Making a Case for Green High-Performance Visualization via Embedded Graphics Processors

Position Paper

Vignesh Adhinarayanan
Dept. of Computer Science
Virginia Tech
Blacksburg, VA 24061
avignesh@vt.edu

Bishwajit Dutta
Dept. of Electrical & Computer Engineering
Virginia Tech
Blacksburg, VA 24061
bdutta@vt.edu

Wu-chun Feng
Dept. of Computer Science
and Electrical & Computer Engineering
Virginia Tech
Blacksburg, VA 24061
wfeng@vt.edu

***Abstract*—This paper makes a case for using low-power embedded GPUs for the purpose of executing high-performance scientific visualization tasks. We compare the *greenness* (i.e., power, energy, and energy-delay product → EDP) of an embedded GPU with a CPU for commonly encountered visualization tasks using two real-world applications: (1) Modeling for Prediction Across Scale – Ocean (MPAS-O) and (2) Particular Ensembles (PE). Our preliminary results show that the low-power embedded GPU is capable of handling complex visualization tasks while consuming less than 50% of the energy consumed by a CPU server. In addition, we find that the embedded GPU outperforms the CPU with dynamic voltage-frequency scaling (DVFS) enabled in a majority of the cases.**

I. Introduction

Power and energy have emerged as first-order design criteria in high-performance computing (HPC) systems. For example, the U.S. Department of Energy seeks to build an exascale system with a power budget of 20 MW [17]. Operating such a machine at the planned power budget would incur an energy bill that is one-quarter of its acquisition cost even if we consider the least expensive power in the U.S. (≈ $0.05/kWh) [10]. Under a more typical energy cost (≈ $0.10/kWh) and a realistic projection for power (≈ 40 MW in 2020 [2]), the energy bill for an exascale supercomputer would be as high as its acquisition cost. As a consequence, researchers should explore opportunities to reduce the energy consumption of high-performance supercomputing systems while still delivering high performance — in other words, improved greenness (i.e., power, energy, and energy efficiency). In this paper, we make a case for greenness on an important class of HPC applications, namely, high-performance scientific visualization.

A recent trend in high-performance scientific visualization seeks to move away from post-processing visualization to in-situ visualization due to I/O constraints [5]. In a traditional post-processing pipeline (as shown in Fig. 1), the scientific simulation runs on the primary CPU cluster. The CPU cluster then transfers the generated simulation data to the storage cluster periodically. After the simulation completes, the data is then transferred to a visualization cluster where images are rendered using discrete graphics processing units (GPUs). In contrast, an in-situ pipeline avoids the data transfer into and out of the storage cluster by running both the simulation and visualization on the primary CPU cluster. Recent studies have shown that this approach can save as much as 55% energy by minimizing I/O-related stalls [6], [4].

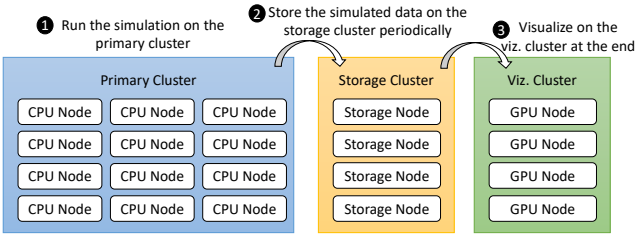


Fig. 1. Traditional post-processing visualization pipeline

However, such an in-situ approach would result in losing the ability to run the visualization task on a processor that is specialized for visualization (i.e., GPU). An obvious solution is to employ graphics processing units (GPUs) on the primary cluster as well. However, if cost is an issue, data centers may restrict (or reduce) the availability of discrete GPUs for application scientists. In a post-processing visualization pipeline, only the visualization nodes (which are fewer in number) need graphics-processing capabilities, thus using a more expensive discrete GPU for this post-processing pipeline is a viable and cost-effective option. For in-situ visualization, however, the visualization task runs on the *same* nodes as the simulation. Thus, graphics-processing capabilities must be provided on *each* node, which may not be viable from a cost perspective.

In this paper, we make a case for modifying the in-situ pipeline to run visualization tasks on low-power embedded graphics processors. Due to the shared processor and motherboard designs for desktop and server computing, embedded graphics processors are readily available on CPU nodes (but reserved for driving the display on a desktop computer). For instance, recent server-based Intel Skylake CPUs, such as the Intel HD Graphics P530 and Intel Iris Pro Graphics P580,

come equipped with embedded graphics, thereby reducing the need for discrete GPUs [3]. As a consequence, these embedded graphics processors do *not* incur any additional hardware cost if used for visualization tasks. Furthermore, they operate at a *fraction* of the thermal power envelope of discrete GPUs and thus do not consume any significant energy (or power) when the visualization task is not running. Our preliminary experiments with two real-world scientific visualization applications — (1) Modeling for Prediction Across Scale – Ocean (MPAS-O) and (2) Particular Ensembles (PE) — show that these embedded GPUs can handle complex scientific visualization tasks comfortably while consuming less than 50% of the energy consumed by the CPUs.

II. Related Work

The problem of improving the *greenness* of scientific visualization pipelines has received recent attention due to the increasing energy cost of data movement [17]. The solutions previously explored can be classified into four categories.

- **In-situ Techniques:** These techniques reduce data movement by processing most of the data when it resides in memory rather than allowing it to *move* to the storage subsystem. Specific examples include an image-based approach to scientific visualization [5], in-situ sampling [4], in-situ analytics [8], and data compression [7].
- **Work Distribution:** The basic premise here is to carefully redistribute the simulation and visualization tasks across the various resources (e.g., nodes or cores) to minimize data movement [16]. For instance, the two tasks may be distributed to dedicated nodes (or cores) or the same set of resources may be used for the two tasks, but in a time-shared manner, which affects intra- and inter-node data movement and hence affects energy.
- **Memory Optimizations:** New types of memory such as NVRAM exhibit lower data-access energy. Some research papers have looked at optimizing the memory hierarchy to reduce the energy consumption of scientific visualization [9], [11].
- **DVFS-based Techniques:** The frequency of the cores and the other components of a compute node are varied to tune power and performance. Labasan et al. have looked at both DVFS [13] and techniques built atop DVFS such as power sloshing [14].

In contrast to the above work, we explore an *underutilized* compute component inside a node (i.e., embedded GPU) and compare its efficacy versus classical green computing techniques such as DVFS.

III. Methodology

In this paper, we perform and present some preliminary experiments to characterize the *greenness* of different types of devices for some commonly encountered visualization tasks via two real-world applications: Modeling for Prediction Across Scale - Ocean (MPAS-O) [15] and Particular Ensembles (PE) [1]. Here, *greenness* may refer to power, energy, or the energy-delay product (EDP). The visualization tasks proceed in four steps, as illustrated in Fig. 2:

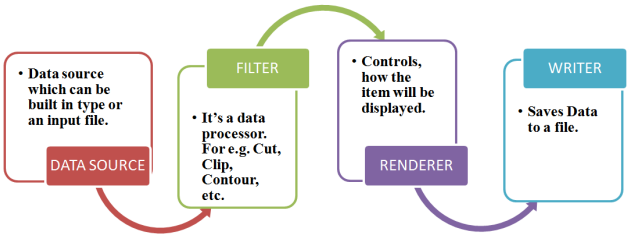


Fig. 2. Four stages of the visualization pipeline

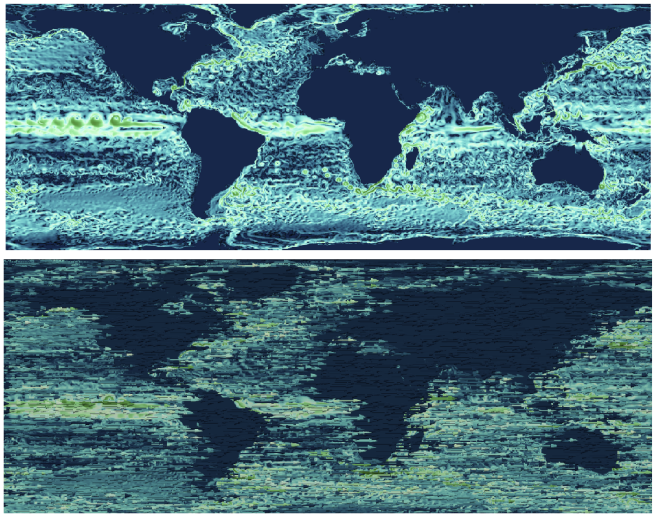


Fig. 3. Glyph filter applied to the MPAS-O data set

- **Data Sources:** The raw data is represented as structured grids for MPAS-O and a point cloud for PE and is read as a binary file from disk.
- **Filter Operations:** Filters are functional units that process the input data to extract interesting features and form the core of the visualization pipeline. The visualization tasks evaluated in this paper include *clip*, *contour*, *glyph*, and *slice*. In addition, we studied *stream tracer* and *warp by filter* which are relevant only to PE.
- **Rendering:** Rendering is the process of generating an image from a 2D or 3D model. In this paper, we use volume rendering for the PE data set and surface rendering for the MPAS-O data set. Fig. 3 shows a sample surface rendering for MPAS-O.
- **Writers:** The final output is written as a raster graphics image file onto the disk.

We performed our experiments on a SuperMicro workstation equipped with two 8-core Intel Xeon E5-2665 processors and a Matrox G200 graphics processor, which is integrated

onto the motherboard. Relevant details of the hardware platform are presented in Table I.

TABLE I
HARDWARE DETAILS

	CPU	GPU
Device	2x Intel Xeon E5-2665	Matrox G200
Operating Frequency	1200 MHz - 2400,MHz	84 MHz
Memory	64 **GB** DDR3	32 **MB** SGRAM
TDP	230 W (i.e., 115 W*2)	4 W

We measured the power consumption of these platforms using a *WattsUp Pro* power meter. All experiments were repeated three times and their median values reported in the subsequent section. Because the CPU supports dynamic voltage and frequency scaling (DVFS), we also ran our experiments at different frequencies, ranging from 1200 MHz to 2400 MHz in steps of 200 MHz, and compared the CPU's *greenness* versus that of the embedded GPU. For software, we used ParaView v5.0.1 compiled with MESA libraries for the CPU and OpenGL libraries for the embedded GPU.

IV. PRELIMINARY RESULTS

Table II presents the execution time, dynamic power, dynamic energy, and EDP for the embedded GPU for various visualization tasks and data sets. The values presented in this table are normalized with respect to the CPU running at its base frequency of 2.4 GHz.

TABLE II
NORMALIZED EXECUTION TIME, DYNAMIC POWER, ENERGY AND EDP FOR THE EMBEDDED GPU.

Visualization Task	Data Set	Time	Power	Energy	EDP
Clip	MPAS-O	0.37	0.05	0.02	0.01
	PE	4.61	0.10	0.48	2.20
Contour	MPAS-O	4.21	0.09	0.37	1.57
	PE	5.37	0.24	1.27	6.79
Glyph	MPAS-O	1.15	0.10	0.11	0.13
	PE	1.01	0.06	0.01	0.01
Slice	MPAS-O	4.62	0.08	0.37	1.65
	PE	4.76	0.15	0.69	3.29
Stream Tracer	PE	3.65	0.17	0.57	2.09
Wrap By Vector	PE	4.87	0.14	0.66	3.23

We clearly see that there exists a trade-off between performance and power for the two devices. While the CPU exhibits better performance for nearly all of our visualization tasks, the GPU consistently consumes less power. This trade-off results in widely different energy and EDP characteristics for the two devices. The embedded GPU consumes lower energy than the CPU in *nine out of ten* cases whereas the CPU exhibits a lower EDP in *seven out of ten* cases. Moving forward, we expect power and energy to have a greater weight in the $E^m D^n$ energy-delay model [12]. Therefore, we expect embedded GPUs to be increasingly useful for high-performance visualization in the future.

Next, we investigate the extent to which DVFS affects the performance and greenness of the devices. Results are only presented for the PE data set due to space limitations. Fig. 4 presents the performance, power, energy, and EDP for the CPU using DVFS with different filters, respectively. In particular, Fig. 4(a) shows mixed results for visualization performance in that the DVFS-controlled CPU (solid line) outperforms the embedded GPU (dotted line) on four of the six filters: clip, contour, slice, and stream tracer, while the embedded GPU convincingly outperforms the CPU for the glyph and wrap-by-vector filters. Fig. 4(b) shows the power consumed by the CPU at various frequencies for different filters. Note that the GPU frequency cannot be changed and is fixed in these experiments. While DVFS helped reduce CPU's power consumption significantly for all six filters, the embedded GPU remained the *greener* device in terms of power.

While previous experiments suggest that the embedded GPU consumed lower energy than the CPU for all visualization tasks, if we reduce the operating frequency of the CPU, we can make the CPU into the *greener* device with respect to energy, as in the case with the *slice* and *wrap-by-vector* filters (as shown in Fig. 4(c)). However, DVFS tuning for CPUs may also negatively affect their EDP as observed in the case of the *stream tracer* filter (as shown in Fig. 4(d)). Overall, our results show a complex interplay between the various parameters under study and suggest that the embedded GPUs deserve further investigation for high-performance scientific visualization.

V. CONCLUSION

In this paper, we evaluated the *greenness* of CPU and the embedded GPU across four different metrics—performance, power, energy, and EDP. We found that the embedded GPU outperforms the CPU in terms of power and energy even when DVFS is enabled on the CPU. On the other hand, the CPU does better in terms of performance and energy. Moving forward, with power and energy becoming first-class citizens, we believe embedded GPUs will form a major part of in-situ visualization pipelines. While much work is still yet to be done in the area of energy-efficient high-performance visualization, we believe our initial work in this paper spurs interest in low-power embedded GPUs for scientific visualization.

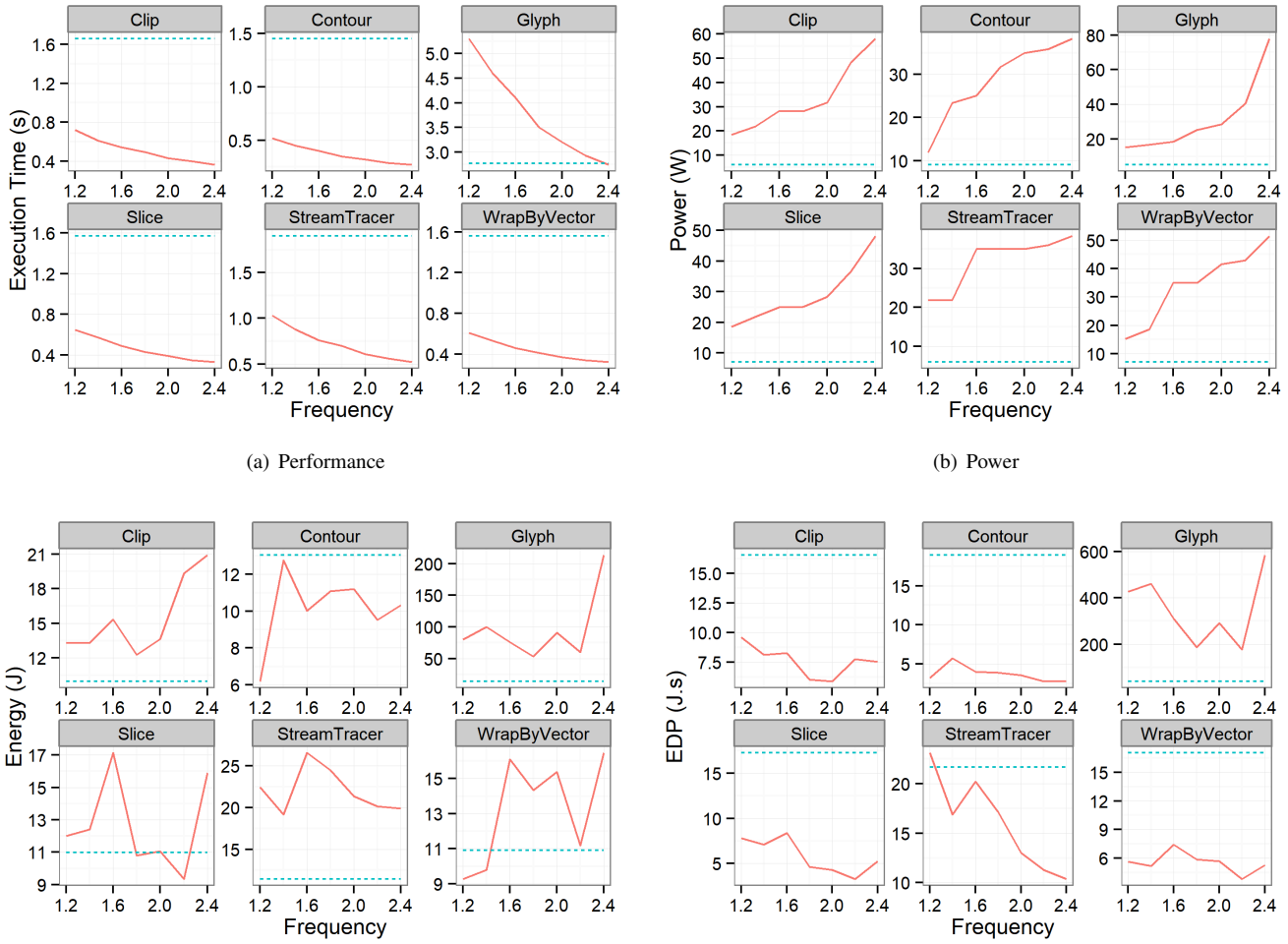
In the future, we plan to expand the initial characterization presented in this paper to include more applications, more visualization filters, and more hardware devices including discrete GPUs. We believe such a study will help visualization scientists in making greener and less expensive simulations in the context of high-performance computing (HPC).

ACKNOWLEDGEMENTS

This work is supported by a grant from the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) via DE-SC0012637.

REFERENCES

[1] Particular Ensembles. http://www.uni-kl.de/sciviscontest/, 2016.
[2] The Next Generation of HPC Technology Dominates Green500. *Scientific Computing World*, Nov 2016.



(a) Performance

(b) Power

(c) Energy

(d) EDP

Fig. 4. Greenness results for PE data set at different CPU frequencies. Solid line = CPU, Dotted Line = GPU

[3] Intel HD Graphics P530 & Iris Pro Graphics P580 Performance Guide. https://www.intel.com/content/dam/www/public/us/en/documents/guides/hd-graphics-p530-p580-performance-guide.pdf, 2017.

[4] V. Adhinarayanan, W. Feng, D. Rogers, J. Ahrens, and S. Pakin. Characterizing and Modeling Power and Energy for Extreme-Scale In-situ Visualization. In *IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, 2017.

[5] V. Adhinarayanan, W. Feng, J. Woodring, D. Rogers, and J. Ahrens. On the Greenness of In-Situ and Post-Processing Visualization Pipelines. In *IEEE Int'l Parallel & Distributed Processing Symp. Workshops*, 2015.

[6] V. Adhinarayanan, S. Pakin, D. Rogers, W. Feng, and J. Ahrens. Performance, Power, and Energy of In-Situ and Post-Processing Visualization: A Case Study in Climate Simulation. In *SC15*.

[7] A. S. Berres, V. Adhinarayanan, T. Turton, D. Rogers, and W. Feng. A Pipeline for Large Data Processing Using Regular Sampling for Unstructured Grids. Technical report, Los Alamos National Laboratory (LANL), 2017.

[8] M. Gamell, I. Rodero, M. Parashar, J. C. Bennett, H. Kolla, J. Chen, P.-T. Bremer, A. G. Landge, A. Gyulassy, P. McCormick, et al. Exploring Power Behaviors and Trade-offs of In-situ Data Analytics. In *IEEE/ACM Int'l Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2013.

[9] 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 *IEEE Int'l Conference on High Performance Computing (HiPC)*, 2013.

[10] N. Gholkar, F. Mueller, and B. Rountree. A Power-aware Cost Model for HPC Procurement. In *IEEE Int'l Parallel & Distributed Processing Symposium Workshops (IPDPSW)*, 2016.

[11] 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*, 30(2):207–218, 2015.

[12] C.-H. Hsu, W. Feng, and J. S. Archuleta. Towards Efficient Supercomputing: A Quest for the Right Metric. In *IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, 2005.

[13] S. Labasan, M. Larsen, and H. Childs. Exploring Tradeoffs Between Power and Performance for a Scientific Visualization Algorithm. In *IEEE Symp. on Large Data Analysis and Visualization (LDAV)*, 2015.

[14] S. Labasan, M. Larsen, H. Childs, and B. Rountree. PaViz: A Power-Adaptive Framework for Optimizing Visualization Performance. In *EuroGraphics Symposium on Parallel Graphics and Visualization (EGPGV)*, 2017.

[15] 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*, 69(0):211 – 232, 2013.

[16] 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 *IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2016.

[17] J. Shalf, S. Dosanjh, and J. Morrison. Exascale Computing Technology Challenges. In *Int'l Conference on High Performance Computing for Computational Science (VECPAR)*, 2010.