Exploring Energy-Consistency Trade-offs in Cassandra Cloud Storage System

Houssem-Eddine Chihoub, Shadi Ibrahim, Yue Li, Gabriel Antoniu, María Pérez, Luc Bougé

Exploring Energy-Consistency Trade-offs in Cassandra Cloud Storage System

Houssem-Eddine Chihoub
LIG, Grenoble Institute of Technology
Grenoble, France
houssem.chihoub@imag.fr

Shadi Ibrahim[1]
Inria Rennes - Bretagne Atlantique
Rennes, France
shadi.ibrahim@inria.fr

Yue Li
Orange Lab
Cesson-Sevigne, France
yue2.li@orange.com

Gabriel Antoniu
Inria Rennes - Bretagne Atlantique
Rennes, France
gabriel.antoniu@inria.fr

María S. Pérez
Universidad Politécnica de Madrid
Madrid, Spain
mperez@fi.upm.es

Luc Bougé
ENS Rennes / IRISA
Rennes, France
luc.bouge@ens-rennes.fr

***Abstract*—Apache Cassandra is an open-source cloud storage system that offers multiple types of operation-level consistency including eventual consistency with multiple levels of guarantees and strong consistency. It is being used by many data-center applications (e.g., Facebook and AppScale). Most existing research efforts have been dedicated to exploring trade-offs such as: consistency *vs.* performance, consistency *vs.* latency and consistency *vs.* monetary cost. In contrast, a little work is focusing on the consistency *vs.* energy trade-off. As power bills have become a substantial part of the monetary cost for operating a data-center, this paper aims to provide a clearer understanding of the interplay between consistency and energy consumption. Accordingly, a series of experiments have been conducted to explore the implication of different factors on the energy consumption in Cassandra. Our experiments have revealed a noticeable variation in the energy consumption depending on the consistency level. Furthermore, for a given consistency level, the energy consumption of Cassandra varies with the access pattern and the load exhibited by the application. This further analysis indicates that the uneven distribution of the load amongst different nodes also impacts the energy consumption in Cassandra. Finally, we experimentally compare the impact of four storage configuration and data partitioning policies on the energy consumption in Cassandra: interestingly, we achieve 23% energy saving when assigning 50% of the nodes to the hot pool for the applications with moderate ratio of reads and writes, while applying eventual (quorum) consistency. This study points to opportunities for future research on consistency-energy trade-offs and offers useful insight into designing energy-efficient techniques for cloud storage systems.**

***Index Terms*—Cloud storage; replications; Cassandra; consistency; energy; Hot-N-Cold;**

I. Introduction

To meet the ever-growing user needs, large cloud providers have equipped their infrastructure with millions of servers distributed on multiple physically separate data-centers. This results in a tremendous increase in the energy consumed to operate these data-centers (i.e., electricity used for operating and cooling them) and ends up with high money bills in the order of millions of dollars (e.g., the annual electricity usage and bill of Google are over 1,120 GWh and $67 M, and they are over 600 GWh and $36 M for Microsoft [18]). As the cost is still increasing as both prices of energy and scale of data centers are on the rise (e.g., Hamilton [10] estimated that, in 2008, money spent on power consumption of servers and cooling units had exceeded 40 percent of total cost for data centers, which reached more than 1 million per month for a single data center.), many work have been proposed to reduce the energy consumption in the cloud through the Dynamic Voltage Frequency Scaling (DVFS) support in hardware [11], virtualization [22], or exploiting green energy [9].

Meanwhile, we have entered the era of Big Data, where the size of data generated by digital media, social networks and scientific instruments is increasing at an extreme rate. With data growing rapidly and applications becoming more data-intensive, many organizations have moved their data to the cloud, aiming to provide cost-efficient, scalable, reliable and highly available services through replicating their data across geographically diverse data-centers and direct users to the closest or least loaded site. A particularly challenging issue that arises in the context of storage systems with distributed data replication is how to ensure a consistent state of all the replicas.

As strong consistency by the mean of synchronous replication may limit the performance of some cloud applications, relaxed consistency models (e.g., weak, eventual, casual, etc) therefore have been introduced to improve the performance while guaranteeing the consistency requirement of the specific application. Furthermore, given that cloud applications are significantly varying in their consistency requirements (e.g., an e-shop requires strong consistency while social applications tolerate retrieving not up-to-date data), many cloud storage systems have adopted flexible (e.g., configurable) consistency models: giving the users the freedom to select the consistency level according to their application requirement. *Cassandra* is an open-source cloud storage system that offers multiple types of consistency per-operation including multiple levels of eventual consistency [6, 20] and strong consistency. It is

[1]Corresponding Author

being used by many data-center applications (e.g., Facebook (Instagram) [3] and AppScale [2]). Many research efforts have been dedicated to exploring and improving consistency-performance, consistency-latency and consistency-monetary cost trade-offs in Cassandra storage system [8, 16]. In contrast, little work is focusing on consistency-energy trade-off.

As power bills have become a substantial part of the monetary cost for operating a data-center, this paper aims to provide a clearer understanding of the interplay between consistency and energy consumption and offers useful insight into designing energy-efficient techniques for the Cassandra cloud storage system. *To the best of our knowledge, this is the first study on consistency-energy trade-off in storage systems.* Accordingly, a series of experiments have been conducted to explore the implication of different factors including the selected consistency level, the access pattern, and the load of the running applications on the energy-consumption. We do so through deploying Cassandra on 40-node on the Grid'5000 platform [12], powered by three power distribution units (PDUs). Each node is mapped to an outlet, thus we can export fine-grained power monitoring. Our experiments have revealed that the energy consumption varies not only according to the selected consistency level but also according to the access pattern and load exhibited by the application. Moreover, we observe that this variation is contributed to by the obtained throughput of the applications and also by the uneven distribution of the loads amongst different nodes in the Cassandra system.

The primary contributions of this paper are as follows:

1) *A study of the energy consumption of cloud applications in Cassandra cloud storage systems.* We find that variation in energy consumption cloud applications appears when different consistency settings are applied. Moreover, within the same consistency level, there is a noticeable variation in the energy consumption when varying the service load and the access pattern of the running applications.
2) *A micro-analysis to explain this variation and its cause.* We show that the total energy consumption of Cassandra cluster when adopting the *same consistency level* strongly depends on the running applications. More importantly, we observe a high variation in the load amongst the different storage nodes. This variation however increases when degrading the consistency level and therefore results in high variation in the energy consumption.
3) *Hot-N-Cold.* We investigate the impact of storage configuration and data partitioning on energy consumption and performance of cloud applications in Cassandra cloud storage systems. We show that by separating the cluster into *Hot* and *Cold* pools (i.e., the hot pool has larger data range and therefore hosts more data and is highly active, the cold pool however hosts less data and therefore is not highly active), we can reduce the energy consumption by up to 23%, for the quorum level, without any adversary impacts on the desired consistency.

It is important to note that the work we present here neither is limited to the Cassandra nor specific to the key/value store and can be applied to different cloud storage systems that are featured with flexible consistency rules.

Paper Organization. The rest of this paper is organized as follows: Section 2 briefly presents Cassandra consistency management, and discusses the related work. Section 3 describes the overview of our methodologies, followed by the experimental results in Section 4. Section 5 presents and experimentally discusses the impacts of partitioning Cassandra cloud storage systems into hot and cold pools on energy consumption. Finally, we conclude the paper and propose our future work in section 6.

II. Background and Related Work

In this section, we briefly introduce consistency management in Cassandra cloud storage system and then discuss the prior research work on consistency/energy management in cloud storage systems.

A. Consistency management in Cassandra

The way consistency is handled has a big impact on performance. Traditional synchronous replication (strong consistency) dictates that an update must be propagated to all the replicas before returning a success. In contrast, eventual consistency by means of asynchronous quorum replication [6, 17, 19] propagates data lazily to other replicas. Here the consistency level is, commonly, chosen on a per-operation basis and is represented by the number of replicas in the quorum (a subset of all the replicas). A quorum is computed as: $\lfloor (replication\ factor/2) + 1 \rfloor$. Data accesses and updates are performed to all replicas in the quorum. Thus, using this level for both read and write operations guarantees that the intersection of replicas involved in both operations contains at least one replica with the last update. A partial quorum has a smaller subset of replicas, hence returning the most recent data when read is issued, is not guaranteed.

In the Cassandra storage system, several consistency levels [23] are proposed per-operation. A write of consistency level *one* implies that data has to be written to the commit log and memory table of at least one replica before returning a success. Moreover, a read operation with consistency level *ALL* (strong consistency) implies that the read operation must wait for all the replicas to reply with consistent data in order to return the data to the client. However, this will introduce higher latency if some replicas are inconsistent with the most current version. In contrast, a read consistency level of quorum, 2 of the 3 replicas are contacted to fulfill the read request and the replica with the most recent version would return the requested data. In the background, a read repair will be issued to the third replica and will check for consistency with the first two. If inconsistency occurs, an asynchronous process will be launched to repair the stale nodes at a latter time.

B. Related Work

While there have been many research work in addressing consistency management and energy consumption in cloud

storage systems, but *none of them has addressed consistency and energy as a whole*.

Consistency management in cloud storage systems: Multiple analysis studies related to consistency were conducted over the years [5, 21]. *Wada et al.* [21] investigate the level of consistency provided by the commercial cloud storage platforms. Accordingly, they analyze the correlation between the consistency provided and both the performance and the cost. In [5], the authors study past workload executions in order to verify consistency properties and the level of guarantees provided by the underlying key/value store.

Recently few studies have been concentrated on improving the monetary cost of consistency in the cloud [8, 16]. The goal of these studies is to minimize the monetary cost of leased resources in the cloud through selecting the most appropriate consistency level that copes with both the dynamic state of the system and the change of service load. In contrast, we focus on the energy consumption: we aim to provide an in-depth study of energy-consistency trade-offs in cloud storage system.

Energy management in cloud storage systems: Energy consumption in the datacenter is an issue of extremely high importance. In this context, few approaches that attempt to reduce energy consumption for storage systems (underlying file systems for Hadoop [1, 13] mostly) were proposed [4, 15]. GreenHDFS [15] is an energy-conserving variant of HDFS. GreenHDFS divides the Hadoop cluster into *Hot* and *Cold* zones where a zone temperature is defined by its power usage as well as the performance requirements. Within GreenHDFS, data is classified in order to be placed in either zone. The classification's aim is to enlarge the idle time of servers within the cold zones by assigning to them the least solicited data. In contrast to related work, we introduce a first study that analyzes and shows how consistency can affect the energy consumption of the storage system.

III. Methodology Overview

The experimental investigation conducted in this paper focuses on exploring the implications of consistency management on the energy consumption of Cassandra under different workloads. We conducted a series of experiments in order to assess the impact of various consistency levels on both energy consumption and application performance.

A. Platform

The experiments were carried out on the Grid'5000 [12] testbed. The Grid'5000 project provides the research community with a highly-configurable infrastructure that enables users to perform experiments at large scales. The platform is spread over 10 geographical sites: 9 sites are located on French territory and 1 in Luxembourg. For our experiments, we employed nodes belonging to the Nancy site of the Grid'5000. These nodes are outfitted with a 4-core Intel 2.53 GHz CPU and 16 GB of RAM. Intra-cluster communication is done through a 1 Gbps Ethernet network. It is worth mentioning that only 40 nodes of the Nancy site are equipped with power monitoring hardware consisting of 2 Power Distribution Units (PDUs), each hosting 20 outlets. Since each node is mapped to a specific outlet, we are able to acquire coarse and fine-grained power monitoring information using the Simple Network Management Protocol (SNMP). It is important to state that Grid'5000 allows us to create an isolated environment in order to have full control over the experiments and the obtained results.

B. Benchmarks

We aim at a micro benchmark representing typical workloads in current services hosted in clouds. Based on case studies, we have selected the Yahoo! Cloud Serving Benchmark (YCSB) framework [24]. YCSB is used to benchmark Yahoo! cloud storage system "PNUTS". It is extended to be used with a variety of *open-source* data stores such as Amazon mongoDB, Hadoop HBase, and Cassandra. YCSB provides the features of a real cloud serving environment such as scale-out, elasticity, and high availability. For this purpose, several workloads have already been proposed in order to apply a heavy read load, heavy update load, and read latest load, among other workloads. Also, the benchmark is designed to make the integration of new workloads very easy.

We use YCSB-0.1.4 and we run three types of workloads that mimic real-life workloads. In particular,

- Heavy Reads (e.g., photo tagging). By setting the reads/updates ratio to 20/80%.
- Heavy Updates (e.g., user's bids at last minutes sell/auction). By setting the reads/updates ratio to 80/20%.
- Moderate Reads/Updates (e.g., session store recording recent actions). By setting the reads/updates ratio to 60/40%.

Furthermore, to assemble the variation in service loads (i.e., diurnal and monthly loads), we also vary the number of threads (i.e., concurrent clients accessing the system) from 20 to 100 threads.

C. Cassandra deployment

On the testbed described in Section III-A, we configured and deployed a Cassandra cluster — using 39 nodes on Nancy site — using the 1.1.4 Cassandra stable version. The replication factor was set to 5. Since we are exploring the energy-consumption within a single data-center, our replication strategy uses *SimpleStrategy* to enforce replication within one data-center.

Prior to running the benchmarks, we have inserted 2 millions $1KB$ records which represent totally $2GB$ of data into Cassandra. Each node will have $250MB$ of inserted data after replication. Each workload runs with 20 million operations on these nodes. We run each benchmark under three consistency levels: *One*, *Quorum* and *All*.

D. Metrics

In addition to the energy monitoring tools, described in section III-A, we gathered information about resource metrics related to the CPU usage which is a crucial component for

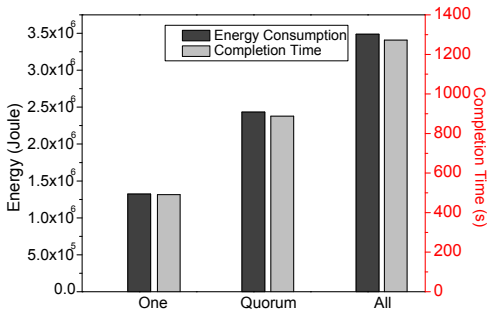


Fig. 1. **Energy vs completion time with different consistency levels:** *We run an application with moderate read/update ratio and 70 concurrent users.*

Fig. 2. **Energy consumption when applying eventual consistency with diverse applications**

many applications. These statistics are all gathered using Dstat. Therefore, minimizing the effects of the monitoring system on our resource measurements.

In each run we monitor the CPU usage for each node per second and we also measure the energy consumption with a resolution of one second.

IV. Experiments Results

The goal of our experiments is to measure the variance in energy-consumption when applying different consistency levels and to analyze the impact they may have on different applications. With this aim, we monitor the metrics described in Section III-D (i.e., the CPU usage) when applying eventual consistency with different applications (i.e., we covered 9 scenarios where applications differ in their access pattern and load as discussed in section III-B).

A. Consistency vs Energy

To give a general idea about the impacts of consistency management on the energy consumption of an application, we run an application with moderate read/update ratio and 70 concurrent users. The results of energy consumption for the three consistency levels are shown in Figure 1. These results show that the energy consumption for the three consistency levels (i.e., *One*, *Quorum*, and *All*) varies considerably.

As shown in Figure 1, the total energy consumption decreases when degrading the consistency level: the energy consumption reduces from 3.5×10^6 — when the consistency level is set to *All* — to 1.32×10^6 when the consistency level is *One* (*i.e.*, weak consistency reduces the energy consumption by almost 62%). This result was expected as a lower consistency level involves fewer replicas in the operations, and thus maintaining low latency, less I/O requests to the storage devices, and less network traffic in general (the run-time of the application varies from 491 to 1272 seconds according the consistency level). This energy reduction comes at the cost of a significant increase in the stale reads rate: 43% of the reads are stale reads — only 57% of the reads are fresh reads — when the consistency level is set to *One*. The stale read rate is an estimation metric proposed in our previous work [7]. This metric relies on the read rate, write rate in the storage cluster and the network state in order to perform probabilistic computation as to provide an estimation of potential stale reads rate.

To sum up, the total energy consumption of Cassandra cluster when running the *same application* strongly depends on the consistency level adopted: stronger consistency has higher energy consumption but higher rate of fresh reads and vice versa.

B. Energy consumption for diverse applications with Eventual consistency

Figure 2 depicts the energy consumption of Cassandra cluster for the six applications with *eventual* consistency (*Quorum* level). We can observe that in the case of *Heavy updates* workload Cassandra cluster consumes the least energy, meanwhile the highest consumption of Cassandra cluster is observed for *Moderate Reads/Updates* workload, exceeding that for the *Heavy Reads* workload. This results can be explained due to internal mechanisms of the Cassandra storage system: Cassandra, much like many NoSQL data stores, is optimized for write throughput. This is mainly because write operations are considered as extremely important and should always be available at a low latency. Therefore, write latency within Cassandra is very small since a write success is issued when data is written to the log file and the memory (not the disk). In this context, the small consumption of the *Heavy updates* workload is due to writes small latency. Write latency is even smaller than the read latency. Data in Cassandra is written to *memtables* in memory and flushed later to *sstables* that are written sequentially to disks. The *sstables* might however contain data rows that diverge overtime. In order to handle this issue, Cassandra implements a *Compaction* process in the background to merge *sstables*. This in turn, introduces extra latency when fetching data for read operations and thus explains why *Moderate Reads/Updates* workload consumes more energy. In the worst case scenario, a *Moderate Reads/Updates* workload results in a more frequent *compaction*, because of the high number of update operations, and therefore it

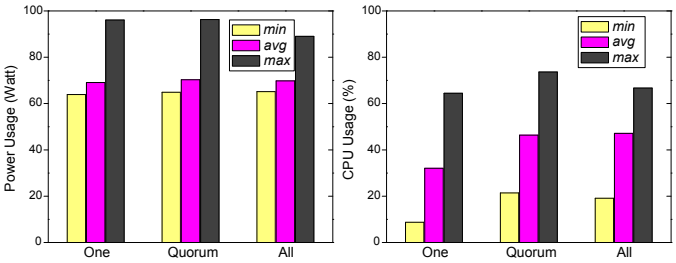


Fig. 3. **Minimum, Average, and Maximum usage of Power and CPU usage of the storage nodes**

further increases the read latency in comparison to the *Heavy Reads* workload. When the updates number is high more data is written to the *sstables* that grow high very fast at high probability of diverging rows (always because of potential updates to the same rows) thus increasing the frequency of the compaction operation that affect the read operations (which are far more numerous compared to the *Heavy updates* workload).

On the other hand, when the number of concurrent clients is small, the throughput in terms of served operations per second is small and therefore the run-time of application is long and the energy consumption is high.

To sum up, the total energy consumption of Cassandra cluster when adopting the *same consistency level* strongly depends on the running applications: applications with the same access pattern (i.e., either heavy updates or heavy reads) have lower latency and higher observed throughput and therefore lower energy consumption — better for write dominant workloads — however when the access pattern of the application is fluctuating between Reads and Updates, the latency grows and the energy consumption increases.

C. Energy consumption vs Uneven distribution with different consistency level

To further understand these results, we have compared the energy consumption of each node with Cassandra cluster. Here we run the same application Moderate Reads/Updates workload with 70 concurrent clients. Surprisingly, as shown in Figure 3, our results indicate that the average power usage (the average of power usage of all nodes at all time periods) differs slightly between the consistency levels. However, the gap between the max value and the min value is relatively large, and largest with the One level. This is an indicator of a potential variation of energy usage between nodes. The average of CPU usage on the other hand, is higher with the stronger consistency levels. This is mainly because CPUs are more loaded since more replicas (and thus nodes) are involved in data access operations. Moreover, there exists a huge gap between the max usage and the min usage, in particular for the One level where the min usage is roughly 8% (indicating that the node is almost idle) and the max value is approximately 64%. This gap is higher with CPU usage than the power usage

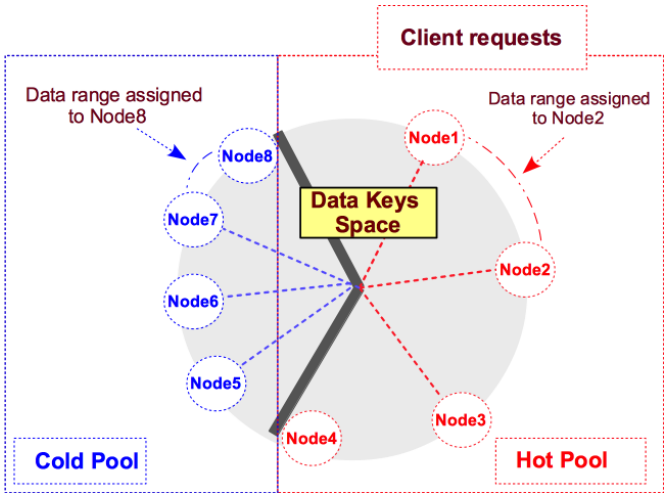


Fig. 4. **Data distribution for hot/cold Pools**: *hot nodes are exclusively responsible for handling client requests and acting as coordinators for user queries; every hot node is assigned a data range two times larger than the data range of a cold node.*

because the latter has a steady consumption portion of roughly 44 Watt (even at the idle time).

In summary, we observe a high variation in the load amongst the different storage nodes. This variation however increases when degrading the consistency level and therefore results in high variation in the energy consumption).

V. Hot-n-Cold

Our observations about the bias in resource and power usage in the storage cluster, in particular with low consistency levels, have led us to consider the investigation of the impact of storage configuration and data partitioning on energy consumption. In this section, we propose the reconfiguration of the storage cluster and conduct a series of experiments to demonstrate the impact on energy consumption with strong and eventual consistency.

A. Reconfiguration Approach

The results presented in the previous section show a variation in CPU usage and power usage between the different nodes of the storage cluster, in particular with low consistency levels. As previously explained, the issue is caused by considering all the nodes in the cluster as equals while not all of them are equally involved in data access operations. This results in inactive or lightly–active nodes consuming much power. In order to circumvent this situation, we propose an alternative cluster configuration of the storage system in order to save energy under certain scenarios. This configuration consists of dividing the storage nodes into two pools: the *hot pool* and the *cold pool*.

Although data is evenly distributed on nodes in the storage ring, not all of them are accessed in the same frequency. Moreover, with replication, applying a low level of consistency

when accessing data can highly contribute to the unbalancing of the load on nodes. Low consistency levels involve only a subset of replicas/nodes in operations which amplify the load-unbalancing impact of having more frequently-accessed data than other.

In Apache Cassandra, much as any consistent hashing scheme [14], each node should be assigned an equal data range based on a hashing function where data is partitioned according to the row key. To achieve this, each node is given a token that determines its position in the ring and its data range. In our new configuration, as shown in Figure 4, we generate new token for each node considering its nature: hot or cold. If the node is hot then its token is two times larger than a token of a cold node.

The hot pool. This pool includes the cluster nodes that are most active and highly consuming. Nodes within the hot pool are assigned with more responsibilities and larger data ranges as shown in Figure 4. Data partitioning in this case is reconfigured in order for hot nodes to be responsible of larger number of keys, possibly double assuming a uniform distribution of keys. In Cassandra, this is accomplished by assigning larger tokens in the ring to these nodes. Moreover, the hot nodes are exclusively responsible of handling client requests and acting as coordinators for queries. Upon a request arrival, a hot node will determine which node hosts data for the requested key. Since larger data ranges are assigned to hot nodes, the probability that the data–hosting node will fall within the hot pool is high.

The cold pool. In contrast to the hot pool, the cold pool includes nodes that are not / will not be highly active. These nodes are thus given less tasks and put on a low consumption mode (possibly using DVFS *Dynamic Voltage and Frequency Scaling* [11] technique to lower the CPU frequency). Therefore, data ranges assigned to the cold pool are smaller in order to reduce their involvements in operations as shown in Figure 4. Nodes in this case, are involved in operations when data fall in their (small) ranges, to respond to other replicas request with strong consistency levels, and with internal mechanisms of the storage system (Read Repair mechanism, and the Gossip protocol for Cassandra). As a result, with eventual consistency, the probability for a request to be served exclusively within the hot pool is high, especially when data keys are created carefully to insure uniform distribution of keys with tokens of equal sizes.

This configuration of Hot and Cold pools should be dynamic and adaptive. Cold nodes could join the hot pool during peak load times and hot nodes could join the cold pool during the not–so–busy periods. Moreover, the dynamic configuration has to consider the mostly-applied consistency level in the workload to adapt properly. The low levels introduce more variation between the nodes than the strong ones and provide better performance.

Reminder. However, the primary goal of this paper is to point to opportunities for future research on consistency-energy trade-offs and to offer useful insight into designing energy-efficient techniques for the Cassandra cloud storage system. As a first step towards energy-efficient consistency management in the cloud, we empirically analyze the energy-consistency trade-off of the Hot-N-Cold approach. *Therefore*, we keep the plan of building an adaptive reconfiguration approach that dynamically adapts the hot and the cold pool sizes to efficiently serve data while reducing the energy consumption, as a part of our future work.

B. Experimental Setup

In order to investigate the cluster reconfiguration and data range skew impact on energy consumption, we have run a moderate reads–updates workload (with a read/write ratio of 60/40) and 70 client threads. Moreover, we have used four storage cluster configurations where every node in the hot pool is assigned twice the size of data range assigned to a node in the cold pool:

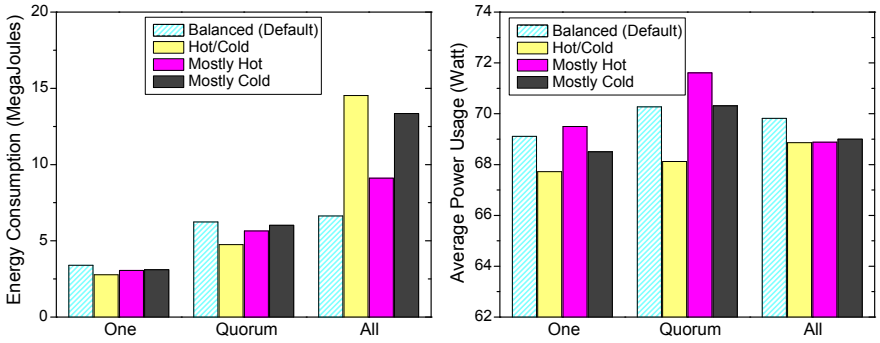
- *Balanced (default).* This is the native configuration with one pool of nodes that share the same tasks and host the same data size.
- *Hot/cold.* This configuration divides the nodes set into two equal subsets one assigned to the hot pool and the other to the cold pool.
- *Mostly hot.* 2/3 of nodes are assigned to the hot pool within this configuration and only the remaining 1/3 of nodes are assigned to the cold pool.
- *Mostly cold.* 2/3 of nodes belong to the cold pool while only 1/3 of nodes belong to the hot pool.

All the nodes in both the hot pool and the cold pool have the same hardware setup. As part of future experiments, we intend to lower the CPU frequency of all the nodes in cold pool.

C. Hot-N-Cold: Impact on Energy Consumption with Eventual Consistency

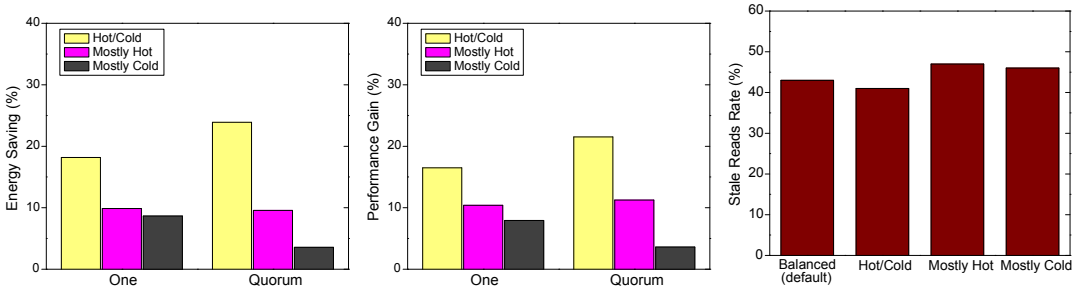
In this section, we discuss the impact of storage cluster reconfiguration and data range assigning strategies when consistency is eventual: the *One* consistency level and the *Quorum* consistency level. Although *Quorum* consistency yields consistent data to the user, a small subset of replicas can be in an inconsistent state. Instead, their convergence to a consistent state is eventual.

Energy Consumption and Power Usage Evaluation. Figures 5(a) and 5(b) show the overall energy consumption, and the average power usage, respectively, of our applied four configurations. Both the energy consumption and average power usage are lowest when the hot and the cold pools are of equal size (hot/cold configuration) with eventual consistency (The *One* and the *Quorum* consistency levels). Moreover, both the mostly hot and the mostly cold configurations consume less energy than the balanced configuration. This clearly shows that, for eventual consistency, a balanced configuration with a balanced data distribution client requests is not the best solution to reduce the energy consumption. Since only a subset of replicas are involved in data access operations, some nodes are more loaded than others. Thus, energy is wasted on *lazy* nodes. The average power usage for the mostly hot configuration is higher than the balanced one because of the high



(a) Energy Consumption (b) Average Power Usage

Fig. 5. **Energy consumption and average power usage of different configurations**



(a) Energy savings of unbalanced configurations (b) Performance gains of unbalanced configurations (c) Stale read rate of the One consistency level

Fig. 6. **Energy saving, performance gains, and stale read rate of eventual consistency levels with different configurations**

number of nodes in the hot pool. However, the overall energy consumption is smaller because of the performance gains (as shown in Figure 6(b)) that result in a smaller execution time. Moreover, the average power consumption for both the mostly hot configuration and the mostly cold configuration is higher than the average of hot/cold configuration, which explains, in addition to the performance gains, why energy consumption is lower with the latter.

Gain Analysis. In Figures 6(a), and 6(b), we show the energy savings and the performance gains (improved throughput in terms of served operations per second) respectively, compared to the balanced configuration (for eventual consistency). The configuration with equally–sized hot/cold pools achieves the highest savings that reach up to 19% of what is already a low consumption when the level of consistency is One, and up to 23% for the *Quorum* level. For this type of the workload applied, the configuration where most of the nodes belong to the cold pool is the one with the lowest saving (8% for the level One and only 3% for the level *Quorum*). Similarly, the performance gains are highest with the equally divided pools configuration and lowest with the mostly cold configuration.

Moreover, Figure 6(c) shows the stale reads rate of all the configurations with the consistency level One. The stale read rate is an estimation metric proposed in our previous work [7]. Both energy savings of the mostly hot and the mostly cold configurations come at the cost of adding a very small portion of stale reads to the one of balanced configuration (4% for the mostly hot and 3% of the mostly cold). Interestingly, we observe that the hot/cold configuration exhibits the lowest stale reads rate (41% vs. 43% for the balanced configuration) with the level *One*. Therefore, and for this type of workloads, the hot/cold configuration is the *best* choice for *eventual* consistency providing the highest energy saving, the highest performance, and the lowest stale reads rate.

From these results, we conclude that a self–adaptive reconfiguration is necessary to reduce the energy consumption related to storage. The self–adaptive approach must consider the observed throughput of the storage system and the most applied consistency level in the workload. Accordingly, the sizes of the hot pool and the cold pool should be computed.

VI. Discussion and Future Work

In the era of Big Data and with the continuous growth of the datacenter scale, energy consumption has become a pressing factor in recent years. Similarly, consistency management has become of even higher importance for storage systems that operate at massive scales. In this study, we have highlighted, for the first time, the impact of consistency on energy consumption in the datacenter. Therefore, we have shown by means of experimental evaluation how the choice of the consistency level affects the energy consumption of the storage cluster. We have demonstrated that the energy consumption is much higher with strong consistency levels. In contrast, the weakest consistency levels reduce (significantly) the energy

consumption but at the cost of high rates of inconsistency. Quorum–based levels are middle ground consistency levels that save a reasonable amount of energy without tolerating stale reads. We conclude that, when update conflicts are efficiently handled, the basic eventual consistency is the best choice to save energy with just a small fraction of stale reads under light accesses while quorum–based levels are a better choice under heavy accesses. In addition, in our analysis, we have demonstrated the presence of bias in the storage cluster with eventual consistency levels. Thereafter, we have introduced a cluster reconfiguration into hot and cold pools as an adaptive solution to further save energy with eventual consistency. Our experimental evaluation has shown that such a solution leads to energy–saving, enhanced performance, and reduced stale reads rate.

In considering future work, we plan to design a self-adaptive reconfiguration of Cassandra storage system. The main goal of this approach is to minimize the energy consumption for the runtime workload by automatically resizing the hot and the cold pools of nodes. The proposed approach must monitor the data access to keep track of peak load times as well as compute the read/write ratio. In addition, it must monitor the used consistency levels in the workload. Consequently, all these data are processed in order to determine the best configuration that achieves the adequate performance while reducing the energy consumption. Thereafter, it dynamically moves nodes from the hot pool to the cold pool and vice versa according to the needs.

ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see http://www.grid5000.fr/).

REFERENCES

[1] The Apache Hadoop Project. http://www.hadoop.org.

[2] Appscale 2.1: Putting the scale in appscale. http://www.appscale.com/tag/cassandra/, June 2015.

[3] Cassandra summit 2013: Instagrams shift to cassandra from redis by rick branson. http://planetcassandra.org/blog/cassandra-summit-2013-instagrams-shift-to-cassandra-from-redis-by-rick-branson/, June 2015.

[4] AMUR, H., CIPAR, J., GUPTA, V., GANGER, G. R., KOZUCH, M. A., AND SCHWAN, K. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM symposium on Cloud computing* (2010), SoCC '10, ACM, pp. 217–228.

[5] ANDERSON, E., LI, X., SHAH, M. A., TUCEK, J., AND WYLIE, J. J. What consistency does your key-value store actually provide? In *Proceedings of the Sixth international conference on Hot topics in system dependability* (Berkeley, CA, USA, 2010), HotDep'10, USENIX Association, pp. 1–16.

[6] CHIHOUB, H.-E., IBRAHIM, S., ANTONIU, G., AND PÉREZ, M. Consistency management in cloud storage systems. *Large Scale and Big Data - Processing and Management* (2014).

[7] CHIHOUB, H.-E., IBRAHIM, S., ANTONIU, G., AND PÉREZ-HERNÁNDEZ, M. S. Harmony: Towards automated self-adaptive consistency in cloud storage. In *2012 IEEE International Conference on Cluster Computing (CLUSTER'12)* (Beijing, China, 2012).

[8] CHIHOUB, H.-E., IBRAHIM, S., ANTONIU, G., AND PÉREZ-HERNÁNDEZ, M. S. Consistency in the cloud:when money does matter! In *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013)* (Delft, the Netherlands, 2013).

[9] GOIRI, I. N., LE, K., HAQUE, M. E., BEAUCHEA, R., NGUYEN, T. D., GUITART, J., TORRES, J., AND BIANCHINI, R. Greenslot: Scheduling energy consumption in green datacenters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis* (2011), SC '11, ACM, pp. 20:1–20:11.

[10] HAMILTON, J. Cost of Power in Large-Scale Data Centers. http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx, 2008.

[11] IBRAHIM, S., PHAN, T.-D., , CARPEN-AMARIE, A., CHIHOUB, H.-E., MOISE, D., AND ANTONIU, G. Governing energy consumption in hadoop through cpu frequency scaling: An analysis. *Future Generation Comp. Syst.* (2015).

[12] JÉGOU, Y., LANTÉRI, S., LEDUC, J., MELAB, N., MORNET, G., NAMYST, R., PRIMET, P., QUETIER, B., RICHARD, O., TALBI, E.-G., AND IRÉA, T. Grid'5000: a large scale and highly reconfigurable experimental Grid testbed. *International Journal of High Performance Computing Applications 20*, 4 (2006), 481–494.

[13] JIN, H., IBRAHIM, S., QI, L., CAO, H., WU, S., AND SHI, X. The mapreduce programming model and implementations. *Cloud Computing: Principles and Paradigms* (2011), 373–390.

[14] KARGER, D., LEHMAN, E., LEIGHTON, T., PANIGRAHY, R., LEVINE, M., AND LEWIN, D. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (1997), STOC '97, ACM, pp. 654–663.

[15] KAUSHIK, R. T., AND BHANDARKAR, M. Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster. In *Proceedings of the 2010 international conference on Power aware computing and systems* (2010), HotPower'10, USENIX Association, pp. 1–9.

[16] KRASKA, T., HENTSCHEL, M., ALONSO, G., AND KOSSMANN, D. Consistency rationing in the cloud: pay only when it matters. *Proc. VLDB Endow. 2* (August 2009), 253–264.

[17] LADIN, R., LISKOV, B., AND SHRIRA, L. Lazy replication: exploiting the semantics of distributed services. In *Proceedings of the ninth annual ACM symposium on Principles of distributed computing* (1990), PODC '90, ACM, pp. 43–57.

[18] QURESHI, A. Power-demand routing in massive geo-distributed systems. In *Ph.D.dissertation, MIT* (2010).

[19] SAITO, Y., AND SHAPIRO, M. Optimistic replication. *ACM Comput. Surv. 37*, 1 (march 2005), 42–81.

[20] VOGELS, W. Eventually consistent. *Commun. ACM* (2009), 40–44.

[21] WADA, H., FEKETE, A., ZHAO, L., LEE, K., AND LIU, A. Data consistency properties and the trade-offs in commercial cloud storage: the consumers' perspective. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings* (2011), pp. 134–143.

[22] XU, J., AND FORTES, J. A multi-objective approach to virtual machine management in datacenters. In *Proceedings of the 8th ACM International Conference on Autonomic Computing* (2011), ICAC '11, ACM, pp. 225–234.

[23] About Data Consistency in Cassandra. http://www.datastax.com/docs/1.0/dml/data_consistency, June 2015.

[24] Yahoo Cloud Serving Benchmark. https://github.com/brianfrankcooper/YCSB/wiki, June 2015.