Frequency Selection Approach for Energy Aware Cloud Database

Chaopeng Guo, Jean-Marc Pierson

2018 30th International Symposium on Computer Architecture and High Performance Computing

Frequency Selection Approach for Energy Aware Cloud Database

Chaopeng Guo
Institut de Recherche en Informatique de Toulouse
University of Toulouse
France
chaopeng.guo@irit.fr

Jean-Marc Pierson
Institut de Recherche en Informatique de Toulouse
University of Toulouse
France
pierson@irit.fr

***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 over-provisioning. In this paper, using Dynamic Voltage and Frequency Scaling (DVFS), a frequency selection approach is introduced to improve the energy efficiency of cloud systems in terms of resource over-provisioning. In the approach, two algorithms, Genetic Algorithm (GA) and Monte Carlo Tree Search Algorithm (MCTS), are proposed. Cloud database system is taken as an example to evaluate the approach. The results of the experiments show that the algorithms have great scalability which can be applied to a 120-nodes case with high accuracy compared to optimal solutions (up to 99.9% and 99.6% for GA and MCTS respectively). According to an optimality bound analysis, 21% of energy can be saved at most using our frequency selection approach.**

***Index Terms*—Frequency Selection, Energy Efficiency, DVFS, Cloud Database, 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 data volume and the construction of data centers, the problem of energy waste becomes more serious. Energy bills for major cloud service providers are typically the second largest item in their budgets [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, for example, HBase [2], Hive [3] and Cassandra [4]. 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 over-provisioning.

Dynamic Voltage and Frequency Scaling (DVFS) [5] is an efficient technology to control power consumption of processors. By means of DVFS, power control policies can be made. 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 [6]. Therefore, it makes sense to control frequency in a fine-grained way.

To cope with the resource over-provisioning for cloud systems especially for cloud database systems, a frequency selection approach is introduced. In our approach, the frequencies of a system are chosen according to the workload predictions. Energy wasted by idle machines and violation of Service-Level Agreement (SLA) due to overloaded machines are reduced, which together improves 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 our environment, the optimal solution can be found by the entire searching within 5 hours. However, a common case has hundred nodes within the cluster. For example, the technical group from instagram claimed that their biggest cluster contains 100 nodes in Cassandra summit 2016 [7]. Therefore, searching of the optimal solution is unrealistic and proposing an approach for searching a suboptimal solution with high accuracy and good performance is the goal in this paper.

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

II. Frequency Selection Model

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 V.

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

DOI 10.1109/SBAC-PAD.2018.00026

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 $|w_i|$. 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.

$$ee = \textstyle\sum_i^n |w_i| / e \tag{2}$$

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 $|b_g|$, 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 $\mathbf{D}^r = \{b_{11}, b_{12}..., b_{1r}...b_{h1}, b_{h2}...b_{hr}\}$, in which $b_{gk} \in \mathbf{D}^r, k \leq r$ is the k^{th} replica of b_g. In order to avoid the effect of Δt's duration, 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. 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 [8]. 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 IV-A).

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. In this work, we only consider the first two types. Let $\mathbf{M}_x$ and $\mathbf{M}_y$ denote the block sets migrated within a rack and between racks, respectively. Let e_x and e_y denote the energy costs of the migration within a rack and between racks for one block unit, respectively, which are obtained through the benchmark experiment. The migration cost estimation function $mc(\widetilde{s})$ is defined by:

$$mc(\widetilde{s}) = e_x \times \sum_{b_{g_x k_x} \in \mathbf{M}_x} |b_{g_x k_x}| + e_y \times \sum_{b_{g_y k_y} \in \mathbf{M}_y} \left| b_{g_y k_y} \right| \tag{6}$$

Since the migration function m is not a focus of this paper, we do not introduce the details of the migration. m consists of two phases, block selection and block migration. Using approximation algorithms, a migration plan can be obtained within polynomial time.

III. Frequency Selection Algorithm

Two algorithms for frequency selection are introduced: Genetic Algorithm (GA) and Monte Carlo Tree Search Algorithm (MCTS). Both algorithms have their advantages and disadvantages which are discussed in Section V. 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. Using the approximation algorithms, 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 $\forall i, j \ {}^{\left(c_{f_i}^{max} - c_{f_i}^{idle}\right)}/_{z(c_i,f_i)} \geqslant {}^{\left(c_{f_j}^{max} - c_{f_j}^{idle}\right)}/_{z(c_j,f_j)}$, 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 (7). 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).

$$p^{max} = \left(z\left(c_{\widetilde{i}+1}, f_{\widetilde{i}+1}\right) - \left(\sum_{i}^{\widetilde{i}+1} z\left(c_i, f_i\right) - \sum_{i}^{n} w_i\right)\right) \times \frac{c_{f_{\widetilde{i}+1}}^{max} - c_{f_{\widetilde{i}+1}}^{idle}}{z\left(c_{\widetilde{i}+1}, f_{\widetilde{i}+1}\right)} + c_{f_{\widetilde{i}+1}}^{idle} + \sum_{i=0}^{\widetilde{i}} c_{f_i}^{max} + \sum_{i=\widetilde{i}+2}^{n} c_{f_i}^{idle} \tag{7}$$

2) Migration Cost: After the block selection phase, the blocks to be migrated are selected. However, migration cost is still unknown before the migration function m completes because the migration cost within a rack and the migration cost between racks are different. The upper bound of migration cost can be obtaine ing the relaxation that the blocks are continuous as well. 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 $\widetilde{\mathbf{M}}_p = (\mathbf{M}_p, \leqslant) = \{b_1, b_2, ...\}$ 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. $\widetilde{\mathbf{M}}_p^x$ and $\widetilde{\mathbf{M}}_p^y$ are partitions of $\widetilde{\mathbf{M}}_p$. $\widetilde{\mathbf{M}}_p^x$ and $\widetilde{\mathbf{M}}_p^y$ satisfy the following conditions:
   - $\widetilde{\mathbf{M}}_p^x = \{b_1, ..., b_q\}$, $\widetilde{\mathbf{M}}_p^y = \{b_{q+1}, b_{q+2}, ...\}$.
   - Let the set of partial nodes in γ_p that have extra capacity be $\mathbf{C}_p^{in} = \{c_i | c_i \in \gamma_p \text{ and } w_i < z(c_i, f_i)\}$. Then, the index q of $\widetilde{\mathbf{M}}_p$ satisfies $\sum_{j=0}^{q} l \times \varphi_j \leqslant \sum_{c_i \in \mathbf{C}_p^{in}} (z(c_i, f_i) - w_i) < \sum_{j=0}^{q+1} l \times \varphi_j$.

With above conditions, the maximum migration cost can be obtained by Equation (8), 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.

$$mc^{max} = e_x \times \sum_{b_x \in \bigcup_p^u \widetilde{\mathbf{M}}_p^x} |b_x| + e_y \times \sum_{b_y \in \bigcup_p^u \widetilde{\mathbf{M}}_p^y} |b_y| \tag{8}$$

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 IV-B) respectively and the execution time for each case is shown in Table (I). 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 IV-B3).

TABLE I: Evaluation of Model Simplification

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

B. Genetic Algorithm

Genetic Algorithm (GA) is a type of algorithms for randomly searching suboptimal solutions, which is guided by evaluation and natural genetics [9]. 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

Algorithm 1 Fitness Function of Genetic Algorithm

```
1: function EVALUATE(chromosome)
2:     F ← DECODE(chromosome)
3:     p^max ← MAXPOWERCONSUMPTION(F)
4:     mc^max ← 0
5:     for p ← [1, ..., u] do
6:         M_p^x, M_p^y ← SELECTMIGRATIONBLOCKS(γ_p)
7:         mc^max ← mc^max + MIGRATIONCOST(M_p^x, M_p^y)
8:     end for
9:     return p^max × |Δt| + mc^max
10: end function
```

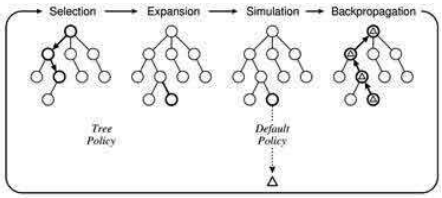

Fig. 1: One iteration of the general MCTS approach

parts of GA—encoding and evaluation—are introduced. The influence of parameters of GA is discussed in Section IV.

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 $< I_\eta(f_1), I_\eta(f_2), ..., I_\eta(f_n) >$.

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 (7) (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 (8). 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 [10].

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 $policy$ 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 (2) 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 (2), $< 1, 2, 2 >$ is one of the frequency vectors. The task of MCTS is to find the frequency vector which produces the minimum energy consumption.

Figure (1) from the survey [10] 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 (9), 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.

$$UCT = \frac{Q(v')}{N(v')} + C\sqrt{\frac{2\ln N(v)}{N(v')}} \tag{9}$$

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 (10), is adopted. The difference between Equation (9) and (10) 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 $1-(Q(v')/e^{max})$, the value is converted within 0 and 1. The highest value of $1-(Q(v')/e^{max})$ indicates the path with the minimum energy consumption.

$$UCT = \frac{1-(Q(v')/e^{max})}{N(v')} + C\sqrt{\frac{2\ln N(v)}{N(v')}} \tag{10}$$

In Equation (10), 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.

IV. Experiment

A. Maximum throughput benchmark

In this experiment, the maximum throughput under each frequency of a cloud database system is measured. A Cassandra system with one single node is established in Nancy site of Grid5000 testbed [11]. The benchmark is Yahoo! Cloud Serving Benchmark(YCSB) [12]. A couple of sample workloads (95% read ratio and 5% update ratio) with different workload sizes are executed on the system. In the experiment, more and more requests are loaded onto the system to test the maximum throughput of the node.

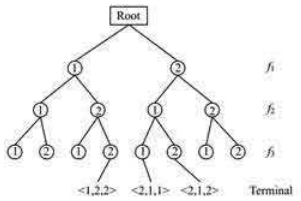

Fig. 2: Frequency Selection Tree

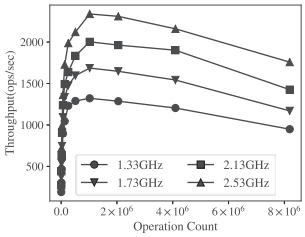

Fig. 3: Relationship between Operation Count and Throughput

TABLE II: Maximum Throughput

Frequency	Maximum Throughput	Frequency	Maximum Throughput
2.53GHz	2339 opt/sec	1.73GHz	1686 opt/sec
2.40GHz	2214 opt/sec	1.60GHz	1557 opt/sec
2.13GHz	2002 opt/sec	1.33GHz	1322 opt/sec
2.00GHz	1884 opt/sec	1.20GHz	1205 opt/sec

Figure (3) shows the relationship between throughputs and operation counts. With the increment of operation counts, system throughput increases rapidly at first and declines steadily afterwards. More requests lead to higher request throughput. When system throughput is lower than request throughput, system throughput increases to meet requests. However, when request throughput exceeds the capacity of the node, system throughput declines due to resource limitation and operation failure. Figure (3) shows that a maximum throughput value exists under each frequency option. The maximum throughputs for all frequency options are shown in Table (II).

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 $(d10, 1800, GA)$ indicates a test case, in which GA is applied to dataset $d10$ and the workload is set to 1800 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 1800 opt/sec by default if not otherwise specified, for example case $(d10, 1800, GA)$ 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

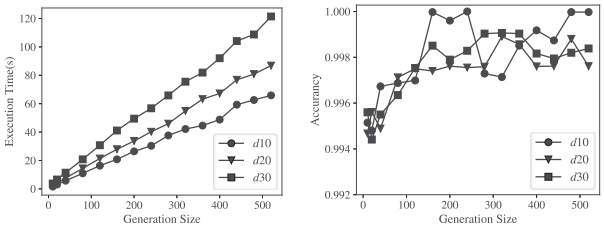

Fig. 4: The Influence of Generation Size on Accuracy

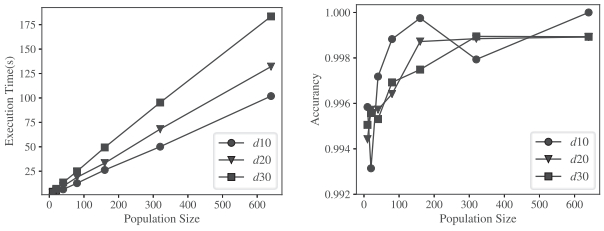

Fig. 5: The Influence of Population Size on Accuracy

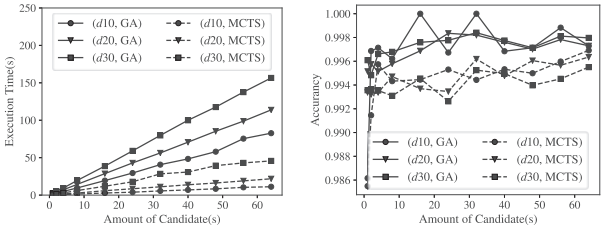

Fig. 6: The Influence of Amount of Candidates on Accuracy

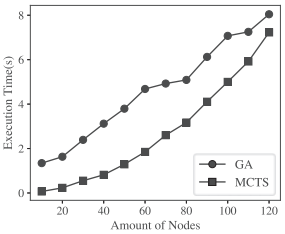

Fig. 7: Scalability of the Frequency Selection Algorithm

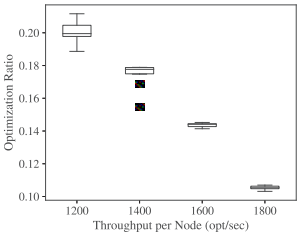

Fig. 8: Optimization Bound of the Frequency Selection Algorithm

$T(case)$. The accuracy of a case $A(case)$ is defined by Equation (11) in which $d \in [d10, d20, d30]$ and $a \in [GA, MCTS]$.

$$A(d,a) = 1 - \frac{(E(d,a) - E(d,Optimal))}{E(d,Optimal)} \tag{11}$$

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 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 IV-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 (2) 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 situation, 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. 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.

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 (12) 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 1200 opt/sec, 1400 opt/sec, 1600 opt/sec and 1800 opt/sec.

$$O(d, l, a) = \frac{E(d, l, Performance) - E(d, l, a)}{E(d, l, Performance)} \quad (12)$$

The result is shown in Figure (8). The optimization ratio depends on the value of throughput per node. For $\forall d \in \{d10, d20, ..., d120\}$ $\forall l_1 > l_2$ $O(d, l_1, GA) < O(d, l_2, GA)$. With the same value of throughput per node, the optimization ratios are concentrated. With the increment of throughput per node, the optimization ratio increases. The maximum optimization ratio is 21% for the case $(d80, 1200, GA)$, and the minimum optimization ratio is 10% for the case $(d100, 1800, GA)$. 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 1205 opt/sec and the node is set to the performance mode. Therefore, $O(s1, 1205) = \frac{p(<2.53Ghz,1205>)-p(<1.20Ghz,1205>)}{p(<2.53Ghz,1205>)} = \frac{c^{idle}_{2.53Ghz}+\frac{1205}{2339}\times(c^{max}_{2.53Ghz}-c^{idle}_{2.53Ghz})-c^{max}_{1.20Ghz}}{c^{idle}_{2.53Ghz}+\frac{1205}{2339}\times(c^{max}_{2.53Ghz}-c^{idle}_{2.53Ghz})} = 22.89\%$. However, there exists block migrations in the real case, which consume energy.

E. Comparison with Hot-N-Cold

Houssem-Eddine Chihoub *et al.* [13] 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 1200 opt/sec, 1400 opt/sec, 1600 opt/sec and 1800 opt/sec respectively.

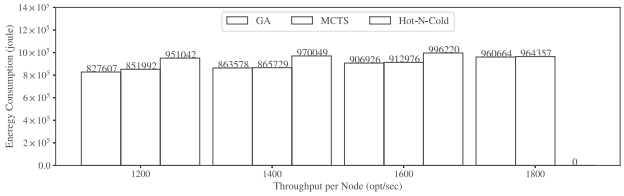

Fig. 9: Comparison with Hot-N-Cold

The results are shown as Figure (9). 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 10.9%, and the average improvement of MCTS is 9.8%. When the value of the throughput on each node is set to 1800 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 35440 opt/sec, however with the setting 1800 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.

V. 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 can be applied to a large cluster which contains 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.
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.

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

1) The nodes in an 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.

VI. Related Work

A variety of the past researches have dealt with improving the energy efficiency in cloud systems. Georges Da Costa *et al.* [14] gave a lot of possible techniques of reducing energy consumption in large scale distributed systems. Willis Lang *et al.* [15] studied the interaction among energy management, load balancing, and replication strategies for the data-intensive cluster computing. Willis Lang *et al.* [16] gave guiding principles for building up the energy-efficient cloud DBMS with query properties and scalability taken into account. Balaji Subramaniam *et al.* [17] measured the power consumption and the performance of a Cassandra cluster, and used power and resource provisioning techniques to improve the energy proportionality. Daniel Schall *et al.* [18] designed and implemented WattDB, which is a distributed DBMS that dynamically adjusts itself switching nodes on and off to the present workload and reconfigures itself to satisfy the performance demands. Prasad Saripalli *et al.* [19] studied the loading algorithm for the cloud platform to support the sampling requirements, the measurement and the characterization for the load prediction. Houssem-Eddine Chihoub *et al.* [13] 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.

Compared with above literatures, we use DVFS technique to cope with the resource over-provisioning problem for cloud systems especially for cloud database systems by means of GA and MCTS, and as a result the energy efficiency of the system is improved as well. According to Section (IV-E), compared with Hot-N-Cold approach [13], frequency selection approach has a better performance (10.9% improvement from GA and 9.8% improvement from MCTS) and flexibility.

VII. Conclusion and Future Work

In this paper, a frequency selection approach with the corresponding model and algorithms is proposed to cope with the resource over-provisioning problem by means of Genetic algorithm and Monte Carlo Tree Search algorithm. The results of the experiments show that the corresponding algorithms have good scalability which can be applied to a 120-nodes case with a reasonable accuracy (up to 99.9% and 99.6% for GA and MCTS respectively).

In the future work, the first direction is to optimize the algorithms to improve their scalability. Moreover, more interesting findings can be found if the energy consumption in the whole life-cycle is taken into account.

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).

This research also is supported by the scholarship from China Scholarship Council (CSC, see http://en.csc.edu.cn) under the Grant CSC N201506080029.

References

[1] M. Zakarya and L. Gillam, "Energy efficient computing, clusters, grids and clouds: A taxonomy and survey," *Sustainable Computing: Informatics and Systems*, vol. 14, pp. 13–33, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210537917300707

[2] M. N. Vora, "Hadoop-HBase for large-scale data," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, vol. 1. Harbin, China: IEEE, Dec. 2011, pp. 601–605.

[3] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, and R. Murthy, "Hive - a petabyte scale data warehouse using Hadoop," in *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*. Long Beach, California, United State: IEEE, Mar. 2010, pp. 996–1005.

[4] J. Han, H. E, G. Le, and J. Du, "Survey on NoSQL database," in *2011 6th International Conference on Pervasive Computing and Applications*. Port Elizabeth, South Africa: IEEE, Oct. 2011, pp. 363–366.

[5] A. P. Florence, V. Shanthi, and C. B. S. Simon, "Energy Conservation Using Dynamic Voltage Frequency Scaling for Computational Cloud," *The Scientific World Journal*, vol. 2016, p. 13, 2016. [Online]. Available: http://dx.doi.org/10.1155/2016/9328070

[6] 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 Generation Computer Systems*, vol. 54, pp. 219 – 232, 2016.

[7] D. Gu, "Cassandra at Instagram 2016," 2016. [Online]. Available: https://www.slideshare.net/DataStax/cassandra-at-instagram-2016

[8] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *Journal of Network and Computer Applications*, vol. 82, pp. 93–113, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517300231

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. New York, USA: Addison-Wesley, 1989.

[10] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, Mar. 2012.

[11] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard, "Grid'5000: a large scale and highly reconfigurable grid experimental testbed." Seattle, USA: IEEE, Nov. 2005, p. 8 pp.

[12] B. Cooper, *Yahoo! Cloud Serving Benchmark*, Jul. 2010. [Online]. Available: https://github.com/brianfrankcooper/YCSB

[13] H.-E. Chihoub, S. Ibrahim, Y. Li, G. Antoniu, M. Perez, and L. Bouge, "Exploring Energy-Consistency Trade-Offs in Cassandra Cloud Storage System." Florianópolis-SC, Brazil: IEEE, Nov. 2015, pp. 146–153.

[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," IRIT, Technical Report IRIT/RT–2010-2–FR, 2010.

[15] W. Lang, J. M. Patel, and J. F. Naughton, "On energy management, load balancing and replication," *ACM SIGMOD Record*, vol. 38, no. 4, pp. 35–42, Dec. 2009.

[16] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis, "Towards Energy-Efficient Database Cluster Design," *Proceedings of the VLDB Endowment*, vol. 5, pp. 1684–1695, Jul. 2012.

[17] 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.

[18] D. Schall and T. Härder, "WattDB - A Journey towards Energy Efficiency," *Datenbank-Spektrum*, vol. 14, no. 3, pp. 183–198, Sep. 2014.

[19] P. Saripalli, G. Kiran, R. S. R, N. Narware, and N. Bindal, "Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing." Melbourne, Victoria, Australia: IEEE, Dec. 2011, pp. 397–402.