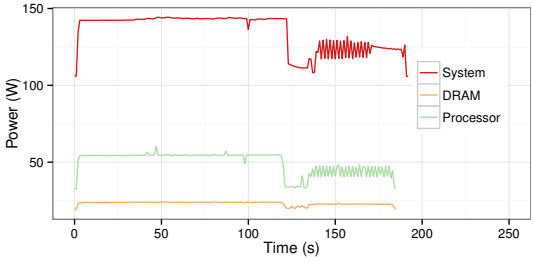
Figures 5a, 5c, 5e and Figures 5b, 5d, 5f show the power profile of post-processing and in-situ pipelines, respectively, for the three different application configurations presented in Section IV-C. These graphs present the instantaneous power consumed by the processor, the memory, and the full system over time. Power profiles for the post-processing pipeline, shown in Figures 5a, 5c, 5e, indicate the presence of distinct power phases in the application. The first major phase, in which the simulation is performed and the data is written to the disk, consumes about 143 W of power on an average. The second major phase, where the simulation data is read back from the disk and visualized, consumes about 121 W of power on an average. Since the average power consumed by the reads and the writes is nearly the same as shown in Figure 6, we can infer that the simulation phase consumes 22 W more power



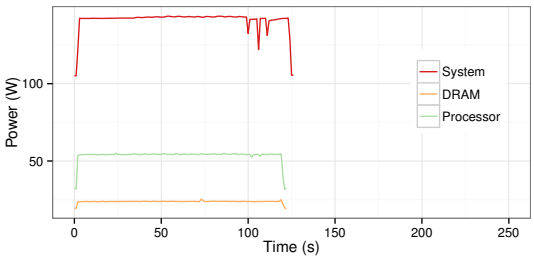
(a) Power profile of post-processing pipeline for case study #1



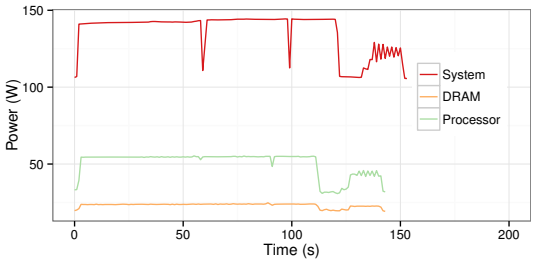
(b) Power profile of in-situ pipeline for case study #1



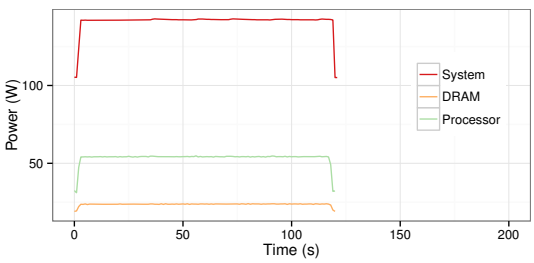
(c) Power profile of post-processing pipeline for case study #2



(d) Power profile of in-situ pipeline for case study #2



(e) Power profile of post-processing pipeline for case study #3



(f) Power profile of in-situ pipeline for case study #3

Fig. 5: Power consumed by processor, memory, and full system over time for post-processing and in-situ pipelines

than the visualization phase. We also observe that there are no distinct power phases for the in-situ pipeline.

Another observation from Figure 5 is that power consumed by the memory subsystem is significantly lower than the processor. Since the maximum power consumed by the disk is also low (about 15 W), we expect that the energy overhead from off-die data movement is not high. We quantitatively demonstrate this in another set of experiments.

Performance, power, energy, and energy-efficiency values are derived from the power profiles to compare the two types of visualization pipeline.

B. Comparison of Pipelines

Figure 7 shows the execution time of the post-processing and the in-situ pipelines for three different problem sizes. The execution time of the in-situ pipelines were 92%, 52%, and 26% lower than the post-processing pipelines for the three problem sizes, respectively.

Figure 8 and Figure 9 shows the average power and the peak power consumption for the post-processing and in-situ pipelines. There is no significant difference in the peak power, which is an important metric for power-capped systems. The in-situ pipelines consumed 8%, 5%, and 3% more power on an average.

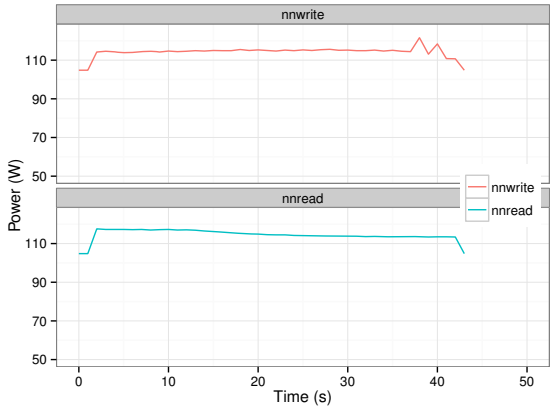


Fig. 6: Power profile of nnread and nnwrite stages

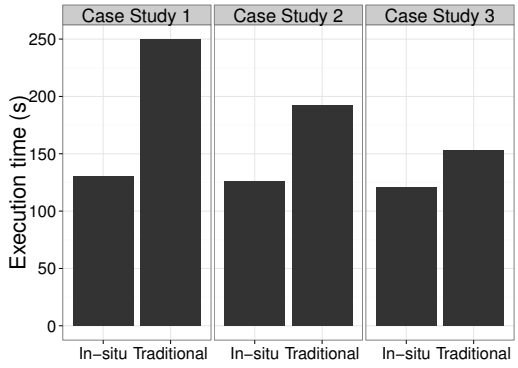


Fig. 7: Execution time of post-processing and in-situ pipelines

Fig. 9: Peak power of post-processing and in-situ pipelines

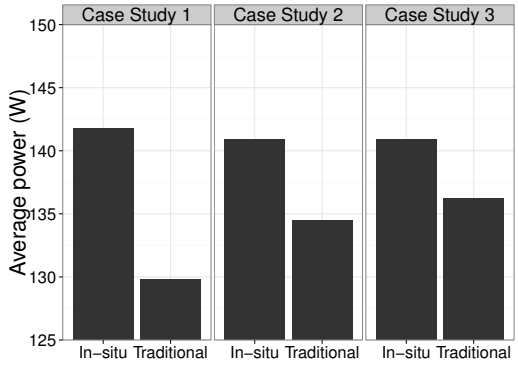


Fig. 8: Average power of post-processing and in-situ pipelines

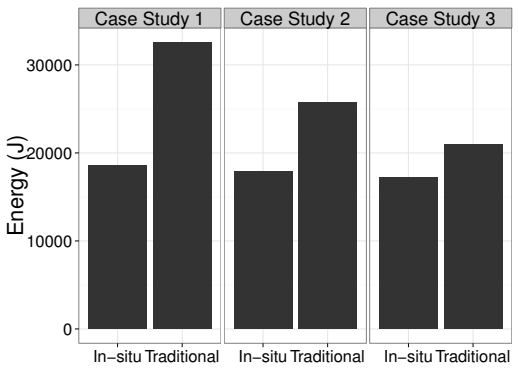


Fig. 10: Energy consumption of post-processing and in-situ pipelines

Energy consumption, which is the integral of instantaneous power over time, is presented in Figure 10. Even though the average power is higher for in-situ pipeline, its energy consumption is 43%, 30%, and 18% lower than post-processing pipeline. This is because of the significantly lower execution time. The improvement in energy-efficiency from adapting an in-situ pipeline varies from 22% to 72% depending on the time spent in I/O operations (Figure 11).

Overall, the in-situ pipelines are greener than post-processing pipelines. However, the advantages in energy efficiency tapers out as the time spent in I/O lowers. Since many real-world simulation-visualization applications spend a substantial amount of time doing disk I/O [27]–[29], their energy-efficiency improvement will be significant if they adapt in-situ pipelines.

C. Energy Savings Breakdown

Figure 10 showed that in-situ pipelines saves as much as 43% of the total energy consumed by post-processing pipeline. In this section, we show the breakdown in energy savings, showing how much energy was saved by (i) reducing data transfers, and (ii) reducing idle time by reducing data transfers. In the first case, the energy savings come from the dynamic component where the power is consumed due to data accesses. In the second case, the savings come from reducing the static component of energy. It is important to make this distinction because the power optimization techniques used for the two situations will be vastly different. If the source of energy savings is significant for the dynamic component, *data sampling* technique is preferred, which may result in loss of useful information. If the energy savings mostly come from the static component, other techniques such as frequency scaling and data rearrangement may help.

To estimate the energy savings breakdown, we first obtain the power profile for nnread and nnwrite stages of our proxy application, which is shown in Figure 6. From the profile, we extract relevant metrics, namely average total power consumption, and average dynamic power consumption for the two stages. This information is shown in Table II. The dynamic energy savings is calculated by multiplying the average dynamic

TABLE II: Properties of nnread and nnwrite stages

Metric	nnread	nnwrite
Avg. Power (Total)	115.1	114.8
Avg. Power (Dynamic)	10.3	10.0

TABLE III: Performance, power, and energy consumption for the *fio* tests

Metric	Sequential Read	Random Read	Sequential Write	Random Write
Execution time (s)	35.9	2230.0	27.0	31.0
Full-system power (W)	118	107	115.4	117.9
Disk dynamic power (W)	13.5	2.5	10.9	13.4
Disk dynamic energy (KJ)	0.4	5.5	2.9	0.4
Full-system energy (KJ)	4.2	238.6	3.1	3.6

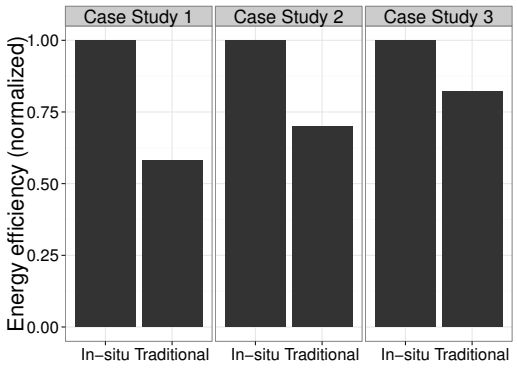


Fig. 11: Energy efficiency of post-processing and in-situ pipelines

power shown in Table II with the corresponding time spent, i.e. the difference in execution time between in-situ and post-processing pipelines shown in Figure 7. The static savings are computed by subtracting the dynamic savings computed from the total energy savings shown in Figure 10. For case study #1, the energy saved by avoiding idling (from static sources) is 12.8 KJ, and the energy saved by reducing data accesses is 1.2 KJ. That is, as much as 91% of the energy is saved by avoiding system idling.

D. Discussion

Our test application performs I/O operations sequentially. This is not always the case in real applications, where I/O operations may occur in a random fashion. To account for this, we consider the sequential and random tests from the *fio* disk benchmark. We read and write 4 GB of data to sequential and random locations in the disk using this benchmark. The power, energy, and execution time for these cases are shown in Table III. For an application exhibiting random I/O behavior, we could save 242.2 KJ (238.6 KJ+3.6 KJ) of energy by adopting in-situ visualization. However, we will lose the capability for exploratory analysis. But, if we were to adopt data-rearrangement techniques [30], [31] on the post-processing pipeline, we will lose out only 7.3 KJ (4.2 KJ+3.1 KJ) of energy, instead of 242.2 KJ, while at the same time retaining all of the exploratory analysis capabilities. This presents many interesting possibilities for reducing the power consumption of the traditional post-processing pipelines without having to lose out on exploratory analysis.

VI. Conclusion and Future Work

In this study, we found that the energy saved by adopting in-situ visualization is as high as 43% on a proxy application configured for realistic I/O load. We also found that as much as 91% of the energy savings comes from reducing the system idle time. Comparatively, only 9% of the energy is saved by reducing off-chip accesses. Using *fio* benchmark, we showed cases where the I/O bottleneck and the associated power overhead could be overcome even in the post-processing pipelines when techniques such as software-directed data reorganization is used.

A. Future Work

One of the major limitations of this work is that our tests are based on only one application, running on a single node, using traditional hard disks and file system. In future, we plan to work on the following:

- Evaluation of real-world applications such as MPAS [32] and xRAGE [33].
- Evaluation on a multi-node system to study the effect of network I/O in addition to disk I/O.
- Evaluation on systems using RAID disks, solid-state drives, and other flash-based devices such as NVRAM.
- Evaluation on multi-node systems running parallel file systems to understand the impact of file system on energy consumption.
- We would also like to develop a runtime system that makes use of our characterization studies. Such work would entail the development of power models that estimates the hard disk power based on the number of disk accesses, size of each access, and the corresponding access pattern. Using this model, the runtime will decide the power optimization technique to be used.

Acknowledgment

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 and by infrastructure provided by Supermicro. This paper is a Los Alamos Unclassified Release LA-UR-15-21414.

References

[1] S. R. Sachs, K. Yelick *et al.*, "Exascale Programming Challenges," *2011 Workshop on Exascale Programming Challenges*, 2011.

[2] R. Lucas *et al.*, "Top Ten Exascale Research Challenges," *DOE ASCAC Subcommitte Report, February*, 2014.

[3] 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.

[4] J. N. Reddy and D. K. Gartling, *The Finite Element Method in Heat Transfer and Fluid Dynamics*. CRC press, 2010.

[5] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory Power Estimation and Capping," in *Proceedings of the 2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, Aug 2010, pp. 189–194.

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

[7] H. Childs, M. Duchaineau, and K.-L. Ma, "A Scalable, Hybrid Scheme for Volume Rendering Massive Data Sets," in *Proceedings of the 6th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 153–161.

[8] H. Yu, C. Wang, and K.-L. Ma, "Massively parallel volume rendering using 2-3 swap image compositing," in *International Conference on 2008 High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2008, pp. 1–11.

[9] 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.

[10] 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)*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 49:1–49:9.

[11] 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)*. New York, NY, USA: ACM, 2013, pp. 57:1–57:8.

[12] 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)*. Piscataway, NJ, USA: IEEE Press, 2014, pp. 424–434.

[13] 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.

[14] 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.

[15] 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.

[16] 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)*, 2011, pp. 101–109.

[17] T. Tu, H. Yu, L. Ramirez-Guzman, J. Bielak, O. Ghattas, K.-L. Ma, and D. O'Hallaron, "From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing," in *Proceedings of the 2006 International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, Nov 2006, pp. 12–12.

[18] 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.

[19] 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*. ACM, 2008, pp. 15–24.

[20] 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.

[21] J. Woodring, J. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann, "In-situ Sampling of a Large-scale Particle Simulation for Interactive Visualization and Analysis," in *Proceedings of the 13th Eurographics / IEEE - VGTC Conference on Visualization (EuroVis)*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2011, pp. 1151–1160.

[22] C. Wang, H. Yu, and K.-L. Ma, "Application-Driven Compression for Visualizing Large-Scale Time-Varying Data," *IEEE Computer Graphics and Applications*, vol. 30, no. 1, pp. 59–69, 2010.

[23] ——, "Importance-Driven Time-Varying Data Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1547–1554, Nov 2008.

[24] 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 Tradeoffs of In-situ Data Analytics," in *Proceedings of the 2013 International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*. New York, NY, USA: ACM, 2013, pp. 77:1–77:12.

[25] 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.

[26] 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.

[27] T. Peterka, H. Yu, R. Ross, and K.-L. Ma, "Parallel Volume Rendering on the IBM Blue Gene/P," in *Proceedings of the 8th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2008, pp. 73–80.

[28] R. B. Ross, T. Peterka, H.-W. Shen, Y. Hong, K.-L. Ma, H. Yu, and K. Moreland, "Visualization and Parallel I/O at Extreme Scale," *Journal of Physics: Conference Series*, vol. 125, no. 1, p. 012099.

[29] T. Peterka, H. Yu, R. Ross, K.-L. Ma, and R. Latham, "End-to-End Study of Parallel Volume Rendering on the IBM Blue Gene/P," in *Proceedings of the 2009 International Conference on Parallel Processing (ICPP)*, Sept 2009, pp. 566–573.

[30] Y. Zhang, J. Liu, and M. Kandemir, "Software-Directed Data Access Scheduling for Reducing Disk Energy Consumption," in *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2012, pp. 596–605.

[31] S. W. Son and M. Kandemir, "Integrated Data Reorganization and Disk Mapping for Reducing Disk Energy Consumption," in *Proceedings of the 2007 IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*. IEEE, 2007, pp. 557–564.

[32] 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.

[33] M. Gittings, R. Weaver, M. Clover, T. Betlach, N. Byrne, R. Coker, E. Dendy, R. Hueckstaedt, K. New, W. R. Oakes, D. Ranta, and R. Stefan, "The RAGE radiation-hydrodynamic code," *Computational Science and Discovery*, vol. 1, no. 1, p. 015005, 2008.