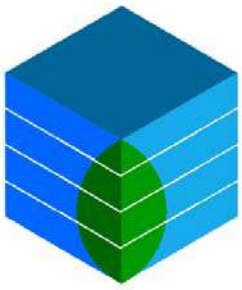


BigStorage

BigStorage: MSCA-ITN-2014-ETN-642963

Storage-based convergence between HPC and Cloud to handle Big Data

Deliverable number	D5.2
Deliverable title	WP5. Energy savings – Final Report
Main Authors	Y. Alforov, G. Antoniu, A. Brinkmann, T. Cortes, S. Ibrahimi, M. Kuhn, J. Kunkel, T. Ludwig, Y. Taleb

Grant Agreement number	642963
Project ref. no	MSCA-ITN-2014-ETN-642963
Project acronym	BigStorage
Project full name	BigStorage: Storage-based convergence between HPC and Cloud to handle Big Data
Starting date (dur.)	1/1/2015 (48 months)
Ending date	31/12/2018
Project website	http://www.bigstorage-project.eu

Coordinator	María S. Pérez
Address	Campus de Montegancedo sn. 28660 Boadilla del Monte, Madrid, Spain

Reply to	mperez@fi.upm.es
Phone	+34- 910672857

Document Identifier	D5.2
Class Deliverable	Public Report
Version	1.0
Document due date	M48
Submitted	31.12.2018
Responsible	André Brinkmann, JGU
Reply to	brinkman@uni-mainz.de
Document status	final
Nature	R(Report)
Dissemination level	(Public)
WP/Task responsible(s)	André Brinkmann, JGU
Contributors	Y. Alforov, G. Antoniu, A. Brinkmann, T. Cortes, S. Ibrahimi, M. Kuhn, J. Kunkel, T. Ludwig, Y. Taleb
Distribution List	Consortium Partners
Reviewers	All advisors
Document Location	http://bigstorage-project.eu/index.php/deliverables

Executive Summary

Big Data processing on HPC and Cloud infrastructures has an increasing effect on energy consumption. This motivated our choice to dedicate a full WP to energy-related aspects that are present at all levels of the data storage and processing stack. Research on energy efficiency of storage systems mostly focused on energy-proportional storage (unused systems are turned off and the energy consumption is proportional to bandwidth requirements) and on "integrating" new storage technologies, e.g., flash devices. Techniques to reduce the storage footprint by deduplication or compressions have been extensively studied, nevertheless very little is known on their energy consumption impact as well as on the potential impact of using hints to the storage system. Similar observations hold for the energy-consistency trade-offs. Research work focused on exploring consistency-performance and consistency-availability trade-offs in cloud environments but has not included energy aspects. There is also no work on how much energy a supercomputer consumes while running a scientific simulation when adopting different data management approaches or how to use simulation to predict energy-consumption before building or modifying a data center architecture.

Work package 5 therefore focused on

- the development of smarter IO interfaces, so that semantic information can be used to determine the best ways to decrease the storage footprint and to intelligently place and move data,
- the impact of reducing the amount of stored data on energy efficiency by investigating new techniques to reduce the storage footprint and to evaluate the trade-off between energy savings and costs involved in reducing the storage footprint, and
- the impact of the most common Cloud and HPC usage scenarios, including new storage architectures, like Flash, NVRAM, or memory-based storage systems.

Main results of this work package have been published at ICDCS, IEEE Cluster, and SBAC-PAD [TalebIAT17][TalebIAT17b][AlforovNKKL18]. Results have been able to:

- reveal that well-known in-memory storage systems are scalable for read-only applications, but exhibit non-proportional power consumption. We also find that the current replication scheme implemented in the studied RAMCloud-environment limits the performance and results in high energy consumption and surprisingly show that replication can also play a negative role in crash-recovery,
- provide empirical evidence of the impact of client's locations and network impact on the performance and energy consumption of in-memory storage systems, and
- propose data reduction strategies that can decrease the data volume transferred and stored by as much as 80% in some cases, consequently leading to significant savings in storage and networking costs.

The experimental results as well as derived frameworks can help to better understand energy consumption related to the storage system as well as directly optimize energy usage and therefore help to reduce overall data center costs.

Document Information

IST Project Number	MSCA-ITN-2014-ETN-642963	Acronym	BigStorage
Full Title	BigStorage: Storage-based convergence between HPC and Cloud to handle Big Data		
Project URL	http://www.bigstorage-project.eu		
Document URL	http://bigstorage-project.eu/index.php/deliverables		
EU Project Officer	Szymon Sroda		

Deliverable	Number	D1.12	Title	Benchmarks defined & tested
Workpackage	Number	WP1	Title	Use Case Analysis and Evaluation

Date of Delivery	Contractual	M48	Actual	31.12.2018
Status	version 1		final ■	
Nature	report			
Dissemination level	public			

Authors (Partner)	Y. Alforov, G. Antoniu, A. Brinkmann, T. Cortes, S. Ibrahimi, M. Kuhn, J. Kunkel, T. Ludwig, Y. Taleb			
Responsible Author	Name	André Brinkmann	E-mail	brinkman@uni-mainz.de
	Partner	JGU	Phone	+49 6131 3926390

Abstract	Big Data processing on HPC and Cloud infrastructures has an increasing effect on energy consumption. This motivated our choice to dedicate a full WP of the BigStorage ITN to energy-related aspects that are present at all

(for dissemination)	levels of the data storage and processing stack. Main results of energy-related research in BigStorage have been able to reveal that well-known in-memory storage systems are scalable for read-only applications, but exhibit non-proportional power consumption provide empirical evidence of the impact of client's locations and network impact on the performance and energy consumption of in-memory storage systems, and propose data reduction strategies that can decrease the data volume transferred and stored by as much as 80% in some cases, consequently leading to significant savings in storage and networking costs.
Keywords	Benchmarking, Energy, In-Memory Storage, HPC data formats

Version	Modification(s)	Date	Author(s)
01	First proposal	<22/12/2018>	André Brinkmann
02	Final version reviewed	<31/12/2018>	Reviewed by María S. Pérez

Project Consortium Information

Participants		Contact
Universidad Politécnica de Madrid (UPM), Spain		María S. Pérez Email: mperez@fi.upm.es
Barcelona Supercomputing Center (BSC), Spain		Toni Cortes Email: toni.cortes@bsc.es
Johannes Gutenberg University (JGU) Mainz, Germany		André Brinkmann Email: brinkman@uni-mainz.de
Inria, France		Gabriel Antoniu Email: gabriel.antoniu@inria.fr Adrian Lebre Email: adrien.lebre@inria.fr
Foundation for Research and Technology - Hellas (FORTH), Greece		Angelos Bilas Email: bilas@ics.forth.gr
Seagate, UK		Sai Narasimhamurthy Email:sai.narasimhamurthy@seagate.com

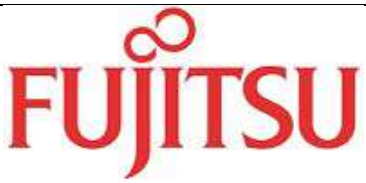
DKRZ, Germany		Thomas Ludwig Email: ludwig@dkrz.de
CA Technologies Development Spain (CA), Spain		Victor Muntes Email: Victor.Muntes@ca.com
CEA, France		Jacque Charles Lafoucriere Email: Charles.LAFOUCRIERE@CEA.FR
Fujitsu Technology Solutions GMBH, Germany		Sepp Stieger Email: sepp.stieger@ts.fujitsu.com

Table of Contents

Introduction

Big Data processing on HPC and Cloud infrastructures has an increasing effect on energy consumption. This motivated our choice to dedicate a full WP to energy-related aspects that are present at all levels of the data storage and processing stack. Research on energy efficiency of storage systems mostly focused on energy-proportional storage (unused systems are turned off and the energy consumption is proportional to bandwidth requirements) and on "integrating" new storage technologies, e.g., flash devices. Techniques to reduce the storage footprint by deduplication or compressions have been extensively studied, nevertheless very little is known on their energy consumption impact as well as on the potential impact of using hints to the storage system. Similar observations hold for the energy-consistency trade-offs. Research work focused on exploring consistency-performance and consistency-availability trade-offs in cloud environments but has not included energy aspects. There is also no work on how much energy a supercomputer consumes while running a scientific simulation when adopting different data management approaches or how to use simulation to predict energy-consumption before building or modifying a data center architecture.

A main focus has been set on modern in-memory storage systems like RAMCloud. Most large popular web applications, like Facebook and Twitter, have been relying on large amounts of in-memory storage to cache data and offer a low response time. As the main memory capacity of clusters and clouds increases, it becomes possible to keep most of the data in the main memory. This motivates the introduction of in-memory storage systems. While prior work has focused on how to exploit the low-latency of in-memory access at scale, there is very little visibility into the energy-efficiency of in-memory storage systems. Even though it is known that main memory is a fundamental energy bottleneck in computing systems (i.e., DRAM consumes up to 40% of a server's power). By means of experimental evaluation, we have studied the performance and energy-efficiency of RAM-Cloud – a well-known in-memory storage system. We reveal that although RAMCloud is scalable for read-only applications, it exhibits non-proportional power consumption. We also find that the current replication scheme implemented in RAMCloud limits the performance and results in high energy consumption. Surprisingly, we show that replication can also play a negative role in crash-recovery.

More recently, in-memory databases are able to leverage high-performance networks, e.g., InfiniBand. To be able to leverage these systems, it is essential to understand the trade- offs induced by the use of high-performance networks. We aimed to provide empirical evidence of the impact of client's location on the performance and energy consumption of in-memory storage systems. Through a study carried on RAMCloud, we focused on two settings: 1) clients are collocated within the same network as the storage servers (with InfiniBand interconnects); 2) clients access the servers from a remote network, through TCP/IP. We compare and discuss aspects related to scalability and power consumption for these two scenarios which correspond to different deployment models for applications making use of in-memory cloud storage systems.

Moving from Cloud architectures to HPC systems, we see a comparable deluge of data generated by scientific applications and simulations or large-scale experiments [GrawinkelNMPBS15]. The upscaling of supercomputers generally results in a dramatic increase of their energy consumption. We argue that techniques like data compression can lead to significant gains in terms of power

efficiency by reducing both network and storage requirements. However, any data reduction is highly data specific and should comply with established requirements. Therefore, unsuitable or inappropriate compression strategies can utilize more resources and energy than necessary. To that end, we propose a novel methodology for achieving on-the-fly intelligent determination of energy efficient data reduction for a given data set by leveraging state-of-the-art compression algorithms and meta data at application-level I/O. We motivate our work by analyzing the energy and storage saving needs of data sets from real- life scientific HPC applications and review the various lossless compression techniques that can be applied. We find that the resulting data reduction can decrease the data volume transferred and stored by as much as 80% in some cases, consequently leading to significant savings in storage and networking costs.

HW Trade-Offs

Large scale web applications, such as Facebook, are accessed by billions of users [NishtalaFGK+13]. These applications must keep low response times even under highly concurrent accesses. To do so, they strongly rely on main memory storage through caching [NishtalaFGK+13] [AtikogluXF+12].

The increasing size of main memories has led to the advent of new types of storage systems. These systems propose to keep all data in distributed main memories [OusterhoutGG+15] [DragojevicNCH14]. In addition to leveraging DRAM speed, they can offer: low-latency by relying on high speed networks (e.g., InfiniBand) [OusterhoutGG+15][DragojevicNCH14][LimHAK14]; durability by replicating data synchronously to DRAM or asynchronously to disk [OusterhoutGG+15][DragojevicNCH14][OnargoRSOR11][ZhangDC16]; strong consistency and transactional support [OusterhoutGG+15][LeePKMO15]; and scalability [OusterhoutGG+15], [LimHAK14]. RAMCloud [OusterhoutGG+15] is a system providing the aforementioned features. It is now used in various fields: analytics [TinnefeldKGBRSP13] or used as a low-latency storage for SDNs [BerdeGHH+14][KablanAKL17], or for scientific workflows [BlomerG15].

While energy-efficiency has long been an important goal in storage systems, e.g., disk-based storage [GrawinkelSBHP11][GrawinkelNB11][ThereskaDN11][AmurCGGKS10][WongA12] and flash-based storage [AndersenFKPTV09], prior literature has not investigated enough the energy-efficiency of in-memory storage systems. Some studies reported that DRAM-based main memories consume from 25% up to 40% of a server's total power consumption [UdipiMCBDJ10].

Given the increasing concerns about data center energy and the prevalence of in-memory storage systems, we aim to provide a clearer understanding of the main factors impacting performance and energy-efficiency of in-memory storage systems. We carry out an experimental study on the Grid'5000 [BaloukAC+13] testbed and use the RAMCloud in-memory key-value store. We design and run various scenarios that help us to identify the main performance bottlenecks and sources of energy-inefficiency.

More precisely we answer the following questions:

- What is the energy footprint of peak performance of RAMCloud? We reveal issues of power non- proportionality that appear with read-only applications.
- What is the performance and energy footprint of read-update workloads? We focus on these workloads (mixed reads and updates) as they are prevalent in large scale Web applications [AtikogluXF+12].
- How does RAMCloud's replication scheme impact performance and energy? Though it is supposed to be transparent in RAMCloud, we find that replication can be a major performance and energy bottleneck.
- What is the overhead of RAMCloud's crash-recovery in terms of availability and energy consumption? In addition, we study how replication affect these metrics. Surprisingly, we show that replication can play a negative role in crash-recovery.
- What can be improved in RAMCloud, and for in-memory storage? We discuss possible improvements for RAM- Cloud and derive some optimizations that could be applied in in-memory storage systems.

Additionally, high-performance networks are becoming more accessible in the Cloud. For example, Amazon is recently offering VM instances with the support of enhanced networking (i.e., by using the Amazon EC2 Elastic Network Adaptor with up to 20 Gbps of aggregate network bandwidth). However, clients can access the storage in two setups: (1) in an HPC-like setup, clients access the storage system through the local network; (2) For Internet-based applications, which are typically hosted in the Cloud, clients access the system remotely through the typical TCP/IP stack. In this case clients will have slower access to the system, and more importantly it is not clear whether they will benefit or not from the fast access of data stored in memory.

We argue that the type of network used by clients to access the storage system impacts the performance. Moreover, given that main memories of DRAM-based servers consume between 25% up to 40% of the total energy of the servers, it is as vital to understand the network impact on energy efficiency as on performance.

Through an experimental study carried on GRID'5000, we investigate the performance and energy consumption of RAMCloud storage system in both cases where clients are sharing the same network as the storage system and when they are not. We find that the location of the clients could be a scalability and performance limiting factor of in-memory storage systems. For example, 10 RAMCloud servers obtain a throughput of up to 2 Million op/s when accessed by 90 clients who are using InfiniBand, while the throughput is limited to 200 Kop/s for the same settings but when the clients are connected to the cluster using TCP/IP.

It is important to note that, although RAMCloud is used as an experimental platform in this work, our target is more general. Our findings can serve as a basis to understand the behavior of other in-memory storage systems sharing the same features and provide guidelines to design energy-efficient in-memory storage systems.

Background

A representative system: RAMCloud

Ideally, the main attributes that in-memory storage systems should provide are performance, durability, availability, scalability, efficient memory usage, and energy efficiency. Most of today's systems target performance and memory efficiency [LimHAK14][MitchellGL13][FanAK13]. Durability and availability are also important as they free up application developers from having to backup in-memory data in secondary storage and handle the synchronization between the two levels of storage. On the other hand, scalability is vital, especially with today's high-demand Web applications. They are accessed by millions of clients in parallel. Therefore, large scale clustered storage became a natural choice to cope with such high demand [NishtalaFGK+13].

In contrast with most of the recent in-memory storage systems, RAMCloud main claims are performance (low-latency), durability, scalability, and memory efficiency. The other closest system to provide all these features to be found in the literature is FaRM [DragojevicNCH14]. Unfortunately, it is neither open-source nor publicly available.

The RAMCloud Storage System

Architecture: RAMCloud's cluster consists of three entities: a coordinator maintaining metadata information about storage servers, backup servers, and data location; a set of storage servers that expose their DRAM as storage space; and backups that store replicas of data in their DRAM temporarily and spill it to disk asynchronously. Usually, storage servers and backups are collocated within a same physical machine [OnargoRSOR11].

Data management: Data in RAMCloud is stored in a set of tables. Each table can span multiple storage servers. A server uses an append-only log-structured memory to store its data and a hash-table to index it. The log-structured memory of each server is divided into 8MB segments. A server stores data in an append-only fashion. Thus, to free unused space a cleaning mechanism is triggered whenever a server reaches a certain memory utilization threshold. The cleaner copies a segment's live data into the free space (still available in DRAM) and removes the old segment.

Data durability and availability: Durability and availability are guaranteed by replicating data to remote disks. More precisely, whenever a storage server receives a write request, it appends the object into its latest free segment, and forwards a replication request to the backup servers randomly chosen for that segment. The server waits for acknowledgements from all backups to answer a client's update/insert request. Backup servers will keep a copy of this segment in DRAM until it fills. Only then, they will flush the segment to disk and remove it from DRAM.

Fault-tolerance: For each new segment, a random backup in the cluster is chosen in order to have as many machines performing the crash-recovery as possible. At run time each server will compute a will where it specifies how its data will be partitioned if it crashes. If a crashed server is detected, then each server will replay the segments that were assigned to it according to the crashed server's will. As the segments are written to a server's memory, they are replicated to new backups. At the end of the recovery the segments are cleaned from old backups.

Experimental Settings

Metrics

We took as main metrics the throughput of the system (i.e., the number of requests served per second) and the power consumption of nodes running RAMCloud. We used the energy efficiency metric as well [WongA12], i.e., (Number of requests served)/joule.

Platform

The experiments were performed on the Grid'5000 [BaloukAC+13] testbed. The Grid'5000 platform provides researchers with an infrastructure for large-scale experiments. It includes 9 geographical sites spread across French territory and one located in Luxembourg.

We used Nancy's site nodes to carry out our experiments. More specifically, the nodes we used have 1 CPU Intel Xeon X3440, 4 cores/CPU, 16GB RAM, and a 298GB HDD. Additionally, each node includes an Infiniband-20G and a Gigabit Ethernet cards. We have chosen these nodes as they offer capability to monitor power consumption: 40 of these nodes are equipped with Power Distribution Units (PDUs), which allow to retrieve power consumption through an SNMP request. Each PDU is mapped to a single machine, allowing fine grain power retrieval. We run a script on each machine which queries the power consumption value from its corresponding PDU every second. We start the script right before running the benchmark and stop it after all clients' finish.

RAMCloud's configuration: Throughout all our experiments we make sure to reserve the whole cluster, which consists of 131 nodes, to avoid any interference with other users of the platform. We dedicate the 40 nodes equipped with PDUs to run RAMCloud's cluster, i.e., master and backup services. One node is used to run the coordinator service. The remaining 90 nodes are used as clients. We have fixed the memory used by a RAMCloud server to 10GB and the available disk space to 80GB. We have fixed the memory and disk to much larger sizes than the workloads used.

We configured RAMCloud with the option ServerSpan equal to the number of servers. As RAMCloud does not have a smart data distribution strategy, this option is used to manually decide how many servers each table will span. Data is then distributed uniformly.

Throughout all our experiments, we used RAMCloud's Infiniband transport only. The impact of the network on performance and energy efficiency has been studied in [TalebIAT17].

Benchmark

We used the industry standard Yahoo! Cloud Serving Benchmark (YCSB) benchmarking framework [CooperSTRS10]. YCSB is an open and extensible framework that allows to mimic real-world workloads such as large-scale web applications, to benchmark key-value store systems. YCSB supports a large number of databases and key-value stores, which is convenient for comparing different systems.

To run a workload, one needs to fill the data-store first. It is possible to specify the request distribution, in our case we use uniform distribution. Executing the workload consists of running clients with a given workload specification. We used three basic workloads provided by YCSB:

Workload A which is an update-heavy workload (50% reads, 50% updates), Workload B which is a read-heavy workload (95% reads, 5% updates), and Workload C which is a read-only workload. This combination of workloads has already been used in other systems evaluations [LimHAK14] [ZhangDC16][MitchellGL13]. Other workloads could be used to assess different features of the system, e.g., scans to evaluate the indexing mechanism, however, we leave it as future work.

When assessing RAMCloud's peak performance, we use a workload of 5M records of 1KB, and 10M requests per client. Running the benchmark consists of launching simultaneously one instance of a YCSB client on each client node. With 30 clients it corresponds to 300M requests which represents 286GB of data requested per run. In some runs, e.g., when running 30 clients with a single RAMCloud node, the execution time reaches in average 4300 seconds, which we believe outputs enough and representative results to achieve our goals.

For the rest of the experiments, where we use update-workloads, we pre-load 100K records of 1KB in the cluster. Each client issues 100K requests, which corresponds to the total number of records. Having each client generate 100K requests results in having 1M requests with 10 clients for example, and 9M requests with 90 clients, which corresponds to 8.58GB of data requested per run. With workload A, it corresponds as well to 4.3GB of data inserted per run. Therefore, we avoid saturating the main memory (and disk when using replication) of servers and trigger the cleaning mechanism.

In our figures, each value is an average of 5 runs with the corresponding error bars. When changing the cluster configuration (i.e., number of RAMCloud servers), we remove all the data stored on servers as well as in backups, then we restart all RAMCloud servers and backups in the cluster to avoid any interference with prior experiments. Overall, we made roughly 3000 runs with a total run time of approximately 1000 hours.

The energy footprint of the peak performance

In this section we present our experiments on understanding of the maximum achievable performance. It is important since it gives us a landmark that can help us compare with our next experiments and identify potential performance bottlenecks. Furthermore, given that baseline we can compute energy-efficiency as well.

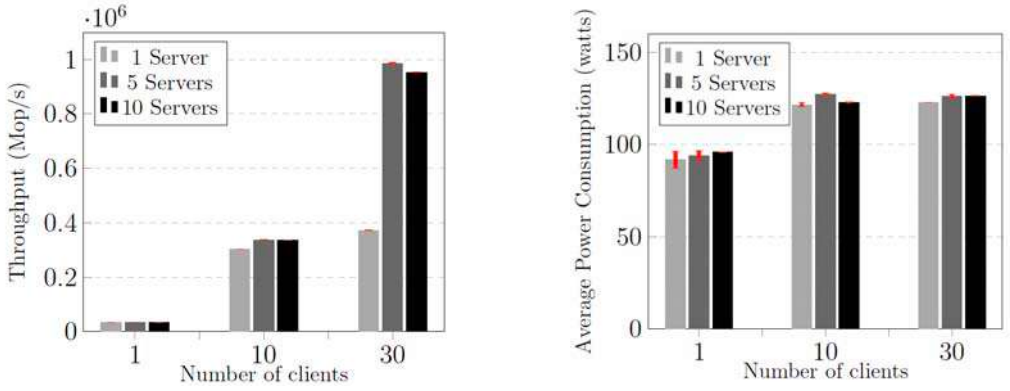
Methodology

To allow RAMCloud to deliver its maximum performance, we configured our experiments as follows:

1. Disabling replication to avoid any communication between RAMCloud servers. In that way we have server-client communication only.
2. We use read-only workloads to avoid write-write race condition, and more importantly to prevent appending data to memory and triggering the cleaning mechanism of RAMCloud.
3. We use Infiniband network.
4. We make sure that the dataset is always smaller than memory capacity.

5. We distribute data uniformly (i.e., at RAMCloud cluster level) and requests (i.e., at the clients' level) over the cluster to distribute work equally among servers and avoid hotspots.
6. We use a single client per machine to avoid any network interference at the NIC level of a machine or any CPU contention between client processes within a same machine.

We varied the RAMCloud cluster size from a single server to 5 and 10 servers. We did as well vary the clients' number from one to 10 and 30 clients. We used the workload consisting of 5M objects and 10M read-only request per client.



a) b)

Figure 1: The aggregated throughput (a) and average power consumption per server (b) as a factor of the cluster size

The maximum throughput per server: Figure 1a) shows the total aggregated throughput for this scenario. In one node experiment, the system scales up to 10 clients until it reaches its limit at 30 clients for a throughput of 372Kreq/s (this is in line with the results presented in [RumbleKO14]). Increasing the number of servers to 5 improves the total throughput for 10 clients and 30, achieving linear scalability. However, increasing the number of servers from 5 to 10 does not bring any improvement to the aggregated throughput achieved, which is due to the clients' limited rate at this point.

The corresponding power consumption: Figure 1b shows the average power consumption. With one server and a single client the average power is 92W. It increases to 93W and 95W for 5 and 10 servers respectively. Even with smaller amount of work for 5 and 10 servers we can see that the power consumption remains at the same level. The same behavior happens when increasing the load. For 10 clients the average power is between 122W and 127W. More surprisingly for 30 clients it stays at the same levels, i.e., 122W and 127W. This suggests that the servers are used at the same level (i.e., CPU) under different loads.

Non-proportional power consumption?: To confirm or invalidate this statement we looked into CPU usage of individual servers when running the workload. Table I displays the minimum and maximum CPU usage values. First, we notice that the difference between the minimum and maximum values is not bigger than 2% for all scenarios. What is striking is to see that the

difference between CPU usage of one node and 5 and 10 nodes is very small. When digging into details we discovered that RAMCloud hogs one core per machine for its polling mechanism, i.e., polling new requests from the NIC and delivering them to the application to achieve low-latency. Since we are using 4-core machines, this ends up using 25% of CPU all times, even without any clients.

Surprisingly, increasing the RAMCloud cluster size from 1 to 5 servers increases the aggregated throughput by 10% only while keeping the same CPU usage per node. We think this is an overhead to handle all concurrent incoming requests. By looking at the scenario of a single node, if we look carefully at the cases of 10 clients and 30 clients, we can see a difference of 23% in throughput for a 1% difference in CPU usage. This suggests that this issue relates to the threads' handling. First, each additional running server has a dedicated polling-thread, which in our case translates to 25% of additional CPU usage. Moreover, not all threads are effectively utilized. For instance, the aggregated throughput and the average CPU usage per node are very similar for 1, 5, and 10 servers when servicing 10 clients. To mitigate the energy non-proportionality issue, one can think of carefully tuning the number of nodes in a cluster to meet the needs of the workloads.

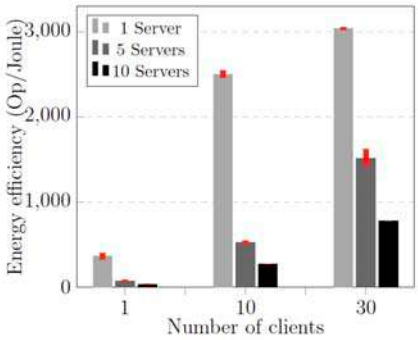


Figure 2: The energy efficiency of different RAMCloud cluster sizes when running different number of clients.

Figure 2 shows the energy-efficiency: As expected the energy efficiency is at its highest with a single server and with the largest number of clients. With 5 servers, the energy-efficiency can barely reach half of the one of a single server. Further increasing the RAMCloud cluster size to 30 decreases the energy efficiency by a factor of 7.6x compared to the one of a single server.

Finding 1: RAMCloud is scalable in throughput for read-only applications. However, we find that it can have non-proportional power consumption, i.e., the system can deliver different values of throughput for the same power consumption. The reason is that servers reach their maximum CPU usage before reaching peak performance. Energy-proportionality can be achieved by adapting the number of servers according to their peak performance.

The Energy Footprint with Read-Update Workloads

As demonstrated in [AtikogluXF+12], today's workloads are read-dominated workloads with a GET/SET ratio of 30:1. Hence, the questions we would like to answer are: How good is RAMCloud in terms of performance when running realistic workloads? Does it scale as well as it does for

read-only workloads? How energy efficient is it when running realistic workloads? What are the architectural choices that can be improved?

Methodology

We used the same methodology as before except that we changed the workloads to have both read-update (95% reads, 5% updates) and update-heavy (50% reads, 50% up- dates) workloads. We recall that replication is disabled. It is important to note that we use different number of servers and clients compared to the previous experiment since the goal is different. Thereby, we do not compare the following scenario with the previous one. We use from 10 to 40 servers and from 10 to 90 client nodes. For space's sake we do not show all scenarios, but they all converge towards the same conclusions we give.

Table I: Total aggregated throughput (Kop/s) of 10 servers when running different with different numbers of clients. We used the three YCSB by default workloads A, B, and C

Clients \ Workload	A	B	C
	avg — err	**avg** — err	**avg** — err
10	**98K** — 4K	**236K** — 11K	**236K** — 18K
20	**106K** — 11K	**454K** — 11K	**482K** — 31K
30	**64K** — 4K	**622K** — 20K	**753K** — 16K
60	**63K** — 2K	**816K** — 26K	**1433K** — 38K
90	**64K** — 2K	**844K** — 19K	**2004K** — 31K

Comparing the performance with the three workloads: Table I shows the aggregated throughput for 10 servers when running the three workloads A, B, and C. For read-only workloads, the throughput increases linearly, reaching up to 2Mop/s for 90 clients. For read-heavy workload the throughput increases linearly until 30 clients where it reaches 622Kop/s. It increases in a much slower pace up to 844Kop/s for 90 clients. The worst case is for update-heavy workload that surprisingly reaches its best throughput of 106Kop/s when running 20 clients, then performance starts declining down to 64Kop/s for 90 clients. At 90 clients, the throughput with workload C is 31x more than the one with heavy-update workload.

To have a better view on this phenomena, Figure 3 shows the ratio of the throughput when taking 10 clients as a baseline. We can see clearly that read-only applications have a perfect scalability, while read-heavy collapses between 30 and 60 clients. With heavy-update workload, throughput does not increase at all and degrades when increasing the number of clients.

Interestingly, we can see what impact do update operations have on performance. Since RAMCloud organizes its memory in a log-structured fashion writes should not have that impact on performance, because every update/insert operation results in appending new data to the memory log and updating the hash-table. However, bringing up that much concurrency, e.g., 90 clients leads to a lot of contention within a single server, and therefore servers will queue most of the incoming requests. This suggests that there is a poor thread handling at the server level when requests are queued.

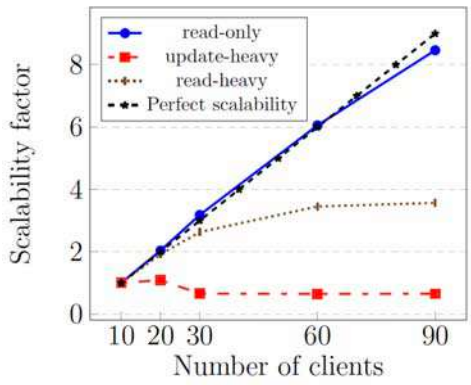
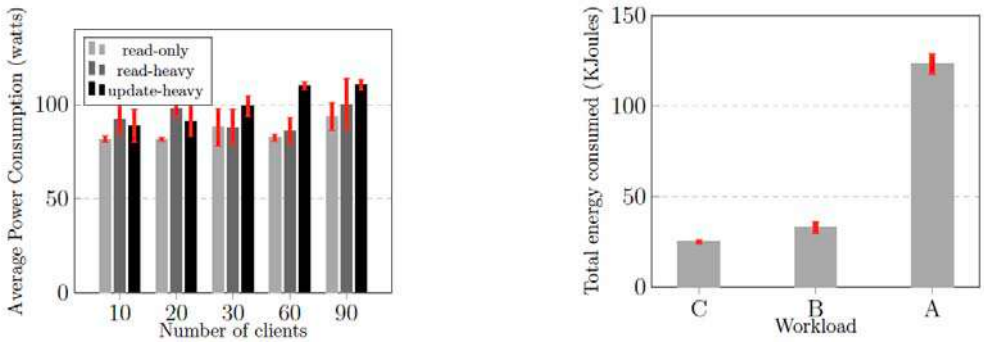


Figure 3: Scalability of 10 RAMCloud servers in terms of throughput when varying the number of clients. The ”perfect” line refers to the expected throughput when increasing the number of clients. We take the baseline as the throughput achieved by 10 clients.

More updates means more power per node?: The energy impact is straightforward when looking at figure 4a that represents the average power consumption of the RAMCloud cluster. The power consumption per server when executing read-only workloads stays at the same level, i.e., 82Watts up to 60 clients, after that it jumps to 93Watts. We can observe the same pattern for read-heavy except that it stands at a higher power consumption than read-only, which is around 92Watts, then it goes up to 100Watts for 90 clients. Power consumption of the heavy-update workload is the highest, though it starts with 10 and 20 clients around 90Watts, it continues to grow with the number of clients to reach 110Watts for 90 clients which is the highest value for all experiments.

To complement these results, we show Figure 4b which displays the total energy consumed when running the three workloads with 90 clients. The total energy consumed is computed by multiplying the power consumption, of every server, collected every second by the execution time. We can see that read-heavy has 28% percent more energy consumption than read-only. The more surprising fact is the difference between heavy-update and read-only which is 492% more energy consumed for the heavy-update workload.



a) b)

Figure 4: (a) represents the average power consumption (Watts) per node of 20 RAMCloud servers as a function of the number of clients. (b) represents the total energy consumed by the same cluster for 90 clients as a function of the workload.

While it is expected that update operations take more time and processing than read operations, it is noteworthy to recall that RAMCloud relies on an append-only log-structured memory. By design updates are equivalent to inserts, therefore the additional overhead should mostly come from updating the in-memory hash-table and appending new data to the log. Consequently, we expected the gap between reads and updates to be very low as we have disabled replication in this particular case. We suspect the main cause being the thread management at the server level. If all threads of a server are busy then upcoming requests will be queued, generating delays. More- over, the number of servicing threads impacts performance as well. This number can depend on the hardware type (e.g., number of cores). Worse, it depends also on the workload type, e.g., in Figure 3 read-only scale perfectly while only update workloads experience performance degradation. For instance, performance will decrease under write-heavy workloads with more threads due to useless context switches, since all work could be done by a single thread. Whereas it is the opposite for read-only applications. Identifying this number empirically can bring substantial performance improvements. This issue was confirmed by RAMCloud developers.

Finding 2: We find that RAMCloud loses up to 57% in throughput with YCSB's read-heavy workload com- pared to read-only. With heavy-update workloads, the performance degradation can reach up to 97% at high concurrency (with replication disabled in both cases). Furthermore, we found that heavy-update workloads lead to a higher average power consumption per node, which can lead to 4.92x more total energy consumed compared to read-only workloads. We find this issue is tightly related to the number of threads servicing requests. Finding the optimal number can improve performance and energy- efficiency, however, this is not trivial since it depends on the hardware and workload type.

Replication's Impact on Performance and Energy-Efficiency

RAMCloud uses replication to provide durability and availability. There are two main techniques used to replicate data: Primary-Backup or Symmetric [ZhangDC16]. Since the per bit cost of memory is very high, primary-backup replication (PBR) is usually preferred to symmetric replication, especially when all data is kept in DRAM. RAMCloud uses PBR and keeps a single replica in memory to serve requests and pushes eventually replicas to disk. Thus, the main

concern about replication is at what extent it impacts the performance and energy consumption of the system and eventually look for possible improvements.

Methodology

We measure the transparency of the replication scheme by stressing it with an update-heavy workload (50% reads, 50% updates). This workload was preferred to update-only (100% updates) workload since the latter is far from what in-memory storage systems and Web storage systems were designed for [RablGSMJM12]. We use the same parameters as before, except the replication factor that we vary from 1 to 4. We change the number of RAMCloud servers from 10 to 40 and the clients from 10 to 60 nodes.

Replication impact on throughput: In Figure 5 we plot the total aggregated throughput when running heavy-update with different replication factors for 20 RAMCloud servers. When running with 10 clients we can see a clear decrease in throughput whenever the replication factor is increased, e.g., from replication factor 1 to 4 the throughput drops from 78Kop/s to 43Kop/s which corresponds to a 45% decrease. Further increasing the number of clients to 30 and 60 leads to an increase of throughput for replication factor 1 and 2 which means that RAMCloud's cluster capacity has not been reached yet. Ultimately, for 30 and 60 clients setting the replication factor to 4 leads to a throughput of 41Kop/s and 50Kops, respectively, which means that we saturated RAMCloud's cluster capacity.

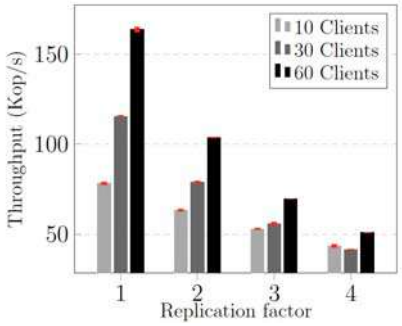


Figure 5: The total aggregated throughput of 20 RAMCloud servers as a factor of the replication factor.

While the impact of replication on performance might seem substantial, since the normal replication factor at which the system should run with is at least 3 or 4, the explanation lies on the replication scheme itself. In RAMCloud for each update request, a server will generate as many requests as its replication factor, e.g., if the replication factor is set to 4, upon receiving a request a server will generate 4 replication requests to other servers that have available backup service up and running. For every replication request a server sends, it has to wait for the acknowledgements from the backups before answering the client that issued the original request. This is crucial for providing strong consistency guarantees. Indeed, suppose a server can answer a client's request as soon as it has processed it internally and sent the replication requests without waiting for acknowledgements, then in case of a failure of that specific server, the RAMCloud cluster can end up with different values of the same data.

We plot in Figure 6a the aggregated throughput when running heavy-update at fixed rate of 60 clients. We varied the number of servers as well as the replication factor. It is interesting to see how enlarging the number of servers can reduce the load, e.g., when having replication factor set to 1 the throughput can be increased from 128Kop/s to 237Kop/s when scaling up the cluster from 10 to 40 servers. We remark the same behavior for higher replication factors, though the throughput is lower when increasing the replication factor. It is noteworthy that the figure does not include values for 10 servers when going beyond replication factor 2. The reason is that the experiments were always crashing despite the number of runs because of excessive timeouts in answering requests.

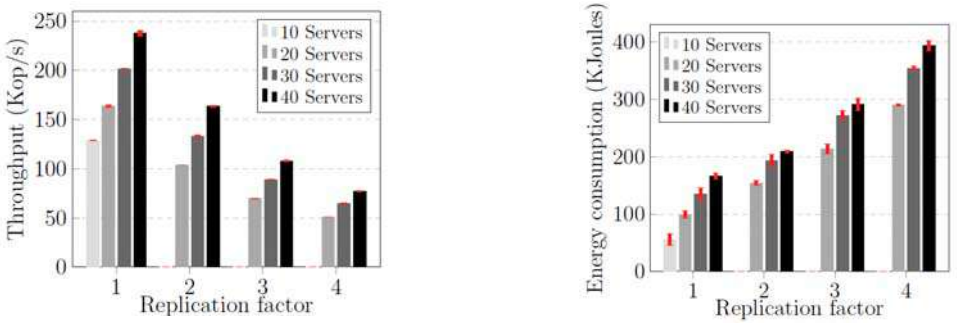


Fig. 6: (a) The total aggregated throughput as a function of the number of servers. The number of clients is fixed to 60. (b) The total energy consumption of different RAMCloud servers numbers as a function of the replication factor when running heavy-update with 60 clients.

Replication impact on energy consumption: To give a general view on the power consumption under different replication factors we plot Figure 7 that shows the average power consumption per node of a cluster of 40 servers when running 60 clients for different replication factors. As expected, the lowest average power is achieved with a replication factor of 1 with an average power of 103Watts per server. Increasing the replication factor leads to an increase up to 115Watts per server for a replication factor of 4.

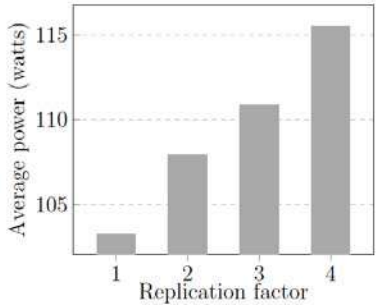


Fig. 7: The average power consumption per node of a cluster of 40 servers when fixing the number of clients to 60.

The rise in the power consumption per server results in a substantial increase in the total energy consumption of the RAMCloud cluster. Figure 6b illustrates the total energy consumption when fixing the number of clients to 60. Firstly, it is interesting to look at the difference in the total

energy consumption when increasing the replication factor. For instance, with 20 RAMCloud servers and replication factor set to 1 the total energy consumed is 81K Joules. It rises up to 285K Joules which corresponds to an increase of 351%. For 40 servers the extra energy consumed reaches 345% whenever tuning the replication factor from 1 to 4. Secondly, it is noteworthy to compare the energy consumption when resizing the number of servers. As an example, with a replication factor of 1, the difference between the energy consumed by 20 and 30 servers is 16%. The increase in the energy consumed when scaling the cluster from 20 to 40 servers is 28%. When increasing the replication factor these ratios tend to decrease. More specifically, when the replication factor is fixed to 2, the difference in the total energy consumed by 20 and 30 servers is 10%, but this difference reaches 17% only when comparing 20 and 40 servers.

Finding 3: Increasing the replication factor in RAM-Cloud causes a huge performance degradation and more energy consumption. For instance, changing the replication factor from 1 to 4 leads to up to 68% throughput degradation for update-heavy workloads while it leads to in 3.5x additional total energy consumption. This overhead is due to the CPU contention between replication requests and normal requests at the server level. Waiting for acknowledgements from backups increases the overhead.

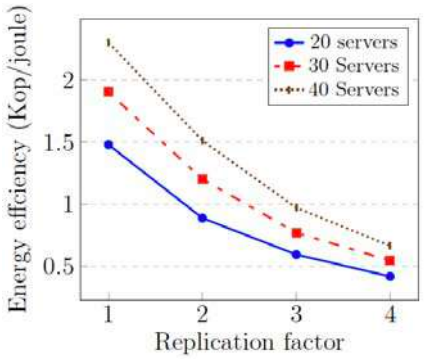


Fig. 8: The energy efficiency of different configurations as a function of the replication factor when running heavy-update with 60 clients

Changing the cluster's size. Figure 8 illustrates the energy efficiency when fixing the number of clients to 60. In a very surprising way, we can see that having more servers leads to better energy efficiency. As an illustration, when the replication factor is set to 1 the energy efficiency of 20 servers equals 1500 while for 30 servers it is 1900, finally, for 40 server it reaches 2300. The ratio tends to decrease whenever the replication factor is increasing.

It is noteworthy that the relative differences in the energy-efficiency between different number of servers shrinks when increasing the replication factor. The explanation resides in Figure 6a where we can see that the relative differences in throughput decreases, therefore the fraction throughput/power goes down.

In contrast with the previous results, where the best energy efficiency for read-only workloads was achieved with the lowest number of servers, with heavy-update and replication enabled, it appears that provisioning more servers not only achieves better performance, but also leads to a better energy efficiency.

Finding 4: Surprisingly, in contrast with Finding 1, increasing the RAMCloud cluster size results in a better energy efficiency with update-heavy workloads when replication is active. Therefore, the type of workload should be strongly considered when scaling up/down the number of servers to achieve better energy-efficiency.

Crash Recovery

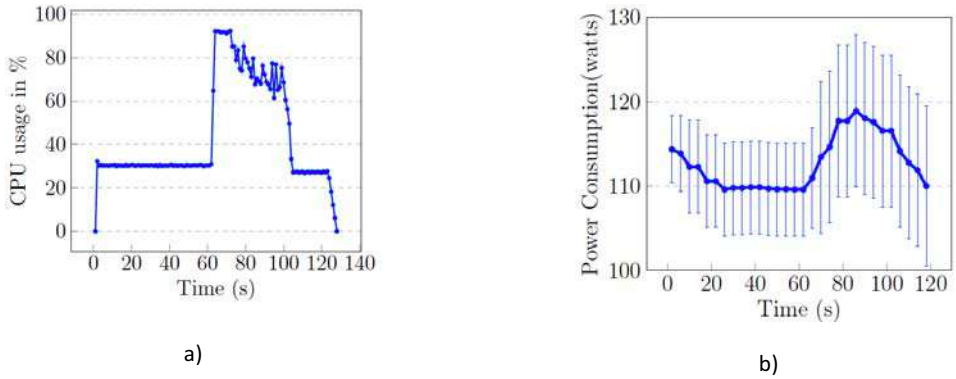
Data availability is as critical as performance during normal operations since there is only a single primary replica in RAMCloud. Studying RAMCloud's crash recovery can help in understanding the factors that need to be considered when deploying a production system, e.g., replication factor, size of data per server, etc. Therefore, our main concern in this section is to answer the following questions: What is the overhead of crash-recovery? What are the main factors impacting this mechanism?

Methodology

Two important metrics we consider are recovery time and energy consumption. We perform the assessment by inserting data into a cluster and then killing a randomly picked server (after 60 seconds) to trigger crash-recovery, with different number of servers. First, we assess the overhead of a crash-recovery, and then we vary the replication factor to observe the impact on the recovery-time and the corresponding energy consumption.

The overhead of crash-recovery. For our first scenario we setup a RAMCloud cluster of 10 servers. We then inserted 10M records which corresponds to 9.7GB of data uniformly split across the servers, i.e., each server receives 1M records. It is noteworthy that we have set the replication factor to 4 which corresponds to the normal replication factor used in production systems [CidonRSKOR13].

Figure 9a shows the average CPU usage of the cluster. RAMCloud monopolizes one core for its polling mechanism, which results in our case in a 25% CPU usage even when the system is idle. When the crash occurs the CPU usage jumps to 92%, then gradually decreases. This is mainly due to loading data from disks and replaying it at the same time. Figure 9b shows the overall average power consumption per node for the same scenario. The jump in resource usage translates in a power consumption of 119W. Since the power measured every second is an average, the increase in power consumption is not as sudden as the CPU usage.



a) b)

Fig. 9: The average CPU usage (a) and power consumption (b) of 10 (idle) servers before, during, and after crash-recovery. At 60 seconds a random server is killed.

The overhead of crash-recovery on normal operations. Figure 10 shows the average latency of two clients requesting data in parallel in the same configuration as the previous scenario. However, we chose manually a server to kill and make sure one of the clients always requests the subset of data held by that server. It is interesting first to notice that after the crash happens, the client which requests lost data is blocked for the whole duration of crash recovery, i.e., 40 seconds. Second, it is important to point out the jump in latency for the client which requests live data, i.e., from 15µs to 35µs just after the crash recovery starts. The average increase is between 1.4X and 2.4X in latency during crash recovery. This can be explained if we consider that in figure 9a crash recovery alone causes up to 92% CPU usage. Consequently, normal operations are impacted during crash recovery.

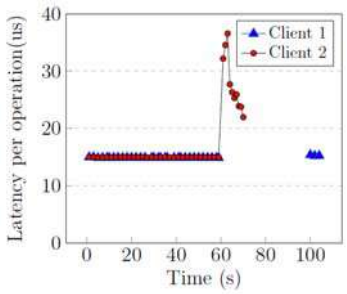


Fig. 10: The latency per operation before, during, and after crash recovery for two clients running concurrently. Client 1 requests exclusively the set of data that is on the server that will be killed. Client 2 requests the rest of the data.

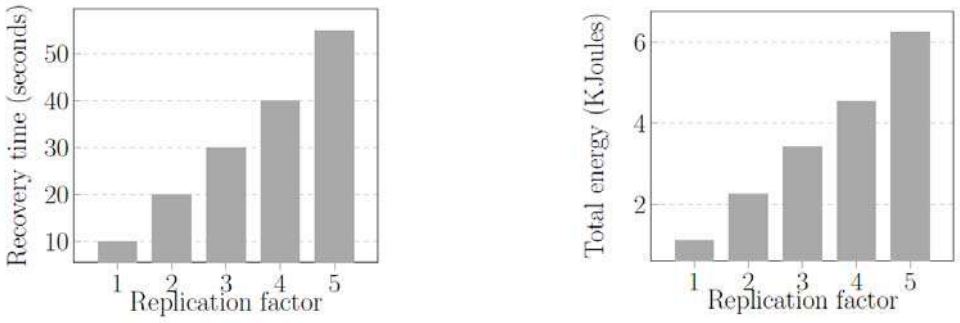
Finding 5: As expected, crash recovery induces CPU (in addition to network and disk) overhead and up to 8% additional power consumption per node. We find that during crash recovery: (1) lost data is unavailable, causing latencies equal to the recovery time (e.g., 40 seconds for replication factor 4); (2) The performance of normal operations can have up to 2.4X additional latency in average due to the overhead of crash recovery.

Replication impact on crash-recovery. Random replication in RAMCloud intends to scatter data across the cluster to have as much resources as possible involved in recovery. We have setup an experiment with 9 nodes and inserted 10M records which corresponds to 9.765GB of data. Since

we are inserting data uniformly in the RAMCloud cluster, each server has up to 1.085GB of data. Same as previous experiment, we choose a random node after 1 min, then we kill RAMCloud process on that node.

Figure 11a shows the recovery time taken to reconstruct data of the crashed node. Surprisingly, increasing the replication factor increases the recovery time. With a replication factor of 1, only 10 seconds are needed to reconstruct data, this number almost grows linearly with the replication factor up to replication factor 5, where the time needed to reconstruct the same amount of data takes 55 seconds.

In order to shed the light on these results it is important to recall in more details how crash recovery works in RAMCloud. When a server is suspected to be crashed, the coordinator will check whether that server truly crashed. If it happens to be the case, the coordinator will schedule a recovery, after checking that the data held by that server is available on backups. Recovery happens on servers selected a-priori. After these steps, the recovery starts, first by reading lost data from backups disks, then replaying that data as in normal insert operations, i.e., inserting in DRAM, replicating it to backup replicas, waiting for acknowledgement and so on. We argue that the performance degradation shown in Finding 3 similarly impacts crash recovery. Since data is re-inserted in the same fashion, increasing the replication factor generates additional overhead and concurrency, leading to higher recovery time.



a) b)

Fig. 11: (a) Recovery time as a function of the replication factor. (b) Average total energy consumption of a single node during crash recovery as a factor of the replication factor. The size of data to recover corresponds to 1.085GB in both cases.

Impact of disk contention. While investigating the factors impacting the crash recovery performance, we find that disk contention can have an impact. For instance, Figure 12 corresponds to the total aggregated disk activity of the servers during crash recovery. We can see that right after the crash, there is a small increase in the read activity corresponding to reading backup data. Shortly after, there is a huge peak in write activity corresponding to the re-replication of the lost data. Read and write activities continue in parallel until the end of crash recovery.

We think that at small scale this issue is exacerbated. Since the probability of disk-interference between the backup performing a recovery, i.e., reading, and a server replaying data, i.e., writing, is high.

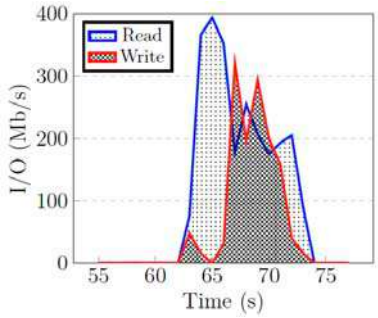


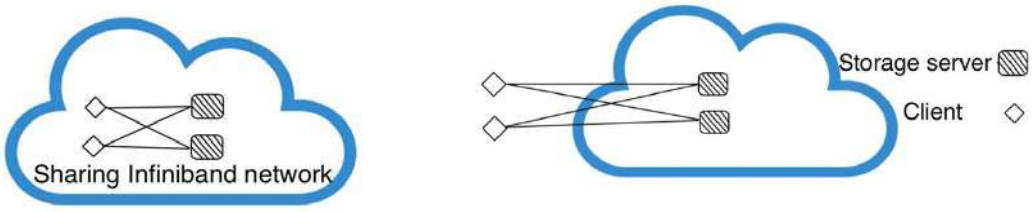
Fig. 12: Total aggregated disk activity (read and write) of 9 nodes during crash recovery. The size of data to recover corresponds to 1.085GB. The dark green part is the overlap between reads and writes.

Finding 6: Counterintuitively, increasing the replication factor badly impacts the performance and the energy consumption of crash-recovery in RAMCloud. The replication factor should be carefully tuned according to the cluster size in order to avoid the phenomena discovered in Finding 3 since during a crash-recovery data is replayed.

In Figure 11b we report the total energy consumed by a single node during crash recovery within the precedent experiment. As expected, the energy consumed grows when increasing the replication factor. The energy increases almost linearly with the replication factor. It is noteworthy that the average power consumption of a node is comprised between 114 and 117 watts during recovery. Thus, the main factor leading to increased energy consumption is the additional time took to replay the lost data.

Network Impact on Energy Efficiency of In-memory Stores

To understand the importance of the network, we plot Figure 13 that shows the two possible ways of using RAMCloud. In Figure 13a) the clients are sharing the same datacenter network as the storage system. In this case the clients can take full advantage of the storage system and benefit from high speed networks (when available). A typical example is an HPC setup or in-memory BigData analytics. On the other hand, a second use case is displayed in Figure 13b), where the clients use a TCP/IP network stack to access the storage system. In this case clients will have slower access to the system, and more importantly it is not clear whether they will benefit or not from the fast access of data stored in memory. This is the case for a Cloud deployment for instance.



a) Clients and servers share the same network

b) Clients access servers from outside the datacenter

Figure 13: The two possible scenarios for accessing RAMCloud. Figure 1a refers to the case where the clients also use InfiniBand transport. Figure 1b represents the case where the clients are not collocated within the same data center with the storage servers. We refer to this case in RAMCloud as the one where the client uses TCP/IP

Through an experimental study, we attempt to answer this question by reproducing both scenarios displayed in Figure 13.

Experimental Methodology

The experiments have been performed, again, on the Grid'5000 platform at Nancy. In our experiments each client runs on a single machine to avoid interferences. Each node ran as server and backup at the same time. We have fixed the memory used by a RAMCloud server to 10 GB and the available disk space to 80 GB. We have fixed the memory size to an acceptable limit to carry the whole data in the cluster. Before running each benchmark, we pre-load 100 K records of 1 KB in the cluster. Running the benchmark consists of launching simultaneously one instance of a YCSB client on each client node. Each client issues 100 K requests, which corresponds to the total number of records. Having an increasing number of requests with the number of clients enabled us to study the impact of concurrency on RAMCloud's scalability. Having each client generate 100 K requests results in having 1M requests with 10 clients for example, and 9 M requests with 90 clients, which corresponds to 8.58 GB of data requested per run.

It is noteworthy that we use read-only workloads in our experiments. Since our focus is on understanding how the client location can impact the energy efficiency of the system, read-only workloads are more suitable. When a client issues an update request it is first processed by the server holding the primary-replica. Then the server creates a replication request to the servers holding the secondary-replicas of the object being written. The client gets the acknowledgement only after all secondary-replicas have replied to the primary-replica. Therefore, update/insert operations are not suitable in our study because of the intra- cluster communication they induce between primary- and secondary-replicas. On the other hand, read operations are processed only by the primary-replica. In this case, there is a client-server communication exclusively, which is better suited for our study.

In our figures, each value corresponds to an average of 5 runs with the corresponding error bars. When changing the cluster configuration (i.e., number of RAMCloud servers), we remove all data stored on servers as well as in backups, then we restart the RAMCloud servers and backups on the whole cluster to avoid any interference with prior experiments.

We configured RAMCloud with the option *ServerSpan* that is equal to the size of the cluster. As RAMCloud does not have a smart data distribution strategy, this option is used to manually decide the distribution of data across the servers, i.e., how many servers each table will span. Moreover, we used uniform data distribution in YCSB clients to insert and request data. At the end, all the data is evenly distributed across the servers and each object is requested at the same frequency.

Results

Performance and Scalability: Figure 14 shows the aggregated throughput for a 10-node cluster with both InfiniBand and TCP. A considerable gap can be observed in the throughput achieved when clients access the system with InfiniBand and TCP. When running 10 clients with InfiniBand, the system achieves 7.38x more throughput compared to when clients use TCP. Whenever increasing the number of clients the gap widens and reaches 11.40x more throughput when clients use Infiniband. What stands out here is the limited scalability of TCP, for example between 60 and 90 clients, while the scalability of Infiniband-accessed cluster increases by a factor of 1.39x, TCP-accessed cluster only increases by 1.10x. This suggests that since the transport protocol is slowing down the operations, RAMCloud nodes are subject to more concurrency, and thus scalability might suffer.

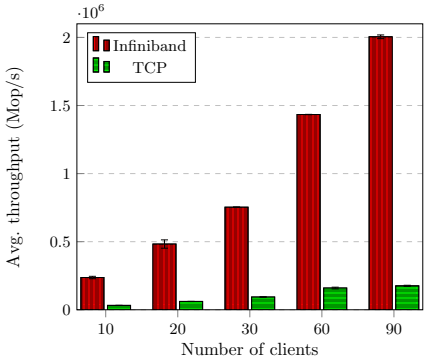


Figure 14: Total aggregated throughput as a factor of clients number with a fixed cluster size of 10 servers. InfiniBand and TCP represent two scenarios where clients use respectively InfiniBand or TCP to access RAMCloud.

To further investigate that behavior, we plot in Figure 15 the scalability of each of the scenarios, i.e., varying the cluster size from 10 to 40 nodes and we fix the clients to 90. The scalability factor presents the ratio of throughput increase when taking 10 clients as a baseline. When clients are collocated with RAMCloud, it can achieve linear scalability as we already showed in Figure 14, even under high concurrent accesses. On the other hand, we can witness the difference widening between TCP scalability and the expected ratio. For the cluster of 10 nodes with TCP the scalability factor is 5.4 while the expected scalability factor is 9. This difference reduced when increasing the cluster size, for example it is 6.9 for 20 nodes and 7.3 for 30 nodes. This confirms our observation about RAMCloud being less scalable in throughput when accessed by TCP compared to InfiniBand. We suspect this is due to lower load on resources as we increase cluster size.

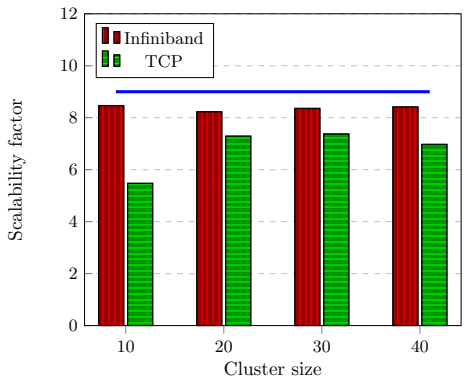


Figure 15: Throughput scalability factor as a function of the cluster size and when running 90 clients. The blue line represents the expected increase in throughput according to the baseline of 10 clients. Infiniband and TCP represent two scenarios where clients use respectively Infiniband or TCP to access RAMCloud.

As expected, when clients are collocated on the same network as RAMCloud it can achieve more throughput compared to when clients access the system from a different network. In the same lines, when clients are collocated with RAMCloud it achieves better scalability.

Per-node power consumption:

Figure 16 shows the average power consumption of the cluster for the same experiment described above. First, it is important to notice the horizontal blue bar that represents the average power consumed by the machines when idle, which is in average 50 Watts. The horizontal black dashed line represents the average power per server when running the RAMCloud service. There is an increase of 1.6X in the average power consumed per node since RAMCloud hogs one core per node, even when idle. This is due to the polling mechanism of RAMCloud, which polls continuously requests from the Network interface controller (NIC) in order to handle packets as fast as possible.

In Figure 16, when clients use InfiniBand, we can see a stable power consumption of 94Watts up to 60 clients, while it increases a little bit when going up to 90 clients. More likely, this is the point where the cluster starts demanding more resources to cope with the load, which matches our previous observations. If we look at the power consumption when clients use TCP, the results are more surprising. First, we see that the average power consumption is constantly increasing from 97Watts for 10 clients up to 109Watts for 90 clients. Moreover, in all cases it is higher than the average power consumption of InfiniBand-accessed cluster.

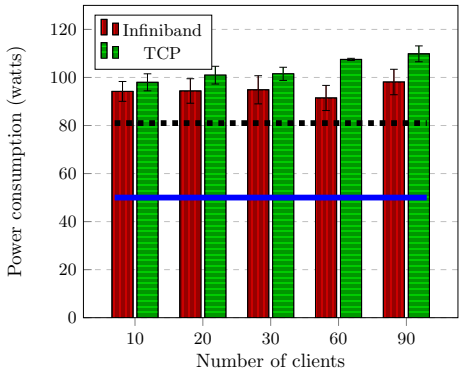


Figure 16: Average power consumption per server in Watts. The cluster size is fixed to 10. The horizontal blue line represents the average power consumed by servers when idle. The horizontal dashed black line represents the average power per server when RAMCloud is running and idle. InfiniBand and TCP represent two scenarios where clients use respectively InfiniBand or TCP to access RAMCloud.

To understand why the system consumes more power when accessed by TCP compared InfiniBand, we show Table II. It represents the minimum and maximum CPU usage with different numbers of clients for a cluster of 10 RAMCloud servers. We remark a slight additional maximum CPU usage when clients use TCP compared to InfiniBand. What is more interesting is to see the minimum CPU usage of the servers when clients use TCP (bold) compared to when they use InfiniBand. There is between 9% additional CPU usage and up to 15%. We have noted that in some scenarios, servers reach the same maximum CPU usage. More importantly, we observed high variation in CPU usage in the case of InfiniBand. For example, when running 10 clients, there is a 9% difference between the minimum and maximum CPU usage when clients are using InfiniBand.

Table II: The minimum and maximum of the average CPU usages (in percentage) of all servers when running read-only workload on 10 RAMCloud servers.

Clients \ Transport	TCP	IB
	min — max	min — max
0	25 — 25	25 — 25
10	**52,933** — 54,543	44,193 — 53,391
20	**55,044** — 57,244	46,985 — 57,921
30	**56,638** — 58,943	40,534 — 46,960
60	**59,917** — 62,234	47,974 — 62,235
90	**65,129** — 67,676	50,626 — 65,958

As a result, when clients use TCP to access RAMCloud, the system exhibits higher CPU usage compared to when accessed through InfiniBand. This increased CPU usage results in additional average per-node power consumption.

The energy efficiency

We plot Figure 17 to show the overall energy consumption when clients use both InfiniBand and TCP. Up to 60 clients the average difference is roughly 7X more energy consumed when clients use TCP. The difference goes up to 9X when running 90 clients.

Latency per operation: While Figure 16 has shown that accessing the system through TCP can result in more average per-node power consumption compared to when accessed through InfiniBand, the expected difference in total energy consumption is not as big as shown in Figure 17. To explain such a phenomenon, we plot Figure 18 which represents the latency of a single operation for the same scenario, i.e., 10 servers running up to 90 clients. When looking into InfiniBand's cluster latency one can see a stable latency around 40 µs. When clients use TCP, the latency is stable around 315 µs up to 60 clients. It reaches more than 512 µs for 90 clients.

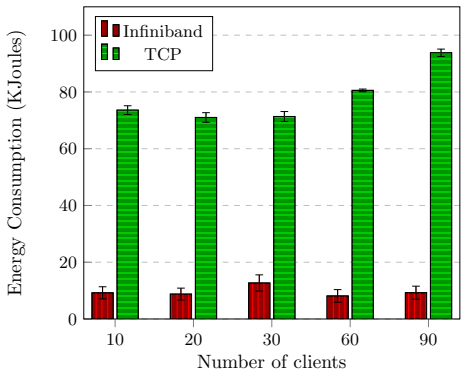


Figure 17: The total energy consumption when running read-only workload with a cluster of 10 server. InfiniBand and TCP represent two scenarios where clients use respectively InfiniBand or TCP to access RAMCloud.

Consequently, when a client issues a request with TCP it takes longer to be processed at the server level compared to when it is issued with InfiniBand. While this is not surprising, it can explain the wide gap that appears in Figure 17 between the cases where clients access RAMCloud through TCP and InfiniBand.

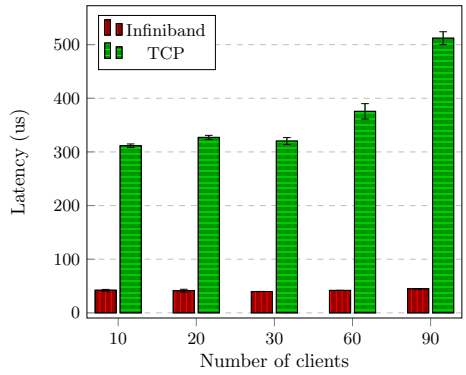


Figure 18: The latency per single operation when running read-only workload with a cluster of 10 nodes. InfiniBand and TCP represent two scenarios where clients use respectively InfiniBand or TCP to access RAMCloud.

Figure 19 represents the energy efficiency of different number of RAMCloud servers when running 90 clients. Consequently, to the increased power consumption and high latency when clients access the system with TCP, the energy efficiency is very low compared to when clients use InfiniBand. In average the system has 10X better energy efficiency when accessed with InfiniBand compared to TCP.

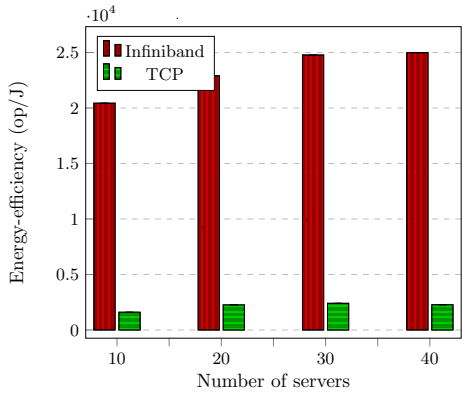


Figure 19: The energy efficiency when running read-only workload with different number of servers. InfiniBand and TCP represent two scenarios where clients use respectively InfiniBand or TCP to access RAMCloud. The number of clients is fixed to 90.

To summarize, we find that RAMCloud has a better energy efficiency when clients use InfiniBand network to access it compared to when they use TCP/IP. The reasons are that when clients access RAMCloud through TCP, the system has a higher CPU usage compared to when accessed through InfiniBand. Moreover, using TCP leads to higher latencies, therefore the system has a much higher energy consumption compared to when using InfiniBand. All these reasons contribute to the energy inefficiency of RAMCloud when accessed through TCP.

Energy Efficient Data Management

Energy saving is an imminent problem for supercomputing centers to reduce costs as far as possible without decreasing the runtime performance. Even though a number of approaches and mechanisms to reduce energy consumption in supercomputers have been suggested at the different levels of computing systems (CPU and GPU, storage disks, I/O, network, etc.), more and more HPC applications still produce enormous volumes of data sets that need more storage space to keep information saved. This also results in additional costs for energy because new hardware used in computing systems (for example, SSDs and HDDs) requires additional power. Therefore, storage, energy and overall costs increase for supercomputing facilities. For instance, for each PB of HDD storage space, the German Climate Computing Center (Deutsches Klimarechenzentrum, DKRZ) roughly has to pay investment costs of 100,000 € and annual electricity costs of 3,600 €; for its 54 PiB storage system, this amounts to almost 200,000 € per year for electricity alone. These costs do not even include maintenance (approximately 15 % of storage costs) and tapes for long term archives (70 000 tape slots) [HuebbeK13].

Reduction of data volumes is a straight-forward solution to minimize energy consumption in storage systems. It can be achieved by leveraging different data reduction techniques like compression, transforms or deduplication [MeisterKBCKK12] [KaiserBSM15] [XiaJFDSHFZZ16]. For storage systems, data reduction directly results in less storage hardware that has to be procured and operated. The main benefits of reduction techniques are storage capacity optimization, network bandwidth reduction, minimization of operational costs and, of course, energy saving.

In this connection, HPC users have a great interest in data reduction. Especially in those methods and algorithms that are the most appropriate for their data sets. Of course, the chosen data reduction must be able to provide the highest reduction ratio without decreasing the whole runtime performance or using additional resources including energy consumption. In our paper, we are focusing on scientific applications (typical users of HPC), where defining reduction strategies with high performance and energy efficiency suitable for the generated deluge of scientific data is really a challenging task today. For these users, the choice of a compression algorithm is a technical decision that is difficult to make, since a well-suited algorithm for one data set might be suboptimal for another data set or on another machine. That means, compression schemes and options are highly data specific. Therefore, our ultimate goal is to automatize the decision-making process on behalf of the users.

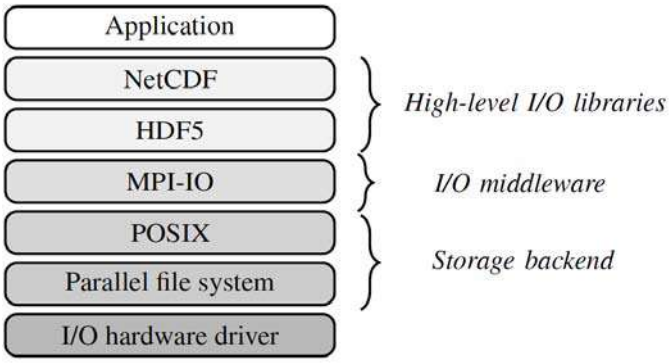


Figure 20: HPC I/O stack commonly used in life sciences

In this work, we aim to define the methodology for intelligent selection of algorithms from a variety of state-of- the-art reduction techniques with an emphasis on their energy consumption. Here, energy consumption expands the set of criteria to pick up the most suitable compression strategy for a data set, user and data center with its resources. Elaborated methodology will be applied in the further development of our framework for scientific data reduction. Regarding this purpose, we make the following contributions in this section:

- We highlight main drawbacks and benefits of reduction techniques deployment on determined levels of data path through the common HPC I/O stack. After that we introduce our framework for scientific data compression though high-level I/O interfaces.
- We introduce experimental setup with a single node for investigation of application-level data compression techniques. Here, different lossless compressors were taken to evaluate their performance.
- We present the preliminary results obtained from a series of experiments and outline the main principles of our methodology for reduction method selection.

Data Reduction through high-level IO

All data have to cross the full I/O stack during manipulation or retrieval. It consists of a file system, middleware and I/O libraries as depicted in Figure 20. Two of the most popular and common high-level I/O interfaces in the scientific community to access data in both serial and parallel manner

are HDF5 (Hierarchical Data Format 5) and NetCDF (Network Common Data Form). They allow HPC applications written in various programming languages (e.g., C, C++, Fortran, Python, etc.) to manipulate and store data in a self-describing portable way by using multidimensional arrays [BartzCKNL15]. Leverage of self-describing data formats gives the opportunity to store a description of the data file layout as an additional meta information in the header part.

HDF5 and NetCDF perform I/O in a layered manner. The NetCDF programming interface delegates data storing to HDF5, and HDF5 uses the I/O implementation of MPI (Message Passing Interface) [BartzCKNL15]. MPI employs the I/O operations of the underlying parallel file system (backends for specific file systems or the more generic POSIX backend). In the end, I/O is performed by the I/O driver. If the application performs data writing, it uses the high-level I/O library, and the data are going through the stack down until they are placed in the driver layer. A data read works in the opposite direction.

It must be also clearly noticed that NetCDF and HDF5 interfaces provide parallel I/O. In this case it is necessary to use HDF5's MPI-IO backend and have an underlying parallel file system which allows multiple processes to access a file. Otherwise, the I/O operations will be serialized.

Table III: Drawbacks and benefits provided by deployment of reduction techniques in high and low levels of IO

	SYSTEM LEVEL	APPLICATION LEVEL
DRAWBACK	**Uncertainty** due to the lack of access to application-specific semantic information (e.g., data structures, important variables, etc.) only lossless reduction can be considered	**Clarity** insight into the code and requirements of applications is needed for tuning the performance of data reduction techniques
BENEFIT	**Transparency** no need to modify applications, even if they are very diverse or do not use a common I/O software stack	**Flexibility** semantic information is easily accessible, hence more reduction techniques can be leveraged (even for specific portions of data)

On the basis of these considerations, it is possible to determine in general two main levels of the data path where data reduction mechanisms can be deployed. They are system (low) and application (high) levels. Depending on where in the I/O stack data reduction is employed, different benefits and drawbacks become apparent.

As can be seen from Table III, data reduction usage on higher levels of HPC I/O stack is advantageous. Unlike low layers, it is possible to access and exploit additional meta information stored in the header part of files like data types. Different HPC applications (e.g. for climate change and weather forecasting, bioinformatics, etc.) are using a common I/O stack (Figure 20), making it easier to employ application-level data reduction for them. Thus, usage of data reduction at the application level can be fine-tuned by taking application requirements and metadata into account. Techniques which can be deployed in a way that is transparent for users of these applications are deduplication, compression and transforms [SchlachterCE16].

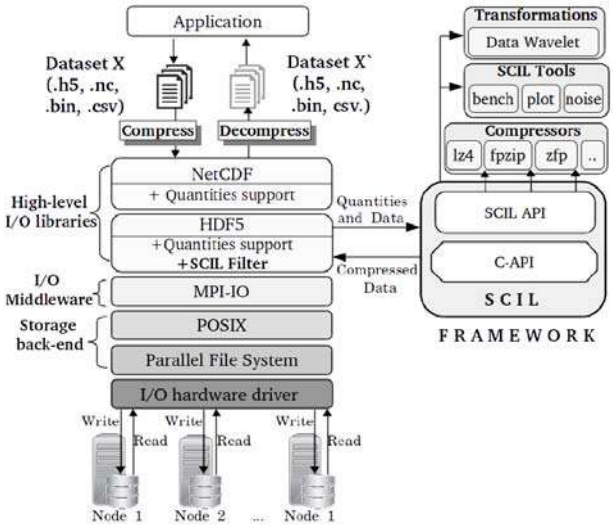


Figure 21: General architecture of Scientific Compression Library SCIL

With regards to the problem mentioned before and bearing in mind the data path, we are developing Scientific Compression Library (SCIL) 2 – a framework for data reduction though high-level I/O interfaces (see Figure 21). Its main goal is providing the most appropriate data reduction strategy for a given scientific data set on the basis of semantic information and performance of algorithms. It currently supports different lossless and lossy techniques.

Users requirements to the compression are represented in SCIL as optional criteria, and meta-information is already available in self-describing NetCDF or HDF5 files. So, a user is not required to accurately document any specific information that describes a data set. Moreover, insight into the application code is not needed here, too, unless application developers want to use SCIL as a separate library in their workflow. Otherwise, any application with HDF5 data model can use its filters. An application can either use the NetCDF4, HDF5 or the SCIL C interface, directly. Majority of scientific data, for instance, in climate science are stored in NetCDF file format. However, NetCDF relies on the HDF5 filters for compression and only supports with its API the DEFLATE lossless algorithm. Using SCIL filter and some others we will launch our tests further in this paper.

In addition, SCIL is rich of many useful features. It provides several tools to:

- Create random patterns or add noise.
- Compress/Decompress data.
- Plot the results

In general, SCIL is a meta-compressor that aims to exploit knowledge on the application level [KunkelNBS17][KunkelNBS17b]; it decouples the selection of various error quantities and the expected performance behavior from the selection of the algorithm. For example, a newer and better algorithm could be selected by the library without change in the application code once it becomes available. The library should ultimately pick a suitable chain of algorithms yielding the user's requirements. Initially, this is done based on the capabilities of the algorithms, but the

ongoing work is a preliminary stage for the design of an improved algorithm selector that could benefit from energy- aware selection of the algorithms.

In the following, we focus on data compression techniques widely used to safe storage space, and set the following tasks:

- Investigate application-level data reduction techniques with their energy consumption.
- Compare performance of various HDF5 compression filters and the SCIL filter chain.
- Utilize meta information about data set for compression tuning.
- Define a methodology for determination of appropriate reduction strategy that accounts energy consumption.

Experimental Setup

Apart from the points described above, it is necessary to examine whether there are dependencies between performance of compressor (including its energy efficiency) and the structure of the data. If they exist, then it will be extremely useful to know how such dependencies can be employed in selection of power-aware reduction techniques for a given data set when its metadata are available at the hand.

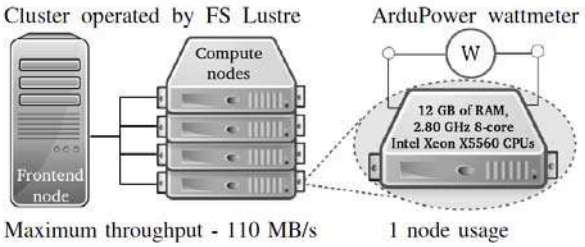


Figure 22: Experimental Setup

In the next sections, we will try to check this through evaluation of HDF5 filters at the high-level I/O and their distinct performance and energy consumption characteristics. Algorithms like LZ4 and Zstd are fast and provide high throughputs. However, their compression ratios can be lower compared to slower algorithms that consume more energy (such as LZMA).

Environment setup. In order to investigate the performance of data compression at the application level, we used a cluster which operates with the parallel distributed file system Lustre (Figure 22). The maximum throughput was limited to roughly 110 MB/s due to using only one node (outfitted with two 2.80 GHz quad-core Intel Xeon X5560 processors and 12 GB of RAM). The experiments were conducted 10 times by repacking the source files located on the same node and storing them in a local file system. For the energy consumption measurements, the ArduPower wattmeter was used. It is designed to simultaneously measure the DC (Direct current) power consumption of different components (e.g., motherboard, CPU, GPU, disks) inside computing systems even at very large scale. ArduPower provides 16 channels to monitor the power consumption with a sampling rate varying from 480 to 5,880 Hz.

Metrics. The main metrics in which we were interested are the compression ratio (CR), defined as the original site divided by the compressed sizte, to quantify the data reduction, runtime of each algorithm to see how slow or fast is it, average CPU utilization and consumed energy.

Data set and workload. For the evaluation of data reduction techniques at the high-level I/O, two data sets with roughly the same size have been chosen and one smaller data were taken for additional experiments:

- 17GB data set of 3-dimensional ecosystem model for the North Sea ECOHAM [GroßeGKLMPST15] [LorkowskiPMK12] (from Climate Science)
- 14 GB data set of tomography experiments from PETRA III's PCO 4000 detector (from High Energy Physics)
- 4 GB data set of ECHAM atmospheric model [RoecknerBBBEGHKKM+03] (from Climate science)

Evaluated techniques. To perform the reduction of data sets, different HDF5 compression filters have been leveraged. In experimental evaluation we compared the following algorithms:

- *off:* No filtering is applied. This represents the baseline.
- *blosc:* The Blosc meta-compressor using LZ4 compres- sor. Additionally, Blosc's shuffle pre-conditioner was used.
- *mafisc:* The MAFISC compression algorithm that uses several pre-conditioners and LZMA compressor [21].
- *lz4:* The LZ4 compression algorithm using its default acceleration factor.
- *zstd:* The Zstd compression algorithm using its default aggression parameter. The zstd-11 and zstd-22 variants represent Zstd with aggression parameters of 11 and 22, respectively.
- *scil:* HDF5 plugin applying LZ4 compression algorithm with some pre-conditioners to each variable in a data set.

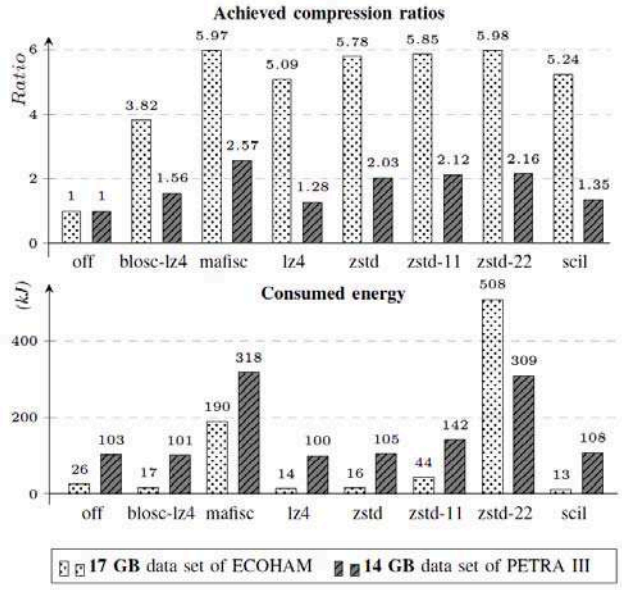


Figure 23: Average compression ratios and energy consumption depending on the HDF5 filter used for data compression

The pre-conditioners used in some filters can reorder or reformat data to increase the compression ratio in algorithm applied to a data set. Therefore, they improve efficiency in compression/decompression process.

Results and Discussion

The obtained results of compression ratio and consumed energy for the ECOHAM and PETRA III data sets are plotted in Figure 23. Average CPU utilization (after 10 experimental runs) for both data sets are plotted in Figure 24 and Figure 25 accordingly. Here, only one CPU core was used, others were idle.

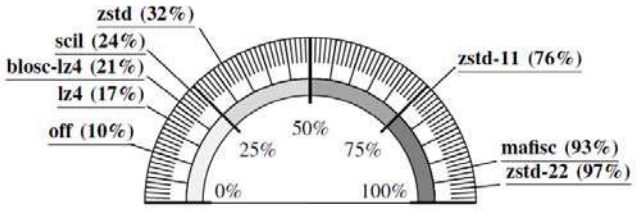


Figure 24: Average CPU utilization with ECOHAM data set (only 1 core)

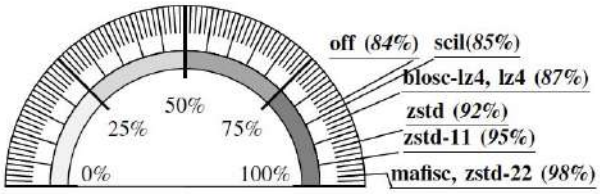


Figure 25: Average CPU utilization with PETRA data set (only 1 core)

From the figures, one can see that PETRA III data set requires more resources than ECOHAM and SCIL performance is acceptable. Figure 26 shows that overall the runtimes vary wildly, even though both data sets have roughly the same size (17GB for ECOHAM and 14GB for PETRA III). This gives rise to the view that it is related to the different data set structures: while the ECOHAM data set contains more than 300 4-dimensional variables of double precision floats, the PETRA III data set contains around ten 3-dimensional variables of 16-bit integers. Consequently, the pre-conditioners were tuned for the data sets where appropriate. Blosc was set up to use an 8-byte data size for ECOHAM and a 2-byte data size for PETRA III. In addition, ECOHAM contains many (repeating) fill values, which explains higher compression ratios in comparison to the PETRA III data set.

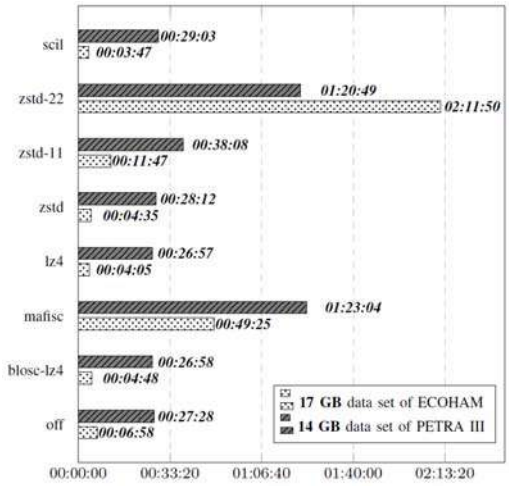


Figure 26: Runtime T(HH:MM:SS) of evaluated compressors

Blosc's pre-conditioners shuffle the data in such a way that compressors should achieve higher compression ratios. In case of PETRA III, this approach works well because blosc-lz4 achieves a compression ratio of 1.56 while lz4 only achieves a compression ratio of 1.28. Due to a very similar runtime and CPU utilization (and thus, energy consumption), the additional pre-conditioning provides benefits. However, in case of ECOHAM, Blosc's pre-conditioner significantly reduces the achievable compression ratio from 5.09 for lz4 to 3.82 for blosc-lz4. In addition to that, it also slightly increases the runtime and CPU utilization and, therefore, consumes 16 % more energy.

For both data sets, Zstd achieves significantly better compression ratios than LZ4 even with its lowest compression level. The increases in runtime and energy consumption are negligible for the

PETRA III data set. Zstd has also been tested with levels 11 and 22 besides its lowest compression level. As can be seen from Figure 23 and Figure 26, level 11 improves both the achieved compression ratios and runtimes moderately. Level 22 gives even higher compression ratios but with significantly increased runtime and, thus, energy consumption.

MAFISC achieves the highest compression ratios in both cases due to its advanced pre-conditioners. However, this achievement is expensive because runtime and CPU utilization are increased significantly. Regarding energy consumption, it needs 8x and 3.5x the energy compared to running without compression for ECOHAM and PETRA III, respectively.

Concluding, SCIL was run with only one algorithm - LZ4 for all variables. Comparing to LZ4, its compression ratio is slightly better. This happened because more pre-conditioners were applied to algorithm in SCIL filter. However, as for the average energy consumption, also for the average CPU utilization, the values are almost the same as in the case of LZ4. Thus, obtained results show that SCIL itself does not have a negative impact on the performance of chosen algorithm (comparing to LZ4 or Blosc-LZ4). This is a highly important indicator in our work because this meta-compressor does not imply additional resources or costs.

Table IV: ECOHAM, PETRA III, and ECHAM data sets compressed using different HDF5 filters

Data set	Filter	Comp.	Runtime	CPU	Energy
ECOHAM	off	1.00	06:58	10 %	24 J
	blosc-lz4	3.82	04:48	21 %	16,5 J
	mafisc	5.97	49:25	93 %	189,4 J
	lz4	5.09	04:05	17 %	14,2 J
	zstd	5.78	04:35	32 %	16,2 J
	zstd-11	5.85	11:47	76 %	44,3 J
	zstd-22	5.98	2:11:50	97 %	508 J
	scil	5.24	03:47	24 %	15 J
PETRA III	off	1.00	27:28	84 %	103 J
	blosc-lz4	1.56	26:58	87 %	101 J
	mafisc	2.57	1:23:04	97 %	318 J
	lz4	1.28	26:57	87 %	100 J
	zstd	2.03	28:12	92 %	105.4 J
	zstd-11	2.12	38:08	95 %	142 J
	zstd-22	2.16	1:20:49	98 %	309 J
	scil	1.35	29:03	85 %	108 J
ECHAM	off	1.00	2:59	99 %	11 kJ
	blosc-lz4	1.95	3:05	99 %	11,7 kJ
	mafisc	2.36	20:12	99 %	77 kJ
	lz4	1.5	3:03	99 %	11,7 kJ
	zstd	1.8	3:17	99 %	12,6 kJ
	zstd-11	1.82	4:50	99 %	18,2 kJ
	zstd-22	1.91	34:30	99 %	132 kJ
	scil	1.5	4:58	71 %	18 kJ

Additionally, we have tested ECHAM data which are smaller (4 GB only) than ECOHAM and PETRA III data (see Table IV). ECHAM data set contains 135 double and float variables. Average CPU utilization is 99% for all the algorithms, excluding SCIL, where CPU utilization equals 71 %. The Blosc-LZ4 saves more energy for this data set than others.

Outcome and Methodology

After collecting all the metrics for each of the applied compressors, now it is possible to look at the dependencies between metrics. We can compare results for all combinations (CR vs. Time, CR vs. Energy, CR vs. CPU, Time vs. CPU, Time vs. Energy, CPU vs. Energy), however most of the graphs show chaotic results, except of one, which is depicted on Figure 27 where energy *E* is directly

proportional to time *T*. This can be easily explained by the formula: *Energy=Power*Time*. Therefore, the less time an algorithm needs for compression, the less energy it consumes. This can be taken as a base line for a power-aware compression selection. To this end, we can now observe that the most efficient by energy, time and CPU usage for both data sets is LZ4 compressor with acceptable compression ratio (CR). Hence, this algorithm can be employed for data compression by default.

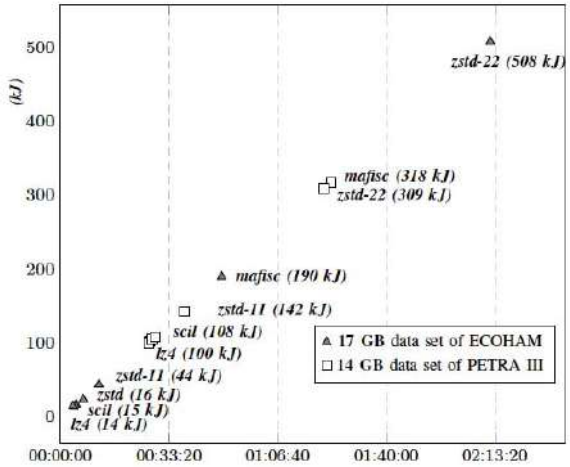


Figure 27: Consumed Energy vs. Runtime of compressors

As can be seen from the evaluation results, there is a trade-off between compression ratio and energy consumption. With defined accuracy for these metrics (when small increases may be negligible) more appropriate algorithms can be mapped for each data set by all the parameters: Zstd-11 for ECOHAM with CR=5.85 and Zstd for PETRA III with CR=2.03. In this way, if the input data have approximately the same data structure as ECOHAM for instance, SCIL can leverage Zstd-11. Thus, based on the results of performed compression for given data sets we can describe basic steps at the beginning of reduction strategy selection:

- LZ4 is set up by default. This option takes into account all the metrics.
- If only CR is important for a user, then MAFISC can be applied to the data set.
- If CR is important for a user with taking into account the energy consumption, then the data structure must be checked and compared to the obtained results (here user can establish the accuracy and semantic meta information should be available): Regarding the characteristics of input data set one of the following compressors can be used (Blosc-LZ4, Zstd, Zstd-11).

All these observations are of great significance for the methodology of compression algorithm selection. MAFISC can be employed when only the ratio matters, and LZ4 or Zstd when runtime, energy or CPU load are also important. Thus, if the given data set at the input has a similar structure as ECOHAM or PETRA III, we already have a defined reduction strategy. In order to make the strategy selection more precise, evaluation of various compressors on different data must be performed with collection of various metrics. For taking an intelligent decision on what compression can be applied to the given data, it is possible to leverage a trained decision tree

from machine learning which can provide to the user these different options in algorithm selection based on the meta data and performance of algorithms. Design and implementation of such a decision support unit is an ongoing work that is a part of a future work.

In addition to the elaborated methodology, measured energy consumption of every evaluated compressor gives also an opportunity to calculate and see how costs for data center maintenance are influenced by applying reduction to the stored data. Left part of Table III presents electricity costs in C of each HDF5 filter used to compress all three data sets. Prices are based on example of DKRZ from Introduction section, where every used 1 kWh costs 0,14 €. As can be seen, all the costs for energy that we spend during compression runs are very low. The highest price (almost 2 cents) for compression has Zstd-22 applied to ECOHAM and the majority of other costs are way less than 1 cent.

Table V: Expenditures of evaluated compression and annual storage costs for reduced data set sizes (DKRZ example)

Filter	Compression costs (€)			Annual storage costs (€)		
	ECOHAM	PETRA III	ECHAM	ECOHAM	PETRA III	ECHAM
off	0.0009	0.0040	0.0004	60.00	50.00	14.00
blosc-lz4	0.0006	0.0039	0.0005	15.71	32.05	7.18
mafisc	0.0074	0.0124	0.0030	10.05	19.46	5.93
lz4	0.0006	0.0039	0.0005	11.79	39.06	9.21
zstd	0.0006	0.0041	0.0005	10.38	24.63	7.78
zstd-11	0.0017	0.0055	0.0007	10.26	23.58	7.69
zstd-22	0.0198	0.0120	0.0051	10.03	23.15	7.33
scil	0.0006	0.0042	0.0007	11.45	37.04	9.33

The right part of the Table V shows costs (again, based on DKRZ example) for storing every data set for one year. Prices to keep ECOHAM, PETRA III and ECHAM uncompressed in storage system of DKRZ are approximately 60 €, 50 € and 14 € accordingly. However, applying LZ4 compressor even by default, which costs less than 1 cent, reduces these prices to approximately 12 C, 40 C and 10 € accordingly. Thus, couple of cents spent to reduce a user's data with the highest compression ratio will save much more money needed for an annual storage.

Related Work

Many research efforts have been dedicated to improve the performance of in-memory storage systems. However, very little visibility exists on their energy-efficiency although there have been a lot of works on improving the energy-efficiency of disk-based storage systems.

In-memory storage systems More and more companies are adopting in-memory storage systems. Facebook leveraged Memcached to build a distributed in-memory key-value store. Twitter used Redis and scaled it to 105TB of RAM. Alternatively, a lot of work has been proposed pushing the limits of performance of storage systems to a next level. As an example, MemC3 [FanAK13] is an enhancement of Memcached. Through a set of engineering and algorithmic improvements, it outperformed the original implementation by 3x in terms of throughput. Pilaf [MitchellGL13] is

an in-memory key-value store that takes advantage of RDMAs (Remote Direct Memory Access). Similarly, FaRM [DragojevicNCH14] is a distributed in-memory storage system based on RDMAs with transactional support.

Evaluating storage systems performance Many studies have been conducted to characterize the performance of storage systems and their workloads. In [AtikogluXF+12] a deep analysis is made on traces from Facebook's Memcached deployment and gives insight about the characteristics of a large-scale caching workload. In another dimension, the work in [RablGSMJM12] proposes to study the throughput of six storage systems. However, our goal is different since we study features implemented in RAMCloud and relate them to performance and energy-efficiency. It is noteworthy that performance evaluations of other in-memory storage systems corroborate our results [LimHAK14]. More specifically, for the throughput of read-only and read-write workloads we see the same trends. These studies however do not discuss energy aspects.

Energy efficiency Many systems have been proposed to tackle the energy efficiency issue in storage systems. For instance, LoneStar RAID, a disk-based storage architecture, focuses on high reliability and low energy consumption by applying massive array of independent disks (MAID) techniques with up to hundreds of disk drives per server [GrawinkelNB11]. Rabbit [AmurCGGKS10] is a distributed file-system built on top of HDFS (Hadoop File System) that tries to turn off the largest possible subset of servers in a cluster to save energy with the least performance decrease. It tries to minimize the number of nodes to be turned-on in case of primary replica failure by putting data in a minimal subset of nodes. In a similar way Sierra [ThereskaDN11] is a distributed object store that aims at power-proportionality. Through a three-way replication, it allows servers to be powered-down or put in stand-by in times of low I/O activity. This is made possible through a predictive model that observes daily I/O activity and schedules power-up or power-down operations. In [HassanCN15], authors propose a placement strategy for hot and cold objects in DRAM and NVRAM respectively. On the hardware level, MemScale [DengMRWB11] proposes to scale down DRAM frequency to save energy, while Grawinkel et al. investigate the impact of on-disk erasure codes on energy efficiency [GrawinkelSBHP11]. In [ChihoubIKAPB15] authors explore the energy-consistency trade-off and reveal that energy can vary considerably according to the level of consistency. In contrast with these works, we aim at characterizing the energy footprint of a set of features implemented in the RAMCloud storage system.

Compression The impact of compression on I/O throughput is studied in [WeltonKCPIR11]. The results show that the achievable throughput is highly dependent on the chosen algorithm and data properties because slow algorithms or incompressible data can decrease throughput significantly. One way to compensate for this drawback is to implement these algorithms in hardware [BeniniBMM02]. Authors of [AbdelfattahHS14] have implemented gzip on FPGAs using OpenCL. Their implementation offers a throughput of 3 GB/s in comparison to 300MB/s for a highly-optimized CPU implementation. Moreover, performance-per-watt ratio from the FPGA implementation is twelve times better than the one from the CPU implementation. However, not all accelerator-based implementations are faster than CPU-based implementations. In [PatelZMDO12], the authors have implemented bzip2 on an NVIDIA GTX 460 and found that their implementation is more than two times slower than the original. One of the reasons for this is the fact that all data have to be transferred via the PCIe bus to the GPU and back. Thus, compressing

data already on the compute nodes can be much more beneficial than porting compression methods to the accelerators.

It is furthermore important to foresee which reduction method will produce the best results. For example, [ChenGK10] presents a decision algorithm for MapReduce users to decide whether to use compression or not. The key factor here is a data compressibility which determines the cases when compression is worthwhile. With the introduced algorithm, the MapReduce framework becomes a more powerful tool for data centers. After studying the impact of compression on performance and energy efficiency for MapReduce data-intensive workloads, the authors of this work reported that compression provides up to 60 % energy savings for some jobs. Therefore, prediction is crucial.

In [RatanaworabhanKB06], the authors present a compression algorithm for arrays of 64-bit floating-point values. It predicts the next value in the array based on previous values and uses XOR to encode the difference between the predicted and actual value. For this, data analysis is very useful and also helps to determine appropriate reduction and to avoid negative information loss. In [BakerHMX+16], the authors examine properties of every individual variable of a data set produced by Community Earth System Model (CESM) [HurrelHG+13] and suggest applying compression on a per variable basis. Further, they target to implement an automated tool for appropriate lossy compressor identification which, however, focuses only on CESM's workflow [BakerXHLC17]. Suchlike approach has been considered in [Kuhn13] when semantic data from high-level I/O can be taken into account, which is unfortunately problematic for the underlying file system.

However, to the best of our knowledge, almost none of the previous works focuses on data reduction at the high levels of the HPC I/O stack and uses additional meta information for tuning the chosen technique or for identifying an appropriate reduction strategy. Moreover, energy consumption is not considered as a rule. It is precisely this gap we aim to address in our work.

Conclusions and Future Work

Our work on the in-memory storage system RAMCloud characterized the performance and energy consumption of a representative in-memory storage system to reveal the main factors contributing to performance degradation and energy-inefficiency. Firstly, we revealed that although RAMCloud scales linearly in throughput for read-only applications, it has a non-proportional power consumption. Mainly because it exhibits the same CPU usage under different levels of access.

Secondly, we have shown that prevalent Web workloads [AtikogluXF+12], i.e., read-heavy and update-heavy workloads, significantly impact performance and energy consumption. We relate it to the impact of concurrency, i.e., RAMCloud poorly handles its threads under highly-concurrent accesses.

Thirdly, we have shown that replication can be a major bottleneck for performance and energy. With update-heavy workloads, it can lead to 68% throughput degradation and 3.5x more energy

consumption when increasing the replication factor from 1 to 4. The root cause of this issue is the contention at the server level between clients' requests and replication requests. Moreover, the replication scheme, which implies to wait for acknowledgements from all backups exacerbates this problem.

Finally, we have quantified the overhead of a crash-recovery, which can end up in 90% CPU usage and 8% more power consumption on its own. We studied the impact of replication on this mechanism and find that, surprisingly, it has a negative effect on it, i.e., increasing the replication factor increases recovery time and energy consumption.

A natural extension for this work is to consider more workloads in order to cover more aspects of the system. E.g., one could think of scans to assess the indexing mechanism of the system. We consider as well evaluating the system with different request distributions. We are also considering to evaluate more systems, e.g., Memcached, Redis, etc. It can help in building a broader vision of the energy efficiency in in-memory storage systems.

Finally, we will explore the possibility to mitigate the replication overhead (in terms of performance and energy) as suggested by leveraging RDMAs. Naturally this implied tackling the availability vs durability trade-off under crashes. An interesting aspect to consider then would be correlated failures [LiL15].

Investigating the network impact of RAMCloud, we have been able to confirm foreseen results such as the scalability and throughput when using InfiniBand. However, we have discovered that using RAMCloud with TCP does not scale as well as with InfiniBand. This phenomenon becomes dramatic at high loads. Moreover, we discovered that using TCP leads to more energy consumption compared to using InfiniBand. When investigating the issue, we remarked that it was due to higher CPU occupation times whenever running with TCP.

As future work we plan to complete the picture of the main factors impacting performance and energy efficiency. To do so we plan to break down the design principles of RAMCloud such as the polling mechanism, log-structured memory, replication, crash-recovery mechanism, and study the impact of each of them on the performance and energy efficiency. Doing so can help system designer to build more energy-aware systems in the future. It can also open new research opportunities in investigating the trade-offs between energy efficiency and performance in in-memory storage systems.

Our ultimate goal would be to model the performance and energy consumption for in-memory storage systems. While this is a very hard task, it can have a huge impact as it can enable system designers to be aware of the impact of each feature in their system on the performance and energy consumption. Moreover, it can help system administrators tune their system in order to achieve better energy efficiency.

The results on data compression for HPC environments show that the amount of data which can be saved after using reduction techniques like compression heavily depends on the structure of data. Pre-conditioners such as byte- and bit-shuffling might work well for one data set, but they might worsen data savings for others. Moreover, one can observe that there is a delicate trade-off between compression ratio and energy consumption. Data reduction algorithms like MAFISC

can provide high compression ratios but at the same time increase energy costs and reduce throughput. In addition, different approaches are appropriate depending on the use case. Files for archival can be compressed with slower algorithms while parallel I/O should be handled as fast as possible.

In the presented work, we did not intend to run real-life applications and, thus, not accounted the energy consumption of this computation during the execution process. Instead, we leveraged the HDF5 compression filters to compare the performance with SCIL compressor which was used as HDF5 filter too. Measurements with real applications runs have been out of scope and are part of further research.

In the future, we plan to extend our evaluation of compression algorithms on various data sets in order to build more strategies and to discover what exactly influences on energy consumption during reduction process. Besides this, we aim also to experiment with additional application-specific data reduction techniques. Using semantic information (available at the application level) about data enables other techniques such as lossy compression or other transforms to reduce their precision. Moreover, deploying data reduction at the high levels of the HPC I/O stack allows their fine-tuning to take application requirements into account (different compression algorithms could be used for different types of variables). For instance, lossy compression could be used for less important variables. We also plan to experiment with techniques that are complex and/or expensive to employ at the system level such as deduplication. Using it only on a per-variable basis could help significantly reduce the costs in terms of memory that is required for the deduplication tables.

Obtained preliminary results in our paper now can be taken into account during ongoing work when implementing the decision unit for intelligent algorithms selection in a SCIL framework for scientific data reduction. It will identify appropriate data reduction strategies for HPC users based on relevant semantic information and performance metrics. For our purposes, collected metrics of LZ4, MAFISC and Zstd algorithms will be employed at the start.

References

[AbdelfattahHS14] M. S. Abdelfattah, A. Hagiescu, and D. Singh, “Gzip on a chip: high performance lossless data compression on FPGAs using OpenCL,” in Proceedings of the International Workshop on OpenCL, IWOCL 2013 & 2014, May 13-14, 2013, Georgia Tech, Atlanta, GA, USA / Bristol, UK, May 12-13, 2014, 2014, pp. 4:1–4:9.

[AlforovNKKL18] Y. Alforov, A. Novikova, M. Kuhn, J. Kunkel, T. Ludwig, “Towards green scientific data compression through high-level I/O interfaces.” In: SBAC-PAD 2018, Lyon, France. (In Press)

[AndersenFKPTV09] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, “Fawn: A fast array of wimpy nodes,” in Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles, ser. SOSP ’09, 2009, pp. 1–14.

[AmurCGGKS10] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan, “Robust and flexible power-proportional storage,” in Proceedings of the 1st ACM Symposium on Cloud Computing, ser. SoCC ’10, 2010, pp. 217–228.

[AtikogluXF+12] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, “Workload analysis of a large-scale key-value store,” in Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, ser. SIGMETRICS ’12, 2012, pp. 53–64.

[BakerHMX+16] A. H. Baker, D. M. Hammerling, S. A. Mickelson, H. Xu, M. B. Stolpe, P. Naveau, B. Sanderson, I. Ebert-Uphoff, S. Samarasinghe, F. De Simone et al., “Evaluating lossy data compression on climate simulation data within a large ensemble,” Geoscientific Model Development, vol. 9, no. 12, p. 4381, 2016.

[BakerXHLC17] A. H. Baker, H. Xu, D. M. Hammerling, S. Li, and J. P. Clyne, “Toward a multi-method approach: Lossy data compression for climate simulation data,” in High Performance Computing - ISC High Performance 2017 International Workshops, DRBSD, ExaComm, HCPM, HPC-IODC, IWOPH, IXPUG, P^3MA, VHPC, Visualization at Scale, WOPSSS, Frankfurt, Germany, June 18-22, 2017, Revised Selected Papers, 2017, pp. 30–42.

[BaloukAC+13] D. Balouek, A. C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Perez, F. Quesnel, C. Rohr, and L. Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In Cloud Computing and Services Science, volume 367 of Communications in Computer and Information Science, pages 3-20. Springer International Publishing, 2013.

[BartzCKNL15] C. Bartz, K. Chasapis, M. Kuhn, P. Nerge, and T. Ludwig, “A Best Practice Analysis of HDF5 and NetCDF-4 Using Lustre,” in High Performance Computing, pp. 274–281, 2015.

[BeniniBMM02] L. Benini, D. Bruni, A. Macii, and E. Macii, “Hardware-Assisted Data Compression for Energy Minimization in Systems with Embedded Processors,” in 2002 Design, Automation and Test in Europe Conference and Exposition (DATE 2002), 4-8 March 2002, Paris, France, 2002, pp. 449–453.

[BerdeGHH+14] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O’Connor, P. Radoslavov, W. Snow, and G. Parulkar, “Onos: Towards an open, distributed sdn os,” in Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ser. HotSDN ’14, 2014, pp. 1–6.

[BlomerG15] J. Blomer and G. Ganis, “Large-scale merging of histograms using distributed in-memory computing,” Journal of Physics: Conference Series, vol. 664, no. 9, p. 092003, 2015.

[ChenGK10] Y. Chen, A. Ganapathi, and R. H. Katz, “To compress or not to compress - compute vs. IO tradeoffs for mapreduce energy efficiency,” in Proceedings of the 1st ACM SIGCOMM Workshop on Green Networking 2010, New Delhi, India, August 30, 2010, 2010, pp. 23–28.

[ChihoubIKAPB15] H. E. Chihoub, S. Ibrahim, Y. Li, G. Antoniu, M. S. Pérez, and L. Boug, “Exploring energy-consistency trade-offs in cassandra cloud storage system,” in 2015 27th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2015, pp. 146–153.

[CidonRSKOR13] A. Cidon, S. Rumble, R. Stutsman, S. Katti, J. Ousterhout, and M. Rosenblum, “Copysets: Reducing the frequency of data loss in cloud storage,” in Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13), San Jose, CA, 2013, pp. 37–48.

[CooperSTRS10] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with ycsb,” in Proceedings of the 1st ACM Symposium on Cloud Computing, ser. SoCC ’10, 2010, pp. 143–154.

[DengMRWB11] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "Memscale: Active low-power modes for main memory," SIGPLAN Not., vol. 47, no. 4, pp. 225–238, Mar. 2011.

[DragojevicNCH14] A. Dragojević, D. Narayanan, M. Castro, and O. Hodson, "Farm: Fast remote memory," in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), Seattle, WA, Apr. 2014, pp. 401–414.

[FanAK13] B. Fan, D. G. Andersen, and M. Kaminsky, "Memc3: Compact and concurrent memcache with dumber caching and smarter hashing," Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, 2013, pp. 371–384.

[GrawinkelSBHP11] M. Grawinkel, T. Schäfer, A. Brinkmann, J. Hagemeyer, M. Porrmann, "Evaluation of Applied Intra-disk Redundancy Schemes to Improve Single Disk Reliability", 19th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Singapore, 25-27 July, pages 297 – 306, 2011.

[GrawinkelNMPBS15] M. Grawinkel, L. Nagel, M. Mäsker, F. Padua, A. Brinkmann, and L. Sorth, "Analysis of the ECMWF Storage Landscape", Proceedings of the 13th {USENIX} Conference on File and Storage Technologies (FAST), Santa Clara, CA, USA, February 16-19, 2015, pp. 15 – 27, 2015

[GrawinkelNB11] M. Grawinkel, L. Nagel, A. Brinkmann, "LoneStar RAID: Massive Array of Offline Disks for Archival Systems, ACM Transactions on Storage (ToS), 12(1), 5:1--5:29, 2016.

[GroßeGKLMPST15] F. Große, N. Greenwood, M. Kreus, H. Lenhart, D. Machoczek, J. Pätsch, L. A. Salt, and H. Thomas, "Looking beyond stratification: a model-based analysis of the biological drivers of oxygen depletion in the North Sea," Biogeosciences Discussions, pp. 2511–2535, 2015.

[HassanCN15] A. Hassan, H. Vandierendonck, and D. S. Nikolopoulos, "Energy- efficient hybrid dram/nvm main memory," in 2015 International Conference on Parallel Architecture and Compilation (PACT), 2015, pp. 492–493.

[HuebbeK13] N. Hübbe and J. M. Kunkel, "Reducing the HPC-datastorage footprint with MAFISC - Multidimensional Adaptive Filtering Improved Scientific data Compression," Computer Science - R&D, vol. 28, no. 2-3, pp. 231–239, 2013.

[HurrelHG+13] J. W. Hurrell, M. M. Holland, P. R. Gent, S. Ghan, J. E. Kay, P. J. Kushner, J.-F. Lamarque, W. G. Large, D. Lawrence, K. Lindsay et al., "The community earth system model: a framework for collaborative research," Bulletin of the American Meteorological Society, vol. 94, no. 9, pp. 1339–1360, 2013.

[Kuhn13] M. Kuhn, "A semantics-aware I/O interface for high performance computing," in Supercomputing - 28th International Supercomputing Conference, ISC 2013, Leipzig, Germany, June 16-20, 2013. Proceedings, 2013, pp. 408–421.

[KunkelNBS17] J. M. Kunkel, A. Novikova, E. Betke, and A. Schaare, "Toward decoupling the selection of compression algorithms from quality constraints", in ISC High Performance 2017 International Workshops

[KunkelNBS17b] J. Kunkel, A. Novikova, and E. Betke, "Towards Decoupling the Selection of Compression Algorithms from Quality Constraints – an Investigation of Lossy Compression Efficiency," Supercomputing Frontiers and Innovations, pp. 17–33, 12 2017.

[KablanAKL17] M. Kablan, A. Alsudais, E. Keller, and F. Le, “Stateless network functions: Breaking the tight coupling of state and processing,” in 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), Boston, MA, 2017, pp. 97–112.

[KaiserBSM15] J. Kaiser, A. Brinkmann, T. Süß, D. Meister, „Deriving and comparing deduplication techniques using a model-based classification”, Proceedings of the Tenth European Conference on Computer Systems (EuroSys), Bordeaux, France, April 21-24, pp. 11:1--11:13, 2015.

[LeePKMO15] C. Lee, S. J. Park, A. Kejriwal, S. Matsushita, and J. Ousterhout, “Implementing linearizability at large scale and low latency,” in Proceedings of the 25th Symposium on Operating Systems Principles, ser. SOSP ’15, 2015, pp. 71–86.

[LiL15] J. Li and B. Li, “Beehive: Erasure codes for fixing multiple failures in distributed storage systems,” in 7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15), Santa Clara, CA, 2015.

[LimHAK14] H. Lim, D. Han, D. G. Andersen, and M. Kaminsky, “Mica: A holistic approach to fast in-memory key-value storage,” in 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14), Seattle, WA, Apr. 2014, pp. 429–444.

[LorkowskiPMK12] I. Lorkowski, J. Pätsch, A. Moll, and W. Kühn, “Interannual variability of carbon fluxes in the North Sea from 1970 to 2006–Competing effects of abiotic and biotic drivers on the gas-exchange of CO_2,” Estuarine, Coastal and Shelf Science, vol. 100, pp. 38–57, 2012.

[MeisterKBCKK12] D. Meister, J. Kaiser, A. Brinkmann, T. Cortes, M. Kuhn, J. M. Kunkel, “A study on data deduplication in HPC storage systems”, Conference on High Performance Computing Networking, Storage and Analysis, (SC), Salt Lake City, UT, USA, November 11 - 15, 2012

[MitchellGL13] C. Mitchell, Y. Geng, and J. Li, “Using one-sided rdma reads to build a fast, cpu-efficient key-value store,” in Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13), San Jose, CA, 2013, pp. 103–114.

[NishtalaFGK+13] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, “Scaling memcache at facebook,” in Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13), Lombard, IL, 2013, pp. 385–398.

[OnargoRSOR11] D. Ongaro, S. M. Rumble, R. Stutsman, J. Ousterhout, and M. Rosenblum, “Fast crash recovery in ramcloud,” in Proceedings of the Twenty- Third ACM Symposium on Operating Systems Principles, ser. SOSP ’11, 2011, pp. 29–41.

[OusterhoutGG+15] J. Ousterhout, A. Gopalan, A. Gupta, A. Kejriwal, C. Lee, B. Montazeri, D. Ongaro, S. J. Park, H. Qin, M. Rosenblum, S. Rumble, R. Stutsman, and S. Yang, “The ramcloud storage system,” ACM Trans. Comput. Syst., vol. 33, no. 3, pp. 7:1–7:55, Aug. 2015.

[PatelZMDO12] R. A. Patel, Y. Zhang, J. Mak, A. Davidson, and J. D. Owens, “Parallel lossless data compression on the GPU,” in 2012 Innovative Parallel Computing (InPar), May 2012, pp. 1–9.

[RablGSMJM12] T. Rabl, S. Gómez-Villamor, M. Sadoghi, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii, “Solving big data challenges for enterprise application performance management,” Proc. VLDB Endow., vol. 5, no. 12, pp. 1724–1735, Aug. 2012.

[RatanaworabhanKB06] P. Ratanaworabhan, J. Ke, and M. Burtscher, “Fast Lossless Compression of Scientific Floating-Point Data,” in 2006 Data Compression Conference (DCC 2006), 28-30 March 2006, Snowbird, UT, USA, 2006, pp. 133–142.

[RoecknerBBBEGHKKM+03] E. Roeckner, G. Bäuml, L. Bonaventura, R. Brokopf, M. Esch, M. Giorgetta, S. Hagemann, I. Kirchner, L. Kornblueh, E. Manzini et al., “The atmospheric general circulation model echam 5. part i: Model description,” Max Planck Institute for Meteorology, 2003.

[RumbleKO14] S. M. Rumble, A. Kejriwal, and J. Ousterhout, “Log-structured memory for dram-based storage,” in Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14), Santa Clara, CA, 2014, pp. 1–16.

[SchlachterCE16] J. Schlachter, V. Camus, and C. C. Enz, “Design of energy-efficient discrete cosine transform using pruned arithmetic circuits,” in IEEE International Symposium on Circuits and Systems, ISCAS 2016, Montréal, QC, Canada, May 22-25, 2016, 2016, pp. 341–344.

[TalebIAT17] Y. Taleb, S. Ibrahim, G. Antoniu, and T. Cortes, “Understanding how the network impacts performance and energy-efficiency in the RAMCloud storage system,” Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, (CCGRID), Madrid, Spain, May 14-17, pages 1027 – 1034, 2017

[TalebIAT17b] Y. Taleb, S. Ibrahim, G. Antoniu, and T. Cortes, “Characterizing Performance and Energy-Efficiency of the RAMCloud Storage System”, 37th {IEEE} International Conference on Distributed Computing Systems (ICDCS), pp. 1488 – 1498, Atlanta, GA, USA, June 5-8, 2017

[ThereskaDN11] E. Thereska, A. Donnelly, and D. Narayanan, “Sierra: Practical power- proportionality for data center storage,” in Proceedings of the Sixth Conference on Computer Systems, ser. EuroSys ’11, 2011, pp. 169–182.

[TinnefeldKGBRSP13] C. Tinnefeld,D. Kossmann,M. Grund, J.-H. Boese, F. Renkes, V. Sikka, and H. Plattner, “Elastic online analytical processing on ramcloud,” in Proceedings of the 16th International Conference on Extending Database Technology, ser. EDBT ’13, 2013, pp. 454–464.

[UdipiMCBDJ10] A. N. Udipi, N. Muralimanohar, N. Chatterjee, R. Balasubramonian, A. Davis, and N. P. Jouppi, “Rethinking dram design and organization for energy-constrained multi-cores,” in Proceedings of the 37th Annual International Symposium on Computer Architecture, ser. ISCA ’10, 2010, pp. 175–186.

[WangZTLGGMM15] Y. Wang, L. Zhang, J. Tan, M. Li, Y. Gao, X. Guerin, X. Meng, and S. Meng, “Hydradb: A resilient rdma-driven key-value middleware for in-memory cluster computing,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC ’15, 2015, pp. 22:1–22:11.

[WeltonKCPIR11] B. Welton, D. Kimpe, J. Cope, C. M. Patrick, K. Iskra, and R. B. Ross, “Improving I/O Forwarding Throughput with Data Compression,” in 2011 IEEE International Conference on Cluster Computing (CLUSTER), Austin, TX, USA, September 26-30, 2011, 2011, pp. 438–445.

[WongA12] D. Wong and M. Annavaram, “Knightshift: Scaling the energy proportionality wall through server-level heterogeneity,” in Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, ser. MICRO-45, 2012, pp. 119–130.

[XiaJFDSHFZZ16] W. Xia, H. Jiang, D. Feng, F. Douglis, P. Shilane, Y. Hua, M. Fu, Y. Zhang, and Y. Zhou, “A Comprehensive Study of the Past, Present, and Future of Data Deduplication,” Proceedings of the IEEE, vol. 104, no. 9, pp. 1681–1710, 2016

[ZhangDC16] H. Zhang, M. Dong, and H. Chen, “Efficient and available in-memory kv-store with hybrid erasure coding and replication,” in 14th USENIX Conference on File and Storage Technologies (FAST 16), Santa Clara, CA, Feb. 2016.