Frequency Selection Approach for Energy Aware Cloud Database

Chaopeng Guo, Jean-Marc Pierson, Liu Hui, Jie Song

Digital Object Identifier 10.1109/ACCESS.2018.2885765

Frequency Selection Approach for Energy Aware Cloud Database

CHAOPENG GUO[1], JEAN-MARC PIERSON[1], HUI LIU[2], AND JIE SONG[3]

[1]Institut de Recherche en Informatique de Toulouse, University of Toulouse, Toulouse 31062, France
[2]School of Metallurgy, Northeastern University, Shenyang 110819, China
[3]Software College, Northeastern University, Shenyang 110819, China

Corresponding author: Jie Song (songjie@mail.neu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672143, Grant 61662057 and Grant 51606031, in part by the Fundamental Research Funds for the Central Universities under Grant N161602003, and in part by the scholarship from the China Scholarship Council under Grant CSC N201506080029.

ABSTRACT A lot of cloud systems are adopted in industry and academia to face the explosion of the data volume and the arrival of the big data era. Meanwhile, energy efficiency and energy saving become major concerns for data centers where massive cloud systems are deployed. However, energy waste is quite common due to resource provisioning. In this paper, using dynamic voltage and frequency scaling (DVFS), a frequency selection approach is introduced to improve the energy efficiency of the cloud systems in terms of resource provisioning. In the approach, two algorithms, genetic algorithm (GA) and Monte Carlo tree search algorithm (MCTS), are proposed. A cloud database system is taken as an example to evaluate the approach. The results of the experiments show that both algorithms have its advantages. The algorithms have great scalability, in which the GA can be applied to thousands of nodes and the MCTS can be applied to hundreds of nodes. Both algorithms have high accuracy compared with optimal solutions (up to 99.9% and 99.6% for GA and MCTS, respectively). According to an optimality bound analysis, 26% of energy can be saved at most using our frequency selection approach.

INDEX TERMS Cloud database system, dynamic voltage and frequency scaling, energy efficiency, frequency selection, optimization.

I. INTRODUCTION

To cope with challenges of Big Data, an increasing number of data centers are constructed. Cloud systems are developed to meet users' rapidly growing data processing needs. With the explosive growth of the construction of data centers, the problem of energy waste becomes more serious. The energy bills for major cloud service providers are typically the second largest item in their budgets due to the increased number of computational resources. [1].

Cloud database system is one of the typical cloud systems. To cope with the huge data storage and query needs, massive cloud databases are established, such as, HBase [2], Hive [3] and Cassandra [4]. To build energy efficiency cloud database systems, a lot of researches have been done. For example, to achieve green computing, the measurement model and approach of the energy efficiency of cloud database systems were defined by Song *et al.* [5], and the EE characteristics of cloud database were investigated. Similarly, Lang *et al.* [6] studied the tradeoff between performance and energy efficiency for parallel DBMS to give principles of designing an energy efficiency database system. Typical cloud databases are wasting energy. Since users' activities are dynamic, the workloads of the system vary with time: users' activities are more intense in daytime, whereas they do little at night. In this case, part of the energy is wasted if the system's configuration at night remains the same as the one during daytime. The energy waste comes from resource provisioning.

To cope with resource provisioning in cloud systems, some self adjusting systems emerged, i.e. Ursa [7] and WattDB [8]. WattDB is a self adjusting distributed database management system, in which it tries to switch nodes on and off according to current workload. Ursa is a self adjusting cloud storage system, in which it tries to migrate data from the hotspot node to underutilized nodes with minimizing latency and bandwidth manner, and it tries to turn off the underutilized node to save energy if possible. However, these strategies might not suitable for cloud database systems since switching nodes on and off would cause an unacceptable migration cost and damage system availability. In WattDB, Schall and

Härder took advantages of SDD to reduce the IO cost during the migration process. In Ursa, You *et al.* tried to turn off the underutilized nodes which do not contain primary replicas. Since there are different consistence policies in cloud database systems, the approach is not suitable either. Chihoub *et al.* [9] introduced another idea to improve the energy efficiency of cloud database systems, in which they analyzed the tradeoff between energy and consistency of the system by means of Dynamic Voltage and Frequency Scaling (DVFS). DVFS [10] is an efficient technology to control power consumption of processors. Power control policies can be made with DVFS technique. In modern Linux operating systems, five different power schemes (governors) are available to dynamically scale CPU frequency according to CPU utilization. The dynamic tuning strategies use CPU's running information to adjust its frequency, which does not reflect the state of cloud databases' workload lifecycle including memory and disk transfers. The current power schemes do not exactly match their design goal and may even become ineffective in improving energy efficiency of cloud database systems [11]. Therefore, it makes sense to control frequency in a fine-grained way. In [9], Chihoub *et al.* proposed a property approach. They assigned highest frequency to half of the nodes and lowest frequency to another half, and they achieved 23% of energy saving. However, this approach is a static method for assignment frequencies, and the usage scenario is limited.

To extend Houssem-Eddine's work, a frequency selection approach is introduced in this work to cope with the resource provisioning for cloud systems especially for cloud database systems. Instead switching nodes on and off, we try to assign the frequencies which are chosen according to the workload predictions of the system. Meanwhile, if there are underutilized nodes and overloaded nodes, a migration approach is introduced to avoid violation of Service-Level Agreement (SLA). Energy wasted by idle machines and SLA violation due to overloaded machines are reduced, which together improve the energy efficiency of the system.

The main challenge of the approach is its scalability. In a small case with 30 nodes and 8 available frequency options, there are 8^{30} frequency combinations in total. In our environment, the optimal solution of above case can be found by the entire searching within 5 hours. However, a common case has hundred nodes within a cluster. For example, the technical group from instagram claimed that their biggest Cassandra system contains 1000+ nodes and biggest cluster contains 100+ nodes in Cassandra summit 2016 [12]. Therefore, searching the optimal solution is unrealistic under this huge search space. Proposing an approach for searching a suboptimal solution with high accuracy and good performance is our goal. Besides that, when SLA violation occurs, the migration approach should be introduced which causes more effort to find the proper solution.

This paper is an expanded version of our previous conference paper [13]. In the previous work, we focus on the resource provisioning problem within cloud system, specially for cloud database system, and we made following contributions:

- By means of the DVFS technology and workload prediction, a general frequency selection model is proposed in aspect of energy efficiency for cloud system. Then, the model is specialized for cloud database system.
- Based on the frequency selection model, a genetic based algorithm and a monte carlo tree search based algorithm are proposed.
- Corresponding experiments are designed and implemented to verify the model and algorithms.

In this expanded version we add following contents:

- The model is described with more details about its constraints and objective.
- In model simplification section (IV-A), proofs of the upper bounds are given.
- Compared with the short version, we give more details and new result of the experiments.
- Since the prediction errors are inevitable, the robustness of proposed algorithms is analyzed in this extension (see Section VI).

The remainder of the paper is organized as follows: Section II reviews related works. A generic frequency selection model and a specialized model for cloud databases are proposed in Section III. Section IV proposes two algorithms under the frequency selection model. Section V evaluates the proposed algorithms, and Section VI analyzes the robustness of the algorithms. Section VII carries out some discussions. Section VIII concludes the paper and points out future research directions.

II. RELATED WORK

A variety of the past researches have been done to study and analyze the energy efficiency in cloud systems and cloud database systems. In general parts, Da Costa *et al.* [14] gave a lot of possible techniques of reducing energy consumption in large scale distributed systems. Lang *et al.* [6] gave guiding principles for building up the energy-efficient cloud DBMS with query properties and scalability taken into account. Subramaniam and Feng [15] measured the power consumption and the performance of a Cassandra cluster, and used power and resource provisioning techniques to analyze the energy proportionality of the cloud database system. Tsirogiannis *et al.* [16] analyzed the energy efficiency in distributed RDBMS systems, and they concluded that the most energy efficiency configuration is the highest performance one. In our work, their energy efficiency model is applied for cloud database system and the resources are provided according to the current workload to make sure the system is at its energy efficiency point. In this work, we focus on the energy efficiency for cloud systems especially for cloud database systems. The main idea of the work is to take advantage of DVFS technique to optimize the resource provisioning.

To improve the energy efficiency of cloud systems and cloud database systems, a straightforward idea is to switch underutilized nodes off and switch it on when necessary.

Based on this idea, there are lots of research have been done in cloud systems. Schall and Härder [8], [17], [18] designed and implemented WattDB, which is a distributed DBMS that dynamically adjusts itself switching nodes on and off according to the present workload, and reconfigures itself to satisfy the performance demands. You *et al.* [7] proposes system Ursa which scales to a large number of storage nodes and objects and aims to minimize latency and bandwidth costs during system reconfiguration. At first, Ursa tries to detect the hot spots in the system and re-balance these data with minimized transformation cost. Based on the data migration approach, Ursa implement power management approach in which they use a threshold strategy to maintain the amount of nodes to save energy.

Similarly machine virtualization-based technology that consolidating VMs dynamically and turning off idle servers has been proved effective also within its domain. Han *et al.* [19] proposed a remaining utilization-aware (RUA) algorithm for virtual machine placement, and a power-aware algorithm to find proper hosts to shut down for energy saving. Savinov and Daudjee *et al.* [20] proposed virtualization-driven database provisioning system, Dolly. Dolly took advantage of a new database replica spawning technique that leverages virtual machine cloning in which Dolly makes more replicas when the system is overloaded, and reduces the amount of replicas otherwise. Chen *et al.* [21] proposed scheduling approach for real time tasks within virtualization environment based on interval number theory, in which they also proposed scale functions to switch nodes on and off according to the workload.

Above works achieved energy efficiency in their domains. However, such approaches naturally require to migrate a large number of data from one node to another in order to switch off underutilized nodes or switch on more nodes. In WattDB [8], Schall and Härder took advantage of SDD to reduce the migration cost. In Usra [7], Gae-Won You *et al.* took advantage of replica strategy to shutdown nodes which do not contains any primary replicas. However, in cloud database systems, this technique might not suitable since with different implement of system, they have different consistency strategy. For example, Cassandra provides 5 consistency levels, and HBase provides strong consistency only. In Guangjie Han *et al.*'s work [19], their virtual machine consolidation policy aims to improve resource utilization and reduce the number of active physical servers, therefore they did not consider the migration cost. In Dolly [20], Emmanuel Cecchet *et al.* only considered the down time of the database but not the migration cost. Meanwhile, they only applied the technique for a web application's database layer in which 12 GB amount of data was used in the extreme case of the experiments. In Huangke Chen *et al.*'s work [21], they scheduled real-time tasks to the virtual machines. When there are overloaded nodes, they scale up the computing resource, namely turn up more virtual machines, and otherwise they reduce the amount of virtual machines. In their work, virtual machines are computing resources to execute tasks, therefore there is not migration cost at all. Compared with above related works, we do not chose to switch the nodes on and off, but assign a frequency according to the workload amount. Meanwhile, when SLA violation occurs and there is no higher frequency can be assigned, a migration approach is introduced.

In this work, we try to use DVFS technique to improve the energy efficiency of cloud systems, especially cloud database systems. There are some researches have been done where they used Dynamic Voltage and Frequency Scaling (DVFS) or Dynamic Voltage Scaling (DVS) technique. Yu *et al.* [22] studied the power efficiency scheduling problem of real-time tasks in an identical multi-core system, and presented Node Scaling model to achieve power-aware scheduling. Liu and Guo [23] studied energy efficient scheduling of periodic real-time tasks on multi-core processors with voltage islands, in which cores are partitioned into multiple blocks and each block has its own power source to supply voltage. Above works proposed heuristic algorithms to cope with voltage scaling problem in power saving manner. Compared our work, we try to assign the frequency to each node of cloud database systems according to the workload. Thus, we proposed a meta-heuristic algorithm (Genetic Based Algorithm) and heuristic search algorithm (Monte Carlo Search Tree Based Algorithm). In term of the use cases, in our approach, we do not switch nodes on and off to avoid the unacceptable migration cost and unavailability of the system, and for each time window the workloads would not be total reassigned but be migrated according to the assigned frequencies and previous workloads.

Chihoub *et al.* [9] explored the tradeoff between consistency and energy efficiency on the energy consumption in Cassandra. Meanwhile a prototype model, Hot-N-Cold, is introduced to reduce energy consumption in Cassandra by means of setting the frequencies manually. In our work, we try to extend this idea. The frequencies are assigned through frequency selection.

In practice, this work is related to workload prediction and knapsack problem also. Survey made by Maryam and Mohammad-Khanli [24] reviewed the state of the art application prediction methods in different aspects. Through a meticulous literature review of the state of the art application prediction schemes, a taxonomy for the application prediction models is presented that investigates main characteristics and challenges of the different models. Martello and Toth [25] introduce the knapsack approaches and multi-capacity bin-packing problem. Multi-capacity bin-packing is a generalization of the classical one-dimensional bin-packing problem and the approach is used to solve multi-resource allocation and scheduling solutions. In our work, the frequency selection approach is based on the workload prediction approach. According to workload predictions, the corresponding algorithms chose the frequencies for the system. Multi-capacity bin-packing technique is used to solve the migration problem.

III. FREQUENCY SELECTION MODEL

In this section, a generic model is introduced to abstract frequency selection model within cloud systems. Then, the generic model is specialized to cloud database systems by redefining the key concept. In the end, the objective and frequency selection related constraints are concluded.

A. GENERIC MODEL

A cluster $\mathbf{C}$ consists of n nodes. To simplify the description, the nodes are considered homogeneous. The extension of the model to heterogeneous nodes is discussed in Section VII.

The total running time of a system is made up of time windows. The length of a time window Δt is denoted as $|\Delta t|$. In Δt, the state of the system, $s_{\Delta t}(\mathbf{F}_{\Delta t}, \mathbf{W}_{\Delta t})$, is the state where the nodes are assigned to a frequency vector $\mathbf{F}_{\Delta t}$ and a workload vector $\mathbf{W}_{\Delta t}$. Since the following discussion is focusing on one time window, the notation Δt is omitted to simplify the description. A frequency f_i, ($f_i \in \mathbf{F}$), is assigned to a node c_i. Similarly, a workload w_i, ($w_i \in \mathbf{W}$), is assigned to c_i. The amount of w_i is denoted as $|w_i|$. It should be noticed that $\mathbf{W}$ is a prediction value. The influence of prediction errors is analyzed in section VI. The maximum amount of workload that can be handled by a node under a frequency is defined as its capacity. Let the capacity measurement function be $z(c_i, f_i)$. When the current workload exceeds the node's capacity, the workload cannot be completed, which causes SLA violation. In this paper, we consider that SLA violation is not allowed, namely all requests of a workload must be completed during the time window.

In order to avoid SLA violation, a migration process is introduced. Let the workload migration function be m. The migration process is considered as a state transformation process, namely $s(\mathbf{F}, \mathbf{W}) \xrightarrow{m} s^*(\mathbf{F}, \mathbf{W}^*)$. The process is denoted as $\widetilde{s}$. The workload for c_i after the migration is denoted as w_i^*. Energy used by the migration process is defined as migration cost. The migration cost estimation function is denoted $mc(\widetilde{s})$ and the system power consumption estimation function is $p(s)$. The energy consumption e of the system in Δt is:

$$e = p(s^*) \times |\Delta t| + mc(\widetilde{s}). \tag{1}$$

Energy efficiency of a system is defined in Equation (2) in which the energy efficiency in Δt is a ratio between the amount of workload processed and the energy consumption in Δt. Since the amount of workload is constant during a time window, the objective is to minimize e to improve the energy efficiency of the system.

$$ee = \sum_{i}^{n} |w_i|/e. \tag{2}$$

For a given frequency vector, the power consumption and the migration cost can be estimated by $p(s)$ and $mc(\widetilde{s})$ respectively. Finding the most energy efficient configuration is then to find the best frequency for each node: It is a search problem within the frequency combinations space.

The conditions for applying the model are:

1) The running time of a system can be divided into time windows. In a time window, the workload should be stable hence the power consumption can be estimated.
2) The node's capacity can be measured and part of the workload can be migrated when the current workload exceeds its capacity.
3) The workload of the next time window can be predicted according to previous running information.
4) The power consumption and the migration cost can be estimated according to the frequencies and the workloads.

B. SPECIALIZED MODEL FOR ENERGY AWARE CLOUD DATABASE

In this section, the generic model is specialized for cloud database systems. The workload w_i, the capacity measurement function $z(c_i, f_i)$, the migration function m, the power consumption estimation function $p(s)$ and the migration cost estimation function $mc(\widetilde{s})$ must therefore be identified.

In a cloud database system, the dataset $\mathbf{D}$ consists of h data blocks, in which the size of block b_g is denoted as $\left|b_g\right|$, and the blocks are distributed within the cluster. In order to meet data integrity and fault-tolerance requirements, cloud databases use a replication factor to control the number of replicas of the data blocks. The dataset with a replication factor r is denoted as $\mathbf{D}^r = \{b_{11}, b_{12} \ldots, b_{1r} \ldots b_{h1}, b_{h2} \ldots b_{hr}\}$, in which $b_{gk} \in \mathbf{D}^r, k \leq r$ is the k^{th} replica of b_g. The workload w_i of a cloud database system is defined as data query throughput. The probability of b_{gk} being accessed is denoted as φ_{gk}, and the total throughput of the system is denoted as l. For all the blocks $\sum_g^h \sum_k^r \varphi_{gk} = 1$. φ_{gk} and l are prediction values, which can be given by data mining techniques and machine learning techniques, such as time series data mining and linear regression. For the corresponding techniques, we refer readers to the literature [24]. Let the block set assigned to c_i be $\mathbf{D}_i^r = \{b_{gk} \mid b_{gk} \in \mathbf{D}^r \text{and } b_{gk} \text{ is assigned to } c_i\}$. The workload w_i is defined by:

$$w_i = \sum_{b_{gk} \in \mathbf{D}_i^r} l \times \varphi_{gk}. \tag{3}$$

The capacity measurement function $z(c_i, f_i)$ is a discrete function. Using benchmarks, the maximum throughput of a cloud database under each frequency can be obtained (see Section V-A1).

The frequency option set is denoted as η. A frequency f is one of the available frequency options. Let the idle power consumption and the maximum power consumption of a node under a frequency f be c_f^{idle} and c_f^{max} respectively. If $\forall f_p$, $f_q \in \eta$ and $f_p > f_q$, $c_{f_p}^{idle} > c_{f_q}^{idle}$ and $c_{f_p}^{max} > c_{f_q}^{max}$. With a higher frequency, the system provides more resources to support workloads, but consumes more energy. If a fraction ψ of CPU is used under the frequency f, the power consumption estimation function is defined by Equation (4). In cloud database systems, ψ_i is defined by Equation (5), in which

w_i^* is the workload after the migration, hence $\psi_i \leqslant 1$.

$$p(s^*) = \sum_{i}^{n} \left(c_{f_i}^{idle} + \psi_i \times \left(c_{f_i}^{max} - c_{f_i}^{idle} \right) \right), \tag{4}$$

$$\psi_i = \frac{w_i^*}{z(c_i, f_i)}. \tag{5}$$

The migration process refers to the migration of data blocks. According to the network topology, there are 3 types of migration: 1) Migration within a rack; 2) Migration between racks; 3) Migration between data centers. This paper only focuses on the first two types. Since the migration function m is not a focus of this paper, it is described briefly: m consists of two phases, block selection and block migration.

- Block selection can be treated as a single knapsack problem. Considering a node c_i where the workload exceeds its capability, the problem is to find which blocks should be kept in order to maximize the total size of the kept blocks. In the single knapsack problem, the capability of the knapsack is the capability of the node $z(c_i, f_i)$. The items are the blocks assigned to the node. The values of the items are the block sizes. The weights of the items are the throughputs of the blocks $l \times \varphi_{gk}$.
- Block migration can be considered as a 0-1 multiple knapsack problem. Let the rack set be $\{\gamma_1, \ldots, \gamma_u\}$ $u \ll n$. Considering a rack γ_p in which the selected blocks set is $\mathbf{M}_p$, the problem is to find which blocks should be kept within the rack in order to maximize the total size of kept blocks. The knapsacks are the nodes with extra capabilities, namely $\{c_i | c_i \in \gamma_p,\ z(c_i, f_i) > w_i\}$, and the sizes of knapsacks are $z(c_i, f_i) - w_i$. The items are the selected blocks, $\mathbf{M}_p$. The weights and values of items are throughputs and size of the blocks, respectively.

Due to the potentially huge amount of blocks, exact algorithms cannot be applied. A greedy algorithm is used to cope with the knapsack problem, and an approximation algorithm, MTHM algorithm [25], is used to solve the 0-1 multiple knapsack problem. Let $\mathbf{M}_{\text{In}}$ and $\mathbf{M}_{\text{Out}}$ denote the block sets migrated within a rack and between racks, respectively. Let e_{In} and e_{Out} denote the energy costs per mega byte of migration within a rack and between racks respectively, which are obtained through the benchmark experiment (see Section V-A2). The migration cost estimation function $mc(\widetilde{s})$ is defined by:

$$mc(\widetilde{s}) = e_{\text{In}} \times \sum_{b_{gk} \in \mathbf{M}_{\text{In}}} \left| b_{gk} \right| + e_{\text{Out}} \times \sum_{b_{gk} \in \mathbf{M}_{\text{Out}}} \left| b_{gk} \right|. \tag{6}$$

By means of redefining the key conceptions and key functions of the generic model, the model is specialized for cloud database systems to cope with the resource provisioning problem. This subsection is an example to extend the generic model to fit to a cloud system.

C. CONCLUSION

To improve the energy efficiency of cloud database systems, energy consumption of each time window should be minimized according to Equation (2). In could database systems, energy consumption is estimated according to Equation (1) in which the power consumption and the migration cost are given by Equation (4) and Equation (6) respectively. Thus, energy consumption of each time window comes from two parts: energy for running the workload and energy for performing migration. According to the previous sections, the model of frequency selection approach for cloud database systems is shown as follow:

$$\textbf{minimize} \sum_{i=1}^{n} \left(c_{f_i}^{idle} + \frac{w_i^*}{z(c_i, f_i)} \times \left(c_{f_i}^{max} - c_{f_i}^{idle} \right) \right) + e_{\text{In}} \times \sum_{b_{gk} \in \mathbf{M}_{\text{In}}} \left| b_{gk} \right| + e_{\text{Out}} \times \sum_{b_{gk} \in \mathbf{M}_{\text{Out}}} \left| b_{gk} \right|. \tag{7}$$

$$\textbf{subject to } \forall i \in [1, n]\ w_i^* \leq z(c_i, f_i), \tag{8}$$

$$\min(\eta) \geq f^*, \tag{9}$$

$$\forall i \in [1, n]\ f_i \in \eta, \tag{10}$$

$$\text{If } \forall f_p, \quad f_q \in \eta,\ f_p > f_q, \quad \text{then } c_{f_p}^{idle} > c_{f_q}^{idle}, \quad c_{f_p}^{max} > c_{f_q}^{max}. \tag{11}$$

Equation (7) gives the objective of the approach where the energy consumption of each time window should be minimized. Equation (8) to Equation (11) show the constraints that are related to frequency selection process. Constraint (8) indicate that the SLA cannot be violated within each time window, namely the migrated workloads cannot exceed the capacity of each node. In Constraint (9), f^* is the critical speed of the CPU [22], [26]. Constraint (9) shows all the available frequency options are bigger than the critical speed. Constraint (10) shows that every node within the system is assigned with a frequency. Combining Constraint (9) and Constraint (10), nodes are always active within the frequency selection process to avoid unacceptable migration cost and unavailability of the system. Constraint (11) shows that the idle power consumption and maximum power consumption of the node have positive relationship with its assigned frequency.

IV. FREQUENCY SELECTION ALGORITHM

Two algorithms for frequency selection are introduced in this section: Genetic Algorithm (GA) and Monte Carlo Tree Search Algorithm (MCTS). Both algorithms have their advantages and disadvantages which are discussed in Section VII. Before introducing the algorithms, a model simplification is proposed.

A. MODEL SIMPLIFICATION

The power consumption and the migration cost can be estimated by Equation (4) and (6). However, both values are obtained after the migration process. Using the greedy algorithm and MTHM algorithm, a migration plan can be obtained within polynomial time. However, when the total amount of possible frequency vectors increases, the evaluation time becomes unacceptable. The model simplification obtains upper bounds of the power consumption and the migration

cost, which are used to evaluate the frequency vectors. When several candidates are obtained, m is applied and the vector with the minimum energy consumption is chosen. The idea of the model simplification is to reduce costs for computing migration cost values given by the migration cost estimation function mc using a relaxation approach.

1) POWER CONSUMPTION

According to Equation (4), power consumption is related to node's frequency and CPU usage. Considering a block b_{gk} is assigned to c_i, the power consumption of c_i increases because of it, i.e. it increases with the probability access φ_{gk} due to that block. The increment is $\frac{\varphi_{gk}}{z(c_i,f_i)} \times \left(c_{f_i}^{max} - c_{f_i}^{idle}\right)$ in which the constant factor $\frac{\left(c_{f_i}^{max} - c_{f_i}^{idle}\right)}{z(c_i,f_i)}$ is defined as the power consumption contribution factor of c_i. In order to achieve the maximum power consumption, the blocks are assigned to the nodes with the highest power consumption contribution as much as possible.

In order to simplify the assignment, a relaxation is introduced, in which the blocks are continuous. By means of the relaxation, one block can be split and put to multiple nodes. Let the total ordered node set be $\widetilde{\mathbf{C}} = (\mathbf{C}, \leq)$, in which if $\forall i, j \left(c_{f_i}^{max} - c_{f_i}^{idle}\right)/z\left(c_i, f_i\right) \geqslant \left(c_{f_j}^{max} - c_{f_j}^{idle}\right)/z\left(c_j, f_j\right)$, then $c_i \leqslant c_j$. If an index $\widetilde{i}$ of $\widetilde{\mathbf{C}}$ satisfies $\sum_{i=0}^{\widetilde{i}} z(c_i, f_i) \leqslant l \leqslant \sum_{j=0}^{\widetilde{i}+1} z(c_j, f_j)$, then index $\widetilde{i}$ is a pivot node: the blocks are assigned to the nodes c_i with $i \leqslant \widetilde{i}+1$. p^{max} is shown by Equation (12). The nodes c_i with $i \leqslant \widetilde{i}$ reach their maximum power consumption, and the nodes c_i with $i \geqslant \widetilde{i}+2$ reach their idle power consumption because there is no assigned block. The power consumption of $c_{\widetilde{i}+1}$ is computed by Equation (4).

$$p^{max} = \left(z\left(c_{\widetilde{i}+1}, f_{\widetilde{i}+1}\right) - \left(\sum_{i}^{\widetilde{i}+1} z\left(c_i, f_i\right) - \sum_{i}^{n} w_i\right)\right) \times \frac{c_{f_{\widetilde{i}+1}}^{max} - c_{f_{\widetilde{i}+1}}^{idle}}{z\left(c_{\widetilde{i}+1}, f_{\widetilde{i}+1}\right)} + c_{f_{\widetilde{i}+1}}^{idle} + \sum_{i=0}^{\widetilde{i}} c_{f_i}^{max} + \sum_{i=\widetilde{i}+2}^{n} c_{f_i}^{idle}. \tag{12}$$

Proposition 1: For a given frequency vector $\mathbf{F}$*, the power consumption achieved by Equation (12) is the maximum power consumption.*

Proof: For a given frequency vector $\mathbf{F}$, p^{max} is obtained by Equation (12). Assume that p^* exists, and $p^* > p^{max}$. Consider the following situations:

1) When $\widetilde{i}+1 = n$, then $\exists b_{gk}$ is assigned to c_i ($i \in [1, \widetilde{i}]$) in p^{max}, while it is assigned to $c_{\widetilde{i}+1}$ in p^*. In this case, c_i ($i \in [1, \widetilde{i}]$) is denoted as c_{i_1} and $c_{\widetilde{i}+1}$ is denoted as c_{i_2}.
2) When $\widetilde{i}+1 < n$, then $\exists b_{gk}$ is assigned to c_{i_1}, $i_1 \in [1, \widetilde{i}+1]$ in p^{max}, while it is assigned to c_{i_2}, $i_2 \in [\widetilde{i}+1, n]$ in p^*, and $i_1 \neq i_2$.

In above situations, $i_1 < i_2$. $\frac{\varphi_{gk}}{z(c_{i_1}, f_{i_1})} \times \left(c_{f_{i_1}}^{max} - c_{f_{i_1}}^{idle}\right) > \frac{\varphi_{gk}}{z(c_{i_2}, f_{i_2})} \times \left(c_{f_{i_2}}^{max} - c_{f_{i_2}}^{idle}\right)$ can be concluded since the nodes are sorted by descending their power contribution factors. So, $p^{max} > p^*$ which is a contradiction. Therefore, p^{max} is the maximum power consumption under the given frequency vector. □

2) MIGRATION COST

To obtain the migration cost by means of Equation (6), migrated blocks and migration destinations are required. The upper bound of migration cost can be obtained by a relaxation approach, in which the nodes with free capacity in the same rack are combined to form a big knapsack with larger free capacity. To compute the maximum migration cost, two conditions are introduced:

1) For a rack γ_p, the selected block set is denoted as $\mathbf{M}_p$. The total ordered set of $\mathbf{M}_p$ is denoted as $\widetilde{\mathbf{M}}_p = (\mathbf{M}_p, \leqslant) = \{b_1, b_2, \ldots\}$ in which the blocks are sorted by the ratio between their throughputs and sizes, namely $\forall b_i, b_j \in \widetilde{\mathbf{M}}_p$, if $\frac{l \times \varphi_i}{|b_i|} \geqslant \frac{l \times \varphi_j}{|b_j|}$, then $b_i \leqslant b_j$.
2) The blocks with higher ratio values are kept within racks and the other blocks are migrated to other racks. $\widetilde{\mathbf{M}}_p^{\mathrm{In}}$ and $\widetilde{\mathbf{M}}_p^{\mathrm{Out}}$ are partitions of $\widetilde{\mathbf{M}}_p$. $\widetilde{\mathbf{M}}_p^{\mathrm{In}}$ and $\widetilde{\mathbf{M}}_p^{\mathrm{Out}}$ satisfy the following conditions:
   - $\widetilde{\mathbf{M}}_p^{\mathrm{In}} = \{b_1, \ldots, b_q\}$, $\widetilde{\mathbf{M}}_p^{\mathrm{Out}} = \{b_{q+1}, b_{q+2}, \ldots\}$.
   - Let the set of partial nodes in γ_p that have extra capacity be $\mathbf{C}_p^{in} = \left\{c_i | c_i \in \gamma_p \text{ and } w_i < z(c_i, f_i)\right\}$. Then, the index q of $\widetilde{\mathbf{M}}_p$ satisfies $\sum_{j=1}^{q} l \times \varphi_j \leqslant \sum_{c_i \in \mathbf{C}_p^{in}} (z(c_i, f_i) - w_i) < \sum_{j=1}^{q+1} l \times \varphi_j$.

With above conditions, the maximum migration cost can be obtained by Equation (13), in which u is the amount of racks. Since the blocks with lower throughputs but larger sizes are migrated to other racks, the migration cost is the highest among all migration plans.

$$mc^{max} = e_{\mathrm{In}} \times \sum_{b_{\mathrm{In}} \in \bigcup_{p=1}^{U} \widetilde{\mathbf{M}}_p^{\mathrm{In}}} |b_{\mathrm{In}}| + e_{\mathrm{Out}} \times \sum_{b_{\mathrm{Out}} \in \bigcup_{p=1}^{U} \widetilde{\mathbf{M}}_p^{\mathrm{Out}}} |b_{\mathrm{Out}}|. \tag{13}$$

Proposition 2: For a given migrated block set $\mathbf{M}$*, the migration cost achieved by Equation 13 is the maximum migration cost.*

Proof: Consider two migration cases: 1, the migration process achieved by MTHM algorithm (or other 0-1 multiple knapsack algorithms); 2, the migration process achieved by the model simplification approach. The migration cost for case 1 is denoted as mc obtained according to Equation (6), and the migration cost for case 2 is denoted as mc^{max} obtained according to Equation (13).

The amount of rank(s) is denoted as U. When $U = 1$, then the migration cost is constant for the given migrated block $\mathbf{M}$, which is $e_{\mathrm{In}} \times \sum_{b \in \mathbf{M}} |b|$. Therefore $mc^{max} = mc$.

When $U > 1$, we have following discussion. To simply the description, the following discussion is carried out within a rank γ_p, and the migration cost because of γ_p is denoted as mc_p and mc_p^{max} respectively for both cases. For a rank γ_p, the migrated block sets are denoted as $\mathbf{M}_1^{\mathrm{In}}$ and $\mathbf{M}_1^{\mathrm{Out}}$ for

TABLE 1. Evaluation of model simplification.

DataSet	With Migration Process	With Model Simplification
$d10$	$7.86s$	$0.18s$
$d20$	$16.01s$	$0.25s$
$d30$	$37.87s$	$0.34s$

case 1, and the migrated block sets are denoted as $\mathbf{M}_2^{\text{In}}$ and $\mathbf{M}_2^{\text{Out}}$ for case 2. Without loss of generality, we assume all sets are not empty.

The set of partial nodes that have extra capacity is $\mathbf{C}_p^{in} = \{c_i | c_i \in \gamma_p \text{ and } w_i < z(c_i, f_i)\}$. The total capacity which can handle workloads is $\sum_{c_i \in C_p^{in}} (z(c_i, f_i) - w_i)$. Consider the block assignment process, $\frac{\sum_{b_j \in \mathbf{M}_1^{\text{In}}} l \times \varphi_j}{\sum_{b_j \in \mathbf{M}_1^{\text{In}}} |b_j|} \leq \frac{\sum_{b_j \in \mathbf{M}_2^{\text{In}}} l \times \varphi_j}{\sum_{b_j \in \mathbf{M}_2^{\text{In}}} |b_j|}$, because $\mathbf{M}_2^{\text{In}}$ contains the first q blocks from the total ordered set of $\mathbf{M}$, in which the blocks are sorted by the descending ratios between its throughput and size. Therefore, $\sum_{b_j \in \mathbf{M}_1^{\text{In}}} |b_j| \geq \sum_{b_j \in \mathbf{M}_2^{\text{In}}} |b_j|$. Since $\mathbf{M}_1^{\text{In}} \cup \mathbf{M}_1^{\text{Out}} = \mathbf{M}_2^{\text{In}} \cup \mathbf{M}_2^{\text{Out}} = \mathbf{M}$, $\sum_{b_j \in \mathbf{M}_1^{\text{Out}}} |b_j| \leq \sum_{b_j \in \mathbf{M}_2^{\text{Out}}} |b_j|$. According to Equation (6), $mc_p^{max} \geq mc_p$. Therefore, $mc^{max} \geq mc$, and mc^{max} is the maximum migration cost for the given migrated block. □

Using the model simplification, the performance for evaluating frequency vectors is improved. To verify the improvement, a frequency evaluation process with model simplification and a frequency evaluation process with migration process are applied to evaluate 1000 frequency vectors for dataset $d10$, $d20$ and $d30$ (see Section V-B) respectively and the execution time for each case is shown in Table (1). The execution time of evaluating frequency vectors for $d10$, $d20$ and $d30$ are reduced 97.7%, 98.4% and 99.1% respectively. In order to avoid the impact of this simplification on the frequency selection algorithms, the corresponding algorithms are executed multiple times to obtain several frequency vector candidates, and the one with the minimum energy consumption is chosen as the final solution. An experiment is made to evaluate the influence of number of candidates (see Section V-B3).

B. GENETIC ALGORITHM

Genetic Algorithm (GA) is a type of algorithms for randomly searching suboptimal solutions, which is guided by evaluation and natural genetics [27]. Generally, GA includes several phases in each iteration : 1, encoding; 2, generation of initial population; 3, evaluation; 4 selection; 5 crossover; 6 mutation and 7 stopping criteria. In this section, the essential parts of GA—encoding and evaluation—are introduced. The influence of parameters of GA is discussed in Section V.

The objective of frequency selection is to generate the frequency vector $\mathbf{F}$ for the cloud system, which minimizes energy consumption within each time window. An encoded frequency vector $\mathbf{F}$ is regarded as a chromosome. In the encoding process, frequency $f_i \in \mathbf{F}$ is replaced by its index within the frequency option set. Let the frequency option set be η and function $I_\eta(f_i)$ gives the corresponding index of the frequency option f_i. The chromosome is represented by $< I_\eta(f_1), I_\eta(f_2), \ldots, I_\eta(f_n) >$.

Algorithm 1 Fitness Function of Genetic Algorithm

Require: *chromosome* encoded frequencies
Ensure: *score* fitness value
1: **function** evaluate(*chromosome*)
2: $\mathbf{F} \leftarrow$ DECODE(*chromosome*)
3: $p^{max} \leftarrow$ MAXPOWERCONSUMPTION($\mathbf{F}$)
4: $mc^{max} \leftarrow 0$
5: **for** $p \leftarrow [1, \ldots, u]$ **do**
6: $\mathbf{M}_p^{\text{In}}, \mathbf{M}_p^{\text{Out}} \leftarrow$ SELECTMIGRATIONBLOCKS(γ_p)
7: $mc^{max} \leftarrow mc^{max} +$ MIGRATIONCOST($\mathbf{M}_p^{\text{In}}, \mathbf{M}_p^{\text{Out}}$)
8: **end for**
9: **return** $p^{max} \times |\Delta t| + mc^{max}$
10: **end function**

In the evaluation phase, a fitness function is required to qualify chromosomes. The power consumption and the migration cost are estimated according to chromosomes. However, as discussed above, the fitness function may become a bottleneck and the model simplification approach is applied. The pseudocode of the fitness function is shown in Algorithm 1.

The function gives a score for each *chromosome*. Firstly, the *chromosome* is decoded (line 2). Secondly, the maximum power consumption is obtained by Equation (12) (line 3). Thirdly, the maximum migration cost is calculated within the loop (line 4 to line 8). There are u racks, and the migration blocks are selected for each rack (line 6), and the migration cost for each rack is obtained by Equation (13). Finally, the maximum energy consumption is returned as the score.

Since the power consumption and the migration cost are replaced by their upper bound values, the solution of GA may not be optimal. In order to improve the accuracy of the algorithm, GA is applied multiple times to generate several candidates. Afterwards, m is applied to all candidates and the solution with the minimum energy consumption is chosen.

C. MONTE CARLO TREE SEARCH ALGORITHM

Monte Carlo Tree Search Algorithm (MCTS) is a method for finding the suboptimal decision in a given domain by taking random samples in the decision space and building a search tree according to the results. Over the last few years, MCTS has achieved great success with many games, complex real-world planning, optimization and control problems [28].

MCTS is based on Monte-Carlo process model. The model consists of a set of states, a set of actions, a transition model, and a reward function. The decision is presented as a pair of a state and an action, and the next state is chosen by a probability distribution built up by the current state and available actions. The link between state and actions is defined as

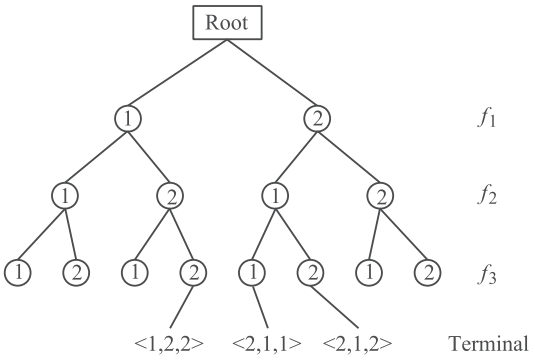


FIGURE 1. Frequency selection tree.

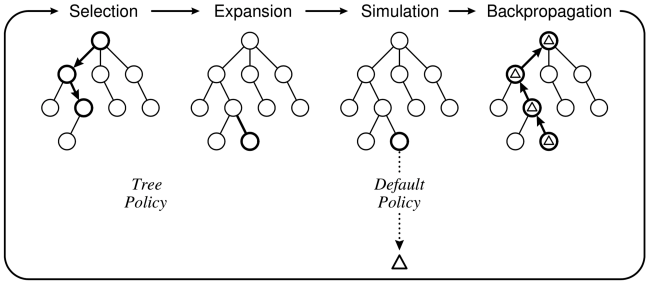


FIGURE 2. One iteration of the general MCTS approach.

policy and the aim is to find the special *policy** generating highest reward.

Under the frequency selection approach, the set of states are the frequency options. The action refers to choosing a frequency option for the next node. Figure (1) shows a structure of the frequency selection tree under a small case with 3 nodes and 2 available frequency options. In each layer, the frequency for the node is chosen. At beginning, f_1 has two options. When f_1 is set to frequency 1, there are 2 actions: set frequency 1 to f_2 and set frequency 2 to f_2. Therefore there are 4 states in second layer. Since the frequency option for the next node is not related to the current state, the frequency selection tree is a complete $|\eta|$-ary tree. When the searching process arrives at leaf nodes, the terminal condition is reached. A path of the tree is denoted as a frequency vector. For example, in Figure (1), $< 1, 2, 2 >$ is one of the frequency vectors. The task of MCTS is to find the frequency vector which produces the minimum energy consumption.

Figure (2) from the survey [28] explains the general process in MCTS including 4 phases in each iteration: selection, expansion, simulation, and back propagation.

1) Selection: Starting at the root node, a child selection policy is recursively applied to descend through the tree until the most urgent expandable node is reached. A node is expandable if it represents a non-terminal state and has unvisited (i.e. unexpanded) children.
2) Expansion: One (or more) child nodes are added to expand the tree, according to the available actions.
3) Simulation: A simulation is run from the new node(s) according to the default policy to produce an outcome.
4) Back-propagation: The simulation result is backed up through the selected nodes to update their statistics.

Basically, the process is controlled by two functions, a tree policy and a default policy. A node on the search tree is denoted as v and a child node of v is denoted as v'. Let function N show how many times the node has been visited. Let function Q give the score of v. The tree policy chooses the node with maximum UCT value. The UCT function is shown by Equation (14), in which C is a constant factor. In UCT function, the exploitation (visiting the expanded nodes) and the exploration (visiting the unexpended nodes) are balanced. If a node is not visited before, the tree policy chooses a node randomly. In the default policy, one of the paths is evaluated by a reward function R. Based on all nodes on the path, an evaluation score is required and the score is back propagated to all nodes on the path.

$$UCT = \frac{Q(v')}{N(v')} + C\sqrt{\frac{2\ln N(v)}{N(v')}}. \tag{14}$$

The base idea of the default policy in this work is the same with the fitness function of GA that evaluates a current solution and gives a score. Therefore, Algorithm (1) is used as the reward function of the default policy. However, the input parameter *chromosome* is replaced by the paths of the tree. In the tree policy, a dedicated UCT function, shown in Equation (15), is adopted. The difference between Equation (14) and (15) is the method for calculating scores. In MCTS, the value of the node's score is between 0 and 1, and the node with highest UCT value is selected. In the default policy of this paper, the maximum energy consumption is returned as score of a node. By means of $1 - \left(Q(v')/e^{max}\right)$, the value is converted within 0 and 1. The highest value of $1 - \left(Q(v')/e^{max}\right)$ indicates the path with the minimum energy consumption.

$$UCT = \frac{1 - \left(Q(v')/e^{max}\right)}{N(v')} + C\sqrt{\frac{2\ln N(v)}{N(v')}}. \tag{15}$$

In Equation (15), the value of e^{max} is required. Equation (1) shows that the energy consumption consist of two parts, namely the energy consumption of the running system and the energy consumption of workload migration. In most of the cases, the energy consumption of the running system is the dominant part. In this case, the maximum energy consumption scenario is that each node is assigned with its highest frequency. By means of Equation (4) and (6), the maximum energy consumption e^{max} can be obtained. In contrast, when migration cost is the dominant part, the frequency vector for achieving e^{max} cannot be constructed directly. In this case, GA is applied to find e^{max}, which makes it meaningless to apply MCTS since the frequency vector for achieving e^{min} can be found by GA also. Therefore, MCTS may be inapplicable for the cases in which the energy consumption of the workload migration is the dominant part. Like the GA approach, it should be noticed that the final solution of MCTS may not be optimal. MCTS is applied multiple times

to obtain several solutions, and the solution with the minimum energy consumption is chosen.

V. EXPERIMENT

In this section, a series test cases are designed and executed to verify the model and the corresponding algorithms. There are 5 objectives for the experiment:

1) Verify the exist of maximum throughout in cloud database system and obtain the corresponding static parameters;
2) Analyze the accuracy and performance of the algorithms;
3) Analyze the scalability of the algorithms;
4) Analyze the optimization bound of the approach;
5) Compare our approach with the exist approach.

Therefore the experiments consist of 5 parts: setup benchmark, parameter influence, scalability analysis, optimization boundary analysis and comparison with Hot-N-Cold approach.

A. SETUP BENCHMARK

In this experiment, two benchmarks are executed to obtain the capacity measurement function $z(c_i, f_i)$, and the energy costs per mega byte of migration within a rack and between racks, namely e_{In} and e_{Out}.

1) CAPACITY BENCHMARK

This benchmark is executed on Grid5000 [29] testbed. Grid5000 is designed to provide a scientific tool for computer scientists similar to the large-scale instruments used by physicists, astronomers and biologists. In the benchmark, a database system Cassandra [4] with 10 nodes belonging to the Nancy site graphene cluster is deployed. The core properties used in YCSB workload profile are shown in Table (2). To be noticed that, the cache related parameters are set to 0 to avoid the influence of cache mechanism to the experiment results.

The nodes are equipped with a 4 cores Intel Xeon X3440 and 16 GB of RAM. The energy consumption values are collected by Power Distribution Units (PDU). There are 8 available frequency options: 2.53GHz, 2.40GHz, 2.13GHz, 2.00GHz, 1.73GHz, 1.60GHz, 1.33GHz, 1.20GHz. In this benchmark, the maximum throughputs under each available frequency option are obtained.

To simulate the real workloads, Yahoo! Cloud Serving Benchmark(YCSB) framework [30] is selected as benchmark framework. YCSB is an open-source specification and program suite for evaluating retrieval and maintenance capabilities of computer programs. The core properties used in YCSB workload profile are shown in Table (3).

To obtain the maximum throughput of the system, more and more requests are loaded into the system. There are 12 workloads in total for each frequency option. The workload is denoted as Q_i $i \in [1, 12]$. For workload Q_i, the total amount of requests is $4000 \times 2^{i-1}$. The result is shown by Figure(3).

TABLE 2. Core properties of Cassandra.

Property	Value
num_tokens	256
max_hints_file_size_in_mb	128
key_cache_size_in_mb	0
row_cache_size_in_mb	0
concurrent_reads	32
concurrent_writes	32

TABLE 3. Core properties of YCBS workload.

Property	Value
recordcount	3000000
fieldlength	1000
readproportion	0.95
updateproportion	0.05
requestdistribution	uniform
threadcount	500

Figure (3a) shows the trends of throughput along with the increasing requests for the system under frequency 2.53GHz, 2.00GHz, 1.60GHz and 1.33GHz. The trends has a same pattern. Along with the increasing requests, the throughputs are increasing at first and then decline. During the fluctuation, the throughputs under different frequencies reach the highest point. At beginning, the throughput is lower than the node's capability. Therefore, the throughput increases as well. However, after the highest point of each line, the throughput tries to exceed the node's capability, but some requests cannot be finished because of the resource competition. The result is that the throughput declines. For all the frequencies, the capabilities are different. When the frequency is higher, the capability is larger. The capabilities for all frequency options are listed in Figure (3d). Note that, the capability is related to the hardware and software configuration of the system. When the configuration changes, the capability need to be reevaluated.

Figure (3b) shows the relationship between the energy efficiency and the throughputs under different frequencies. Along with the increasing of the throughputs, the energy efficiency increases as well. To be noticed that each line has a few coincident parts because when the throughput tries to exceeds the capability, the throughput declines. Each frequency has its maximum energy efficiency value and the energy efficiency value reaches its maximum value at its maximum throughput. To be noticed that each line has a few coincident parts because when the throughput tries to exceeds the capability, the throughput declines. Each frequency has its maximum energy efficiency value and the energy efficiency value reaches its maximum value at its maximum throughput.

Figure (3c) shows the energy consumption for each workloads. Along with the increasing of the requests, the energy consumption increases as well for each frequency configuration. However with the same requests amount, the energy consumptions under different frequency option do not have big diffidence. Since the maximum throughput of the system are tested, a lot of requests are loaded into the system.

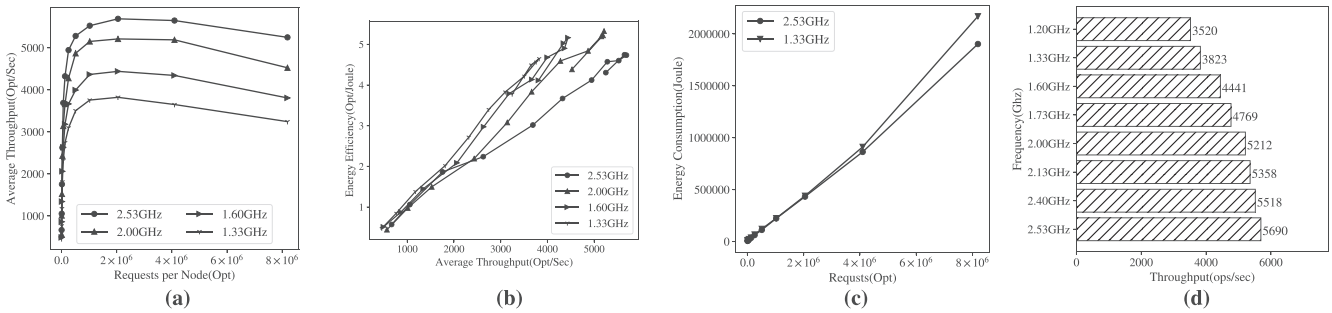


FIGURE 3. Capability benchmark result of Cassandra system. (a) Relationship between request amount and throughput. (b) Relationship between throughput and energy efficiency. (c) Energy consumption for workloads. (d) Node's capability under each frequency option.

TABLE 4. The energy cost per mega byte of the migration.

Type	Value
e_{In}	0.8 Joule/Mb
e_{Out}	1.0 Joule/Mb

With different frequency configuration, the throughput of the system is different. For example with 2.53GHz, the average throughput is 5590 Opt/Sec and with 1.33 GHz, the average throughput is 3822 Opt/Sec. Therefore with the same request amount, the execution time under two frequency option is quite different. For example, with Q_{12}, $t(2.53GHz) = 1778$ sec and $t(1.33GHz) = 2837$ sec. As a consequence, the energy consumptions do not have big difference for both cases.

2) MIGRATION COST BENCHMARK

In Equation (6), the migration cost is obtained by means of two static parameters, the energy costs per mega byte of migration within a rack and between racks, namely e_{In} and e_{Out}. To obtain these values, a benchmark is executed.

This benchmark is executed in Grid5000 platform at Nancy site graphene cluster as well. However, the system is deployed on 2 nodes. After the loading process, the system is waiting for a few minuets (5 mins in the experiment) to obtain the energy consumption in the idle status which is denoted as e_{Idle}. Then, a decommission process is executed on one of the nodes, which causes all the blocks in the node is migrated to another one. The energy consumption within the decommission process is denoted as $e_{\text{Migration}}$. By means of choosing different node combinations, e_{In} and e_{Out} can be obtained.

$$e_{\text{In|Out}} = \frac{e_{\text{Migration}} - e_{\text{Idle}} \times \frac{t_{\text{Migration}}}{t_{\text{Idle}}}}{\sum_{b_{gk} \in \mathbf{M}_{x|y}} |b_{gk}|}. \tag{16}$$

e_{In} and e_{Out} are calculated by Equation (16), in which $t_{\text{Migration}}$ and t_{Idle} indicate the execution time of the idle status and the execution time of the decommission process respectively. The results are shown in Table (4).

B. PARAMETER INFLUENCE

In this section, the influence of parameters on the algorithms is examined. The datasets are denoted as $d\{NodeAmount\}$. For example, $d10$ represents a dataset consisting of 10 nodes. As the number of blocks on each node does not impact the performance of the frequency selection algorithm, the number of blocks is set to 64 for each node, and the access probabilities are generated by Zipf's law (distribution factor is set to 2.5).

A test case is a combination of a dataset(d), a workload(l) and an algorithm(a). For example ($d10$, 5000, GA) indicates a test case, in which GA is applied to dataset $d10$ and the workload is set to 5000 opt/sec for each node. The value of throughput per node is a standard to simulate the total workload for the cases, and the throughput for each node is decided by φ_{gk}. The value of throughput per node is set to 5000 opt/sec by default if not otherwise specified, for example case ($d10$, 5000, GA) is denoted as ($d10$, GA)

In order to evaluate the accuracy of the algorithms, the solutions for the cases ($d10$, $Optimal$), ($d20$, $Optimal$) and ($d30$, $Optimal$) are obtained, in which $Optimal$ indicates the complete search where all possible frequency vectors are evaluated. The energy consumption for a case is denoted as $E(case)$ and the corresponding execution time is denoted as $T(case)$. The accuracy of a case $A(case)$ is defined by Equation (17) in which $d \in [d10, d20, d30]$ and $a \in [GA, MCTS]$.

$$A(d, a) = 1 - \frac{E(d, a) - E(d, Optimal)}{E(d, Optimal)}. \tag{17}$$

1) THE INFLUENCE OF GENERATION SIZE ON GA

The result is shown in Figure (4). In each case, the population size is set to 100 and the amount of candidates is set to 10. In Figure (4), $T(d, GA)$ increases with the increment of generation size for the reason that more generations lead to more iterations. With the same population size, $\forall i > j$ $T(di, GA) > T(dj, GA)$. More nodes lead to longer chromosome in GA, because the length of a chromosome is the number of nodes. In terms of accuracy, the range of $A(d, GA)$ is [0.994, 0.999]. As shown in Figure (4), $A(d, GA)$ increases at beginning with the increment of generation size. However, when generation size exceeds some points (100 for $d10$, $d20$ and 150 for $d30$), $A(d, GA)$ does not increase significantly and sometimes $A(d, GA)$ even decreases a little. More generations lead to more iterations of GA. At beginning, it leads to

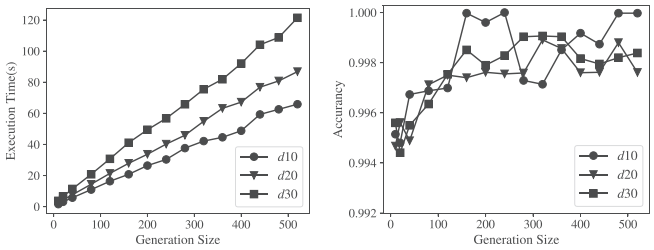


FIGURE 4. The influence of generation size on accuracy.

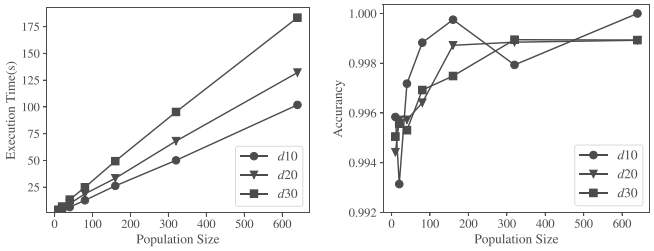


FIGURE 5. The influence of population size on accuracy.

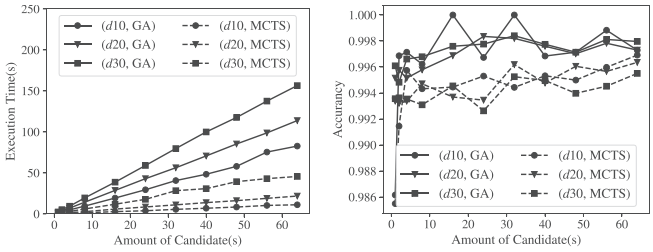


FIGURE 6. The influence of amount of candidates on accuracy.

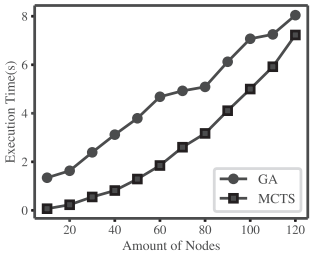


FIGURE 7. Scalability of the frequency selection algorithm.

more evolutions, which improves $A(d, GA)$. However afterwards the search process is close enough to an optimal point and the iterations keep the solution around the optimal point.

2) THE INFLUENCE OF POPULATION SIZE ON GA

The result is shown in Figure (5). In each case, generation size is set to 150 and the amount of candidates is set to 10. In Figure (5), with the same population size, $\forall i > j$, $T(di, GA) > T(dj, GA)$, because more chromosomes are evaluated in one iteration. Increasing population size improves $A(d, GA)$ at the beginning. However, at some points (80 for $d10$, 160 for $d20$ and 320 for $d30$), the increment of population size does not improve the accuracy any more.

3) THE INFLUENCE OF NUMBER OF CANDIDATES

GA and MCTS cannot find the optimal solution because of the model simplification approach. Therefore, both algorithms are executed multiple times to find several candidates, and the solution with the minimum energy consumption is chosen. For GA, generation size is set to 150 and population size is set to 100. The result is shown in Figure (6). In Figure (6), $T(d, GA)$ and $T(d, MCTS)$ increase with the amount of candidates. With the same amount of candidates, $T(d, GA) > T(d, MCTS)$. The reason is that GA is based on the evolutions while MCTS is based on the tree searching technique. The computation cost is higher for GA (see Section V-C). In term of accuracy, the increment of the amount of candidates improves $A(d10, GA)$ and $A(d10, MCTS)$ significantly at beginning. $A(d10, GA)$ goes up and down when more candidates are involved because in some cases, a close to optimal solution is found occasionally. In other cases, the accuracy of both algorithms increases slightly in general when more candidates are involved. Generally, $A(d, GA) > A(d, MCTS)$. In MCTS, the search space is organized by a tree structure. The leaf nodes are not ordered and there is no tendency among all the solutions. Considering Figure (1) and the maximum power consumption of terminal nodes, we have $p^{max}(< 1, 2, 2 >) > p^{max}(< 2, 1, 1 >)$ and $p^{max}(< 1, 2, 2 >) = p^{max}(< 2, 1, 2 >)$. Therefore, the random sampling method doesn't perform well in this scenario, which impacts negatively the overall accuracy. The maximum accuracy of MCTS is 99.6%.

C. SCALABILITY ANALYSIS

In this section, there are 12 datasets ($d10$ to $d120$) involved at first. Only one candidate is required in each case. For GA, generation size is set to 150 and population size is set to 100. The result is shown in Figure (7). Generally, $T(d, GA) > T(d, MCTS)$. When the dataset is smaller, the difference is more dramatic. For example, $T(d10, GA)$ is nearly 19 times $T(d10, MCTS)$. However, the growth rate of the execution time of GA is lower than MCTS. For example, $T(d120, GA)$ is 6 times $T(d10, GA)$ while $T(d120, MCTS)$ is 103 times $T(d10, MCTS)$. In GA, the increasing amount of nodes leads to the longer length of chromosome. When generation size and population size are constant, the length of chromosome only influences the performance in each evaluation. It leads to linear increment. However, in MCTS, when the amount of node is increased by 1, the height of the tree is increased by 1 which leads to exponential growth of the leaf nodes. The search of solutions is an exponential function, which makes the execution time grows exponentially.

In order to examine the scalability of our algorithms, an extreme scalability analysis is done. In this expeimrent, 10 huge datasets, ($d100$ to $d1000$) is used. Only one candidate is required in each case, and the configuration of the algorithms keeps same. The result is shown in Figure (8). At beginning, GA solve $d100$ around $7.05s$, while MCTS find a solution around $5.15s$. With the increment of the problem, both algorithms needs more time for finding a solution.

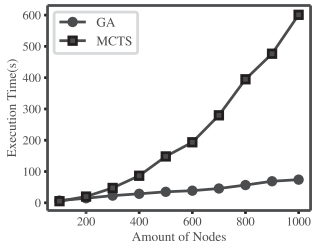


FIGURE 8. Extreme scalability examine.

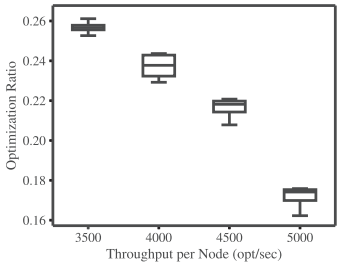


FIGURE 9. Optimization bound of the frequency selection algorithm.

For $d1000$, GA consumes $73.97s$, while MCTS consumes $600.90s$. The execution time of GA is acceptable considering the length of time window. As the conclusion from Figure (7) and Figure (8), GA is more suitable with huge datasets, since the execution time increment of GA is liner relationship with the amount of nodes, while it is exponential for MCTS.

D. OPTIMIZATION BOUNDARY ANALYSIS

In this section, optimization ratio is introduced to evaluate how much energy can be saved using the frequency selection approach. The optimization ratio is defined by Equation (18) in which *Performance* refers to the approach that all nodes are set to the performance mode (i.e., the maximum frequency). The optimization ratio indicates the ratio between the saved energy by frequency selection approach and the energy consumption under performance mode. In this section, each case is solved by GA. In GA, generation size is set to 150, population size is set to 100, and the amount of candidates is set to 10. There are 12 datasets involves ($d10$ to $d120$), and the cases are divided into 4 categories based on their throughputs per each node. The throughputs for each node are 3500 opt/sec, 4000 opt/sec, 3500 opt/sec and 5000 opt/sec.

$$O(d, l, a) = \frac{E(d, l, Performance) - E(d, l, a)}{E(d, l, Performance)}. \quad (18)$$

The result is shown in Figure (9). The optimization ratio depends on the value of throughput per node. For $\forall l_1, l_2 \in \{3500, 4000, 45000, 5000\}$ $l_1 > l_2$, $\mathbf{O}(d, l_1) < \mathbf{O}(d, l_2)$. With the same value of throughput per node, the optimization ratios are concentrated. With the increment of value of throughput per node, the optimization ratio increases. The maximum of optimization ratio is 26.2% for the case $(d10, 3500, GA)$, and the minimum of optimization ratio is 16% for the case $(d110, 5000, GA)$. If the power consumption is the only concern of the system's administration, the maximum optimization ratio can be constructed as follows. The node's throughput is set to 3520 opt/sec and the node is set to the performance mode. According to equation (18), the maximum optimization ratio is calculated as equation (19). 26.43% is the optimization bound of the model theoretically. The higher optimization ratio bound could be obtained by means of decreasing workload. However when the workload is too low, the value is meaningless, because the cluster is fully under-utilized. In the real cases, the optimization ratio might be lower than 26.43% because of the existence of the migration cost.

$$\begin{aligned} &O(d1, 3520) \\ &= \frac{P(< 2.53Ghz, 3520 >) - P(< 1.20Ghz, 3520 >)}{P(< 2.53Ghz, 3520 >)} \\ &= \frac{c^{idle}_{2.53Ghz} + \frac{3520}{5690} \times (c^{max}_{2.53Ghz} - c^{idle}_{2.53Ghz}) - c^{max}_{1.20Ghz}}{c^{idle}_{2.53Ghz} + \frac{3520}{5690} \times (c^{max}_{2.53Ghz} - c^{idle}_{2.53Ghz})} \\ &= 26.43\%. \end{aligned} \quad (19)$$

E. COMPARISON WITH HOT-N-COLD

Chihoub *et al.* [9] proposed a reconfiguration approach called Hot-N-Cold for Cassandra System to demonstrate the impact on energy consumption with strong and eventual consistency, in which half of the nodes are set to highest frequency and another half of nodes are set to lowest frequency. The comparison between Hot-N-Cold, GA and MCTS is made. The dataset used in this section is $d20$. For the cases, the throughput per node is set to 3500 opt/sec, 4000 opt/sec, 4500 opt/sec and 5000 opt/sec respectively.

The results are shown as Figure (10). The results given by GA and MCTS are better than the corresponding results given by Hot-N-Cold. The average improvement of GA compared with Hot-N-Cold is 12.65%, and the average improvement of MCTS is 11.44%. When the value of the throughput on each node is set to 5000 opt/sec, Hot-N-Cold approach cannot produce a valid result. Theoretically, when Hot-N-Cold approach applied, the system with 20 nodes can support any workloads with throughput under 92100 opt/sec, however with the setting 5000 opt/sec for each node, the system does not have enough resources to support it. Therefore the corresponding energy consumption is recorded as 0. The main drawback of Hot-N-Cold is its flexibility. GA and MCTS choose the frequency vector according to the workload predictions, while Hot-N-Cold sets the frequencies statically.

VI. ALGORITHM ROBUSTNESS ANALYSIS

In previous sections, the frequency selection model and corresponding algorithms are based on the predictions of the workload. Because the prediction errors are inevitable, the robustness of the algorithms are analyzed in this section.

To analyze the robustness of algorithms, datasets $d10$, $d20$ and $d30$ (Section V-B) are used in this section. In the specialized model, the block access possibility φ_{gk} is the key to define the workload for cloud database systems.

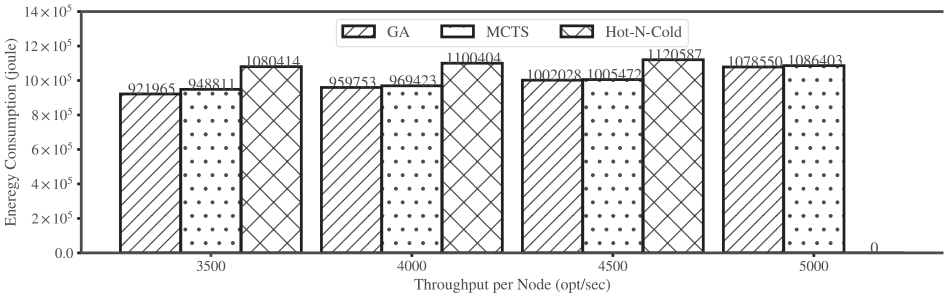


FIGURE 10. Comparison with Hot-N-cold.

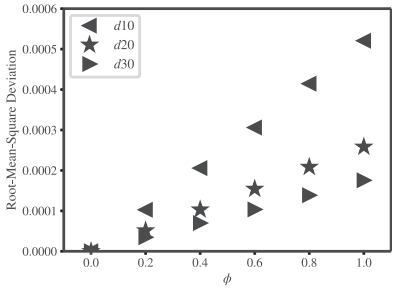


FIGURE 11. Relationship between ϕ and root-mean-square deviation.

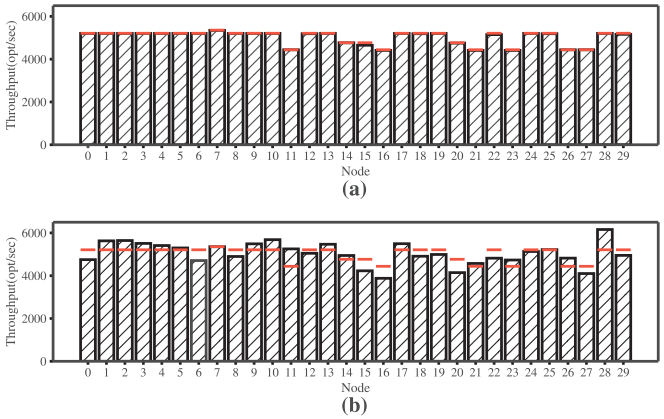


FIGURE 12. The scenarios with error prediction. (a) Based on error predictions. (b) Based on real workload.

At first the prediction errors extracted from the normalize distribution $N(\mu, \sigma^2)$. In each case, μ is set to 0 to make sure half of errors are negative and other half of errors are positive. The range of values extracted from the normalize distribution is within $[\mu + 3\sigma, \mu - 3\sigma]$. To make sure the ranges of errors are the same with corresponding case, and the σ is set to $\overline{\varphi}\phi$ in which ϕ is the value from $[0, 0.2, 0.4, 0.6, 0.8, 1]$ and $\overline{\varphi}$ is average value of φ_{gk}. ϕ controls the total errors. Specially, $\phi = 0$ indicates that there is no error introduced. Secondly, the errors are added to the corresponding φ_{gk} to simulate the cases with prediction errors, and the cases are denoted as $(d\{NodeAmount\}, \phi)$. The root-mean-square deviation (*RMSD*) values are calculated for each case, and the result is shown by Figure (11). With the same node amount, the increment of ϕ leads to the increment of *RMSD*, because larger ϕ value increases the possibility of generating larger error values. With the same ϕ value, when $i > j$ $\mathbf{RMSD}(di, \phi) < \mathbf{RMSD}(dj, \phi)$. This scenario is caused by corresponding $\overline{\varphi}$. For each case, every node is assigned with 64 blocks (see Section V-B) and $\overline{\varphi} = \frac{1}{64 \times NodeAmount}$. When $\overline{\varphi}$ increases, the RMSB increases.

When GA and MCTS are applied to cases with prediction errors, corresponding selected frequencies and migration plans are collected. Because of the prediction errors, there are two scenarios arise.

1) The real workload of a node exceeds its capacity;
2) Some of the resources of a node are wasted since the assigned workloads are too low.

To describe these scenarios, the execution result of GA for case $(d30, 1)$ is shown as Figure (12). In Figure (12), the bars present the workloads for each node, and the lines present the node's capabilities which are selected by GA. In Figure (12a), the prediction errors are pretended unknown, which makes $w_i^* < z(c_i, f_i)$ and a few resources are wasted, for example node 15. In contrary, in Figure (12b), the result is calculated based on the workload without prediction errors. In Figure (12b), some workloads exceed its capability which are the scenario 1, for example node 1, node 2 and so on. Some nodes are wasting their resources which belong to the scenario 2, for example node 0, node 6 and so on.

According to the experiment V-A, when request throughput (workload) try to exceed the node's capacity, the system throughput declines due to the resource limitation and the operation failure. To make sure the node can reach its capability, if the workload exceeds the node capability, part of the requests are refused. The ratio between refused requests and succeeded requests is defined as error ratio for the node. To describe the error ratio for the whole system, maximum error ratio (*MER*) is introduced which is defined by Equation (20). *MER* is used to describe scenario 1.

$$MER = \max \left\{ \frac{\sum_{b_{gk} \in \mathbf{D}_i^r} l \times \varphi_{gk} - z(c_i, f_i)}{z(c_i, f_i)}, i \in [1, n] \right\}. \quad (20)$$

The ratio between wasted resource with the node's capability is defined as waste ratio. Same with the *MER*, *MWR*,

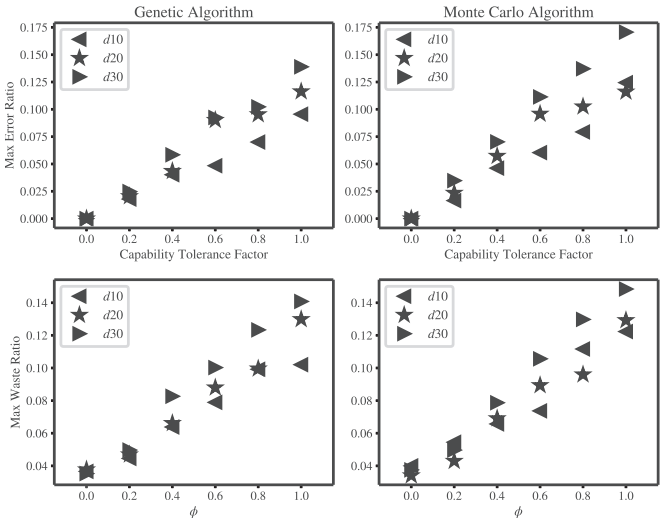


FIGURE 13. The influence of prediction errors.

defined by equation (21), indicates the maximum percentage of resources are wasted amongst all nodes, which is used to describe scenario 2.

$$MWR = \max\left\{\frac{z(c_i, f_i) - \sum_{b_{gk} \in \mathbf{D}_i^r} l \times \varphi_{gk}}{z(c_i, f_i)}, i \in [1, n]\right\}. \quad (21)$$

The results of the influence of prediction errors are shown in Figure (13). By means of GA and MCTS, the frequencies are selected, and the indicators, *MER* and *MWR*, are calculated by Equation (20) and Equation (21). With the same node amount d, if $\phi_1 > \phi_2$, then $MER(d, \phi_1) > MER(d, \phi_2)$ and $MWR(d, \phi_1) > MWR(d, \phi_2)$. For cases with same node amount, the increment of ϕ leads to increment of *RMDS* which indicates more errors are introduced, and more errors lead to higher *MER* and *MWR*. Specifically, when there is no error ($\phi = 0$) for $\forall d \in [d10, d20, d30]$, $MER(d, 0) = 0$ and $MWR(d, 0) > 0$. In perspective of frequency selection model, SLA cannot be violated, therefore no request is refused when no prediction errors are introduced. However, since the blocks are not continues, there are some energy is wasted when the migration process generates migration plan. But the algorithms try to minimize the energy consumption. In our cases, *MWR* is around 0.04 for cases with $\phi = 0$.

According to Figure (13), when prediction error exists, it results in higher *MER* and *MWR*. In order to decrease them, the capability tolerance factor is introduced, which is denoted as C. With the capability tolerance factor, the new capability $z^*(c_i, f_i)$ of node c_i with frequency f_i is shown by Equation (22). When $C < 0$, the capacity of the nodes is regarded lower than its original capacity. When the amount of workloads is constant, the higher frequencies will be selected to make sure enough resources to execute the workloads. In contrast when $C > 0$, the capacity of the nodes is regarded higher than its original capacity and the lower frequencies are likely to be selected to avoid energy wasting.

$$z^*(c_i, f_i) = z(c_i, f_i) \times (1 + C). \quad (22)$$

The values of $C \in [-0.1, -0.05, 0, 0.05, 0.1]$ are adopted to evaluate the influence of the capability tolerance factor. Since the conclusions are quite similar, part of the results, cases with ϕ 0.2 and 1, are shown in Figure (14). Generally, for each case, *MER* increases with the increment of C, while *MWR* is decreasing. The reason for this scenario is that when $C < 0$, the higher frequencies are selected which leads lower *MER*. Correspondingly, more resources are wasted which leads to higher *MWR*. When $C > 0$, the lower frequencies are selected which avoid resource wasting. However, it causes higher *MER*. According to the experiment result, there is a trade off between *MER* and *MWR* by means of tuning the capability tolerance factor. With the increment of capability tolerance factor (from negative to positive), *MER* increases which indicates the increment of SLA violation, and meanwhile *MWR* deceases which indicates the debasement of energy wasting. By means of comparison of results from GA and MCTS, with the same ϕ and C, there are no big difference between them in terms of *MER* and *MWR*. For example, in Figure (14) $\phi(0.2) - Genetic$, the range of *MER* is from 0% to 14% which is the same with $\phi(0.2) - MCTS$. In term of *MWR*, there is the same result. The influence of capacity factor decreases with more prediction errors, because more prediction errors lead to higher *MER* and *MWR*. For example, with $C = -0.01$ and GA, $MER(d30, 0.2) < MER(d30, 1)$.

In this section, the robustness of corresponding algorithms is analyzed. Prediction errors cause the SLA violation and energy wasting. In order to eliminate the influence of the prediction errors, the capability tolerance factor is introduced. By means of tuning the capability tolerance factor, the trade off between *MER* and *MWR* can be found. However, when the case has higher *MER*, it leads to more requests failure. Therefore, in practice, the *MER* should be kept lower value.

VII. DISCUSSION

GA and MCTS have their advantages and disadvantages and should be chosen according to the case.

1) With respect to accuracy, GA has higher accuracy up to 99.9% (only 99.6% for MCTS).
2) In term of scalability, both algorithm can be applied to medium size cluster (120 nodes). The performance of MCTS is better than GA, especially for the small cases. For example, in the case with 10 nodes, MCTS is 19 times faster than GA. However, in face of large cluster (1000 nodes), the performance of GA is more competitive. Since the execution time of GA is linear with the amount of nodes, but it is exponential for MCTS. In our experiment, GA solved the case with 1000 nodes for 74s.
3) The usage scenario of MCTS is limited under the condition that the energy consumption of the running system is the dominant part.
4) GA needs to be tuned with the parameters.

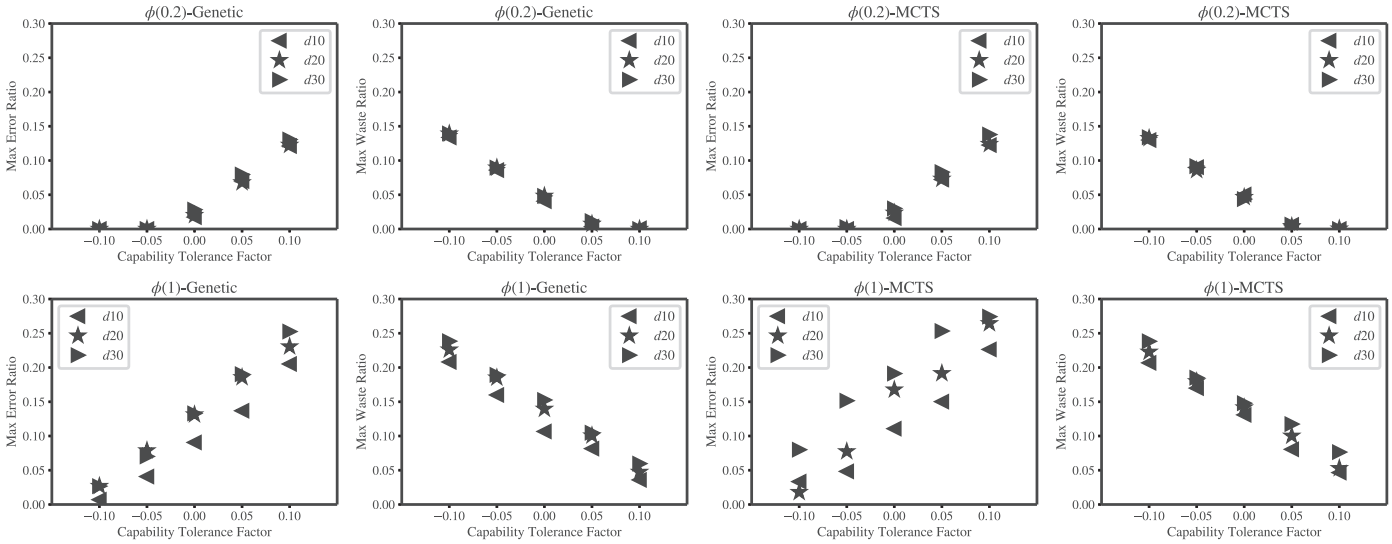


FIGURE 14. The influence of capability tolerance factor.

In Section (III-A), the nodes are considered homogeneous. With the following extensions, the model can be applied to heterogeneous cluster.

1) The nodes in a heterogeneous cluster can be categorized according to their architectures. Otherwise, the efforts for obtaining static parameters are unacceptable.
2) The capacity measurement function $z(c_i, f_i)$ should be specialized for different categories of nodes since the frequency options may not be the same.
3) In the specialized model, the power consumption estimation function should be specialized for different categories of nodes.

In general, the model's static parameters which are related to the node's architecture should be obtained according to the different architectures.

Workload prediction errors could cause SLA violation and energy wasting for both algorithms. In order to reduce impact of prediction errors, the capability tolerance factor is introduced to tune the node capability. According to the experiments, the maximum error ratio can be reduced by means of a negative capability tolerance factor for both algorithms, and the maximum waste ratio can be reduced by means of a positive capability tolerance factor. The capability tolerance factor should be tuned according to the user cases and prediction errors. In practice, the maximum error ratio should be kept lower value avoid the SLA violation.

VIII. CONCLUSION AND FUTURE WORK

At first, the energy efficiency problem in cloud systems, specially for cloud database systems, is discussed in this work. The conclusion is that the key to improve the energy efficiency of the system is to maintain the system at its maximum throughput under a given frequency.

In order to improve the energy efficiency of the system, this paper proposes a frequency selection approach with the corresponding model and algorithms based on the workload predictions and DVFS technique to cope with resource provisioning problem. The proposed algorithms include a genetic based algorithm and a monte carlo tree search based algorithm. Both algorithms have its advantages and disadvantages. The results of the experiment show that both algorithms have great scalability with reasonable accuracy (up to 99.9% and 99.6% for two algorithms respectively).

Since the prediction errors are inevitable, the robustness of the algorithms is analyzed. The prediction errors might cause SLA violation and energy wasting. By means of tuning the capability tolerance factor, the trade off between SLA violation and energy wasting can be found. In practice, the error ratio should be kept in a lower range.

In this work, we only considered the frequency selection and the workload migration for one time window. However, we did not consider the effect for multiple time windows. The optimization within multiple time windows is one of our future research direction. Meanwhile, we only considered using DVFS to solve the resource provisioning for energy aware cloud database systems. In the future work, the other resources can be considered, for example I/O resource can be introduced to further improve the energy efficiency of the system

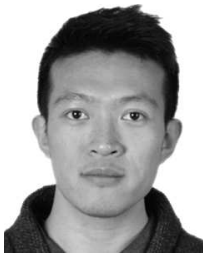
ACKNOWLEDGMENT

Experiments presented in this paper were carried out with 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 https://www.grid5000.fr).

REFERENCES

[1] M. Zakarya and L. Gillam, "Energy efficient computing, clusters, grids and clouds: A taxonomy and survey," *Sustain. Comput., Informat. Syst.*, vol. 14, pp. 13–33, Jun. 2017.

[2] M. N. Vora, "Hadoop-HBase for large-scale data," in *Proc. Int. Conf. Comput. Netw. Technol.*, Harbin, China, vol. 1, Dec. 2011, pp. 601–605.
[3] A. Thusoo *et al.*, "Hive—A petabyte scale data warehouse using hadoop," in *Proc. IEEE 26th Int. Conf. Data Eng. (ICDE)*, Long Beach, CA, USA, Mar. 2010, pp. 996–1005.
[4] J. Han, H. E, G. Le, and J. Du, "Survey on NoSQL database," in *Proc. 6th Int. Conf. Pervasive Comput. Appl.*, Port Elizabeth, South Africa, Oct. 2011, pp. 363–366.
[5] J. Song, T. Li, X. Liu, and Z. Zhu, "Comparing and analyzing the energy efficiency of cloud database and parallel database," in *Proc. 2nd Int. Conf. Comput. Sci., Eng. Appl. (ICCSEA)*, New Delhi, India, vol. 2, May 2012, D. C. Wyld, J. Zizka, and D. Nagamalai, Eds. Berlin, Germany: Springer, 2012, pp. 989–997.
[6] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis, "Towards energy-efficient database cluster design," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1684–1695, Jul. 2012.
[7] G.-W. You, S.-W. Hwang, and N. Jain, "Ursa: Scalable load and power management in cloud storage systems," *ACM Trans. Storage*, vol. 9, no. 1, pp. 1–29, Mar. 2013.
[8] D. Schall and T. Härder, "WattDB—A journey towards energy efficiency," *Datenbank-Spektrum*, vol. 14, no. 3, pp. 183–198, Sep. 2014.
[9] H.-E. Chihoub, S. Ibrahim, Y. Li, G. Antoniu, M. Pérez, and L. Bouge, "Exploring energy-consistency trade-offs in cassandra cloud storage system," in *Proc. SBAC-PAD*, Florianópolis-SC, Brazil, Nov. 2015, pp. 146–153.
[10] A. P. Florence, V. Shanthi, and C. B. S. Simon, "Energy conservation using dynamic voltage frequency scaling for computational cloud," *Sci. World J.*, vol. 2016, p. 13, 2016, Art. no. 9328070. [Online]. Available: https://www.hindawi.com/journals/tswj/2016/9328070/
[11] S. Ibrahim, T.-D. Phan, A. Carpen-Amarie, H.-E. Chihoub, D. Moise, and G. Antoniu, "Governing energy consumption in Hadoop through CPU frequency scaling: An analysis," *Future Gener. Comput. Syst.*, vol. 54, pp. 219–232, Jan. 2016.
[12] D. Gu. (2016). *Cassandra at Instagram*. [Online]. Available: https://www.slideshare.net/DataStax/cassandra-at-instagram-2016
[13] C. Guo and J.-M. Pierson, "Frequency selection approach for energy aware cloud database," in *Proc. 30th Int. Symp. Comput. Archit. High Perform. Comput.*, Lyon, France, Sep. 2018, pp. 1–8.
[14] G. Da Costa, D. Careglio, R. I. Kat, A. Mendelson, J.-M. Pierson, and Y. Sazeides, "Hardware leverages for energy reduction in large scale distributed systems," Inst. Recherche Inform. Toulouse, Tech. Rep. IRIT/RT–2010-2–FR, 2010.
[15] B. Subramaniam and W. Feng, *On the Energy Proportionality of Distributed NoSQL Data Stores*, vol. 8966. New Orleans, LA, USA: Springer, Nov. 2014, pp. 264–274, DOI: 10.1007/978-3-319-17248-4_14.
[16] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the energy efficiency of a database server," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, Indianapolis, IN, USA, 2010, pp. 231–242.
[17] D. Schall and T. Härder, "Approximating an energy-proportional DBMS by a dynamic cluster of nodes," in *Database Systems for Advanced Applications*, S. S. Bhowmick, C. E. Dyreson, C. S. Jensen, M. L. Lee, A. Muliantara, and B. Thalheim, Eds. Cham, Switzerland: Springer, 2014, pp. 297–311.
[18] D. Schall and T. Härder, "Energy-proportional query execution using a cluster of wimpy nodes," in *Proc. 9th Int. Workshop Data Manage. New Hardw. (DaMoN)*, New York, NY, USA, 2013, pp. 1–6.
[19] G. Han, W. Que, G. Jia, L. Shu, and A. Jara, "An efficient virtual machine consolidation scheme for multimedia cloud computing," *Sensors*, vol. 16, no. 2, p. 246, 2016.
[20] S. Savinov and K. Daudjee, "Dynamic database replica provisioning through virtualization," in *Proc. 2nd Int. Workshop Cloud Data Manage. (CloudDB)*, Toronto, ON, Canada, 2010, pp. 41–46.
[21] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *J. Syst. Softw.*, vol. 99, pp. 20–35, Jan. 2015.
[22] L. Yu, F. Teng, and F. Magoulès, "Node scaling analysis for power-aware real-time tasks scheduling," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2510–2521, Aug. 2016.
[23] J. Liu and J. Guo, "Energy efficient scheduling of real-time tasks on multi-core processors with voltage islands," *Future Gener. Comput. Syst.*, vol. 56, pp. 202–210, Mar. 2016.
[24] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, Mar. 2017.
[25] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Hoboken, NJ, USA: Wiley, 1990.
[26] J.-J. Chen and L. Thiele, "Energy-efficient scheduling on homogeneous multiprocessor platforms," in *Proc. ACM Symp. Appl. Comput.*, Sierre, Switzerland, 2010, pp. 542–549.
[27] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York, NY, USA: Addison-Wesley, 1989.
[28] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI in Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.
[29] F. Cappello *et al.*, "Grid'5000: A large scale and highly reconfigurable grid experimental testbed," in *Proc. 6th IEEE/ACM Int. Workshop Grid Comput.*, Nov. 2005, p. 8.
[30] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *Proc. 1st ACM Symp. Cloud Comput. (SoCC)*, Indianapolis, IN, USA, 2010, pp. 143–154.



CHAOPENG GUO received the B.S. degree in software engineering from Northeastern University, China, in 2013. He is currently pursuing the Ph.D. degree with University of Toulouse. His current research focuses on energy efficiency and energy aware cloud database.

JEAN-MARC PIERSON received the Ph.D. degree in computer science from the LIP Laboratory, École Normale Supérieure de Lyon. He has been a Full Professor with University of Toulouse, since 2006. His research interesting includes distributed systems and high-performance computing, distributed data management, security grid, cloud computing, and energy-aware distributed computing.

HUI LIU received the Ph.D. degree from Northeastern University, Shenyang, China, in 2010. She is currently a Lecture with the School of Metallurgy, Northeastern University. Her current research interests include green computing, sustainable computing, and computational fluids dynamics.

JIE SONG received the Ph.D. degree in computer science from Northeastern University, in 2008. He is currently an Associate Professor with the Software College, Northeastern University, China. His main research interests include big data, data intensive computing, and energy-efficient computing.