Energy Efficiency and I/O Performance of Low-Power Architectures

Pablo Pavan, Ricardo Lorenzoni, Vinícius Machado, Jean Bez, Edson Padoin, Francieli Zanon Boito, Philippe Navaux, Jean-François Méhaut

Energy Efficiency and I/O Performance of Low-Power Architectures

Pablo J. Pavan[1], Ricardo K. Lorenzoni[1], Vinícius R. Machado[2],
Jean L. Bez[2], Edson L. Padoin[1], Francieli Z. Boito[3],
Philippe O. A. Navaux[2], Jean-François Méhaut[4]

[1]Department of Exact Sciences and Engineering,
Regional University of Northwest of Rio Grande do Sul (UNIJUI), Ijuí, RS, Brazil
[2]Institute of Informatics,
Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil
[3]Department of Informatics and Statistics,
Federal University of Santa Catarina (UFSC), Florianópolis, SC, Brazil
[4]Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP*, LIG, 38000 Grenoble, France

First submitted in March 2017

Abstract

This paper presents an energy efficiency and I/O performance analysis of low-power architectures when compared to conventional architectures, with the goal of studying the viability of using them as storage servers. Our results show that despite the fact the power demand of the storage device amounts for a small fraction of the power demand of the whole system, significant increases in power demand are observed when accessing the storage device. We investigate the access pattern impact on power demand, looking at the whole system and at the storage device by itself, and compare all tested configurations regarding energy efficiency. Then we extrapolate the conclusions from this research to provide guidelines for when considering the replacement of traditional storage servers by low-power alternatives. We show the choice depends on the expected workload, estimates of power demand of the systems, and factors limiting performance. These guidelines can be applied for other architectures than the ones used in this work.

Keywords: Energy Efficiency, Low-Power Processors, I/O performance, SSD, HDD.

1 Introduction

The increase in processing power of High-Performance Computing (HPC) architectures had as a consequence an increase in power demand, leading to a situation that is economically and ecologically unsustainable. Taking this matter into consideration, a DARPA report suggests a 20 MW limit for power demand of future exascale HPC systems[1]. In order to meet this budget, based on existing technologies, the energy efficiency of the processors needs to be increased by two orders of magnitude. This premise has led HPC researchers to seek alternatives that respect the given power limit. One of such alternatives is the use of low-power architectures, powered by processors such as the advanced RISC machine (ARM) ones. Despite presenting lower performance, these processors provide better energy efficiency for some scientific applications [2].

A similar phenomenon occurs in data centers, caused by the accelerated growth of the amount of data generated and consumed by applications. In this context, there has been some initiatives

*Institute of Engineering Univ. Grenoble Alpes by companies, such as PayPal, to use low-power architectures in their data centers and cloud infrastructures [3, 4].

Typically, the processors are responsible for most of the power demanded by computational systems. Nevertheless, the gap between processing and data access speeds causes many applications to spend most of their time on input/output (I/O) operations [5, 6]. For this reason, increasing the efficiency of the I/O subsystem is also an important step to tackle the energy and power challenge. Particularly, in infrastructures where dedicated machines are used as storage servers, such as parallel file systems for HPC, these machines spend most of their time serving I/O requests. Therefore, the use of low-power architectures as those storage servers could lead to better energy efficiency.

In this paper, we evaluate the viability of such low-power storage servers by conducting an energy efficiency study under heavy I/O workloads. We compare a traditional server with a low-power multi-processor system-on-chip (MPSoC), using different hard disk drives (HDDs) and solid state drives (SSDs). We analyze power demanded by the storage devices and by the whole systems, and show the impact of workloads characteristics — such as request size and access spatiality — in the achieved energy efficiency. Finally, results are used to draw guidelines for the adoption of low-power storage servers. Those guidelines can be applied to different systems than the ones used here.

This paper is organized as follows. Section 2 discusses related work. The experimental methodology is detailed in Section 3. Results are discussed in parts, first focusing on performance, in Section 4, then on power demand, in Section 5. In Section 6 we analyze and discuss the energy efficiency, a metric that accounts for both performance and power demand. The replacement of the traditional server by the low-power one is discussed in Section 7, where the guidelines are detailed. Finally, Section 8 concludes this paper and discusses future work perspectives.

2 Related Work

Many research initiatives focus on energy consumption of I/O operations. In the past, researchers explored the use of multi-speed disks for storage servers [7, 8, 9]. Recent strategies employ Dynamic Voltage and Frequency Scaling (DVFS) to reduce the processor frequency during I/O operations, since these do not require much processing power. DVFS has been a popular technique to save energy consumption by lowering the frequency of idle cores in multicore systems [10], idle resources in cloud environments [11, 12], and during MPI applications' communication phases [13].

Ge, Feng and Sun (2012) [14] propose a strategy for HPC architectures that consists of applying DVFS in the processing nodes during I/O operations to the parallel file system. They consider characteristics of the applications in order to decide the optimal frequency. A similar technique is applied by Shang and Wang (2011) [15] for sequential applications. Saito et al. (2013) [16] also employ DVFS in the processing nodes during checkpointing operations to local storage devices.

Ibrahim et al. (2016) [17] study the performance and energy consumption of MapReduce applications in cloud environments using different DVFS strategies. Other researchers also consider the energy consumption of MapReduce applications in clouds. Cardosa et al. (2012) [18] propose algorithms for virtual machine allocations in real machines to improve energy efficiency.

Amur, Cipar and Gupta (2010) [19] present a distributed file system that places primary data replicas in servers that are always available, and other replicas in servers that can be turned off according to demand. Kim et al. (2011) [20] develop and evaluate algorithms to define the subset of servers to be kept on in order to ensure data availability while minimizing energy consumption.

Nijim et al. (2009) [21] combine SSDs with HDDs to provide energy-efficient storage. This is achieved by using the SSD as a cache for the HDD. This hybrid storage strategy is exploited by others to provide high performance for I/O servers [22]. The higher cost per byte of SSDs often makes the total replacement of HDDs unfeasible.

Regardless of the discussed techniques, the processing power of the storage servers is frequently underutilized, since their main activity is to access storage devices. Therefore, the use of low-power architectures in these servers could be an interesting alternative. This is complementary to the alternatives presented in this section, and to evaluate its feasibility is the goal of this work. In a previous work [23], we have conducted an initial evaluation, and concluded MPSoCs were a good low-power alternative for read workloads. The research presented in this work is more complete — exploring more configurations and different workloads — and reaches different conclusions. This allows us to draw guidelines for the adoption of low-power storage servers.

3 Experimental Methodology

Two environments were used for the experiments discussed in this work. The first, named **PC**, is a traditional desktop, with a quad-core Intel Core-I7 4790 processor of 3.6 GHz base clock frequency, 8 MB L3 cache, four 256 KB L2 caches, and four 32 + 32 KB instructions + data L1 caches. This is a processor of the Haswell architecture and has 14 pipeline stages, executing up to 4 instructions per cycle. The equipment has 16 GB DDR3 RAM memory operating at 1600 MHz.

The second environment is an MPSoC CubieTruck, with a SoC A20 manufactured by AllWinnerTech, and a dual GPU MALI400 MP2, called **MPSoC**. The processor is a dual-core ARM Cortex-A7 at 920 MHz frequency, 1 MB L2 cache and 64 KB L1 cache. The equipment has 2 GB of low-power DDR3 RAM memory at 480 MHz frequency.

The operating system in both environments is Ubuntu with kernel 3.16.0-38 in the PC, and Debian with kernel 3.4.106 in the MPSoC. It is not possible to use a newer kernel in the MPSoC, and we believe using an older kernel in the PC would not lead to a fair comparison. The file system *ext4* was used to access all storage devices.

Four storage devices were used for the experiments, two SSDs and two HDDs, selected to ensure results are not specific to a given device, and to cover different characteristics: HDD1 is a newer model than HDD2, and SSD1 is more recent than SSD2; the HDDs have different speeds, and the SSDs have different capacities. They are presented in Table 1. The names in the first column of the table will be used in the remaining of the text to reference them. Tests were performed with the four storage devices in the two machines, thus totaling eight configurations. All devices were accessed by MPSoC through SATA II, and by PC through SATA III. The fact that HDD2 does not support SATA III is not expected to impact the results, since its bandwidth could not be high enough to surpass the 3 Gb/s transfer rate of SATA II.

Table 1: Storage devices used for the experiments

	Type	Manufacturer/Model	Capacity (GB)	Interface	RPM	Manufacturer specs	
						Voltage (VDC)	Current (A)
HDD1	HDD	Seagate ST1000LM035-1RK172	1000	SATA III	5400	5	0.55
HDD2	HDD	Seagate ST96023AS	60	SATA II	7200	5	0.58
SSD1	SSD	Samsung PM871	240	SATA III	-	5	0.50
SSD2	SSD	Kingston SV300S37A120G	120	SATA III	-	5	1.00

The FIO[1] benchmark was selected for the tests because it allows for the description of the desired access patterns. The **write** experiments were conducted in each configuration **with** and **without** the usage of the buffer cache. For the tests without cache, the benchmark parameter "`-direct=1`" was used so operations would be performed directly to the storage device.

[1] https://linux.die.net/man/1/fio

Regarding the type of operation and the spatiality, four access patterns were generated: sequential write, random write, sequential read, and random read. Additionally, two request sizes were evaluated: 32 KB and 4 MB. Each test was executed for 20 GB of data with an execution time limit of 60 seconds - the test stopped when the first of these two conditions was met.

We include write experiments using the buffer cache in our analysis because the cache plays an important role when writing: data is stored in the cache and transparently spilled to the storage device, which partially hides the write latency of the devices. On the other hand, we **do not** include read experiments with the cache because in this situation the behavior depends on previous accesses. It is not usual that the same data will be written to the persistent storage and then read from it immediately after, so it would not make sense for us to simulate this behavior.

Counting the eight configurations of equipment and storage device, two cache options for half the experiments (with or without it), four operations, and two request sizes, a total of 96 experiments are analyzed in this paper. Each was repeated 10 times, and different experiments in the same configuration were executed in random order, to avoid unexpected effects. A minimum 20-seconds delay is imposed between tests, so the power demand stabilizes. Moreover, the "`sync`" command is used between tests that use the buffer cache to make sure they are independent.

To measure the power demand, we employed an Agilent oscilloscope model DSO6014A. This oscilloscope was connected via USB to a computer, where the BenchVue software[2] logs captured data. A power tip model 1146A, manufactured by Agilent, was used to measure the current for the entire equipment. Current for the storage devices was measured from the Hall effect, with an Allegro solution model ACS712T[3] connected to the oscilloscope. Instantaneous voltage and current measurements are obtained every 500 ms. To each test, we take the arithmetic mean of the power demand measurements (W), obtained by multiplying the voltage (V) by the current (A).

The Kolmogorov-Smirnov statistical test [24] was used — together with Quantile-Quantile Plots — to verify if results of each test followed a normal distribution (bandwidth, average power demand, and energy efficiency were analyzed separately). Since we could not conclude normality for all of them, we use the median to combine results of 10 repetitions and non-parametric statistical methods to compare different experiments, namely the Wilcoxon-Mann-Whitney test [25] to compare pairs and the Dunn test [26] to comparisons between more than two experiments. We use a confidence of 95% for all statistical tests. During the article, unless explicitly said, we only mention a difference between two results if this difference was confirmed by a statistical test.

Some additional experiments were conducted in another machine, the *draco-1*, because in the PC it is not possible to measure energy consumption of the RAM with the Processor Counter Monitor (PCM) [27]. The draco-1 has two Intel Xeon E5-2640 CPUs running at 2 GHz, with 64 GB of memory and a Samsung 840 Series 500 GB SSD. Since this machine is not in the same facility as the PC and the MPSoC, we use IPMI [28] to measure its power demand instead of the oscilloscope.

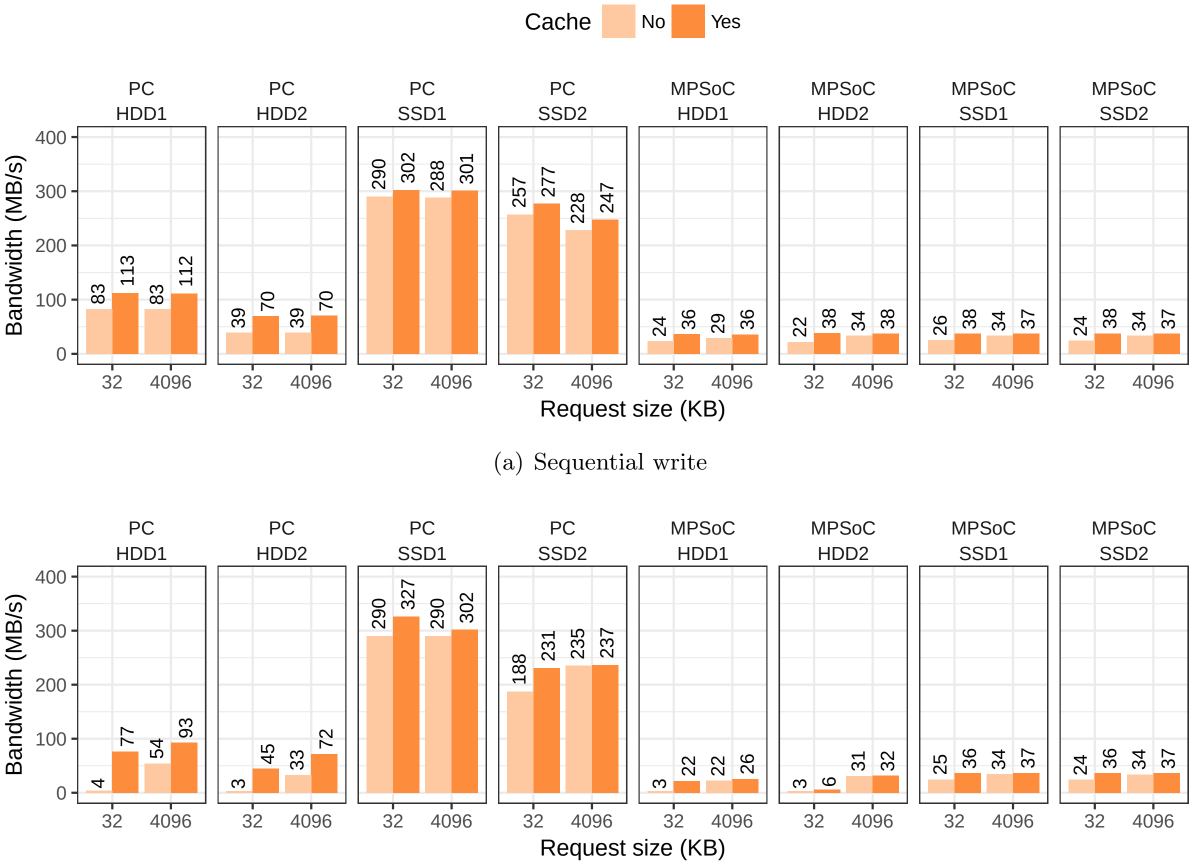
Following the ideas of reproducible research, all codes used to obtain, parse, and analyze results, in addition to results themselves, are publicly available at https://github.com/francielizanon/ccpe2017/.

4 Performance

We use bandwidth as the performance metric because tests execute for a fixed period of time (or until a maximum total size is reached). Most experiments finished in 60 seconds, before reading

[2] http://www.keysight.com/en/pc-2472896/benchvue-software?cc=US&lc=eng

[3] http://www.allegromicro.com/~/media/files/datasheets/acs712-datasheet.ashx



(a) Sequential write

(b) Random write

Figure 1: Write performance

or writing 20 GB of data, except some read experiments in the PC with SSDs. Among these, the fastest took 39 seconds. The slowest experiment read approximately 143 MB. Execution times and amounts of read or written data for all experiments can be seen in the git repository.

Figure 1 compares the performance of sequential (Figure 1(a)) and random (Figure 1(b)) writes. Each box represents a configuration of machine and storage device, the x-axis of each plot represents request size (32 KB and 4 MB), and the colors separate the results with and without cache. The bars show the bandwidth, hence the taller they are, the better performance is.

Write performance was increased by using cache in all situations of sequential write — up to 79% in the PC with HDDs, 71% in the MPSoC with HDDs, 8% in the PC with SSDs, and 55% in the MPSoC with SSDs — and random write — up to 1818% in the PC with HDDs, 611% in the MPSoC with HDDs, 23% in the PC with SSDs, and 48% in the MPSoC with SSDs. This increase in performance was larger for HDDs because their performance was lower, so having the cache to hide write latency becomes more important. Moreover, increases were larger for random write because this access pattern usually presents worse performance than sequential write (especially in HDDs), so the cache also helped by making more sequential accesses to the storage device. Finally, for random write, the higher differences happened for small requests, because, compared with the large requests experiments, more requests were issued during the same test, hence the generated access pattern is “more random”. The only exceptions of cache increasing write performance were sequential and random write in the PC with SSD2 and 4 MB requests, where it was not possible to conclude the results with and without cache are significantly different.

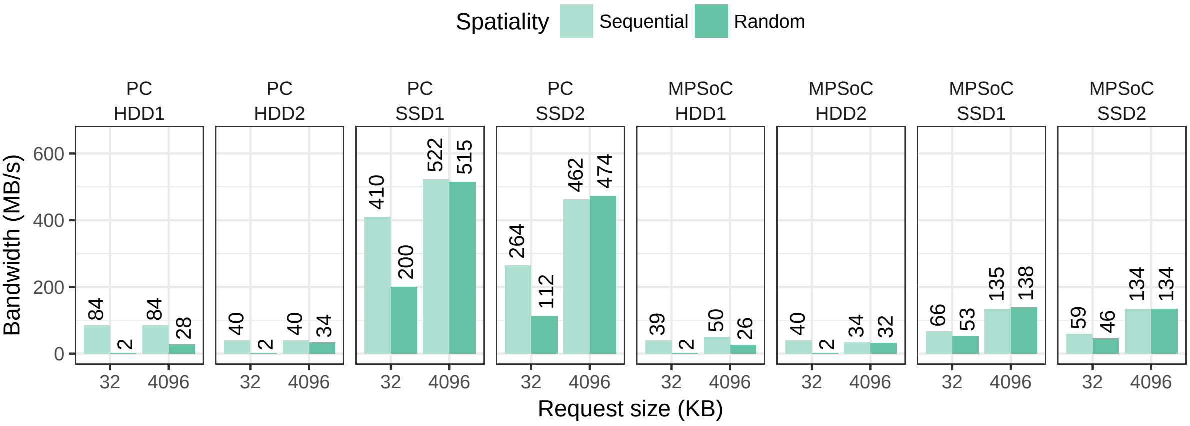


Figure 2: Read performance

Large requests only resulted in higher sequential write performance than small ones in the MPSoC without cache (up to 53%). In the PC with SSD2, the tests with small requests have actually performed better (12% with cache and 13% without it). Nonetheless, **large requests resulted in higher random write performance to most configurations**, except in the PC with SSD1 without cache (no significant difference) and with cache (small requests perform 8% better), and in the PC with SSD2 with cache (no significant difference). The increase in random write performance by issuing large requests when not using cache was of up to 1259% with HDDs and 40% with SSDs. When using cache, these differences were of up to 398% with HDDs and 2% with SSDs. As previously discussed, small requests in random tests result in access patterns that are more random, and the access spatiality is usually more important for performance in HDDs than in SSDs. Moreover, using the cache helped to decrease the randomness of accesses.

Higher bandwidth was achieved by sequential write than random write in most tests with HDDs (up to 1977% in the PC without cache and 56% with cache, and up to 667% in the MPSoC without cache and 495% with cache), except in the PC with HDD2 when issuing large requests and using the cache, where there was no significant difference. Again, using the cache decreased the randomness of the access pattern, so differences between sequential and random were smaller when using the cache. Moreover, differences were larger when using small requests because more random requests were generated during those tests. The only significant differences between sequential and random write in the SSDs were: in the PC with SSD1 issuing small requests using the cache (random write performance is 8% higher), and in the PC with SSD2 issuing small requests (sequential write performance was 20% higher with the cache and 37% without it).

Practically all devices presented lower write performance in the MPSoC than in the PC, the only exception was HDD2 in the random write test with small requests without cache. The increase in sequential write performance when using the PC, compared to using the MPSoC, was of up to 250% with HDD1, 87% with HDD2, 1033% with SSD1, and 955% with SSD2; and in random write performance the increase was of up to 262% with HDD1, 603% with HDD2, 1062% with SSD1, and 669% with SSD2. These differences are investigated in Section 4.1.

Figure 2 shows read performance. Each box represents a combination of machine and storage device, the x-axes shows the request size, and colors compare sequential and random reads.

There were no significant differences between sequential read results for small and large requests with HDDs in the PC. In the MPSoC with HDD1 large sequential requests performed 28% better, and with HDD2 small requests performed 16% better. **Higher read performance was obtained with large requests than with small requests in sequential read ex-**

periments in SSDs and in all random read tests: large sequential reads to SSDs were up to 75% better in the PC and 128% in the MPSoC, large random reads to HDDs were up to 1279% better in the PC and 1226% in the MPSoC, and large random reads to SSDs were up to 321% better in the PC and 189% in the MPSoC. **Higher bandwidth was achieved by sequential read than random read in all tests with HDDs and in tests with small requests to SSDs**: up to 1754% with HDDs in the PC, 1573% with HDDs in the MPSoC, 134% with SSDs in the PC, and 27% with SSDs in the MPSoC. When issuing large requests to SSDs, no significant difference was observed.

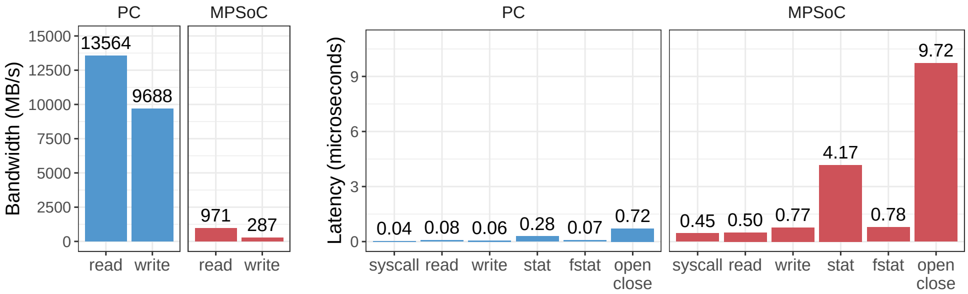
Similarly to what was observed for write performance, **in most situations devices present lower read performance in the MPSoC than in the PC**. The exceptions are random read in HDD1 and sequential read in HDD2, both with small requests, where results are not significantly different. Aside from these, sequential read performance was up to 113% higher in the PC than in the MPSoC with HDD1, 16% with HDD2, 522% with SSD1, and 347% with SSD2; random read was up to 7% higher with HDD1, 7% with HDD2, 274% with SSD1, and 253% with SSD2.

4.1 Investigation of the performance difference between the PC and the MPSoC

The jump from SATA III (available in the PC) to SATA II (in the MPSoC) could limit bandwidth to approximately 384 MB/s, which is higher than what was observed in the MPSoC. We have repeated some of the experiments[4] in the MPSoC while monitoring CPU usage with the *top* command, and observed it reaches up to 25%, what indicates CPU is also not the bottleneck.

To investigate if the RAM memory is limiting performance in the MPSoC, we have executed the *lmbench* benchmark [29] in both types of equipment to measure memory read and write bandwidth, as well as latencies induced by the operating system. The test was repeated four times, and median results are shown in Figure 3. They indicate the RAM is also not to blame for the lower performance observed in the MPSoC, since it would limit write bandwidth to approximately 286 MB/s. Latency, although higher in the MPSoC than in the PC, is not high enough to surpass write or read time.

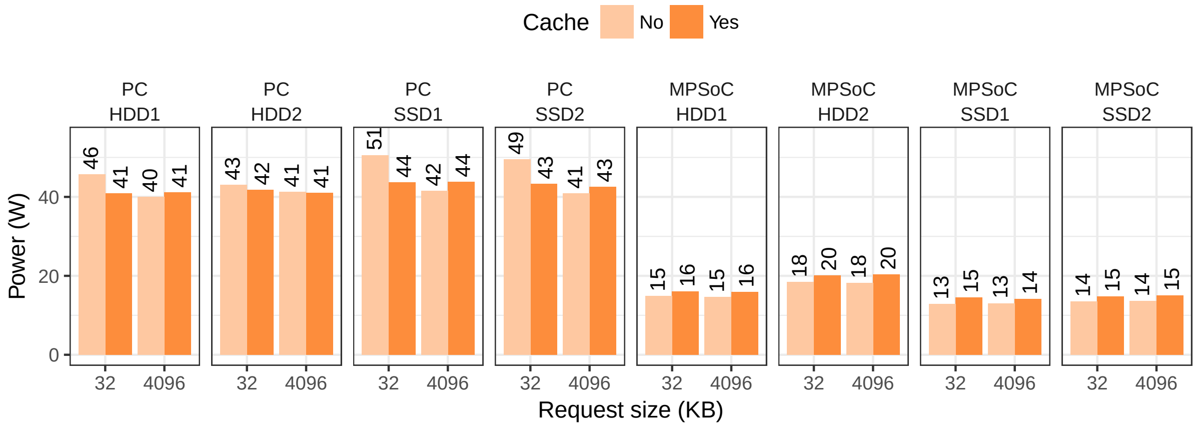
Further investigating this unexpected behavior, we have found reports of issues related to the SATA II bus from these SoCs, possibly caused by their drivers. The reported peak performance for the bus was of approximately 45 MB/s for write and 200 MB/s for read [30].



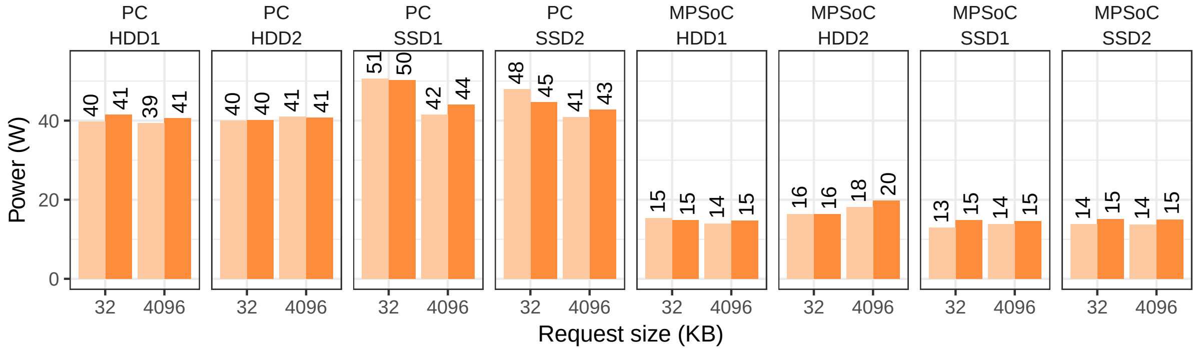
(a) Bandwidth

(b) Latency

Figure 3: Memory bandwidth and latency measurements obtained with *lmbench*



(a) Sequential write



(b) Random write

Figure 4: Power demand of **write** experiments

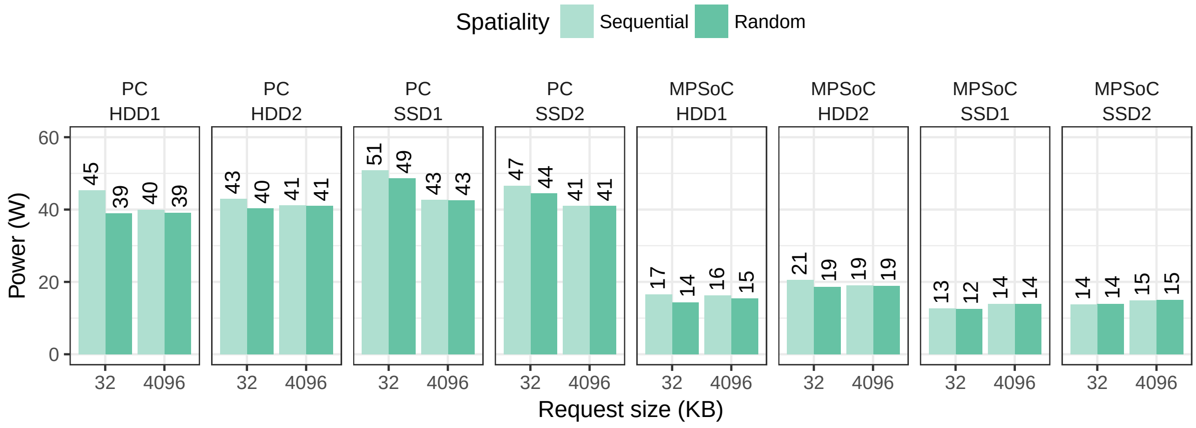
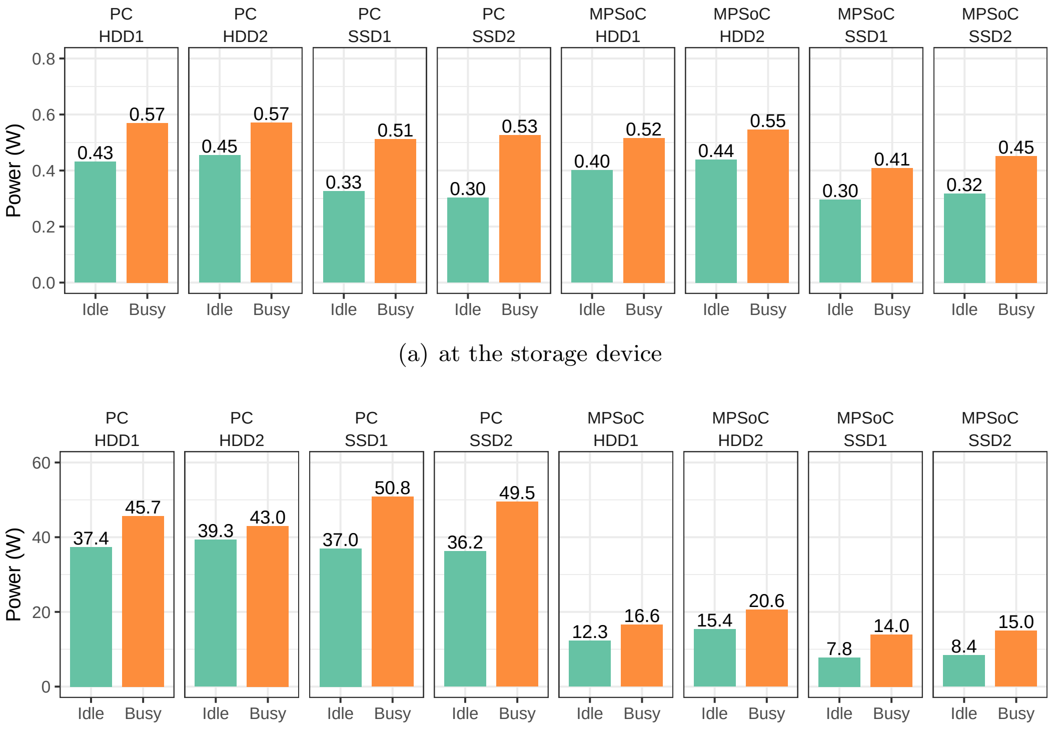


Figure 5: Power demand of **read** experiments

5 Power Demand

Power demand results for all experiments are shown in the Figures 4 and 5. We measured the power demand of each configuration in idle for at least 160 seconds. We collected two measurements per second, and calculated the medians. These results, in Watts, are presented in the graphs from Figure 6 in green, on the left, labeled “idle”. They are compared to the

[4] Sequential read and write with small requests without cache, four repetitions, with HDD1 and SSD2.



(a) at the storage device

(b) the whole system (including storage device)

Figure 6: Power demand in idle compared to the highest observed during experiments

maximum power demand observed during the experiments to the each configuration, in orange, on the right, labeled "busy". Only tests that do not use the buffer cache were considered for this analysis since they impose a heavier workload to the storage devices. Figure 6(a) shows the power demand of the storage device, and Figure 6(b) shows the power demand of the entire system (**including** the storage device). In each figure, each box represents a configuration of equipment and storage device.

The statistical tests indicate that all results presented in this section are significantly different. We can see that each storage device (Figure 6(a)) had very similar power demands in idle in the PC and in the MPSoC (although these results are significantly different). **When busy, the same devices demanded up to 26% more power in the PC than in the MPSoC**. Most configurations showed an increase in power demand of approximately 0.12 W when busy, with the exception of the experiments with SSDs in the PC, where this increase was close to 0.2 W.

Devices of the same type (HDD or SSD) behaved similarly. **SSDs demanded less power than HDDs**, but this difference was smaller in the PC (and in the PC SSDs demanded more power than in the MPSoC). Since the MPSoC environment does not allow the SSDs to perform at their maximum speed, as discussed in Section 4, they also did not reach their peak power consumption.

Considering the power demand of a storage device is approximately 0.5 W, it is clear the differences seen in the power of the whole system (Figure 6(b)) were not due to the device. Despite demanding approximately 3% of the power in the MPSoC and 1.4% in the PC, the storage device behavior affects other components of the machine. We have executed a new experiment with an SSD[5] while monitoring processor and memory energy consumption with

[5] Sequential write with 4 MB requests during 20 seconds, repeated three times, in the draco-1 machine. Its power demand in idle was measured with IPMI to be approximately 98 W, while the median during the experiment was 126 W. During the experiments, PCM reported a median power demand of 38 W for the processor and 5 W for the memory, while in idle these measurements were 13 W and 3 W, respectively.

PCM. We have observed an increase of 28 W in the power demand of the whole system compared to idle. The PCM measurements showed an increase of 25 W for the processor and 2 W for the RAM, indicating the processor was the main responsible for the power demand increase observed during our experiments.

Despite the fact SSDs demand less power than HDDs in the PC (first part of Figure 6(a)), when using them the system demands more power (first part of Figure 6(b)). Since SSDs are faster, they require more work from other components. This does not happen in the MPSoC, where SSDs have lower performance. Figure 7 shows the CPU usage reported by the *top* command during four repetitions of a sequential write test with 4 MB requests and no cache. In the PC we see higher CPU usage when accessing the SSD, but the same cannot be said about the MPSoC.

We continue this discussion on power demand by analyzing how the access pattern affects the power demand. We discuss each access pattern aspect separately: spatiality (sequential or random) in Section 5.1, request size (small or large — 32 KB or 4 MB) in Section 5.2, operation (read or write) in Section 5.3, and using or not the cache in Section 5.4.

5.1 Sequential vs. random

Sequential read demanded more power at the storage device than random read in some experiments, specially with HDDs: all executed with HDD1 (up to 12%); all generating small requests to HDD2 (up to 7%) and to SSD1 in the PC (12%); and with SSD2 in the PC (up to 3%). Other experiments show no significant increase or decrease in performance.

A similar behavior was observed for writes — **sequential write demanded more power at the storage device than random write in HDDs** (up to 15%), except in the MPSoC issuing large requests to HDD2 without cache, where there was no difference; and in the MPSoC issuing small requests to HDD1 without cache, where random write demanded 4% more power than sequential write. With SSDs the only significant difference between sequential and random write was in the MPSoC with SSD1, large requests and no cache, where random write demanded 3% more power.

Looking at the power demand of the whole system **during read tests, the results are similar to what was observed at the storage device**, with sequential reads (Figure 4(a)) demanding more power than random reads (Figure 4(b)): with HDD1 (up to 16%, larger differences in tests with small requests), issuing small requests to HDD2 (up to 10%), and in the PC issuing small requests to SSD1 (4%) and to SSD2 (5%). These results also show a similar tendency to performance results (Section 4). **In many situations, higher bandwidth from the device seems to increase power demanded by the system**.

Comparing write experiments, sequential demanded more power than random: with HDD1 in tests with small requests in the PC without cache (15%) and in the MPSoC (up to 8%), except

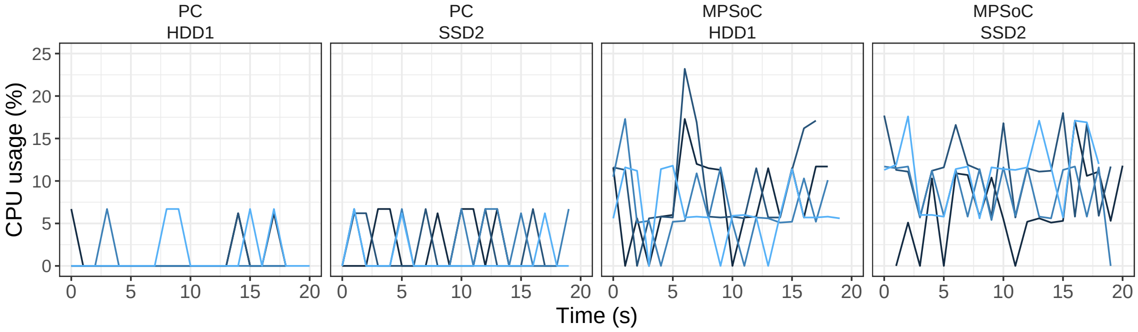


Figure 7: CPU usage reported by the *top* command during four repetitions of a sequential write experiment

in the MPSoC issuing small requests without cache, where random write demanded 3% more power; with HDD2 when issuing small requests in the PC (up to 7%) and in the MPSoC (up to 23%). With SSDs, sequential write only demanded more power than random in the PC issuing small requests to SSD2 without cache (3%). **Random write demanded more power than sequential write in situations where higher performance was previously observed for random write**: with SSD1 in the PC issuing small requests and using the cache (15%) and in the MPSoC issuing large requests (6%), and with SSD2 in the PC when issuing small requests with cache (3%).

5.2 Small vs. large requests

At the device, large read requests demanded more power than small ones in most experiments: all sequential read tests to SSDs (up to 7%) and all random read tests (up to 16%). Large random writes to HDD2 demanded more power than small random writes — up to 15% — but small random writes to HDD1 without the cache demanded more power than large ones — up to 9%. In other experiments there were no significant differences in the storage device power demand.

Considering the whole system, **results for the MPSoC reflect directly the behavior observed for the storage device**. Large read requests demanded more power than small read requests in most tests in the MPSoC (up to 11%), except sequential read to HDD1 and random read to HDD2, where there were no significant differences. Large random write requests demanded more power than small random write requests with the HDD2 (up to 21%), and small random write demanded 10% more power when accessing the HDD1 without the cache.

However, differently from the device behavior, **small requests demanded more power than large ones in most tests in the PC**. The difference in read experiments was of up to 19%, except random read to HDD1 (no difference). Small write requests demanded more power than large ones in all random experiments to SSDs (up to 22%) and sequential tests without the cache — up to 22%. Small requests resulted in more power demand in the PC despite the fact that performance was lower when using them. Since small requests are processed faster, the process stays blocked for shorter periods, impacting the choices of the DVFS mechanism. Figure 8 shows the CPU frequency during eight new repetitions of the sequential experiments without cache in the PC with the HDD1. The graphs illustrate how the time to process each request can affect the DVFS behavior.

5.3 Read vs. write

The general behavior observed when looking at the power demand of the storage device is that reading demanded more power in HDDs (up to 8%) **and writing demanded more power in SSDs** (up to 12%). When looking at the power demand of the

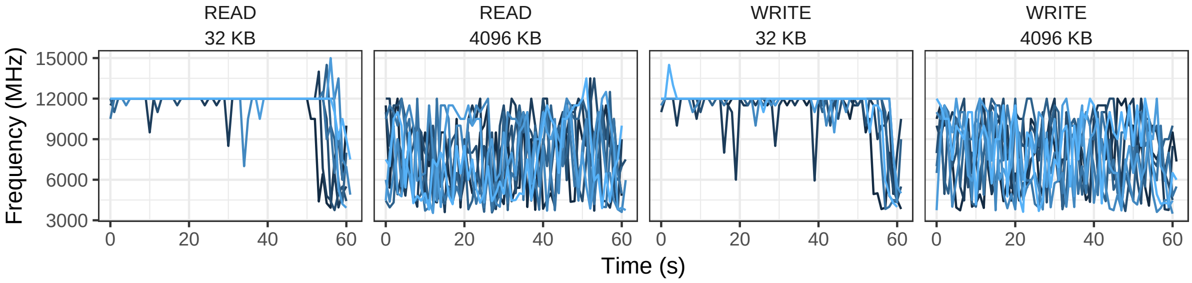


Figure 8: CPU frequency observed during eight tests in the PC with HDD1. From each execution, we plot the sum of the frequencies of the four cores, reported by the *cpufreq-info* command.

whole system, the situation is not always the same as observed for the storage devices. In the MPSoC, issuing small requests to HDD2, reads demanded up to 12% more power than writes. Nonetheless, in the MPSoC with HDD1, write operations demanded up to 11% more power than reads. All other experiments with HDDs, including in the PC, showed no significant difference between read and write results.

With SSDs, **writing demanded more power than reading in some situations where small requests were issued**: in the PC with SSD2 writes demanded up to 14% more power (sequential and random); with SSD1 random writes demanded more power than random reads, in the PC (18%) and in the MPSoC (7%). The opposite — **reading demanding more power than writing — happened in some experiments with large requests**: with SSD1, sequential reads demanded more power than sequential writes, 2.7% in the PC and 7% in the MPSoC; in the MPSoC with SSD2, sequential reads demanded 9% more power than sequential writes. We repeated these experiments in the draco-1 machine twice, one of them removing the effects from the DVFS by setting the CPU frequency to the maximum. The differences observed between read and write experiments were kept, indicating the differences were not caused by the DVFS mechanism. Analyzing power demands by CPU and RAM memory, measured with PCM, we have observed the increases are due to CPU activity.

5.4 Using or not the buffer cache

Considering the power demand of the whole system, **in the MPSoC power demand was always increased by using the cache**: Using the cache results in lower power demand at the storage device in a few random write experiments: issuing large requests to HDD2 in the MPSoC (power is 3% higher when not using the cache) and small requests to HDD1 (up to 9% higher without cache). The opposite — power demand of the device increasing when using the cache — happened in two tests in the MPSoC, issuing sequential small write requests to HDD2 (5%) and random small writes to SSD1 (5%).

up to 11%, except for the random write test with small requests to HDD2, where there was no difference. **The same happened in the PC with large requests to HDD1, SSD1, and SSD2** (up to 6%); **and in the random write test with small requests to HDD1** (4%). Nonetheless, **not using the cache demanded more power in the PC with small**

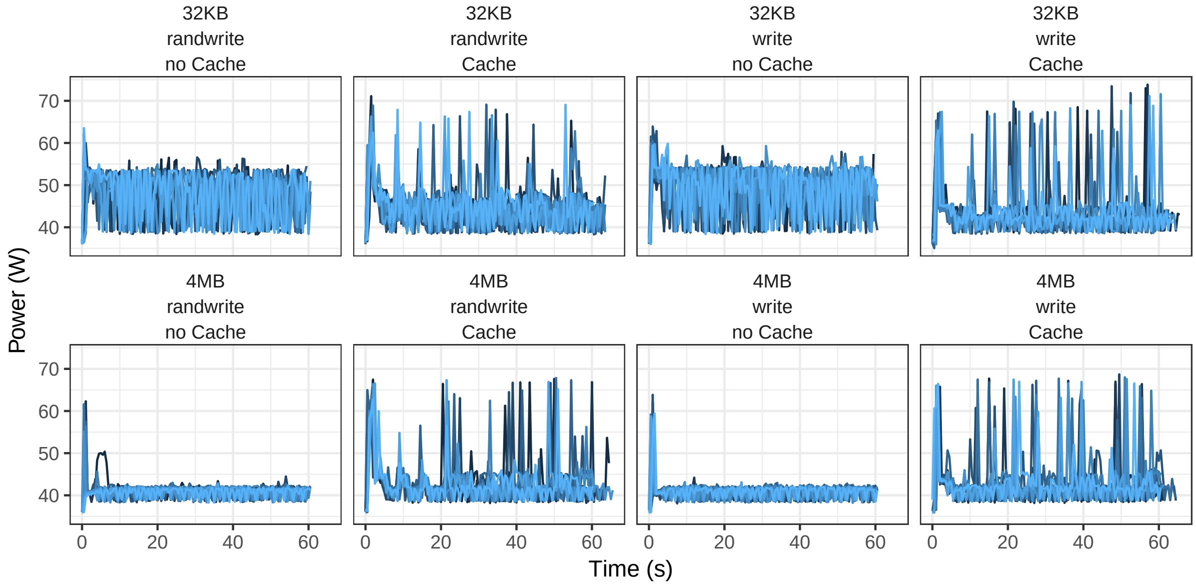


Figure 9: Observed power demand during some experiments on the PC with SDD2.

requests: in all sequential write experiments (up to 16%) and in random writes to SSD2 (7%).

The difference between these scenarios in the PC where using the cache varied the power demand — the request size — is related to the discussion in Section 5.2. In the PC, small requests demanded more power than large ones in most experiments, and this difference happened mostly in tests without cache. Figure 9 shows the power demand during writes in the PC with SSD2. All experiments with cache presented frequent peaks in power demand, but they did not increase the mean power enough to surpass the highest power of the tests with small requests without cache.

6 Energy Efficiency

In this section, we use the bytes per Joule energy efficiency metric, calculated dividing each test's bandwidth (bytes/s) by its average power demand (Watts). This metric allows for a complete comparison between scenarios because it accounts for both performance and power demand.

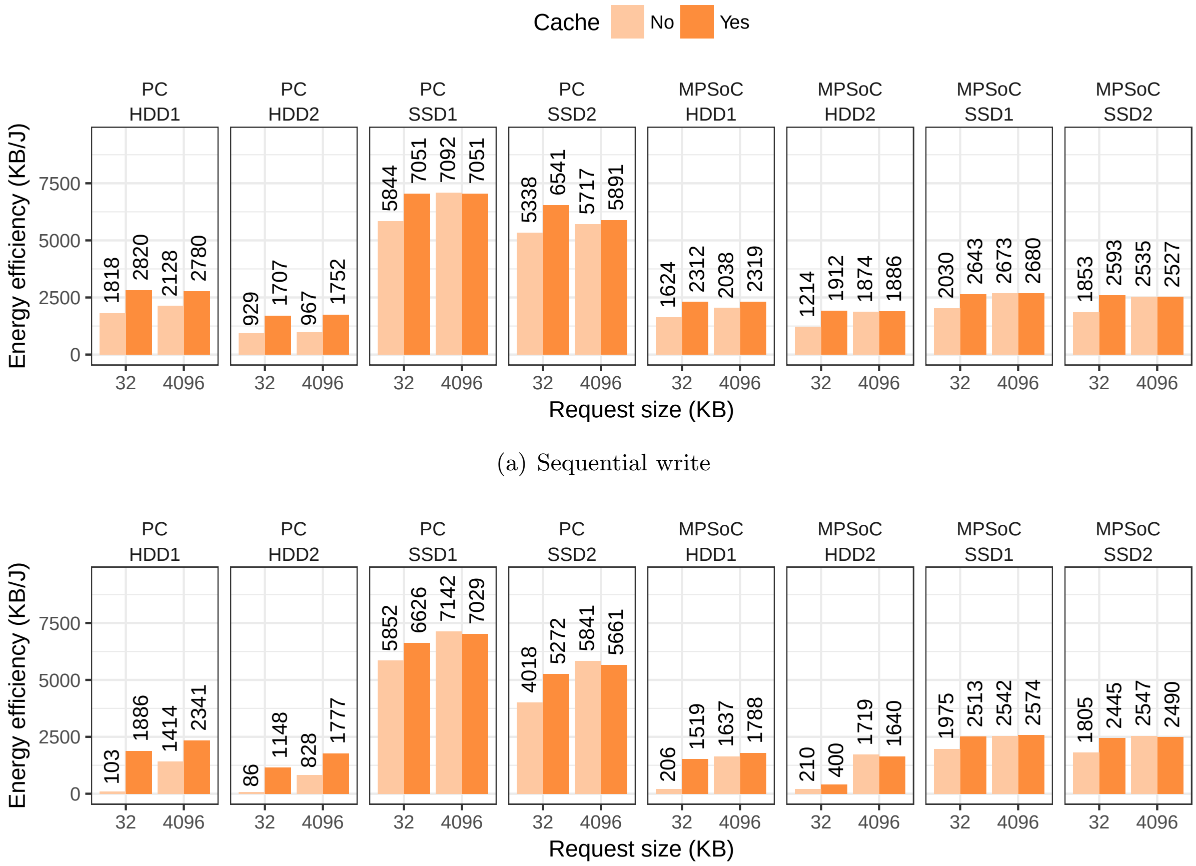
Figure 10 presents energy efficiency results (in KB/J) from write experiments. **Using the cache led to higher energy efficiency in all experiments with small requests, and with large requests to HDD1 in the MPSoC and to both HDDs in the PC**. Sequential write energy efficiency was increased by using the cache in up to 84% in the PC and 57% in the MPSoC. Random write energy efficiency was increased in up to 1737% in the PC and 636% in the MPSoC. Other experiments showed no significant difference in energy efficiency between using or not the cache.

These results can be compared with the performance ones (Section 4). In all situations where energy efficiency was higher using the cache, performance was higher too. On the other hand, as detailed in Section 5.4, power was increased by using the cache in both machines when generating large requests (except in the PC with HDD2), thus in some situations the performance improvement was not large enough to compensate the higher power demand and still result in higher efficiency.

Issuing large requests resulted in higher energy efficiency than small ones in all write experiments without cache. This was a result of better performance obtained with large requests for random writes in both machines (except in the PC with SSD1) and for sequential write tests in the MPSoC, and of higher power demand of small requests for sequential writes in the PC and for random write tests in the PC with SSD1, as mentioned in Section 5.2. Large requests also resulted in higher efficiency **when using cache in random write experiments: with HDDs, SSD2, and in the PC with SSD1**. This happens because large requests resulted in higher performance in random write experiments using the cache with HDDs and in the MPSoC with SSD2, and because small requests demanded more power in random write experiments using the cache in the PC with SSDs. Sequential write energy efficiency was increased in up to 21% in the PC and 54% in the MPSoC, and random write efficiency was increased in up to 1277% in the PC and 717% in the MPSoC.

On the other hand, small requests led to higher efficiency in the PC for the sequential write experiment using the cache with SSD2 (11% higher). As previously detailed, in this experiment issuing small requests also resulted in better performance than issuing large ones.

Higher energy efficiency was achieved in sequential write than in random write experiments when using HDDs because higher performance was achieved for sequential write to HDDs. The difference was of up to 1671% in the PC and 687% in the MPSoC. The exception is the experiment with large requests with cache in the PC with HDD2, where there was no significant difference in efficiency due to not having difference in performance. There was no significant difference between sequential and random write to the SSDs, except **issuing small requests to SSD2: with cache and in the PC without cache** (up to 33% in the PC and 5% in the MPSoC). Differently from other experiments with SSDs, in these cases performance was better for sequential than for random writes.



(a) Sequential write

(b) Random write

Figure 10: Energy efficiency of **write** experiments

In all write experiments, using SSDs resulted in higher energy efficiency than using HDDs — up to 633% for sequential writes in the PC and 67% in the MPSoC, and 6675% for random writes in the PC and 857% in the MPSoC. Higher performance was obtained with the SSDs for random write in both machines and for sequential write in the PC. Additionally, in the MPSoC, SSDs demanded less power than HDDs, as discussed in the beginning of Section 5 (Figure 6).

In all experiments SSDs provided higher energy efficiency when used in the PC (up to 196%**)**. This happened because performance is higher when they are used in the PC. HDD1 provided higher efficiency in the PC than in the MPSoC in sequential write experiments and in random writes with cache (up to 31%) but provided higher efficiency in the MPSoC in random write experiments without the cache (up to 101%). Better performance was reached with HDD1 in the PC for all experiments, but in random write tests without the cache the bandwidth difference was not large enough to compensate the lower power demand of the MPSoC. Finally, HDD2 provided higher efficiency in the MPSoC than in the PC (up to 143%) in all experiments except random write tests using the cache, where efficiency was up to 187% higher in the PC. Those were the only situations where the decrease in performance from the PC to the MPSoC was large enough to compensate the lower power demanded by the MPSoC and result in higher efficiency in the PC.

Figure 11 presents energy efficiency results of read experiments. We can see **issuing large read requests was more energy-efficient in most situations**: up to 1256% with HDDs and 360% with SSDs. In most cases, these differences were due to higher performance. In sequential reads in the PC with HDDs, performance was not better with large requests, but power demand

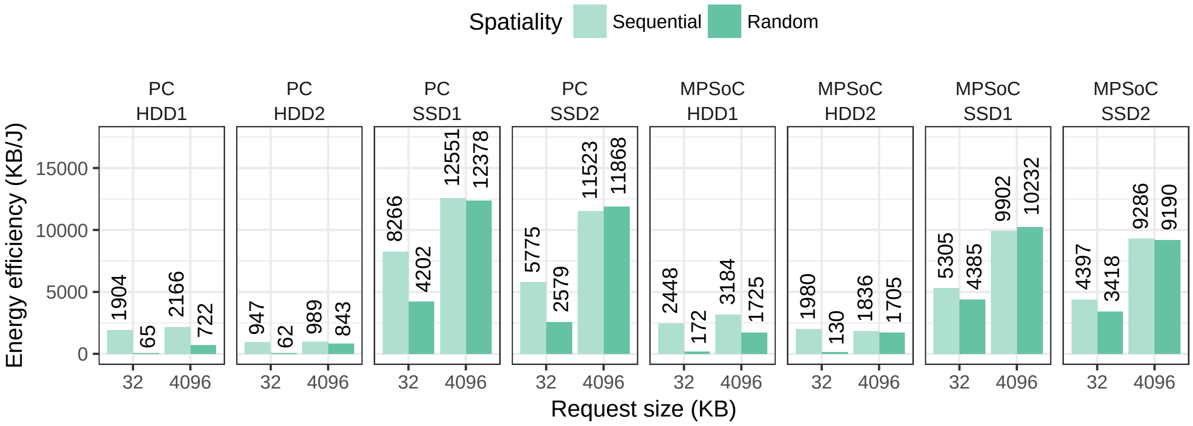


Figure 11: Energy efficiency of **read** experiments

was decreased enough to still result in higher efficiency. The exception was the sequential read test in the MPSoC with HDD2, where small requests led to 8% higher efficiency due to higher performance.

Sequential reads obtained higher energy efficiency than random reads with HDDs (up to 2836%) **and issuing small requests to SSDs** (up to 124%). These differences were caused by higher performance in the sequential read experiments. In tests with large requests to SSDs, differences are not statistically significant, and performance differences were not significant either.

SSDs provided higher energy efficiency than HDDs in all read experiments (up to 6658% in the PC and 3262% in the MPSoC), due to also providing higher performance. **Efficiency for read experiments with HDDs was higher in the MPSoC than in the PC** (up to 166%), despite the fact that in many experiments performance was higher in the PC, because the lower power demand in the MPSoC compensated the bandwidth difference. With SSDs, efficiency was higher in the PC for sequential reads (up to 56%), and random reads with large requests (up to 29%). In random reads with small requests, SSDs have higher efficiency in the MPSoC (up to 32%). In all read experiments, performance with SSDs was higher in the PC, but in random reads with small requests this difference does not compensate the lower power demanded by the MPSoC.

7 Discussion and Guidelines

The results discussed in the previous sections have shown that all devices suffered in performance when used in the MPSoC: writes were up to 1062% faster and reads up to 522% faster in the PC. The power demand analysis has shown SSDs do not demand as much power in the MPSoC than in the PC, as limitations imposed by the MPSoC kept the SSDs from reaching their peak performance.

Workload parameters, such as request size and spatiality, impacted power demand in up to 23%. Since devices demand a small fraction of the power demanded by the system, there were situations where the levels had different behaviors. In many cases, higher I/O performance meant higher power demanded by the system. However, the request serving time affected the DVFS choices, which affects the power demanded by the processor and thus most of the total demanded by the system. Hence in some cases lower performance was accompanied by higher power demand.

Energy efficiency is affected by workloads mainly because they also affect performance. In some situations, using the cache led to up to 1737% higher efficiency, issuing large requests in up to 1277%, and using sequential instead of random in up to 2836%. Using SSDs led to up to

6675% higher energy efficiency than using HDDs. Moreover, when using SSDs energy efficiency was higher in the PC than in the MPSoC— up to 196% for write workloads and 564% for reads.

The results presented in this study indicate that **replacing the traditional server by multiple low-power ones only results in higher energy efficiency if the workload is expected to be mainly of reads, the PC uses HDDs for storage, and the MPSoC uses SSDs**. This replacement could be by 1.4 MPSoC servers (decreasing power demand in up to 51%) to keep the sequential read bandwidth with small requests, and increase performance in up to 61% for sequential reads with large requests and in up to 294% for random reads. **Using the low-power servers also makes sense if both use HDDs for read workloads**: 2.2 MPSoC servers would be required, demanding 20% less power, keeping the same sequential read performance, and increasing the random read bandwidth in up to 120%. Nonetheless, **in both options write workloads would have lower performance**. [6]

In a previous analysis [23], we have compared the same MPSoC to a traditional server, both using SSDs. In that study, the read workload was less intensive than the limitation imposed by the MPSoC, so performance was similar to the traditional server. The write workload was more intensive than the limitation, but not as intense as the one generated here. Finally, that traditional server demanded more power than the PC from this work. Hence that previous analysis concluded that replacing the traditional server by up to four low-power ones would decrease power demand in up to 64% without harming write performance and improving read bandwidth. With results from the most recent analysis, we can see a general rule for the adoption of low-power servers:

- B_{ts} is the bandwidth expected for the traditional server under the expected workload. It will be most likely limited by either the SSD bandwidth or the transfer bus.
- B_{lp} is the bandwidth expected for the low-power server under the expected workload. In the MPSoC the limitation was given by the SATA bus from the SoC. If the bus achieved its theoretical performance peak, then performance could be limited by the RAM memory.
- P_{ts} and P_{lp} are the expected power demanded by the traditional server and by the low-power one, respectively. It is not crucial to have an estimate for the specific workload because, as previously discussed, its impact in power demand is not expected to be of more than 23%.

The replacement of 1 traditional server by $\frac{B_{\text{ts}}}{B_{\text{lp}}}$ low-power alternatives decreases the power demand if $(\frac{B_{\text{ts}}}{B_{\text{lp}}}) \times P_{\text{lp}} < P_{\text{ts}}$. A conclusion to be drawn from our results is that more sophisticated low-power architectures, with more recent processors and higher processing speed, are not necessarily more attractive to work as storage servers — unless their transfer buses and RAM memory are improved as well.

8 Conclusion and Future Work

Among the efforts to improve energy efficiency from large scale systems, low-power ARM-based systems are being considered to replace traditional ones. These low-power alternatives present lower processing capabilities, but offer higher energy efficiency to some classes of applications. In this paper, our goal was to evaluate the use of low-power systems as storage servers.

We conducted a study of I/O performance and energy efficiency, comparing a traditional architecture to a low-power alternative, with HDDs and SSDs under different workloads. Adding

[6] For evaluating replacement options in a fair way, we compare the best results for the PC with the worst results for the MPSoC. For write operations, we always take the results obtained using the buffer cache. We consider non-integer numbers of servers may be acceptable because multiple servers may be replaced at the same time.

on previously obtained results, this analysis provides insight on energy efficiency of I/O intensive tasks, and resulted in guidelines to be used when considering the adoption of low-power storage servers. These guidelines can be used to other systems than the ones used for this study. Possible directions for future work including the network layer to the storage servers analysis.

Acknowledgments

This research has received funding from the EU H2020 Programme and from MCTI/RNP-Brazil under the HPC4E Project, grant agreement number 689772. The authors would like to thank Aishameriane Schmidt and Laércio Pilla from the Federal University of Santa Catarina.

References

[1] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," Tech. Rep., 2008.

[2] E. L. Padoin, L. L. Pilla, M. Castro, F. Z. Boito, P. O. A. Navaux, and J.-F. Mehaut, "Performance/energy trade-off in scientific computing: The case of ARM big.LITTLE and Intel Sandy Bridge," *IET Computers & Digital Techniques*, vol. 2, no. 3, pp. 1–14, 2014.

[3] Ambedded Technology Co., "Software defined storage powered by arm based microserver cluster," http://www.ambedded.com.tw/solutions.php, accessed: Oct 2016.

[4] Y. Sverdlik, "Paypal deploys arm servers in data centers," 2015, accessed: Oct 2016. [Online]. Available: http://www.datacenterknowledge.com/archives/2015/04/29/paypal-deploys-arm-servers-in-data-centers/

[5] R. R. Chandrasekar, A. Venkatesh, K. Hamidouche, and D. K. Panda, "Power-check: An energy-efficient checkpointing framework for HPC clusters," in *Proceedings - IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015.* IEEE, 2015, pp. 261–270.

[6] A.-C. Orgerie, M. Dias de Assuncao, and L. Lefevre, "A Survey on Techniques for Improving the Energy Efficiency of Large Scale Distributed Systems," *ACM Computing Surveys*, pp. 1–35, 2013.

[7] E. V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving disk energy in network servers," *Proceedings of the 17th annual international conference on Supercomputing - ICS '03*, p. 86, 2003.

[8] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: dynamic speed control for power management in server class disks," *30th Annual International Symposium on Computer Architecture, 2003. Proceedings.*, 2003.

[9] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes, "Hibernator: helping disk arrays sleep through the winter," *Proceedings of the 20th ACM symposium on Operating systems principles SOSP'05*, vol. 39, p. 177, 2005.

[10] W. Y. Lee, "Energy-saving dvfs scheduling of multiple periodic real-time tasks on multi-core processors," in *Proceedings...*, International Symposium on Distributed Simulation and Real Time Applications. IEEE Computer Society, 2009, pp. 216–223.

[11] S. Hosseinimotlagh, F. Khunjush, and S. Hosseinimotlagh, “A cooperative two-tier energy-aware scheduling for real-time tasks in computing clouds,” in *Proceedings...*, Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP). IEEE, Feb 2014, pp. 178–182.

[12] A. Younge, G. von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, “Efficient resource management for cloud computing environments,” in *International Conference on Green Computing*. IEEE, 2010, pp. 357–364.

[13] J. Peraza, A. Tiwari, M. Laurenzano, L. Carrington, and A. Snavely, “PMaC’s green queue: a framework for selecting energy optimal DVFS configurations in large scale MPI applications,” *Concurrency and Computation: Practice and Experience*, pp. 1–20, 2013.

[14] R. Ge, X. Feng, and X. H. Sun, “SERA-IO: Integrating energy consciousness into parallel I/O middleware,” in *Proceedings - 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012*, 2012, pp. 204–211.

[15] P. Shang and J. Wang, “A novel power management for CMP systems in data-intensive environment,” *Proceedings - 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2011*, pp. 92–103, 2011.

[16] T. Saito, K. Sato, H. Sato, and S. Matsuoka, “Energy-aware I/O optimization for checkpoint and restart on a NAND flash memory system,” *Proceedings of the 3rd Workshop on Fault-tolerance for HPC at extreme scale - FTXS ’13*, p. 41, 2013.

[17] S. Ibrahim, T.-D. Phan, A. Carpen-Amarie, H.-E. Chihoub, D. Moise, and G. Antoniu, “Governing energy consumption in hadoop through cpu frequency scaling: An analysis,” *Future Generation Computer Systems*, vol. 54, pp. 219–232, 2016.

[18] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, “Exploiting spatio-temporal tradeoffs for energy-aware MapReduce in the cloud,” *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1737–1751, 2012.

[19] H. Amur, J. Cipar, and V. Gupta, “Robust and Flexible Power-Proportional Storage,” *Proceedings of the 1st ACM symposium on Cloud computing*, pp. 217–228, 2010.

[20] J. Kim, J. Chou, and D. Rotem, “Energy proportionality and performance in data parallel computing clusters,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6809 LNCS, pp. 414–431, 2011.

[21] M. Nijim, A. Manzanares, X. Ruan, and X. Qin, “Hybud: An energy-efficient architecture for hybrid parallel disk systems,” *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, vol. 0845257, no. 2005, 2009.

[22] B. Welch and G. Noer, “Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions,” in *2013 IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST)*, 2013, pp. 1–12.

[23] V. Machado, A. Braga, N. Rampon, J. Bez, F. Boito, R. Kassick, E. Padoin, J. Diaz, J.-F. Mhaut, and P. Navaux, “Towards energy-efficient storage servers,” *Proceedings of the 32nd Annual ACM Symposium on Applied Computing (SAC)*, pp. 1554–1559, 2017.

[24] H. W. Lilliefors, “On the kolmogorov-smirnov test for normality with mean and variance unknown,” *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[25] N. Feltovich, “Nonparametric tests of differences in medians: Comparison of the wilcoxon–mann–whitney and robust rank-order tests,” *Experimental Economics*, vol. 6, no. 3, pp. 273–297, 2003.

[26] O. J. Dunn, “Multiple comparisons among means,” *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[27] Intel Corporation, “Processor counter monitor,” https://github.com/opcm/pcm, accessed: Sep 2017.

[28] ——, “Intelligent platform management interface,” http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html, accessed: Sep 2017.

[29] L. McVoy and C. Staelin, “Lmbench: Portable tools for performance analysis,” in *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ser. ATEC ’96. Berkeley, CA, USA: USENIX Association, 1996, pp. 23–23. [Online]. Available: http://dl.acm.org/citation.cfm?id=1268299.1268322

[30] Linux-Sunxi Community, “Sata - current state,” http://linux-sunxi.org/SATA#Current_state, accessed: Sep 2017.