Dynamic Grid Quorum: A Novel Approach for Minimizing Power Consumption without Data Migration in Grid Quorums

Munetoshi Ishikawa, Koji Hasebe, Akiyoshi Sugiki, and Kazuhiko Kato
Graduate School of Systems and Information Engineering
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305–8573, Japan
munetoshi@osss.cs.tsukuba.ac.jp, hasebe@iit.tsukuba.ac.jp, {sugiki, kato}@cs.tsukuba.ac.jp

Abstract

In this paper, we present the dynamic grid quorum, a technique for reducing the power consumption of large-scale storage systems. The key idea is to skew the workload towards a small number of quorums by means of a novel optimization algorithm. Moreover, this system allows reconfiguration by exchanging nodes without any data migration so as to reallocate high-capacity nodes to busier quorums. We demonstrate that the dynamic grid quorum saves, on average, 6-12% energy when compared with the static configurations.

1. Introduction

Service-oriented computing is a key paradigm in achieving the long-awaited, richer environments in computer science. To achieve such platforms for large-scale business computing systems, reliable, available, and cost-effective data management is required.

The use of quorum systems is a promising approach for achieving reliable data management in such platforms. Compared with the traditional read-one write-all technique, the quorum approach allows flexibility in storage configuration and decentralized management for consistency control. Since the first quorum paper [7] was presented, much research has focused on investigating the possibilities of quorum systems. These studies are wide-ranging, from desired configurations in quorum systems [11], [1], [3], [12] to performance optimization methods that minimize the communication delay [6], [15], [10]. On the other hand, little attention has been paid to reducing power consumption in quorum systems, which is necessary in terms of green computing that has recently become of prime interest to practitioners.

In this paper, we present the *dynamic grid quorum*, a technique for reducing storage power consumption. As its name implies, our approach is based on a grid quorum [3], but the key idea of our approach is to skew the workload towards a small number of quorums. This can be realized by our optimization algorithm for strategy (probability distribution over the quorums, determining which quorum is selected for each read/write request). Moreover, the dynamic grid quorum provides reconfiguration by exchanging nodes without any data migration. Although the node exchange idea is used from the study on tree-based quorums [5], much of the work had to be adapted for grid quorums. A major difference between grid and tree quorums is that nodes are naturally exchangeable in tree-based quorums, whereas this is not the case in grids. Since we also consider heterogeneity in nodes, this reconfiguration method allows high-capacity nodes to be allocated to busier quorums so as to reduce the number of active nodes. Thus, our dynamic grid quorum can adapt continuously to any environment, from read-intensive to write-intensive workloads.

To evaluate the effectiveness of the proposed dynamic grid quorum, we compared our approach with two static read-optimized and write-optimized grid configurations. From the simulations, we observed that our dynamic grid quorum saved, on average, 6-12% energy compared with the static configurations when the intensity of the total workload was changed.

Related work. There have been a number of attempts to reduce storage power consumption. A commonly-observed feature in many of these techniques is that they adopt the approach of skewing the load, which is also used in this paper. In MAID [4] and PDC [13], popular data are concentrated on specific disks. In DIV [14], original and redundant data are separated onto different disks, thereby allowing read/write requests to be concentrated on the disks with original data. In RIMAC [18], eRAID [16], and

EERAID [9], a data access on a disk in standby mode is transformed into accesses on active disks or caches, and then the required data are reconstructed from the parities obtained during these accesses. In Hibernator [19] and PARAID [17], data are collected or spread to adapt to changes in operational loads.

Paper organization. Sections 2 and 3 introduce the dynamic grid quorum and its power consumption model, respectively. Section 4 gives the definition of our reconfiguration. Section 5 introduces the algorithm and proves that it minimizes the power consumption of the dynamic grid quorum. Section 6 presents the simulation results. Finally, Section 7 concludes the paper and discusses future research.

2. System Description

Our proposed system is based on the grid protocol [3], which is a special kind of write-read coterie [8], in which the nodes are allocated in a rectangular grid. A write-read coterie $\mathcal{C}$ is a pair $\langle \mathcal{C}_W, \mathcal{C}_R \rangle$ of collections of node groups, called *quorums*, where $\mathcal{C}_W$ and $\mathcal{C}_R$ represent *write quorums* and *read quorums*, respectively. Throughout this paper, we fix the underlying set of nodes $U = \{u_{i,j} \mid 1 \leq i \leq n \wedge 1 \leq j \leq m\}$, where each node is allocated in a grid with n columns and m rows. For readability, we use the following notations. $u_{i,j}$ is used to denote a node at the intersection of the i-th column (from the left) with the j-th row (from the bottom). This point is also referred to using the notation (i, j). Col_i and Row_j are respectively used to denote the sets of nodes in the i-th column (i.e., $Col_i = \{u_{i,j} \mid 1 \leq j \leq m\}$) and in the j-th row (i.e., $Row_j = \{u_{i,j} \mid 1 \leq i \leq n\}$).

The dynamic grid quorum is defined as follows.

Definition 1 (Dynamic grid quorum): Let $\mathcal{C}^f = \langle \mathcal{C}^f_W, \mathcal{C}^f_R \rangle$ be a pair of collections of node groups. $\mathcal{C}^f$ is a dynamic grid quorum (for some $f = 0, 1, \ldots, \min\{m, n\}$) if

- $\mathcal{C}^f_W$ is the set of all possible node groups, each of which (say, Q^f_W) satisfies the following condition.
  $Q^f_W = Col_k \cup \{u_{i,j} \mid \forall i \; \exists! j (f < i < k \wedge 1 \leq j \leq m)\} \cup S_k$
  for some k $(1 \leq k \leq n)$, where
  $$S_k = \begin{cases} \{u_{i,k} \mid 1 \leq i \leq \min\{k-1, f\}\} & 1 \leq k \leq m \\ \{u_{i,j} \mid \forall i \; \exists! j (1 \leq j \leq i \leq f)\} & \text{otherwise} \end{cases}$$
- $\mathcal{C}^f_R$ is the set of all possible node groups, each of which (say, Q^f_R) satisfies the following condition.
  $Q^f_R = \{u_{i,j} \mid \forall i \; \exists! j (1 \leq i \leq n \wedge 1 \leq j \leq m)\}$
  such that if $u_{i,j} \in Q^f_R$ with $i \leq f$ and $i \leq j \leq \min\{n, m\}$ then $u_{j,i} \in Q^f_R$.

In this definition, the number f is called the *degree of flexibility* (or the *degree*, for short). We may omit this if it is clear from the context.

We also use the following notations. The set $\{u_{i,i} \mid 1 \leq i \leq \min\{n, m\}\}$ is called the *diagonal line*. Node $u_{j,i}$ is called the *diagonal point* of $u_{i,j}$. For a given degree f, the set $\{u_{i,j} \mid 1 \leq i \leq f \wedge i < j \wedge j \leq \min\{n, m\}\}$ is called the *left-side exchangeable area*, while the set $\{u_{i,j} \mid 1 \leq j \leq f \wedge i > j \wedge i \leq \min\{n, m\}\}$ is called the *lower exchangeable area*. Clearly, these sets are bijective by the mapping of diagonal points.

Note that each of the write (read, resp.) quorums satisfies the condition that if the quorum includes a node in the lower (left-side, resp.) exchangeable area, then it also includes the diagonal point. (Note, however, that the converse does not always hold.)

To help the readers understand this definition, we present an example of write and read quorums in a dynamic grid quorum with degree $f = 2$. In this example, nodes in the write quorum are denoted by horizontal striped patterns. This write quorum consists of three sets of nodes: the set of all nodes in the fifth column (i.e., Col_5), nodes chosen singly from the i-th column for each $i = 3, 4$ (i.e., $\{u_{3,3}, u_{4,1}\}$), and the set of nodes at the intersection of the fifth row and the i-th column for each $i = 1, 2$ (i.e., $\{u_{1,5}, u_{2,5}\}$). On the other hand, nodes in the read quorum are denoted by gray shading. This read quorum consists of nodes chosen singly from each column (i.e., $\{u_{1,3}, u_{2,4}, u_{3,1}, u_{4,2}, u_{5,4}, u_{6,3}\}$), where $u_{3,1}$ and $u_{4,2}$ must be included because $u_{1,3}$ and $u_{2,4}$ are nodes in the left-side exchangeable area.

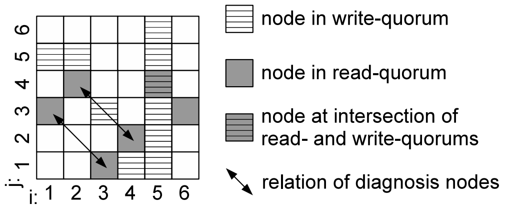
j:
1
2
3
4
5
6
i: 1
2
3
4
5
6
node in write-quorum
node in read-quorum
node at intersection of read- and write-quorums
relation of diagnosis nodes

Fig. 1. Example of quorums on dynamic grid quorum $(f=2)$

3. Power Consumption Model

We introduce a power consumption model that includes the following: *strategy*, *load*, *capacity* of nodes, *states* of nodes, and *power consumption*.

Strategy. Each write (read, resp.) request is assigned to a write (read, resp.) quorum by the probability distribution $P = \langle P_W, P_R \rangle$ over $\langle \mathcal{C}_W, \mathcal{C}_R \rangle$, satisfying the following properties:

- $0 \leq P_W(Q_W) \leq 1$ for any $Q_W \in \mathcal{C}_W$
- $0 \leq P_R(Q_R) \leq 1$ for any $Q_R \in \mathcal{C}_R$

- $\sum_{Q_W \in \mathcal{C}_W} P_R(Q_W) = 1$
- $\sum_{Q_R \in \mathcal{C}_R} P_R(Q_R) = 1$

We call this probability distribution the *strategy*.

Load. We define the load of a system, $ld = \langle ld_{WS}, ld_{RS} \rangle$, as the number of write/read requests per unit time. Using functions $load_W$ and $load_R$, the loads of the write and read quorums (say, Q_W and Q_R) are respectively defined as follows.

- $load_W(Q_W) = ld_{WS} \cdot P_W(Q_W)$ (1)
- $load_R(Q_R) = ld_{RS} \cdot P_R(Q_R)$ (2)

Additionally, we extend these functions to represent the *load of a node* as follows.

- $load_W(u) = \sum_{Q_W \in \mathcal{C}_W, Q_W \ni u} load_W(Q_W)$ (3)
- $load_R(u) = \sum_{Q_R \in \mathcal{C}_R, Q_R \ni u} load_R(Q_R)$ (4)
- $load(u) = load_R(u) + load_W(u)$

Capacity. We assume that each node $u \in U$ has a capacity $cap(u)$, which represents the upper bound of $load(u)$ defined by the following condition.

$$load(u) \leq cap(u) \qquad \text{for any } u \in U$$

States of nodes. We define a Boolean function On over the set of nodes, such that if $load(u) \neq 0$, then $On(u) = 1$. Intuitively, we model the state of nodes by this function: if $On(u) = 1$ then node u is *active*, whereas if $On(u) = 0$ then u is on *standby* and consumes no power. Note that $On(u)$ may be 1 even if $load(u) = 0$. This means that node u is active even though it is not executing any operations.

Power consumption. We assume that every node in an active state consumes the same power in the active state. Thus we define the power consumption of a system as the number of active nodes.

4. Reconfiguration

In order to reduce the power consumption in an environment where the load varies, our system enables reconfiguration by exchanging a node with the corresponding diagonal point.

Definition 2 (Reconfiguration): The dynamic grid quorum $\mathcal{C}'$ is the *reconfigured system* obtained from $\mathcal{C}$ by exchanging node $u_{i,j}$ in the exchangeable areas, if $\mathcal{C}' = \{\pi(Q, u_{i,j}) \mid Q \in \mathcal{C}\}$, where the function π is defined as follows.

$$\pi(Q, u_{i,j}) = \begin{cases} (Q \cup \{u_{j,i}\}) - \{u_{i,j}\} & (u_{i,j} \in Q \wedge u_{j,i} \notin Q) \\ (Q \cup \{u_{i,j}\}) - \{u_{j,i}\} & (u_{i,j} \notin Q \wedge u_{j,i} \in Q) \\ Q & \text{otherwise} \end{cases}$$

Note that any system obtained through this reconfiguration is also a dynamic grid quorum. Also note that this reconfiguration is done without any data migration.

An example of this reconfiguration is presented below. Fig. 2 shows the capacity of all nodes in the 4×4 dynamic grid quorums $\mathcal{C}_1^1$ and $\mathcal{C}_2^1$, where $\mathcal{C}_2^1$ is obtained by exchanging the following pairs of nodes in $\mathcal{C}_1^1$: $\langle u_{1,2}, u_{2,1} \rangle$, $\langle u_{1,3}, u_{3,1} \rangle$, and $\langle u_{1,4}, u_{4,1} \rangle$ (within the heavy-line frames). Here we consider the case that $ld_{WS} = 1$ and $ld_{RS} = 3$. In this instance, as the upper two systems in Fig. 2 indicate, the minimum number of active nodes in $\mathcal{C}_1$ and $\mathcal{C}_2$ are 9 and 7, respectively. On the other hand, where $ld_{WS} = 3$ and $ld_{RS} = 1$, as the lower two systems indicate, the minimum number of active nodes are 7 and 10, respectively. This shows that our reconfiguration may remap nodes with heterogeneous capacity, thereby reducing the power consumption of the system when the ratio of write/read requests varies.

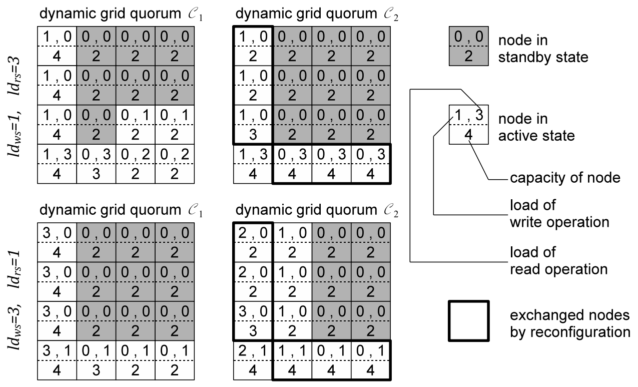

Fig. 2. Example of dynamic grid quorum $(f = 1)$

Although such reconfiguration is indeed useful for reducing the power, it may violate the consistency of the replicated data. Our reconfiguration, however, preserves the property of consistency called *one-copy serializability* (cf. [2]), which guarantees that any read-operation returns the data installed by the last committed write operation. More precisely, the following proposition holds.

Proposition 1: Let $\mathcal{C}' = \langle \mathcal{C}'_W, \mathcal{C}'_R \rangle$ be a dynamic grid quorum obtained from $\mathcal{C} = \langle \mathcal{C}_W, \mathcal{C}_R \rangle$ by any number of reconfigurations. For any $Q_W \in \mathcal{C}_W$, $Q'_W \in \mathcal{C}'_W$, and $Q'_R \in \mathcal{C}'_R$, the following properties hold: $Q_W \cap Q'_R \neq \phi$ and $Q_W \cap Q'_W \neq \phi$.

Proof: Due to space limitations, we only show the first property. In this proof we use the notation $u'_{i,j}$ to indicate the node at (i,j) in the grid of $\mathcal{C}'$. From the definition of a dynamic grid quorum, there exist i and j such that $Col_i \subseteq Q_W$ and $u'_{i,j} \in Q'_R$. Here we consider the following cases: (1) $u'_{i,j}$ is in the lower exchangeable area; (2) $u'_{i,j}$ is in the left-side exchangeable area; (3) otherwise.
Case (1): By the definition of reconfiguration, if $u'_{i,j}$ is exchanged then $u'_{i,j} = u_{j,i}$, otherwise $u'_{i,j} = u_{i,j}$. By the definition of a dynamic grid quorum, $u_{i,j}, u_{j,i} \in Q_W$, because Q_W has the corresponding diagonal node if Q_W has a node in the lower exchangeable area. Thus $Q_W \cap Q'_R \ni u'_{i,j}$.
Case (2): By the definition of reconfiguration, if $u'_{i,j}$ is exchanged then $u'_{j,i} = u_{i,j}$, otherwise $u'_{i,j} = u_{i,j}$. By the definition of a dynamic grid quorum, $u'_{i,j}, u'_{j,i} \in Q'_R$, because Q'_R has the corresponding diagonal node if Q'_R has a node in the left-side exchangeable area. On the other hand, $u_{i,j} \in Q_W$. Thus $Q_W \cap Q'_R \ni u_{i,j}$.
Case (3): Because $u'_{i,j}$ is never exchanged, $u_{i,j} = u'_{i,j}$. Thus $Q_W \cap Q'_R \ni u_{i,j}$. ■

5. Optimization Algorithm

Our idea of power optimization is to skew the load towards a small number of quorums. This is realized by our optimization algorithm which shall be introduced below. To make our discussion simpler, throughout this section we consider the following restrictions on the states of nodes.

(C1) For any i $(1 < i \leq n)$ and j $(1 \leq j \leq m)$, if $load_W(u_{i,j}) \neq 0$ then $On(u_{k,l}) = 1$ for all $1 \leq k < i$ and $1 \leq l \leq m$.

(C2) For any i $(1 \leq i \leq n)$ and j $(1 < j \leq m)$, if $load_R(u_{i,j}) \neq 0$ then $On(u_{i,l}) = 1$ for all $1 \leq l < j$.

Under these restrictions, we consider the following problem.

Problem 1: For a given dynamic grid quorum $\mathcal{C}$, a capacity cap, and a system load ld of $\mathcal{C}$, find a strategy P that minimizes the power consumption of $\mathcal{C}$.

In order to solve this problem, generally, the number of quorums increases exponentially with the number of nodes. This results in an explosion of the search space for the optimization. On the other hand, the power consumption of the system is determined only by the total load of each node. From this observation, the key idea in optimizing the strategy is to first divide each quorum into its components and to consider only the loads thereof. Then we limit the search space to a subset created from a special type of load, denoted by $load'$, from which we obtain our aimed optimal strategy.

In terms of $load'$, we find an optimal solution to Problem 1 by the following procedure: we first find an optimal $load'$ then construct an optimal strategy from this optimal $load'$. Here, P^{opt} is used to denote a strategy obtained by this procedure. Then the following theorem holds.

Theorem 1: P^{opt} is an optimal solution to Problem 1.

The outline of the proof of Theorem 1 is as follows. First, we define $load'$. Next, we prove Lemmas 1 and 2, which verify that an optimal strategy can be obtained from an optimal $load'$. Finally, we prove Lemma 3, which confirms that we can find an optimal $load'$ using our algorithm.

Definition 3 (Special type of load): For a given system $\mathcal{C}^f$ and its load ld, we define $load'$ consisting of the following four functions.

- $load'_{WN} : \{u_{i,j} \mid 1 \leq i < n \wedge 1 \leq j \leq m\} \to \mathbb{R}^+$
- $load'_{WC} : \{Col_i \mid 1 \leq i \leq n\} \to \mathbb{R}^+$
- $load'_{RN} : \{u_{i,j} \mid 1 \leq i \leq n \wedge 1 \leq j \leq i\} \to \mathbb{R}^+$
- $load'_{RT} : T \to \mathbb{R}^+$

where T is the set of $\vec{t} = \langle t_1, \ldots, t_f \rangle$ defined as follows.

- $t_i \in \{0, i+1, i+2, \ldots, m\}$
- If $t_i = j$ then $t_j = 0$ for any i.
- If $t_i = k$ and $k \neq 0$ then $t_j \neq k$ for any $i \neq j$.

Here $load'$ satisfies conditions (5) to (10) given below.

- $$\sum_{j=1}^{m} load'_{WN}(u_{i,j}) = \sum_{k=i+1}^{n} load'_{WC}(Col_k) \quad \text{for any } i\,(f < i < n) \qquad (5)$$
- $$\sum_{j=1}^{i} load'_{WN}(u_{i,j}) = \sum_{k=m+1}^{n} load'_{WC}(Col_k) \quad \text{for any } i\,(1 \leq i \leq f) \qquad (6)$$
- $$load'_{WN}(u_{i,j}) = load'_{WC}(Col_j) \quad \text{for any } u_{i,j} \text{ in the left-side exchangeable area} \qquad (7)$$
- $$\sum_{k=1}^{n} load'_{WC}(Col_k) = ld_{WS} \qquad (8)$$
- $$\sum_{\vec{t} \in T, t_i = j} load'_{RT}(\vec{t}) \leq load'_{RN}(u_{j,i}) \quad \text{for any } u_{i,j} \text{ in the left-side exchangeable area} \qquad (9)$$
- $$\sum_{j=1}^{i} load'_{RN}(u_{i,j}) + \sum_{j=i+1}^{m} \sum_{\vec{t} \in T, t_i = j} load'_{RT}(\vec{t}) = ld_{RS} \quad \text{for any } i\,(1 \leq i \leq f) \qquad (10)$$

The intuitive meaning of $load'$ is as follows. A write quorum can be divided into a single column and the other part. $load'_{WC}(Col)$ represents the total write load of column Col, while $load'_{WN}(u)$ represents the total write load of node u in the other part. On the other hand, $load'_{RT}(\vec{t})$ represents the total read load of that part of the read quorum, included in the left-side exchangeable area. Here $\vec{t}$ represents this part. More precisely, $t_i \in \vec{t}$ denotes that the part includes node $u_{i,j}$ with $j = t_i$. $load'_{RN}(u)$ represents the total read load, where u is not in the left-side exchangeable area.

Using the notion of $load'$, instead of $load_W$ and $load_R$ (introduced by Eqs. 3 and 4), we can also define $load'_W$ and $load'_R$, the total write and read loads, respectively, of each node, as given below.

- $load'_W(u_{i,j}) = load'_{WN}(u_{i,j}) + load'_{WC}(Col_i)$ (11)

- $load'_R(u_{i,j}) =$
$$\begin{cases} \sum_{\vec{t}\in T, t_i=j} load'_{RT}(\vec{t}) & u_{i,j} \text{ is in the left-side exchangeable area} \\ load'_{RN}(u_{i,j}) & \text{otherwise} \end{cases} \quad (12)$$

Moreover, in terms of this definition, we can naturally define the power consumption, similarly to $load_W$ and $load_R$. Then, we obtain the following lemma.

Lemma 1: For any system $\mathcal{C}^f$, any system load ld of $\mathcal{C}^f$, and any $load'$, we can obtain a strategy P whose power consumption is the same as $load'$.

Proof: It is sufficient to show that $load_W(u) = load'_W(u)$ and $load_R(u) = load'_R(u)$ for any u. We define P_W and P_R as (13) and (14), respectively.

- $$P_W(Q_W) = \frac{load'_{WC}(Col_k)}{ld_{WS}} \cdot \prod_{u_{i,j}\in Q_W, f<i<k} \frac{load'_{WN}(u_{i,j})}{\sum_{l=1}^{m} load'_{WN}(u_{i,l})} \cdot \prod_{u_{i,j}\in Q_W, i\le f} g_1(u_{i,j}, k)$$
where k is a number such that $Col_k \subseteq Q_W$ (13)

- $$P_R(Q_R) = load'_{RT}(\vec{t}) \cdot \prod_{u_{i,j}\in Q_R} \frac{g_2(u_{i,j})}{g_3(Q_R, u_{i,j})}$$
where $\vec{t}$ satisfies $t_i{=}j{\neq}0$ iff $u_{i,j}{\in}Q_R$ and $i{<}j$ (14)

Here g_1, g_2, and g_3 are functions defined as follows.

- $$g_1(u_{i,j}, k) = \begin{cases} 1 & k \le m \\ \frac{load'_{WN}(u_{i,j})}{\sum_{l=1}^{i} load'_{WN}(u_{i,l})} & \text{otherwise} \end{cases}$$

- $$g_2(u_{i,j}) = load'_{RN}(u_{i,j}) - \sum_{\vec{t}\in T, t_j=i} load'_{RT}(\vec{t})$$

- $g_3(Q_R, u_{i,j}) =$
$$\begin{cases} \sum_{l=1}^{i} load'_{RN}(u_{i,l}) - \sum_{k=1}^{f} \sum_{\vec{t}\in T, t_k=i} load'_{RT}(\vec{t}) & \text{(Case1)} \\ ld_{RS} - \sum_{k=1}^{f} \sum_{\vec{t}\in T, t_k=i} load'_{RT}(\vec{t}) & \text{(Case2)} \\ load'_{RN}(u_{i,j}) - \sum_{\vec{t}\in T, t_j=i} load'_{RT}(\vec{t}) & \text{(Case3)} \end{cases}$$

where Case 1 covers $i \le f$, $u_{j,i} \notin Q_R$, and $u_{i,j}$ in the lower exchangeable area; Case 2 deals with $i > f$ and, if $u_{j,i} \in Q_R$ then $j > f$; and Case 3 covers all other possibilities.

Then by the definitions of $\langle load_W, load_R\rangle$ (i.e., (1)-(4)), $load'$ (i.e., (5)-(10)), and $\langle load'_W, load'_R\rangle$ (i.e., (11)-(12)), we can derive our required equations. ■

Also, the following lemma, which is the converse of Lemma 1, is provable.

Lemma 2: For any system $\mathcal{C}^f$, any system load ld of $\mathcal{C}^f$, and any strategy P, we can obtain a $load'$ whose power consumption is the same as P.

Proof: We define $load'_{WN}$, $load'_{WC}$, $load'_{RN}$, and $load'_{RT}$ as (15), (16), (17), and (18), respectively.

- $$load'_{WN}(u_{i,j}) = \sum_{Q_W\in\mathcal{C}_W, Q_W\ni u_{i,j}, Q_W\not\supseteq Col_i} load_W(Q_W) \quad (15)$$

- $$load'_{WC}(Col_k) = \sum_{Q_W\in\mathcal{C}_W, Q_W\supseteq Col_k} load_W(Q_W) \quad (16)$$

- $$load'_{RN}(u_{i,j}) = \sum_{Q_R\in\mathcal{C}_R, Q_R\ni u_{i,j}} load_R(Q_R)$$
for any $u_{i,j} \in U$ such that $u_{i,j}$ is *not* in the left-side exchangeable area (17)

- $$load'_{RT}(\vec{t}) = \sum_{Q_R\in\mathcal{T}(\vec{t})} load_R(Q_R) \quad (18)$$

Here $\mathcal{T}(\vec{t})$ in (18) is a class of read quorums such that for any $Q_R \in \mathcal{T}(\vec{t})$, if $t_i = j$ and $j \neq 0$, then $u_{i,j} \in Q_R$ for any i and j, and $u_{i,j}$ is in the left-side exchangeable area.

From Eqs. (15) to (18) and Eqs. (1) to (4) we can derive (5) to (10). This guarantees that $load'$ defined by (15) to (18) satisfies conditions (5) to (10). Moreover, in addition to the above derivation, by (11) and (12) we can obtain our required equations. ■

The algorithm (called *OL*) to obtain an optimal $load'$, is presented in Fig. 3. The algorithm can be divided into three steps. Step 1, comprising lines 1 to 7 in Fig. 3, minimizes the number of active nodes for write requests (called the *write-area*) so as to satisfy (5) to (10). At the end of Step 1, a minimum V for the write-area has been obtained. Step 2, comprising lines 8 to 14, minimizes the number of active nodes for read requests (called the *read-area*) so as to satisfy (5) to (7). At the end of Step 2, V is the union of the minimum write-area and minimum read-area. Because Step 2 disregards (10), some active nodes might be added to satisfy (10). We call this set of nodes the *additional-area*. Step 3, comprising the rest of this algorithm, minimizes the additional-area and then returns an optimal solution.

Next we explain the details of sub-algorithms LP1 and LP2 used in Steps 1 and 3, respectively. LP1 consists of linear programming, which returns true if and only if there exists a basic feasible solution of $load'$ to the constraints represented by the following conditions.

- (5) to (10).
- The condition of capacity.
- $load'_{WC}(Col_i) = 0$ for all $i (K < i \leq n)$.

In Step 1, if there is no feasible area even with $K = n$, then there exists no solution. In this case, the algorithm terminates without finding a solution.

LP2 also consists of linear programming, which returns one of the basic feasible solutions of $load'$, if it exists, to the constraints represented by the following conditions; otherwise it returns nil.

- (5) to (10).
- The condition of capacity.
- $load'_W(u_{i,j}) + load'_R(u_{i,j}) = 0$ for $u_{i,j} \notin V$ and $k < i$.

Now we show that the $load'$ obtained by this algorithm is optimal.

Lemma 3: For any dynamic grid quorum $\mathcal{C}^f$, any system load ld of $\mathcal{C}^f$, and any capacity cap, Algorithm OL returns an optimal solution $load'$, which minimizes the power consumption.

Proof sketch: Let S be the set of active nodes determined by the solution obtained by this algorithm. Here we consider S_W, S_R, and S_A, which are the write-, read-, and additional-areas of S, respectively. Thus, to prove this lemma it is sufficient to show that each of S_W, S_R, and S_A is minimum.

Clearly, for S_W and S_R, these are the minimum sets to handle the given load ld_{WS} and ld_{RS}, respectively. That is because if this were not the case, Step 1 or the inner loop of Step 2 controlled by I would terminate prematurely.

To find S_A, Step 3 repeatedly generates a set (say, $\mathcal{C}_A^{s,t}$) of candidates, by the following operations OP1 and OP2 so as to enumerate all possible candidates.

(OP1) Generate $\mathcal{C}_A^{s,t+1}$ from $\mathcal{C}_A^{s,t}$ by adding $u_{i,j}$ to each candidate $S'_A \in \mathcal{C}_A^{s,t}$, where $u_{i,j}$ satisfies $i \leq f$, $u_{i,j} \notin S'_A \cup S_R$, and $u_{i,l} \in S'_A \cup S_R$ for all $l < j$.

```
input:$U, n, m, cap, ld$

01: {Step 1 starts here.}
02: $K \leftarrow 0$ ;
03: **while** LP1 returns **false**
04:    **if** $K = n$ **then**
05:       **exit** "NO SOLUTION" ;
06:    $K \leftarrow K + 1$ ;
07: $V \leftarrow \bigcup_{l=1}^{K} Col_l$ ;
08: {Step 2 starts here.}
09: **if** $ld_{RS} \neq 0$ **then**
10:    **for** $I = 1$ to $n$
11:       **for** $J = 1$ to $m$
12:          $V \leftarrow V \cup \{u_{I,J}\}$ ;
13:          **if** $\sum_{u \in V, u \in Col_I} cap(u) \geq ld_{RS}$ **then**
14:             **break** ;
15: {Step 3 starts here.}
16: $\mathcal{V} \leftarrow \{V\}$, $V'' \leftarrow U$, $P \leftarrow nil$, $L \leftarrow K$ ;
17: **loop**
18:    **for all** $V' \in \mathcal{V}$
19:       **if** LP2 returns some $load'$ **then**
20:          **if** $|V'| < |V''|$ **then**
21:             $V'' \leftarrow V'$, $P \leftarrow load'$ obtained by LP2 ;
22:          **if** $|V'' - V| \leq \sum_{i=k}^{L+1} (m - |V \cap Col_i|)$ **then**
23:             **return** $P$ ;
24:          **else**
25:             $\mathcal{V} \leftarrow \{V \cup Col_{L+1}\}$, $L \leftarrow L + 1$ ;
26:             **next loop** ;
27:    $\mathcal{W} \leftarrow \phi$ ;
28:    **for all** $V' \in \mathcal{V}$
29:       **for** $I = L + 1$ to $n$
30:          $T \leftarrow \max\{j \mid u_{I,j} \in V'\}$ ;
31:          **if** $T < f$ **then**
32:             $\mathcal{W} \leftarrow \{V' \cup \{u_{I,T}\}\} \cup \mathcal{W}$ ;
33:    $\mathcal{V} \leftarrow \mathcal{W}$ ;
34:    **if** $\mathcal{V} = \phi$ **then**
35:       $\mathcal{V} \leftarrow \{V \cup Col_{L+1}\}$, $L \leftarrow L + 1$ ;
```

Fig. 3. Algorithm OL

(OP2) Generate $\mathcal{C}_A^{s+1,0}$ from $\mathcal{C}_A^{s,0}$ by adding $Col'_{s+1} \subseteq Col_{s+1}$ to each candidate $S'_A \in \mathcal{C}_A^{s,0}$ such that $u \in Col'_{s+1}$ if and only if $u \notin S_R$.

Let $\vec{\mathcal{C}}_A^0, \vec{\mathcal{C}}_A^1, \ldots, \vec{\mathcal{C}}_A^h$ be the sequence of sets of candidates generated in Step3, where for each $0 \leq s \leq h$, $\vec{\mathcal{C}}_A^s = \mathcal{C}_A^{s,0}, \mathcal{C}_A^{s,1}, \ldots, \mathcal{C}_A^{s,h_s}$ for some h_s. Here, for any s and t, each candidate in $\mathcal{C}_A^{s,t}$ satisfies (C1) and (C2). In this sequence, there exists $\mathcal{C}_A^{s,t} \ni S_A$ for some s and t, where S_A is the minimum among all the candidates for each of which LP2 returns a solution. Here, there is no smaller candidate than S_A in $\mathcal{C}_A^{s',t'}$ nor $\mathcal{C}_A^{s'',t''}$ for any s', s'', t', and t'' such that $s' > h$ and $s'' \leq h$ and $t'' > h_{s''}$. Therefore, S_A is a minimum additional-area. ■

Proof of Theorem 1: Let $load'$ be the solution obtained by Algorithm OL. Then, by Lemma 3, $load'$ is an optimal solution in the set of special type of loads. Thus, by Lemma 1, we can obtain a strategy P^{opt} whose power consumption is the same as $load'$. Moreover, P^{opt} is an optimal strategy, because if there exists another strategy (say, P') whose power consumption is smaller than that of P^{opt}, then by Lemma 2, a better solution also exists in the set of special type of loads. This contradicts Lemma 3. ■

Remark: restrictions on the state of nodes. If we disregard either of the restrictions (C1) or (C2), there may be a better solution than that obtained by our algorithm.

For example, let S_R be the read-area, k the number of columns in the write-area, and $load'$ a special type of load obtained by the algorithm. If there is some set of nodes $S'_R \subseteq S_R$ that satisfies $\sum_{u \in (S'_R \cap Col_i)} cap(u) \geq ld_{RS}$ for any Col_i, and $\sum_{\vec{t} \in T, t_i = j} load'_{RT}(\vec{t}) = 0$ for any $u_{i,j}$ such that $1 \leq i \leq k < j \leq \min\{n, m\}$ and $u_{j,i} \notin V_R$, then the nodes $S_R - S'_R$ can be reduced if we disregard assumption (C2).

Remark: computational complexity. By using the notion of $load'$, the computational complexity of our proposed algorithm is decreased. Indeed, as far as our experiments are concerned, we can find optimal solutions in feasible time for all cases of the simulations in the next section. However, the following two factors may still cause a search space explosion: the number of loops in Step 3 (even though this step is not used in many cases) and the number of constraints for $load'_{RT}$, where the computational complexities are $O(f^n)$ and $O(m^f)$, respectively. A possible way to avoid such exponential growth is to restrict the degree f to a small number, but an improvement to our algorithm is still needed. This shall be investigated as one of our future works.

6. Simulation Results

In the evaluation of this section, we consider the following three systems for a 10×10 grid: dynamic grid quorum with reconfiguration (DG), and fixed allocation of nodes optimized for write requests (WG) and read requests (RG), respectively. First we evaluate the impact of the write/read ratio for each case where the system load is low, medium or high. In contrast to the above settings, we also evaluate the impact of system load, where the write/read ratio is fixed but the total number of requests varies.

Parameters and Settings. Each system in the evaluation consists of 70 nodes with capacity c and 30 nodes with capacity $2c$. In the WG and RG, the nodes with $2c$ are allocated in Col_i and Row_i ($1 \leq i \leq 3$), respectively. The DG is a dynamic grid quorum with degree $f = 3$, whose node allocation is the same as in the WG. For the impact analysis of the write/read ratio, we use $2.5c$, $5c$, and $7.5c$ as the values for low, medium, and high loads in the system, respectively. This setting arises from the fact that the write and read capacity for a grid of the same size, comprising homogeneous nodes with capacity c is $10c$ and $6.5c$, respectively. For the impact analysis of the system load, we consider the write/read requests to be $1/3$, which is also used as the default parameter in [14].

Impact of write/read ratio in the case of low load. Fig. 4 shows that the power consumption of the DG is less than that of both the WG and RG for all ratios. Additionally, compared with the WG and RG, our system saves on average 12% and 13%, respectively, and up to 27% and 32% of power, respectively, for all ratios. The results of this simulation show that our system effectively reduces power consumption when compared with both the WG and RG.

Impact of write/read ratio in the case of medium load. Fig. 5 shows that the power consumption of the DG is the least for the range 0% to 70% of the write ratio. Above 70% of the write ratio, our system consumes more power than the WG. The DG, however, does not exceed the power consumption of the RG for all ratios. Considering more detailed results, the DG consumes on average 0.2% more power than the WG and 10% less than the RG for all ratios. Moreover, compared with the WG and RG, our system saves up to 28% and 24% of power, respectively, although it consumes twice as much as the WG in the worst case. The results of this simulation show that our system is less effective for all write ratios with medium load than with low load. However, our system is still efficient for a wide range of write ratios.

Impact of write/read ratio in the case of high load. Where $ld_{WS} + ld_{RS} = 7.5c$, the DG cannot handle a write ratio above 70%, whereas the WG and RG can. However, compared with the WG and RG, our system saves on average 1% and 6%, respectively, for a write ratio in the range 0% to 70%. The results of this simulation show that our system may not be able to handle a high write ratio with high load. However, our system is still able to save some power for other write ratios.

Impact of load growth. Fig. 6 shows that the power consumption of the DG is the least of all the systems for all system loads. Compared with the WG and RG, our system saves on average 12% and 6%, and up to 27% and 22% of power, respectively, for all ratios. The results of this simulation show that our system is effective for any system load.

7. Conclusions and Future Work

In this paper, we presented a power-aware quorum system called the dynamic grid quorum. To reduce power consumption, we introduced an optimization algorithm for skewing the workload towards a small number of quorums. Moreover, we proposed a reconfiguration technique that allows the exchanging of nodes without data migration. We also evaluated our technique by comparing the proposed system with static quorums. The results showed that our system saved on average between 6 and 12% with a $1/3$ write operation ratio.

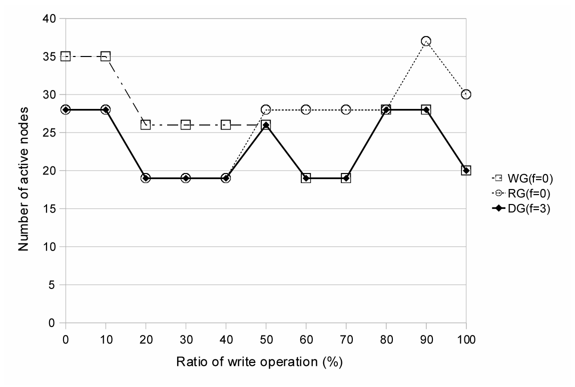

Fig. 4. Number of active nodes$(ld_{WS}+ld_{RS}=2.5c)$

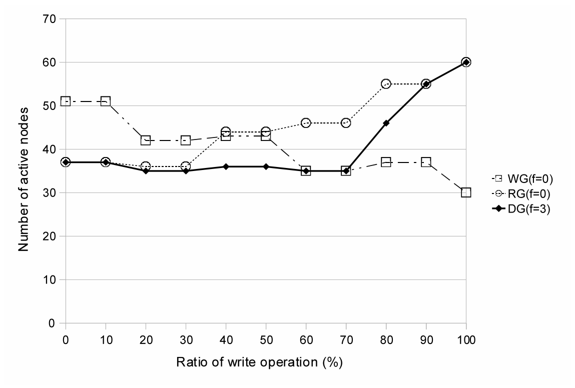

Fig. 5. Number of active nodes $(ld_{WS}+ld_{RS}=5c)$

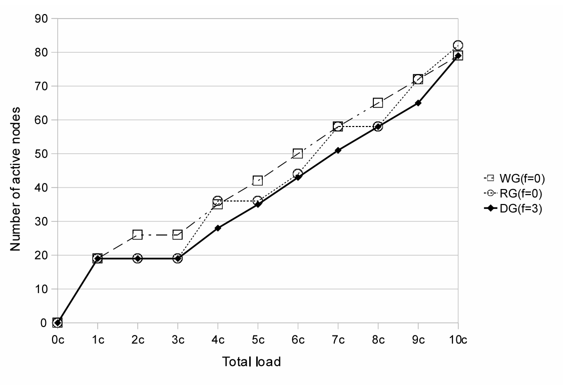

Fig. 6. Number of active nodes $(ld_{WS}:ld_{RS}=1:3)$

There are several possible directions in which this work can be developed. As mentioned in Section 5, we are interested in an improvement to our algorithm to avoid the case that results in a search space explosion. We are also interested in some extensions of our technique, especially by modeling heterogeneity of power consumption and time required for the state transition of nodes, that were not considered in this paper.

Acknowledgments

This study was supported in part by the Global COE Program on "Cybernics: fusion of human, machine, and information systems."

References

[1] D. Agrawal and A.-E. Abbadi. An efficient and fault-tolerant solution for distributed mutual exclusion. *ACM Trans. on Computer Systems*, 9(1):1–20, 1991.

[2] P.-A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison Wesley, 1987.

[3] S. Cheung, M. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. *IEEE Trans. on Knowledge and Data Engineering*, 4(6):582–592, 1992.

[4] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. *Proc. of the ACM/IEEE Conf. on Supercomputing*, pages 1–11, 2002.

[5] I. Frain, R. Basmadjian, J.-P. Bahsoun, and A. M'zoughi. How to improve the scalability of read/write operations with dynamic reconfiguration of a tree-structured coterie. *Proc. of the Int'l. Conf. on Parallel Processing Workshops*, pages 123–134, 2006.

[6] A. Fu. Delay-optimal quorum consensus for distributed systems. *IEEE Trans. on Parallel and Distributed Systems*, 8(1):59–69, 1997.

[7] D. Gifford. Weighted voting for replicated data. *Proc. of the ACM Symposium on Operating Systems Principles*, pages 150–162, 1979.

[8] T. Ibaraki and T. Kameda. A theory of coteries: Mutual exclusion in distributed systems. *IEEE Trans. on Parallel and Distributed Systems*, 4(7):779–794, 1993.

[9] D. Li and J. Wang. EERAID: energy efficient redundant and inexpensive disk array. *Proc. of the ACM SIGOPS European Workshop*, 6 pages, 2004.

[10] X. Lin. Delay optimization in quorum consensus. *Algorithmica*, 38(2):397–413, 2004.

[11] M. Maekawa. A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. on Computer Systems*, 3(2):145–159, 1985.

[12] D. Peleg and A. Wool. Crumbling walls: A class of practical and efficient quorum systems. *Distributed Computing*, 10(2):87–97, 1997.

[13] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. *Proc. of the Int'l. Conf. on Supercomputing*, pages 68–78, 2004.

[14] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. *Proc. of the joint Int'l Conf. on Measurement and Modeling of Computer Systems*, pages 15–26, 2006.

[15] T. Tsuchiya, M. Yamaguchi, and T. Kikuno. Minimizing the maximum delay for reaching consensus in quorum-basedmutual exclusion schemes. *IEEE Trans. on Parallel and Distributed Systems*, 10(4):337–345, 1999.

[16] J. Wang, H. Zhu, and D. Li. eRAID: Conserving energy in conventional disk-based raid system. *IEEE Trans. on Computers*, 57(3):359–374, 2008.

[17] C. Weddle, M. Oldham, J. Qian, A.-I. Wang, P. Reiher, and G. Kuenning. PARAID: A gear-shifting power-aware RAID. *Proc. of the USENIX Conf. on File and Storage Technologies*, pages 245–260, 2007.

[18] X. Yao and J. Wang. RIMAC: a novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. *Proc. of the ACM SIGOPS/EuroSys European Conf. on Computer Systems*, pages 249–262, 2006.

[19] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: helping disk arrays sleep through the winter. *ACM SIGOPS Operating Systems Review*, 39(5):177–190, 2005.