2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems

Exploiting Replication for Energy Efficiency of Heterogeneous Storage Systems

Everett Neil Rush and Nihat Altiparmak
Department of Computer Engineering & Computer Science
University of Louisville, KY 40292, USA
{e.rush,nihat.altiparmak}@louisville.edu

***Abstract*—As a result of immense growth of digital data in the last decade, energy consumption has become an important issue in data storage systems. In the US alone, data centers were projected to consume $4 billion (40 TWh) yearly electricity in 2005. This cost had reached to $10 billion (100 TWh) in 2011, and expected to be around $20 billion (200 TWh) in 2016 by doubling itself every 5 years. In addition to the economic burden on companies and research institutions, these large scale data storage systems also have a negative impact on the environment. According to the EPA, generating 1 KWh of electricity in the US results in an average of 1.55 pounds of carbon dioxide emissions. Considering a projected 200 TWh energy requirement for 2016, energy-efficient data storage systems can have a huge economic and environmental impacts on society. This project exploits replication and heterogeneity existing in modern multi-disk storage systems and proposes an energy-efficient and performance-aware replica selection technique to reduce the energy consumption of data storage systems without negatively affecting their performance. Our proposed technique exploits the difference between active and idle energy consumption in heterogeneous disks holding the same replica and selects replicas by balancing energy and performance.**

***Keywords*-disk energy, replica selection, optimization**

I. Introduction

Advances in computer technology have enabled the generation and storage of massive amounts of data resulting in an increased energy consumption in large scale data storage systems. The following quote from Eric Schmidt, Executive Chairman and former CEO of Google, emphasizes the importance of energy conservation in storage systems dramatically: *"What matters most to the computer designers at Google is not speed but power — low power, because data centers can consume as much electricity as a city."* Current computer technology provides various means of storing digital data. Among them, Hard Disk Drives (HDD) and Solid State Drives (SSD) are the most commonly used devices for permanent data storage. HDDs are mechanical in nature and the motion of the mechanical parts forms a significant portion of the energy needs of an HDD. On the other hand, SSDs are fully electronic and do not contain any moving parts; therefore, SSDs are known to be more energy efficient compared to HDDs. In order to achieve larger storage space, better performance, and data reliability, modern data centers are large multi-disk storage systems composed of storage arrays and clusters of computers.

Replication and heterogeneity are commonly encountered in modern multi-disk storage systems. For instance, hybrid arrays have received the most attention recently among all storage arrays due to their balance of capacity, price, and performance [1]. Replication is commonly utilized in storage arrays through custom replication techniques or various available RAID levels (RAID1/01/10) [2]. As well as hybrid storage arrays, state-of-the-art distributed storage systems in both high-performance computing and big-data processing platforms generally include storage device heterogeneity since such clusters generally evolve over time, as newer storage devices are added to expand storage capacity and older or failing storage devices are replaced. Various parallel file systems including Ceph [3] and GPFS [4], distributed file systems including GFS [5] and HDFS [6], and key-value stores including Cassandra [7] and MongoDB [8] rely on replication for scalability, availability, and reliability purposes. They all divide data into disjoint regions called blocks (stripes/chunks), create multiple replicas for each block (generally three), and distribute them over storage nodes around the cluster.

In addition to the aforementioned benefits, replication has a potential to reduce the energy consumption of the storage system. In this paper, we exploit replication and heterogeneity existing in modern multi-disk storage systems and propose an energy-efficient replica selection technique without causing a major performance degradation. Our proposed technique exploits the difference between active and idle energy consumption in heterogeneous disks holding the same replica and performs the replica selection by balancing energy and performance. The main contributions include:

- Formulating the energy-optimal replica selection in heterogeneous storage systems as an optimization problem and solving it using linear programming techniques.
- Developing a low cost replica selection heuristic balancing energy and performance.
- Providing an extensive performance evaluation of the existing and proposed algorithms on various realistic heterogeneous storage configurations using real-world storage workloads, and investigating their performance compared with optimal performance and energy values.

II. Background and Related Work

A. Replica Selection Problem

Replica selection is an essential task of retrieving replicated datasets, and efficient retrieval of replicated data from multiple disks is a challenging problem. We are given a disk request

2375-0227/16 $31.00 © 2016 IEEE
DOI 10.1109/MASCOTS.2016.70

composed of multiple data blocks and each block can be replicated among multiple disks. A retrieval schedule specifies the replica to be selected for each block in the request and the performance of this selection is determined by the response time (latency) of the request. Response time of a disk request is defined as the time elapsed to complete the request and it can be divided into the following two sub-components:

- ***Service Time:*** The time the storage system spends servicing the request, beginning with the start of the first block and ending with completion of the last block of the request.
- ***Waiting Time:*** The elapsed time between the arrival of the request and the beginning of the service, including the queue wait time and the networking delay, if any.

Definition 1. *Retrieval schedule of a disk request is performance-optimal if the specified retrieval decision results in the minimum response time.*

B. Performance-Optimal Replica Selection

It is possible to achieve the performance-optimal retrieval schedule of a request by running an optimization algorithm. Chen and Rotem first formulated the replica selection problem as a flow network in [9] and solved it in polynomial time using max-flow techniques [10]. They assumed that all the disks are identical and the waiting times of the disks are zero. Authors in [11] generalized this problem and proposed a max-flow solution considering storage system heterogeneity and variable waiting times. This solution is further improved in [12–14] using parallelization and adaptive retrieval techniques.

C. Existing Replica Selection Heuristics

Replica selection heuristics are commonly used in real settings due to their simplicity. However, they do not guarantee the optimal performance and generally assume that the storage devices are homogeneous. Therefore, existing heuristics either employ a static selection mechanism or focus on reducing the waiting time of the requests by concentrating on the network delay or the queue wait time.

Static Replica Selection: Such techniques direct the block requests to a predefined replica called the *primary* replica [15]. This is a common approach in distributed systems implementing a primary back-up protocol that require a strong consistency model [16]. For instance, MongoDB implements this strategy by default. Another example system following a static replica selection strategy is [17], where the replicas are placed to the disks in a round-robin fashion as in RAID 0, and always the first replica is chosen in retrieval. The motivation behind [17] is achieving a better load balancing.

Network-Aware Heuristics: Such heuristics are generally preferred in geographically distributed systems. For instance, HDFS applies a network-delay aware replica selection mechanism such that when a client issues an HDFS read request, HDFS fetches the list of blocks and the locations of replicas from the NameNode, orders the replicas of a block based on their distance from the client, and selects the closest replica [6]. Google File System (GFS) does not explicitly describe the replica selection mechanism; however, it mentions that the read requests should be directed to the nearest replica [5]. Although MongoDB performs a static selection by default, it also provides an option for switching to a network-aware heuristic that uses the round-trip network delay [8].

Load-Aware Heuristics: Load-aware heuristics are generally applied in centralized settings such as accessing a single storage array or a cluster that is connected with high speed interconnects where node-to-node network delays are less than a millisecond. Therefore, such systems focus on improving the load balancing of the disks. One common replica selection technique is the *shortest-queue-first* algorithm [18] . It is implemented in RAID levels including replication such as RAID (RAID1/01/10), and in multimedia servers [19, 20]. In order to decide the replica to be selected for each block, queue lengths of the disks carrying a replica of that particular block are compared, and the disk with the shortest queue length (fewest number of blocks in its queue) is selected.

D. Energy Conservation Techniques for Storage Devices

Energy consumption of storage devices varies depending on their operation mode. For instance, an HDD can be in three different energy states: *active*, *idle*, and *stand-by*. It is defined to be in an active state while performing a Read/Write (R/W) operation, including the seek, rotational latency, and data transfer. Idle state indicates that the disk is not servicing a R/W operation; however, it remains spinning and ready to immediately begin the next request. The disk controller also remains active, but consumes less power because no data is being transferred [21]. Similarly, SSDs can also be in active or idle states; however, different than SSDs, HDDs are also capable of a stand-by state in which the disk is spun down, thus powering off the motor to save more energy.

Energy conservation techniques proposed in the literature for HDD based storage systems generally focus on spinning down the disks and switching to the stand-by mode when a duration of inactivity is experienced. One way to increase the inactivity period of a subset of disks in a multi-disk storage system is grouping popular data in a set of active disks through energy-efficient data (re)organization [22–27]. An alternative approach to increase the inactivity period is exploiting the existing replication and selecting the replicas from a set of active disks [28–30]. However, spinning disks up and down has several associated problems:

- Spin up/down increases wear and reduces lifetime. Well-known HDD manufacturers such as Hitachi and Seagate rate their HDDs for a maximum of 50,000 spin up/down cycles.
- Spin up is time consuming, increases the request response time, and incurs a significant energy penalty. For instance, IBM 36Z15 disk spends 10.9 sec. to spin up and this operation consumes 135 Joules, ~1800 times more energy consumption compared to the average energy spent to retrieve a 4 KB block from the same device.
- In an enterprise setting, the workload is typically such that a disk would not be idle long enough to benefit from such energy savings [21].

TABLE I: Factory Specifications of Common Enterprise Disks

HDD	PA (W)	PI (W)	PΔ (W)	Avg. Seek Time (ms)	Avg. Rotational Latency (ms)	Transfer Rate (MB/s)	EA (mJ)	EI (mJ)	EΔ (mJ)
Hitachi C15K600 (15K SAS - 600GB)	7.5	5.8	1.7	2.9	2.0	271.0	36.9	28.5	8.4
Hitachi C10K1800 (10K SAS - 600GB)	6.2	4.3	1.9	3.8	2.85	247.0	41.3	28.7	12.6
Hitachi 7K6000 (7.2K SATA - 2TB)	9.1	7.1	2.0	7.6	4.16	227.0	107.2	83.6	23.6

SSD	PA (W)	PI (W)	PΔ (W)	Avg. Access Time (ms)	Transfer Rate (MB/s)	EA (mJ)	EI (mJ)	EΔ (mJ)
Intel DC P3700 (PCIe - 400GB)	9.0	4.0	5.0	0.02	1800	0.20	0.09	0.11
Intel DC S3700 (SATA - 400GB)	5.2	0.6	4.6	0.05	300	0.33	0.04	0.29

III. Energy-Aware Replica Selection

In this paper, we target multi-disk heterogeneous storage systems involving replication and aim to decrease the energy consumption of the storage system through an energy-aware replica selection technique without spinning down the disks, and affecting applications' I/O performance significantly.

In Table I, we list the factory specifications of five heterogeneous disks (three HDDs and two SSDs) that are commonly used in current enterprise storage settings. The list shows that that different disks have different performance and power consumption ratings including access time, transfer rate, active power (PA), idle power (PI), and delta power ($P\Delta$) rates. Power rate and I/O performance together determine the energy consumption of a disk to perform a specific I/O task. Using these values, it is possible to calculate the active energy (EA), idle energy (EI), and delta energy ($E\Delta = EA - EI$) consumption of a specific disk to perform an I/O task of size b as follows:

$$\{EA, EI, E\Delta\}_{HDD} = \{PA, PI, P\Delta\} \cdot (t_{seek} + t_{rot} + b/t_{trans}) \quad (1)$$

$$\{EA, EI, E\Delta\}_{SSD} = \{PA, PI, P\Delta\} \cdot (t_{access} + b/t_{trans}) \quad (2)$$

The formulas differ for SSDs and HDDs since SSDs do not include an associated seek time (t_{seek}) and rotational latency (t_{rot}), instead they include a single access time (t_{access}). However, transfer time (t_{trans}) is common in both devices. The last three columns of Table I include these calculated EA, EI, and $E\Delta$ values of disks as a result of performing a 4KB read operation. As it is clear from the table, different disks consume different amount of energy to perform the same I/O task, and this difference can be exploited in the replica selection process to reduce the energy consumption of the storage system.

Although choosing the replica from the disk causing the smallest $E\Delta$ might seem like a reasonable solution at first, it should be noted that such a naive technique may actually affect the I/O performance negatively and cause more energy consumption due to unbalanced loads on the disks, consequently resulting in a larger idle energy consumption and longer application I/O times. In addition, since applications running in the system may take longer to terminate, energy consumption of additional system resources (CPU/Memory/Network) may also be increased. Therefore, in order to minimize the energy consumption of a storage system, we minimize the active energy consumption EA of disk requests, which is the combination of idle and delta energy consumptions ($EI + E\Delta$). We define the energy-optimal retrieval as follows:

Definition 2. *Retrieval schedule of a disk request is energy-optimal if the specified retrieval decision results in the minimum storage system energy consumption during its retrieval.*

The notation used in this paper is described in Table II.

TABLE II: Notation

Notation	Meaning
N	Number of disks in the system
Q	Number of blocks in the disk request (Query size in blocks)
r	Replication factor
PA_j	Active power rate of disk j
PI_j	Idle power rate of disk j
$P\Delta_j$	Delta ($PA - PI$) power rate of disk j
EA	Total active energy consumed to retrieve the request
EI	Total Idle energy consumed to retrieve the request
$E\Delta$	Total delta ($EA - EI$) energy consumed to retrieve the request
C_j	Retrieval cost for disk j
W_j	Waiting time for disk j
W	Maximum waiting time; $MAX\{W_j\}$; $j = 1, \ldots, N$
B_{ij}	1 if block i of the request is retrieved from disk j; 0 otherwise
L_j	Load of disk j for the request; $\sum_{i=1}^{Q} B_{ij}$
S_j	Service time of disk j for the request
R_j	Response time of disk j for the request; $R_j = W_j + S_j$
R	Response time of the request
I_j	0 when L_j is 0; 1 otherwise
I	0 when $R \leq W$; 1 otherwise

A. Energy-Optimal Replica Selection

Energy-optimal replica selection in heterogeneous systems can be formalized as an optimization problem as follows:

$$\begin{aligned}
&Minimize: EI + E\Delta \\
&Subject\ to: \sum_{j=1}^{N} B_{ij} = 1; \quad i = 1, \ldots, Q \\
&L_j = \sum_{i=1}^{Q} B_{ij} \\
&S_j = L_j \cdot C_j \\
&E\Delta = \sum_{j=1}^{N} S_j \cdot P\Delta_j \\
&I_j = 0 \rightarrow L_j = 0 \\
&R_j = I_j \cdot W_j + S_j \\
&R \geq R_j; \quad j = 1, \ldots, N \\
&I = 0 \rightarrow R \leq W \\
&EI = I \cdot (R - W) \cdot (\sum_{j=1}^{N} PI_j)
\end{aligned} \quad (3)$$

The objective function minimizes the active energy consumption of the storage system (EA) during the retrieval

of a request, which is calculated as the summation of idle energy consumption of all disks (EI) and the delta energy consumption ($E\Delta$) of the disks that are actively used in retrieval. In order to achieve this objective, we use $r \cdot Q + 1$ unique regular variables (B_{ij}s and R) and $N+1$ indicator variables (I_js and I). In addition to the regular and indicator variables, $Q+N+2$ regular constraints and $N+1$ indicator constraints are used in the formulation. Please note that PI_js, $P\Delta_j$s, C_js, W_js, and W are constants, and L_js, S_js, R_js, EI, and $E\Delta$ are not unique as they can be written in terms of B_{ij}s.

The constraint $\sum_{j=1}^{N} B_{ij} = 1; \;\; i = 1,\ldots,Q$ ensures that every block i is retrieved only from a single disk j. Service time calculation shown in the formulation above ($S_j = L_j \cdot C_j$) simply uses an average access time value as the retrieval cost of each block and multiplies the disk load (L_j) with this retrieval cost; however, it can be adapted easily to include the specific seek distance of the blocks for more accurate calculation in HDDs. $E\Delta$ is calculated by multiplying the service time S_j of every disk j with its delta power rate $P\Delta_j$. While calculating EI, it is important to subtract the maximum waiting time W from the request response time R before multiplying it with the idle power rate PI to eliminate the reconsideration of overlapping idle energy consumptions of consecutive requests. Finally, R_j holds the response time of each disk j considering its waiting time W_j and service time S_j, and the constraint $R \geq r_j$ for $j = 1,\ldots,N$ guarantees that the I/O response time R is minimized.

The indicator variable I ensures that the idle energy consumption EI never becomes negative, and the indicator variables I_j for every disk j ensure that the waiting time of a disk is only considered if that disk is used in retrieval. In IBM CPLEX solver, indicator constraints can easily be defined as: $I_j = 0 \rightarrow L_j = 0$ (for every j) and $I = 0 \rightarrow R \leq W$.

Since the last constraint of the formulation:

$$EI = I \cdot (R - W) \cdot (\sum_{j=1}^{N} PI_j)$$

multiplies a variable (I) with another variable R, the formulation is not linear in this form. However, it can easily be converted into a linear form by introducing two large constants (M_1 and M_2) and an intermediate variable (EI') to the model, and rewriting the definition of EI using the big-M method [31]. We define the intermediate value EI' such that:

$$EI' = (R - W) \sum_{j=1}^{N} PI_j$$

Then, EI can be defined using the big-M method as follows:

$$EI \geq EI' + M_1 \cdot I - M_1$$
$$EI \leq EI' + M_1 - M_1 \cdot I$$
$$EI \geq -M_2 \cdot I$$
$$EI \leq M_2 \cdot I$$

If $R > W$ (and $I = 1$), then the M_1 terms of the first two restrictions cancel, and only $EI = EI'$ is allowed. The last two would give no restriction ($-M_2 \leq EI \leq +M_2$). If $R \leq W$ (and $I = 0$), then the last two only allow $EI = 0$ and the first two give no restriction ($EI' - M_1 \leq EI \leq EI' + M_1$). Therefore, EI becomes $(R-W)\sum_{j=1}^{N} PI_j$ when $R > W$, and 0 otherwise, as we wanted to achieve in the original non-linear description of EI. Re-writing the definition of EI in a linear form using the big-M method requires the use of one additional variable (EI') and two additional constraints (M_1 and M_2) as above.

The integer linear programming formulation presented above guarantees the energy-optimal retrieval schedule, and it can be solved using an LP solver such as IBM's CPLEX; however, its execution time would generally be unacceptably high in a real world setting due to its NP-hard complexity [32]. *Nevertheless, having an optimal solution is crucial for comparison and evaluation purposes of the heuristic solutions, and to understand how much potential exists to improve the heuristics even further.*

B. Greedy-Energy with Load Balancing (GELB) Heuristic

In this section, we propose a low complexity greedy heuristic called Greedy-Energy with Load Balancing (GELB) for the energy-aware replica selection problem in heterogeneous storage environments. A detailed description of the heuristic is provided in Algorithm 1.

Algorithm 1 Greedy-Energy with Load Balancing (GELB)

Input: $W, N, Q, r,$ replica_to_disk$[\,], W[\,], C[\,], P\Delta[\,], PI[\,]$
Output: selection$[\,]$

```
1:  idle_end ← W
2:  for i ← 1 to N do
3:      load[i] ← 0
4:  for b ← 1 to Q do
5:      min_cost ← ∞
6:      for c ← 1 to r do
7:          d ← replica_to_disk[b,c]
8:          idle_extension ← W[d] + (load[d] + 1) · C[d] − idle_end
9:          if idle_extension > 0 then
10:             cost ← PΔ[d] · C[d] + idle_extension · ( Σ_{i=1}^{N} PI[i] )
11:         else
12:             cost ← PΔ[d] · C[d]
13:         if cost < min_cost then
14:             min_cost ← cost
15:             choice ← d
16:     selection[b] ← choice
17:     load[d]++
18:     disk_end ← W[choice] + load[choice] · C[choice]
19:     if disk_end > idle_end then
20:         idle_end ← disk_end
```

The proposed heuristic is not only considering the energy efficiency of the storage system, but also considers applications' I/O performance. In order to consider the energy efficiency, it includes the delta energy consumption in the replica selection cost function as in line 12. However, in order to eliminate extensive queue length in the lowest energy disk and possible I/O performance degradation, it additionally includes the idle energy consumption of all disks in the cost function of that replica selection when the candidate selection causes an increased waiting time W as in line 10. At the end, it chooses the replica with the lowest cost as in line 16. The function `replica_to_disk[b,c]` returns the disk holding the replica c of the block b, and `load[d]` holds the number of blocks scheduled for disk d. The heuristic has the time complexity of $O(rQ)$.

IV. Evaluation

In this section, we evaluate the performance of the existing and the proposed replica selection algorithms compared to the optimal performance and the energy values calculated using the performance-optimal and the energy-optimal solutions.

A. Experimental Setup

We perform trace-based simulations driven by real world storage workloads using two realistic heterogeneous storage configurations and various replication factors.

1) Disk Configurations: Motivated by the EMC VNXe3200 hybrid storage system's advertised heterogeneous disk configurations, we selected two interesting heterogeneous configurations as shown in Table III. We used the specific disk models provided in Table I in these configurations. We determined the number of disks for each device type in proportion with EMC's provided storage capacity for that particular device type. In addition, we also share EMC's advertised total capacity and dollar value for each configuration.

TABLE III: Heterogeneous Storage Configurations

Disk Type	**Config. 1** 21 TB ($19k)	**Config. 2** 33 TB ($42k)
7.2k HDD	12 (80%)	24 (64.9%)
10k HDD	—	11 (29.7%)
15k HDD	3 (20%)	—
SATA SSD	—	1 (2.7%)
PCIe SSD	—	1 (2.7%)
Total Disks:	15 disks	37 disks

2) Workloads: We perform evaluation using two popular real world storage workloads including block level I/O requests of an online transaction processing applications (OLTP) and a web search engine. These workloads are publicly distributed via the online trace repositories provided by the Storage Networking Industry Association (SNIA) [33] and University of Massachusetts Amherst (UMASS) [34]. Specific details of these workloads are as follows:

- ***TPC-E***: TPC-E is and OLTP benchmark simulating the workload of a brokerage firm. TPC-E is the successor of TPC-C, its transactions are more complex than those of TPC-C, and they more closely resemble modern OLTP transactions. The TPC-E trace covers 84 minutes of workload taken on 10/18/2007 and broken into 6 intervals of 10-16 minutes. We used the TPC-E trace provided by SNIA, which is originally generated by Microsoft.
- ***WebSearch***: The second workload we use is WebSearch, which is taken from a popular search engine. We used the WebSearch1 trace provided by UMASS.

Trace statistics including average/maximum request sizes and average request interarrival times are provided in Table IV.

TABLE IV: Trace Statistics

Trace	**Avg/Max Req. Size (KB)**	**Avg. Interarrival (msec)**
TPC-E	8.43 / 1024	0.04
WebSearch1	15.15 / 1111	2.98

3) Data Placement and Replication: Since the storage traces do not disclose the replica placement details, we need to use a replicated data placement (declustering) strategy to distribute the replicas of the requested blocks in the workloads to the available disks in the system. For this distribution, we use Random Duplicate Allocation (RDA)[35] that stores a block into r disks chosen randomly from all the disks in the system. Our motivation behind choosing this replica placement strategy is that RDA performs equally well for all request/query types while certain allocation strategies are generally optimized for specific query types. For instance, while orthogonal allocations [36] generally perform better for arbitrary queries, periodic allocation strategies [37] are more suitable for range queries. In order to observe the effect of replication factor, we also performed the experiments using various replication factors of $r = 2,3,4,5,6$.

B. Algorithms

We implemented the following algorithms for evaluation:

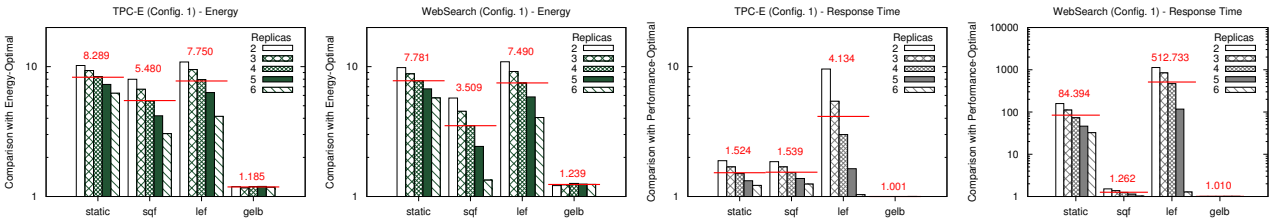
- ***Static*** is a simple heuristic solution following the idea of static replica selection described in Section II-C such that the predefined *primary* replica of each block is always chosen in the retrieval schedule.
- ***Shortest-Queue-First*** is a load-aware heuristic described in Section II-C such that the replica stored in the disk with the lowest number of blocks in its queue is chosen for retrieval.
- ***Lowest-Energy-First*** is a greedy heuristic choosing the replica from the disk causing the smallest $E\Delta$ energy consumption without considering the I/O performance.
- ***Performance-Optimal*** is the max-flow based performance-optimal algorithm described in Section II-B.
- ***Energy-Optimal*** is the integer linear programming based energy-optimal solution described in Section III-A.
- ***GELB*** is our proposed heuristic solution balancing energy and performance as described in Section III-B.

C. Experimental Results

Figures 1 and 2 present the energy consumption (Figures (a) and (b)) and I/O performance (Figures (c) and (d)) of the heuristics compared with the optimal solutions for config. 1 and config. 2, respectively. In these graphs, achieving 1x on the y-axis indicates that the corresponding heuristic performs as good as the the optimal solution, and having a larger y value indicates the deviation from the optimal value as well as indicating the remaining potential to improve the corresponding heuristic. Compared with the energy-optimal solution, performance-optimal solution, and various existing replica selection techniques, our analysis shows that the proposed GELB heuristic outperforms the existing heuristic solutions and achieves performance close to the optimal solutions. Specifically, GELB performs within 24% of the energy-optimal solution without causing significant I/O performance degradation over the performance-optimal solution.

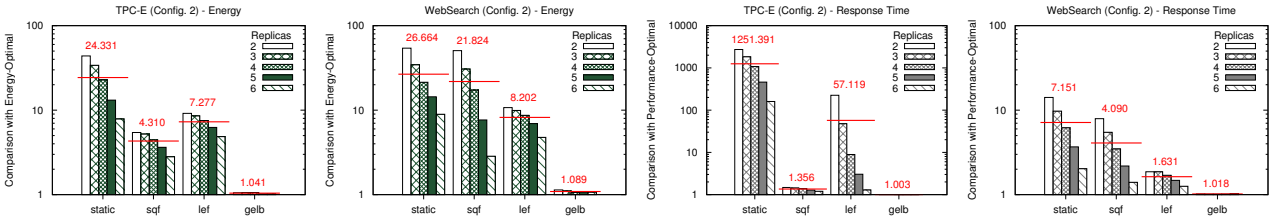
V. Conclusions

In this paper, first we formulate the energy-optimal replica selection as an optimization problem and solve it using linear programming techniques. Next, we propose the low cost GELB heuristic that balances the energy consumption of the



(a) TPC-E - Energy (b) WebSearch - Energy (c) TPC-E - Performance (d) WebSearch - Performance

Fig. 1: Config. 1



(a) TPC-E - Energy (b) WebSearch - Energy (c) TPC-E - Performance (d) WebSearch - Performance

Fig. 2: Config. 2

storage system and the I/O performance of applications. Compared with the energy-optimal solution, performance-optimal solution, and various existing replica selection techniques, our analysis shows that the proposed low-cost GELB heuristic outperforms the existing solutions and performs up to 24% within the optimal energy consumption while not causing a significant I/O performance degradation.

REFERENCES

[1] E. Burgener *et al.*, *Worldwide All-Flash Array and Hybrid Flash Array 20142018 Forecast and 1H14 Vendor Shares*, IDC, Jan 2015.

[2] D. A. Patterson *et al.*, "A case for redundant arrays of inexpensive disks (raid)," ser. SIGMOD '88. ACM, 1988, pp. 109–116.

[3] S. A. Weil *et al.*, "Ceph: A scalable, high-performance distributed file system," ser. OSDI '06. USENIX, 2006, pp. 307–320.

[4] F. Schmuck *et al.*, "Gpfs: A shared-disk file system for large computing clusters," ser. FAST '02. USENIX, 2002.

[5] S. Ghemawat *et al.*, "The google file system," ser. SOSP '03. ACM, 2003, pp. 29–43.

[6] K. Shvachko *et al.*, "The hadoop distributed file system," ser. MSST '10. IEEE, May 2010, pp. 1–10.

[7] A. Lakshman *et al.*, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, Apr. 2010.

[8] K. Chodorow *et al.*, *MongoDB: The Definitive Guide*, 1st ed. O'Reilly Media, Inc., 2010.

[9] L. T. Chen *et al.*, "Optimal response time retrieval of replicated data," ser. SIGMOD/PODS '94. ACM, 1994, pp. 36–44.

[10] L. R. Ford *et al.*, "Maximal Flow through a Network." *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[11] N. Altiparmak *et al.*, "Generalized optimal response time retrieval of replicated data from storage arrays," *ACM Transactions on Storage*, vol. 9, no. 2, pp. 5:1–5:36, Jul. 2013.

[12] N. Altiparmak *et al.*, "Integrated maximum flow algorithm for optimal response time retrieval of replicated data," in *41st International Conference on Parallel Processing (ICPP 2012)*, Pittsburgh, Pennsylvania, September 2012.

[13] N. Altiparmak *et al.*, "Continuous retrieval of replicated data from heterogeneous storage arrays," in *22nd IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2014)*, Paris, France, September 2014.

[14] N. Altiparmak *et al.*, "Multithreaded maximum flow based optimal replica selection algorithm for heterogeneous storage architectures," *IEEE Transactions on Computers*, vol. 65, no. 5, May 2016.

[15] P. A. Alsberg *et al.*, "A principle for resilient sharing of distributed resources," ser. ICSE '76. IEEE, 1976, pp. 562–570.

[16] N. Budhiraja *et al.*, "Distributed systems (2nd ed.)." ACM Press and Addison-Wesley Publishing Co., 1993, pp. 199–216.

[17] S. W. Son *et al.*, "Reliable mpi-io through layout-aware replication," ser. SNAPI '11, Denver, CO, 05/2011 2011.

[18] W. H. Tetzlaff *et al.*, *Block allocation in video servers for availability and throughput*, IBM US Research Centers, 1996.

[19] J. R. Santos *et al.*, "Comparing random data allocation and data striping in multimedia servers," ser. SIGMETRICS '00. ACM, 2000, pp. 44–55.

[20] R. Muntz *et al.*, "A parallel disk storage system for realtime multimedia applications," *International Journal of Intelligent Systems*, vol. 13, 1998.

[21] T. Bostoen *et al.*, "Power-reduction techniques for data-center storage systems," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 33:1–33:38, Jul. 2013.

[22] D. Colarelli *et al.*, "Massive arrays of idle disks for storage archives," ser. SC '02. IEEE Computer Society Press, 2002, pp. 1–11.

[23] E. Pinheiro *et al.*, "Energy conservation techniques for disk array-based servers," in *ACM ICS 25th Anniv. Volume*. ACM, 2014, pp. 369–379.

[24] Q. Zhu *et al.*, "Hibernator: Helping disk arrays sleep through the winter," *SIGOPS Oper. Syst. Rev.*, vol. 39, no. 5, pp. 177–190, Oct. 2005.

[25] T. Xie, "Sea: A striping-based energy-aware strategy for data placement in raid-structured storage systems," *IEEE TC*, vol. 57, no. 6, Jun. 2008.

[26] E. Otoo *et al.*, "Dynamic data reorganization for energy savings in disk storage systems," ser. SSDBM'10, 2010, pp. 322–341.

[27] Y. Chai *et al.*, "Efficient data migration to conserve energy in streaming media storage systems," *IEEE TPDS*, vol. 23, no. 11, Nov 2012.

[28] E. Pinheiro *et al.*, "Exploiting redundancy to conserve energy in storage systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, no. 1, Jun. 2006.

[29] J. Kim *et al.*, "Using replication for energy conservation in raid systems." in *PDPTA*. CSREA Press, 2010, pp. 703–709.

[30] J. C.-Y. Chou *et al.*, "Exploiting replication for energy-aware scheduling in disk storage systems," *IEEE TPDS*, vol. 26, no. 10, Oct. 2015.

[31] W. Winston *et al.*, *Operations Research: Applications and Algorithms*. Thomson Brooks/Cole, 2004.

[32] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, vol. 40, no. 4, pp. 85–103, 1972.

[33] *SNIA IOTTA Repository*, Storage Networking Industry Association, http://iotta.snia.org.

[34] *UMass Trace Repository*, University of Massachusetts Amherst, http://traces.cs.umass.edu/index.php/Storage/Storage.

[35] P. Sanders *et al.*, "Fast concurrent access to parallel disks," in 11^{th} *ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[36] A. S. Tosun, "Replicated declustering for arbitrary queries," in 19^{th} *ACM Symposium on Applied Computing*, March 2004, pp. 748–753.

[37] N. Altiparmak *et al.*, "Equivalent disk allocations," *IEEE Trans. on Parallel and Dist. Systems*, vol. 23, no. 3, pp. 538–546, March 2012.