Active Flash: Performance-Energy Tradeoffs for Out-of-Core Processing on Non-Volatile Memory Devices

Simona Boboila*, Youngjae Kim†, Sudharshan S. Vazhkudai†, Peter Desnoyers* and Galen M. Shipman†
*Northeastern University, †Oak Ridge National Laboratory
{simona,pjd}@ccs.neu.edu, {kimy1,vazhkudaiss, gshipman}@ornl.gov

I. Introduction

In this abstract, we study the performance and energy tradeoffs involved in migrating data analysis into the flash device, a process we refer to as *Active Flash*. The Active Flash paradigm is similar to "active disks", which has received considerable attention. Active Flash allows us to move processing closer to data, thereby minimizing data movement costs and reducing power consumption. It enables *true out-of-core* computation. The conventional definition of out-of-core solvers refers to an approach to process data that is too large to fit in the main memory and, consequently, requires access to disk. However, in *Active Flash*, processing outside the host CPU literally frees the core and achieves real "out-of-core" analysis.

Moving analysis to data has long been desirable, not just at this level, but at all levels of the system hierarchy. However, this requires a detailed study on the tradeoffs involved in achieving analysis tunraround under an acceptable energy envelope. To this end, we first need to evaluate if there is enough computing power on the flash device to warrant such an exploration. Flash processors require decent computing power to run the internal logic pertaining to the Flash Translation Layer (FTL), which is responsible for operations such as address translation, garbage collection (GC) and wear-leveling.

Modern SSDs are composed of multiple packages and several flash chips within a package. The packages are connected using multiple I/O channels to offer high I/O bandwidth. SSD computing power is also expected to be high enough to exploit such inherent internal parallelism within the drive to increase the bandwidth and to handle fast I/O requests. More recently, SSD devices are being equipped with powerful processing units and are even embedded with multicore CPUs (e.g. ARM Cortex-A9 embedded processor is advertised to reach 2GHz frequency and deliver 5000 DMIPS; OCZ RevoDrive X2 SSD has 4 SandForce controllers, each with 780MHz max frequency Tensilica core). Efforts that take advantage of the available computing cycles on the processors on SSDs to run auxiliary tasks other than actual I/O requests are beginning to emerge. Kim et al. [2] investigate database scan operations in the context of processing on the SSDs, and propose dedicated hardware logic to speed up scans. Also, cluster architectures have been explored [1], which consist of low-power embedded CPUs coupled with small local flash to achieve fast, parallel access to data.

Processor utilization on SSD is highly dependent on workloads and, therefore, they can be idle during periods with no I/O accesses. We propose to use the available processing capability on the SSD to run tasks that can be offloaded from the host. This paper makes the following contributions:

- We have investigated *Active Flash* and its potential to optimize the total energy cost, including power consumption on the host and the flash device.
- We have developed analytical models to analyze the performance-energy tradeoffs for *Active Flash*, by treating the SSD as a black-box. This is particularly valuable due to the proprietary nature of the SSD internal hardware.
- We have enhanced a well-known SSD simulator (from MSR) to implement "on-the-fly" data compression using *Active Flash*. Our results provide a window into striking a balance between energy consumption and application performance.

II. Tradeoffs Analysis

We analyze the tradeoffs of using the SSD controller to do part of the computation which would otherwise be carried out on the host CPU. The two scenarios compared are:

- **Baseline:** the entire computation is performed on the host CPU.
- **Hybrid Model:** a part of the computation is performed on the SSD controller; the rest, if any, is run on the host CPU.

Most of the data path segments are common for Baseline and Hybrid Model: in both cases, the data needs to pass through the SSD controller. Data transfer between the controller and the host CPU is very fast with current SATA and PCIe interfaces (e.g. SATA 3.0 at 750MB/s bandwidth; PCIe up to 16GB/s bandwidth), and does not affect performance significantly. Therefore we investigate the tradeoffs due to computation, not data transfer, the latter being ignored to keep the model simple.

Energy and time consumption are mainly affected by the power and speed differences between the SSD controller and host CPU. We estimate energy savings versus slow down, based on the following parameters:

$x =$ *Host CPU utilization for Baseline* (average of a long run).

This represents the percentage of time when the CPU is being utilized and can range from 0 to 100%.

$$s = \frac{Speed\_host\_cpu}{Speed\_ssd\_controller};\ p = \frac{\Delta Power\_ssd\_controller}{\Delta Power\_host\_cpu}$$

Let us suppose the controller does s times less work than the host CPU, as it is s times slower. For example, a 2x slower controller at 100% utilization does the same work in the same amount of time as the host CPU at 50% utilization. Mapping [0%-100%] to [0..1] and for $x = 1/s$, the SSD controller can accommodate the entire computation at 100% controller utilization. This means that the entire computation can be moved to the controller if the baseline CPU utilization is $\leq 1/s$. If the baseline CPU utilization is $> 1/s$, the controller will be fully utilized and able to accommodate only a part of the computation. The percentage of work done is $c\% = (100\% \times Speed\_ssd\_controller)/(x\% \times Speed\_host\_cpu)$, where 100% is the ssd_controller_utilization, and the x% is the host_cpu_utilization. To summarize, the fraction of the computation that can be moved to the SSD controller as a function of x is:

$$c(x) = \begin{cases} 1, & \text{for } x \in [0, 1/s) \\ 1/(sx), & \text{for } x \in [1/s, 1] \end{cases}$$

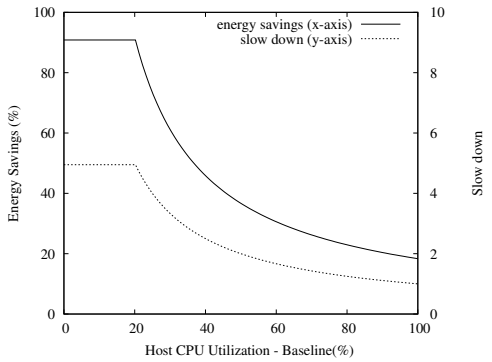

Fig. 1: Energy savings vs slow down in the long run. The SSD controller is used in conjunction with the Host CPU to carry out part of the computation. The x-axis shows host CPU utilization in the baseline, before part of the computation was moved to the controller.

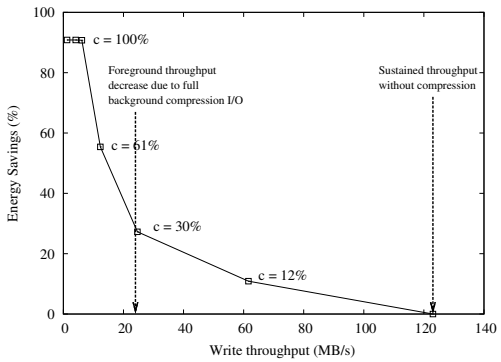

Fig. 2: Compression running in the background on the SSD controller. While the foreground application writes at the specified throughput (x-axis), the SSD controller is able to compress c% of the writes during idle times and save energy (y-axis).

To compute the energy savings, we compute the variation in energy consumption as: $\Delta Energy = Time \times \Delta Power$, where $\Delta Power = Power\_active - Power\_idle$.

Also, time is inversely proportional to speed, which gives:

$$\begin{aligned}\Delta E &= 1 - \frac{\Delta Energy\_ssd}{\Delta Energy\_host} \\ &= 1 - \frac{Speed\_host\_cpu \times \Delta Power\_ssd\_controller}{Speed\_ssd\_controller \times \Delta Power\_host\_cpu} \\ &= (1 - sp)\end{aligned}$$

Finally, since only a fraction, $c(x)$, of the computation is moved to the controller, the energy savings are: $\Delta E(x) = (1 - sp) \cdot c(x)$

Considering that the computation on the SSD controller and the host CPU occur in parallel, the extra time taken by the controller to finish its share of the analysis determines the slow down:

$$S(x) = \frac{Time\_ssd\_hybrid\_model}{Time\_baseline} = s \cdot c(x)$$

Figure 1 gives a concrete estimation of energy savings compared to job slow down. We consider the 800MHz ARM Cortex-A8 embedded processor advertised at a speed of 1600 DMIPS or more, and a host CPU featuring a 2.4GHz Intel Core 2 Duo processor benchmarked at 7922 DMIPS. We estimated p based on the idle and active power consumption values cited in datasheets and benchmarks. For an average host utilization of $\geq 60\%$, the Hybrid Model gives about $20 - 30\%$ energy savings with a negligible slow down (converges to 1). For analyses that are not that computationally intensive, a high fraction of the computation can be accommodated on the controller at some slow down cost, resulting in energy savings of up to 90%.

III. Practical Case Study: Compression

We have analyzed the tradeoffs of *Active Flash* for a widely used application, compression, to show that processing on the SSD is feasible in terms of energy and performancy efficiency. We have extended MSR's SSD simulator to simulate a 64GB SSD and the processor parameters from Section II. The scenario here is that the main application on the host CPU writes out data to the SSD, which is required to compressed. The application issues 6GB of 4K random writes to the SSD. We refer to this I/O traffic to the SSD as a foreground process. The SSD has been simulated to offer 123MB/s random write throughput without compression.

Figure 2 shows energy savings when compression is performed on-the-fly, on the SSD controller, while it also serving the foreground I/O. The compression is run as a background process, during idle periods on the SSD. Since compression is also I/O intensive, the foreground process needs to run at a lower throughput to accommodate the additional I/O due to background compression (i.e. foreground throughput is ≤ 24 MB/s for 100% background compression, due to just additional I/O and no computation on controller).

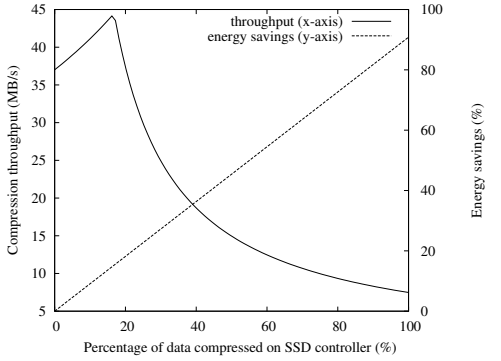

Fig. 3: Compression running in parallel on both host CPU and SSD controller. The data to be compressed on the host CPU is re-read from the SSD; the SSD controller compresses flash-resident data. Both operations are performed after the foreground I/O.

Figure 3 studies throughput and energy savings when compression is performed after the completion of the foreground I/O, in a traditional chaining approach (e.g., main application followed by analytics.) In this case, compression is performed on both the host CPU (re-reading data back from the SSD) and the SSD controller (using SSD-resident data.) The figure shows that the computation time is the bottleneck and the throughput peaks when 20% of the data is compressed on the controller, and the rest on the host CPU. However, if energy cost is to be minimized, then more data needs to be compressed on the SSD controller.

IV. Conclusion and Future Work

We have studied the potential of *Active Flash* as a means to move processing into SSDs and have showed that it can be a vehicle to balance the reduction in energy cost and application throughput. Our future work involves hybrid scheduling on CPU and flash and in optimally running the active computation alongside FTL operations.

References

[1] Andersen, D. G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., and Vasudevan, V. FAWN: A Fast Array of Wimpy Nodes. In *SOSP* (2009).

[2] Kim, S., Oh, H., Park, C., Cho, S., and Lee, S.-W. Fast, Energy Efficient Scan inside Flash Memory SSDs. In *ADMS* (2011).