Reducing Data Movement Costs using Energy-Efficient, Active Computation on SSD

Devesh Tiwari [1], Sudharshan S. Vazhkudai [2], Youngjae Kim [2], Xiaosong Ma [1,2], Simona Boboila [3], and Peter J. Desnoyers [3]
[1]North Carolina State University, [2]Oak Ridge National Laboratory, [3]Northeastern University
{devesh.dtiwari, ma}@ncsu.edu, {vazhkudaiss, kimy1}@ornl.gov, {simona, pjd}@ccs.neu.edu

ABSTRACT

Modern scientific discovery often involves running complex application simulations on supercomputers, followed by a sequence of data analysis tasks on smaller clusters. This offline *approach suffers from significant data movement costs such as redundant I/O, storage bandwidth bottleneck, and wasted CPU cycles, all of which contribute to increased energy consumption and delayed end-to-end performance. Technology projections for an exascale machine indicate that energy-efficiency will become the primary design metric. It is estimated that the energy cost of data movement will soon rival the cost of computation. Consequently, we can no longer ignore the data movement costs in data analysis.*

To address these challenges, we advocate executing data analysis tasks on emerging storage devices, such as SSDs. Typically, in extreme-scale systems, SSDs serve only as a temporary storage system for the simulation output data. In our approach, Active Flash*, we propose to conduct* in-situ *data analysis on the SSD controller without degrading the performance of the simulation job. By migrating analysis tasks closer to where the data resides, it helps reduce the data movement cost. We present detailed energy and performance models for both active flash and offline strategies, and study them using extreme-scale application simulations, commonly used data analytics kernels, and supercomputer system configurations. Our evaluation suggests that active flash is a promising approach to alleviate the storage bandwidth bottleneck, reduce the data movement cost, and improve the overall energy efficiency.*

1 Introduction

High performance computing (HPC) simulations on large-scale supercomputers (e.g., the petascale Jaguar machine, No. 6 on the Top500 list [19]) routinely produce vast amounts of result output and checkpoint data [1, 7]. Examples of such applications include astrophysics (Chimera and Vulcan/2D), climate (POP), combustion (S3D), and fusion (GTC and GYRO) (Table 1.) Deriving insights from these simulation results, towards scientific discovery, often involves performing a sequence of *data analysis* operations.

Application	Analysis data generation rate (per node)	Checkpoint data generation rate (per node)
CHIMERA	4400 KB/s	4400 KB/s
VULCUN/2D	2.28 KB/s	0.02 KB/s
POP	16.3 KB/s	5.05 KB/s
S3D	170 KB/s	85 KB/s
GTC	14 KB/s	476 KB/s
GYRO	14 KB/s	11.6 KB/s

Table 1: Output characteristics of leadership-class parallel simulations, amortized over the entire run. These are calculated based on ORNL Jaguar reports [1, 7].

Traditionally, the simulation jobs and data analyses of the simulation outputs are conducted on different computing resources. Data analysis tasks are run *Offline*, on smaller clusters, *after* the completion of the simulation job (Figure 1(a).) The high-end computing (HEC) machine and the analysis clusters tend to share

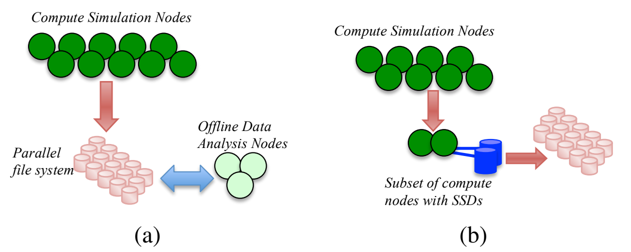

Figure 1: (a) Offline approach to data analysis (b) Data analysis using Active Flash

a high-speed scratch parallel file system (PFS) to read/write input/output data. For example, the machine-room at Oak Ridge National Lab (ORNL) hosts several data analysis clusters that share a PFS with the Jaguar machine. The reason for using *offline* clusters for data analysis is that CPU hours are expensive on HEC machines like Jaguar. Therefore, HPC users generally utilize the allocated CPU hours on HEC machines for highly FLOP-intensive codes such as the simulation job, instead of data analysis tasks.

Unfortunately, this traditional offline approach suffers from both performance and energy inefficiencies. It requires redundant I/O, resulting in excessive data movement across the compute and storage subsystems, which delays the process of scientific discovery. As we transition from petaflop to exaflop systems, the energy cost due to data movement will be comparable, if not more than the computing cost [2]. At the same time, technology projections indicate that energy efficiency will become the primary metric for system design since we need to increase the computing power by 1000x with only 10x increase in the power envelope [17]. Therefore, a key problem to solve is how to expedite the data analysis process in an energy-efficient manner, without degrading or interfering with the main simulation computation.

One promising alternative is to perform *"in-situ"* data analysis [15] on in-transit output data, before it is written to the PFS. Although this eliminates redundant I/O, it uses expensive compute nodes on HEC machines for the relatively less FLOP-intensive data analysis tasks. Moreover, it may even slowdown the simulation job due to interference from data analysis tasks, resulting in potentially inefficient use of expensive resources.

Although "in-situ" data analysis is promising, using HEC nodes for data analysis results in an inefficient utilization of these resources. We observe that emerging storage devices such as Solid-state devices (SSD) have significant computing power on the storage controllers [8]. Fortunately, SSDs are being deployed on large-scale machines such as Tsubame2 [19] and Gordon [16] due to their higher I/O throughput, and are likely to be an integral part of the storage subsystem in next-generation supercomputers. Therefore, we propose to exploit the compute power in SSDs, for in-situ data analysis, which we call *Active Flash.* The active flash approach also has the potential to enable energy-efficient data analysis as the SSDs are equipped with low-power, ARM-based, multi-core controllers. Additionally, computation near the storage reduces the data movement cost, resulting in energy savings.

Figure 1(b) depicts the active flash approach, where data processing is conducted on the collective SSD space, which forms a *staging area*. Such a staging area on HPC machines will be primarily used as a burst buffer, to temporarily hold the output and checkpoint data, before draining them to the PFS. Simulation output data is composed of *analysis data* and *checkpoint data*. Analysis data is the input to post-processing, and data analytics kernels. On the other hand, checkpoint data essentially saves the system state of the simulation at periodic intervals, so that the application can recover from failures, to a previous stable state.

The active flash approach shares the same design philosophy behind active disks [14, 10], which had several technological trends against it. For example, the storage bandwidth bottleneck and the data movement costs are a bigger concern in the current post-petascale era when compared to the gigascale era during active disks. These factors, coupled with SSD's superior capabilities, make a strong case for active flash .

In this paper, we present detailed energy consumption and performance models for the offline and active flash cases. Our model investigates (1) under what conditions is it feasible to offload data analytics on to the SSD controllers, and (2) when is it more energy-efficient to place data analysis on the SSD, without degrading or interfering with the simulation job. We show that active flash can execute many popular data analytics kernels in the background, without degrading simulation performance. In addition, it improves energy-efficiency and accelerates scientific discovery, compared to the offline approach. We analyze the performance/energy tradeoffs of both approaches, using data rates of several leadership HPC applications, different classes of data analysis kernels, and realistic system parameters from the production Jaguar machine. It should be noted that SSDs will not be deployed on future HPC machines specifically for in-situ data analysis, but instead to alleviate the pressure on storage and memory subsystems. However, by providing a mechanism to piggyback computation on the SSD, we argue that the active flash approach reduces the energy consumption involved in data movement, for the HPC center at-large. Another important point to note is that performing active computation on the idle SSD controller is not the same as buying low-power processors. The reason is two fold. First, SSD controllers themeselves are similar to contemporary low-power cores, so we do not need to buy new ones. Instead we need to be able to exploit the SSD controller while it is idle. Second, active computation on the controller also mitigates redundant I/O, unlike processing on low-power cores.

In our prior work [11], we had introduced active flash , and analyzed the tradeoffs in performing some tasks on node-local SSD. Kim et al. [12] presented initial results and experience for performing scanning operations on the SSD controller for a single node case. In contrast, we evaluate and model the active computation paradigm on SSDs in an extreme-scale computing environment for a class of high-end simulation jobs and data analysis kernels. Also, in contrast to previous work, we are unique in considering the case where data is flowing from the compute nodes to the storage system, and hence the active computation needs to be completed within a certain duration to avoid slowing down the data producer.

2 Performance and Energy Model for Data Analysis

In this section, we estimate the energy consumption for the offline and active flash data analysis approaches. We first determine a realistic ratio between the compute nodes and the SSDs, based on capacity and bandwidth constraints, major factors considered while deploying SSD's in HPC environments. Our goal is to *piggyback the active computation of the analysis data on to the SSD controller, while it is on the SSD-based staging area*. Based on the actual ratios, derived from real application data rates, as well as the analysis operation's throughput on the SSD controller, we assess the energy savings due to in-situ data analysis on such devices.

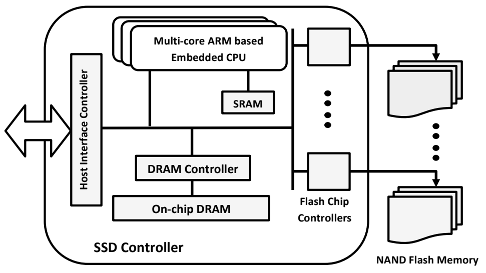

Figure 2: **Detailed view of the SSD controller**

Performance Feasibility of Active Flash: First, we develop an analytical model to examine the feasibility of active flash, by assessing the aggregate processing power from distributed SSD devices. This is based on the realistic assumption that SSDs are likely to be deployed on only a subset of the compute nodes on large machines. Given the enormous size of state-of-the-art supercomputers, it may not be cost-effective to deploy SSDs on each and every compute node. Therefore, we first calculate how many SSDs need to be provisioned for the staging area, to accommodate the peak output data volume at any given point, considering the data generation and consumption (both analytics and I/O) speeds.

The total SSD storage requirement is determined by the following factors: (1) the per-compute-node data production rate for both analysis data and checkpoint data, denoted as λ_a and λ_c respectively, (2) the length of an output iteration (the time between two periodic output operations), t_{iter}, (3) the total number of compute nodes in the system, N, and (4) the number of checkpoints one would like to keep in the staging area, num_{chkpts}.

Taking into account a certain over-provisioning factor, f_{op}, for the SSD space, the total SSD storage requirement at any given time can be estimated as $f_{op} \cdot (\lambda_a + num_{chkpts} \cdot \lambda_c) \cdot N \cdot t_{iter}$. Therefore, the number of SSDs can be estimated as the total SSD storage requirement divided by the capacity of one SSD, C_{SSD}. Consequently, the *maximum staging ratio* (the maximum number of compute nodes sharing one SSD device) based on the capacity requirement can be calculated as:

$$S_{capacity} = \frac{C_{SSD}}{f_{op} \cdot (\lambda_a + num_{chkpts} \cdot \lambda_c) \cdot t_{iter}} \qquad (1)$$

The SSD drive itself will be chosen based on many factors such as IOPS/\$, GB/\$, BW/\$, write-endurance, and C_{SSD}.

$S_{capacity}$ describes the static space constraint. However, the shared SSD store needs to simultaneously satisfy an *I/O bandwidth* constraint: bandwidth to the SSD storage cannot be lower than the PFS bandwidth. Let us suppose the host-to-SSD interface bandwidth is $BW_{host2ssd}$. Then, the maximum staging ratio should be:

$$S_{bandwidth} = \frac{N}{BW_{PFS}} \cdot BW_{host2ssd} \qquad (2)$$

Therefore, considering both the capacity and bandwidth constraints, the final staging ratio is determined as follows:

$$S = \min(S_{capacity}, S_{bandwidth}) \qquad (3)$$

Given an SSD deployment plan based on the staging ratio (derived as above) and a simulation application, we would like to decide which data analytics kernels can be offloaded to SSDs, without delaying the main simulation computation.

Note that only the analysis data from the HPC simulation needs to processed by the SSD controller and not the checkpoint data. This processing involves transferring the data from the flash memory to the flash chip controller at the SSD internal bandwidth BW_{fm2c}, plus transferring from the flash chip controller to the on-device DRAM at bandwidth, BW_{c2m} (refer to Fig. 2 for a detailed view of the SSD controller.) We use T_{SSD_k} to denote the data processing throughput of a single active flash device, running the data analysis kernel, k. Certain data analytics kernels (e.g., compression) may even reduce the analysis data by some factor α. The SSD is responsible for writing the analysis data (possibly reduced) and checkpoint data to the PFS (with an appropriate share of the aggregate file system bandwidth BW_{PFS}). This way, we have the processing plus output time for data generated within unit time:

$$t_a = \lambda_a\left(\frac{1}{BW_{fm2c}} + \frac{1}{BW_{c2m}} + \frac{1}{T_{SSD\_k}} + \frac{\alpha}{\frac{S}{N} \cdot BW_{PFS}}\right) \quad (4)$$

and checkpoint data output time

$$t_c = \frac{\lambda_c}{\frac{S}{N} \cdot BW_{PFS}} \quad (5)$$

Although the draining of the staged checkpoint data to the PFS can be overlapped with the processing of the analysis data, we conservatively assume that the processing and the final draining of the analysis data are carried out on the same SSD controller core, implying that the checkpoint data draining and data analytics happen sequentially. Also, the result of analysis needs to be stored or visualized in-situ as well. However, for simplicity we assume that the result of data analysis is much smaller than the output data itself (i.e. α cannot be larger than one.)

For the active flash approach to be feasible, the device needs to process/output the data from the S nodes before the next I/O iteration, i.e., consuming data at a rate faster than its generation:

$$(t_a + t_c) \cdot S < 1 \quad (6)$$

Solving inequality 6, we obtain the minimum throughput required for the analytics kernels that can be placed on the flash device:

$$T_{SSD\_k} > \frac{\lambda_a \cdot S}{1 - \lambda_a \cdot S \cdot \left(\frac{1}{BW_{fm2c}} + \frac{1}{BW_{c2m}}\right) - \frac{N \cdot (\alpha \cdot \lambda_a + \lambda_c)}{BW_{PFS}}} \quad (7)$$

Later in Section 4, we use it to evaluate the performance feasibility of active flash for a set of representative large-scale applications and data analytics kernels.

Energy Model for Active Flash: In our model, we account for the energy consumed by all SSDs in the staging area, for the entire duration of the application run. It is the sum of the energy consumed by the SSDs during (1) busy time: transferring data from compute nodes to the SSD ($E_{node2ssd}$), processing the analysis data ($E_{activessd}$), and transferring the data to the PFS ($E_{ssd2pfs}$), and (2) idle time ($E_{idlessd}$).

Let P^{SSD}_{busy} and P^{SSD}_{idle} be the SSD busy and idle power level, respectively, and t_{sim} be the total simulation computation time in an application run. $E_{node2ssd}$ can be derived by considering the total time taken to transfer the data from the compute nodes to some SSD in the staging area as: $\frac{S \cdot (\lambda_a + \lambda_c)}{BW_{host2ssd}}$, where $BW_{host2ssd}$ is the host to SSD interface bandwidth. Since there are a total of $\frac{N}{S}$ SSDs, the total energy cost for the data transfer can be expressed as follows:

$$E_{node2ssd} = P^{SSD}_{busy} \cdot \frac{N \cdot (\lambda_a + \lambda_c)}{BW_{host2ssd}} \cdot t_{sim} \quad (8)$$

Similarly, the energy consumed by the data analysis is:

$$E_{activessd} = P^{SSD}_{busy} \cdot N \cdot \lambda_a \cdot \left(\frac{1}{BW_{fm2c}} + \frac{1}{BW_{c2m}} + \frac{1}{T_{SSD\_k}}\right) \cdot t_{sim} \quad (9)$$

After data analysis tasks, both checkpoint and analysis data are written to the PFS by $\frac{N}{S}$ SSDs. Each SSD writes $S \cdot (\alpha \cdot \lambda_a + \lambda_c)$ amount of data at a bandwidth of $S \cdot BW_{PFS}/N$. This amounts to the following energy cost:

$$E_{ssd2pfs} = P^{SSD}_{busy} \cdot \frac{N^2 \cdot (\alpha \cdot \lambda_a + \lambda_c)}{S \cdot BW_{PFS}} \cdot t_{sim} \quad (10)$$

Finally, the idle energy consumption can be calculated using the estimated total idle time, utilizing the busy time estimate above:

$$E_{idlessd} = P^{SSD}_{idle} \cdot \left(\frac{N}{S} \cdot t_{sim} - \frac{E_{activessd} + E_{ssd2pfs}}{P^{SSD}_{busy}}\right) \quad (11)$$

Note that the busy time involved in $E_{node2ssd}$ is not subtracted in the equation above, unlike the other two components. This is because both CPUs and SSDs are involved in the data transfer, which is not a part of t_{sim}, and does not perform data generation.

Interestingly, a faster I/O bandwidth to the SSD may help reduce the I/O time at the compute nodes (currently the primary motivation to deploy SSDs in HPC systems), since cores in a compute node are idle during I/O activities. Consequently, a reduced I/O time lowers the idle time at the compute nodes.

We estimate the per-node idle time reduction as $T_{iosaving} = \frac{N \cdot (\lambda_a + \lambda_c)}{BW_{PFS}} - \frac{S \cdot (\lambda_a + \lambda_c)}{BW_{SSD}}$. Therefore, the energy savings at all compute nodes due to this can be written as:

$$E_{iosaving} = N \cdot T_{iosaving} \cdot P^{server}_{idle} \quad (12)$$

After taking into account all of the components, energy consumption at all SSDs can be expressed as follows:

$$\begin{aligned} E_{SSD} = E_{node2ssd} + E_{activessd} + E_{ssd2pfs} \\ + E_{idlessd} - E_{iosaving} \end{aligned} \quad (13)$$

Performance and Energy Modeling for Offline Processing: With the offline approach, compute nodes on the analysis cluster need to read only the analysis data and write the potentially reduced (by a factor of α) analysis data according to their share of the whole data, i.e., $\frac{N \cdot \lambda_a}{M}$, assuming M nodes are used to perform offline analysis. Similarly, each of the M nodes need to process the data at the processing rate, T_{server_k}, for a given kernel k. Again we assume that each node gets the appropriate share of the PFS bandwidth $\frac{1}{N} \cdot BW_{PFS}$. The following equation captures the runtime of the offline analysis:

$$Time_{offline} = (1 + \alpha) \cdot \frac{\frac{N \cdot \lambda_a}{M}}{\frac{1}{N} \cdot BW_{PFS}} + \frac{N \cdot \lambda_a}{M \cdot T_{server_k}} \quad (14)$$

In some cases, analysis data is transferred over the wide area network for analysis, resulting in more performance and energy penalty than what our optimistic model estimates.

Next, we model the energy cost of offline processing. To be optimistic about the offline approach, we charge only idle power for the compute servers while they read and write the analysis data, and account for busy power during the data analysis. We can obtain this by multiplying the Equation 14 by P^{server}_{idle} for the reading and writing part of the process, and multiplying by P^{server}_{busy} for the analysis part of the process. Also, since Equation 14 represents the runtime of the whole offline analysis process, we have to multiply the equation by P to get the total energy.
The energy cost of offline processing does not depend on the number of offline nodes; it only depends on the total amount of data to be read and processed. This can be expressed as follows:

No. of compute nodes (N)	18,000
PFS bandwidth (BW_{PFS})	240GB/s
Output frequency (t_{iter})	1 hour
Simulation duration (t_{sim})	24 Hours
Overprovisioning factor (f_{op})	1.50
Data reduction factor (α)	1 (no reduction)
SSD model	Samsung PM830
$BW_{host2ssd}$	750MB/s [12]
BW_{fm2c}	320MB/s [12]
BW_{c2m}	3.2GB/s [6]
SSD busy power (P_{busy}^{SSD})	3W [8]
SSD idle power (P_{idle}^{SSD})	.09W [8]
Offline compute node model	2 Quad-core Intel Q6600 [5]

Table 2: Parameters for performance and energy models.

Data Analytics Kernels	ARM-Cortex A9	Intel Q6600 server
Statistics (mean)	416 MB/s	17.7 GB/s
Pattern Matching (grep)	123 MB/s	4.34 GB/s
Data formatting (transpose)	76 MB/s	1.95 GB/s
Dim. reduction (PCA)	10.2 MB/s	80 MB/s
Compression (gzip)	4.1 MB/s	216 MB/s
Dedup. (Rabin fingerprint)	1.81 MB/s	106 MB/s
Clustering (k-means)	0.91 MB/s	13.7 MB/s

Table 3: Processing throughput of common data analytics kernels on different devices. Our offline compute node consists of two Intel Q6600 Quad-core machines.

$$Energy_{offline} = P_{idle}^{server} \cdot (1+\alpha) \cdot \frac{N^2 \cdot \lambda_a \cdot t_{sim}}{BW_{PFS}} + P_{busy}^{server} \cdot \frac{N \cdot \lambda_a \cdot t_{sim}}{T_{server_k}} \tag{15}$$

3 Experimental Methodology

We evaluate our models using the data production rates from “hero” applications on the Jaguar machine at ORNL (Table 1.)

Our model is generic and applies to common supercomputer configurations seen today. Our evaluation is driven by parameters from the Cray XT5 Jaguar supercomputer [9], as shown in Table 2. In the current SSD landscape, there is no support for active computation on the device. To study the viability of active flash, we model after a contemporary SSD such as Samsung PM830, which has a multi-core SSD controller based on the ARM processor [8]. Although such a controller has three ARM-based cores, we adopt a conservative approach, and propose to use only one core for active computation, while leaving the other cores free for typical SSD activities (e.g. error-checking, garbage collection etc.) In the future, more cores are likely to be placed on the same chip, making active computation more promising than what is projected in this study.

To emulate the computing speed of the ARM-based SSD controller, we use an ARM Cortex-A9 processor in the Pandaboard mobile software development platform [18]. We model the data transfer times as follows. We assume 8 flash memory channels, each with 40MB/s bandwidth, to transfer data from the NAND Flash to the chip controller (BW_{fm2c}) [12]. Our numbers are conservative, and modern devices usually have higher channel bandwidth or more channels. Similarly, while the DDR2 SDRAM cache used in these SSDs may have a bandwidth of up to 5.3GB/s [6], we conservatively set BW_{c2m} to 3.2GB/s.

For a comparison with the offline approach, we use two Intel Core 2 Quad 6600 processor [5] as the offline data processing node. We use an input file of 100MB to measure the analysis throughput.

The chosen data analytics kernels cover a wide variety of representative analytics operations on scientific data, including pattern matching, clustering, changing data layout, and compression [3, 4, 13]. We measured their processing throughput on both the ARM-based SSD controller and the Intel server (Table 3.) Note that the throughput of the analysis kernels may be input-dependent as well (e.g. number of clusters, dimensions, search expressions etc.) Therefore, choosing a wide variety of kernels, whose throughput varies from less than one MB/s to a few GB/s helps better understand the limits, and the potential of active Flash.

For energy calculations, we use the thermal design power (TDP), 105W, as the busy power for each offline Quad-core processor (210W for the node.) Idle power is conservatively estimated as one third of the TDP. These are optimistic power estimates, as we do not account for the cooling cost of these offline nodes. Also, we do not account for the power required for memory and network traffic for the offline approach, which is clearly higher compared to active flash, due to additional reading of the analysis data.

4 Evaluation

Our evaluation aims to answer three questions: (1) What is the staging ratio for SSD provisioning, based on the capacity and bandwidth constraints? (2) Is the staging ratio sufficient to support in-situ data processing on the SSDs? (3) If so, what is the energy saving with active flash?

	CHIMERA	VULCAN	POP	S3D	GTC	GYRO
$S_{capacity}(16GB)$	**1**	1285	117	**9**	**3**	83
$S_{capacity}(32GB)$	1	2571	233	18	6	166
$S_{capacity}(64GB)$	1	4500	461	36	12	333
$S_{bandwidth}$	54	**54**	**54**	54	54	**54**

Table 4: Staging ratio is derived from capacity and bandwidth provisioning. More restrictive estimate denoted in the bold.

Table 4 shows the staging ratio based on both capacity and compute node-to-SSD bandwidth constraints given in Equations 1 and 2. The grey cell with bold font indicate the final ratio selected (the most restrictive one). For our target machine, Jaguar, $S_{bandwidth}$ is an application independent constant of 54, while $S_{capacity}$ obviously varies with different applications' data generation rates. Interestingly, the bandwidth constraint is more restrictive for one half of the applications, while the capacity constraint for the other half. This suggests that both factors need to be carefully examined when provisioning active flash devices.

Next, we identify the analysis kernels that can be run on SSDs for a given application, as we vary the staging ratio. Recall that $S_{bandwidth}$ is constant across applications, so we consider multiple staging ratios below it. Figure 3 shows the analysis kernels' compute throughput (measured on our ARM testbed) as flat horizontal lines and the application throughput requirement as slope lines with dots. A data analysis kernel is able to run on the SSD with no simulation performance penalty, if its computational throughput is higher than the required computational throughput of the application (T_{SSD_k} in Equation 7.) As the staging ratio increases, the threshold throughput of the application also increases.

For example, all of the data analysis kernels can be run on the SSD for the fusion application *S3D*, when the staging ratio is 5, as all the horizontal lines are above the S3D dot, with $x = 5$. A staging ratio of 5 means that for every five compute nodes, there is one SSD deployed. In our target setting of 18,000 compute nodes, this amounts to 3,600 SSDs. As we increase the staging ratio to 50, the "gzip, fingerprinting, and kmeans" kernels cannot be run on the SSD as the threshold throughput of *S3D* is *higher* than the compute throughput of these kernels. Figure 3 reveals that even a staging ratio as high as 50 can accommodate all the data analysis kernels for most of the applications, with the exception of CHIMERA and S3D (the most data-intensive.)

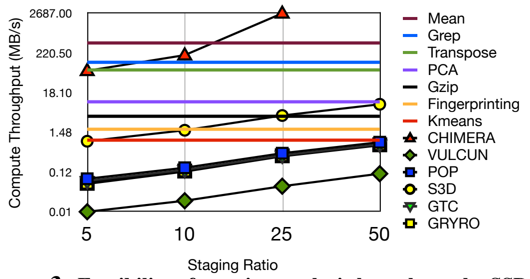

Figure 3: Feasibility of running analysis kernels on the SSD using different staging ratios. A kernel line *higher* than a dot point on the application's slope line is *suitable* for active flash.

Figure 4 plots the energy expense in running data analysis using both active flash and offline strategies. Based on Figure 3, we choose a staging ratio of 10, which seems to serve all the applications well, except *CHIMERA*, which cannot run any of the kernels with this staging ratio. To support a staging ratio of 10, we will need to provision 1800 SSDs, each with 64GB of capacity, due to the most restrictive application, *GTC* (Table 4.) We define $baseline_{PFS}$ ($y = 0$) as just running the simulation job without SSDs: all checkpoint and output data are written to the PFS. In $baseline_{PFS}$, no further data analysis is performed after the simulation run. A negative number indicates the energy saving achieved when compared against $baseline_{PFS}$. We observe that deploying SSDs just for higher I/O throughput, in and of itself ($baseline_{SSD}$, shown as the dotted horizontal line in Figure 4), saves significant energy due to the shortening of the overall application run time.

It is no surprise that active analysis on the SSDs consume extra energy compared to $baseline_{SSD}$. However, what is interesting is that it still results in savings compared to $baseline_{PFS}$, for almost all applications and data analysis kernels, except for *VULCUN* (as it does little I/O, there is no benefit due to SSDs.) In contrast, the offline approach consumes more energy due to the I/O wait time on the offline compute nodes. For example, for *S3D* with *fingerprinting*, we observe up to 1137 kWh of energy savings per simulation run compared to the offline processing as it produces significant analysis data. Overall, active flash makes data analysis virtually "free" in most cases when it is piggybacked on the SSDs, promising significant energy and performance savings compared to the offline approach. However, as noted earlier, active flash is not always feasible for all application data production rates (e.g., *CHIMERA* and *S3D* with *kmeans*), which would require a substantially higher SSD provisioning configuration.

5 Conclusion and Future Work

Based on our evaluation, we conclude that piggybacking data analysis on SSDs is feasible in terms of both capacity and performance, for representative large-scale parallel simulations. Further, it is also more energy-efficient compared to the traditional offline approach. An interesting angle to pursue is if "such energy savings can pay for the infrastructure cost of the SSDs?" This, however, requires a careful estimation of the lifetime of SSDs in our target scenario, considering factors such as I/O access patterns and the specific device model. We plan to investigate this in our future work.

6 Acknowledgments

This work was sponsored partially by ORNL, managed by UT Battelle, LLC for the U.S. DOE (Contract No. DE-AC05-00OR22725.), and partially through NSF awards CNS-0915861 and CCF-0937690, an IBM Faculty Award, and Xiaosong Ma's joint appointment between NCSU and ORNL.

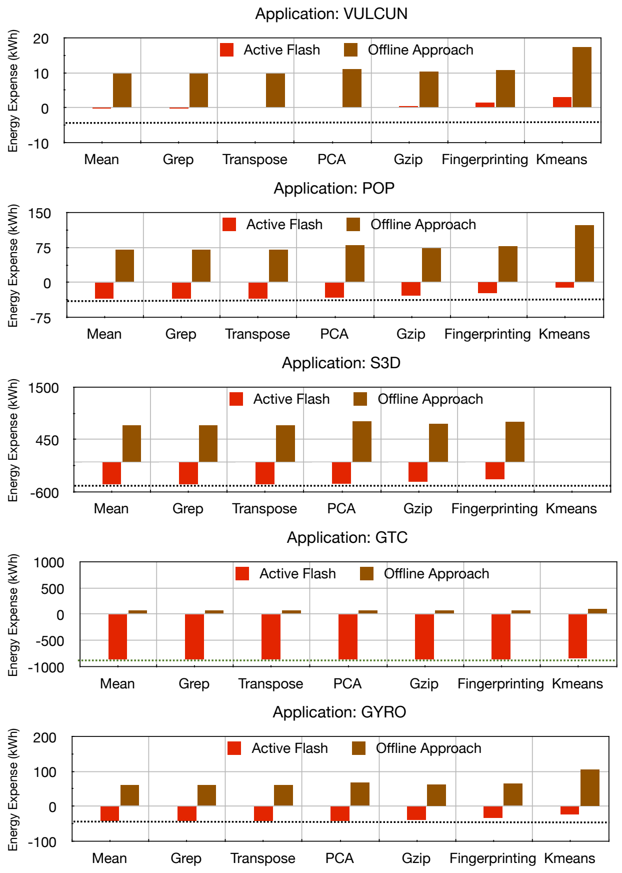

Figure 4: Comparing the energy expenses of all applications for Active Flash and offline approach. Expenses are w.r.t. running only the simulation job using PFS, without SSDs ($baseline_{PFS}$.) Dotted line denotes energy savings due to running only simulation using SSDs without active computation ($baseline_{SSD}$.)

7 References

[1] Computational science requirements for leadership computing, 2007, http://tinyurl.com/nccs2007.

[2] Exascale computing study: Technology challenges in achieving exascale systems, peter kogge, editor and study lead, 2008.

[3] Gnu grep, http://www.gnu.org/software/grep/.

[4] Gnu zip, http://www.gzip.org/.

[5] Intel core2 quad processors , http://tinyurl.com/q6600core.

[6] Micron dram datasheet , http://tinyurl.com/drambw.

[7] Preparing for exascale: Ornl leadership computing facility application requirements and strategy, 2009, http://tinyurl.com/nccs2009.

[8] Samsung pm830 datasheet, http://tinyurl.com/co9zyq7.

[9] National Center for Computational Sciences. http://www.nccs.gov/, 2008.

[10] Acharya et al. Active disks: programming model, algorithms and evaluation. *In ASPLOS*, 1998.

[11] Boboila et al. Active flash: Performance-energy tradeoffs for out-of-core processing on non-volatile memory devices. In *MSST*, 2012.

[12] Kim et al. Fast, Energy Efficient Scan inside Flash Memory Solid-State Drives. *In ADMS workshop with VLDB*, 2011.

[13] Ranger et al. Evaluating mapreduce for multi-core and multiprocessor systems. *In HPCA*, 2007.

[14] Riedel et al. Active storage for large scale data mining and multimedia applications. *In VLDB*, 1998.

[15] Zheng et al. Predata - preparatory data analytics on peta-scale machines. In *IPDPS*, 2010.

[16] Michael L. Norman and Allan Snavely. Accelerating Data-intensive Science with Gordon and Dash. In *TG'10*.

[17] U.S. Department of Energy. DOE exascale initiative technical roadmap, December 2009. http://extremecomputing.labworks.org/hardware/collaboration/EI-RoadMapV21-SanDiego.pdf.

[18] The Pandaboard Development System. `http://pandaboard.org/`.

[19] Top500 supercomputer sites. http://www.top500.org/.