Hot-N-Cold model for energy aware cloud databases

Chaopeng Guo, Jean-Marc Pierson, Jie Song, Christina Herzog

Hot-N-Cold model for energy aware cloud databases

Chaopeng Guo [a,*], Jean-Marc Pierson [a], Jie Song [b], Christina Herzog [c]
[a] IRIT, University Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse CEDEX 9, France
[b] Software College, Northeastern University, No. 159 ChuangXin Road, HunNan District, ShenYang, 110169, LiaoNing, China [c] LIUPPA, Université de Pau et des Pays de l'Adour, Bayonne & Efficit, France

HIGHLIGHTS

- The energy efficiency in cloud database systems is discussed.
- The resource-providing problem in cloud database systems is considered as two bounded problems which are studied in this work.
- Under Hot-N-Cold model, nonlinear programming algorithm and multiphase algorithm are proposed to solve the bounded problems.
- The simulation and real case experiments are designed and implemented to measure the proposed algorithms in aspect of performance and assurance.

ABSTRACT

A lot of cloud computing and cloud database techniques are adopted in industry and academia to face the explosion of the arrival of the big data era. Meanwhile, energy efficiency and energy saving become a major concern in data centers, which are in charge of large distributed systems and cloud databases. However, the phenomenon of energy wasting is related to resource provisioning. Hot-N-Cold model is introduced in this paper, which uses workload predictions and DVFS(Dynamic Voltage and Frequency Scaling) to cope with the resource provisioning problem within energy aware cloud database systems. In this model, the resource provisioning problem is considered as two bounded problems. A nonlinear programming algorithm and a multi-phase algorithm are proposed to solve them. The experimental results show that one of the proposed algorithms has great scalability which can be applied to a cloud database system deployed on 70 nodes. Using Hot-N-Cold model can save up to 21.5% of the energy of the running time.

Keywords:
Energy efficiency
Hot-N-Cold
DVFS
Cloud database
Bounded problem

1. Introduction

In face of Big Data challenges, an increasing number of data centers are constructed. Cloud database systems are developed to meet the users rapidly growing data query needs, for example, HBase [21], Hive [19], Cassandra [8] and MongoDB [2]. With the explosive growth of data volume and construction of data centers, the problem of energy wasting becomes more and more serious. It becomes a major task to improve energy efficiency in cloud systems. It is estimated in [25] that the cost of powering and cooling accounts for 53% of the total operational expenditure of data centers, and the energy bills for major cloud service providers are typically the second largest item in their budgets [24]. However, few researches focus on the problem of energy efficiency in cloud database system.

Typical cloud databases waste energy. Workloads of a cloud database system vary with time, because users' activities are dynamic. For example users' activities are more intense at day time, whereas they do little at night. In this situation, energy is wasted if the configuration of the system at night remains the same as the one during daytime. Sometimes workloads are unbalanced, in which part of system is occupied, whereas the other part remains idle. In this situation, energy is wasted because the workloads are not well scheduled accordingly. The energy wasting mentioned above comes from resource provisioning, namely there is a gap between resource needed and resource provided.

In modern Linux operation systems, there are five different power schemes (governors) available to dynamically scale CPU frequency according to CPU utilization. However, dynamic tuning strategies only use the system's running information, which does not reflect the state of the system lifecycle. Meanwhile the current power schemes do not exactly match their design goal and may even become ineffective at improving power consumption when dealing with cloud databases [9]. Therefore, it makes sense to control the frequency in a fine-grained way.

Energy efficiency in cloud database systems is a ratio between workloads and energy consumption [17]. The workload in cloud database systems refers to users requests. The resource provisioning of energy aware cloud database systems mainly has two

* Corresponding author.
E-mail address: chaopeng.guo@irit.fr (C. Guo).

https://doi.org/10.1016/j.jpdc.2018.09.012

situations in terms of energy efficiency. (1) When the system is provided with more resources than needed, energy is wasted because too much resources are provided but few used to response users' requests, which leads to lower energy efficiency. (2) When the system is assigned a heavy workload, so that the amount of wasted energy is reduced but a few is used to deal with the resource competition, which causes lower energy efficiency as well. By means of a benchmark experiment, Section 4.1, two conclusions are learned about the energy efficiency in cloud database systems. (1) Energy efficiency of cloud database systems has positive relation with the system throughput. When the system achieves a higher throughput, it has better energy efficiency. (2) The cloud database system has a throughput peak under a given frequency configuration. Based on the conclusions, Hot-N-Cold model and corresponding algorithms are proposed in this paper to solve the resource provisioning problem within an energy aware cloud database system, and meanwhile improve the energy efficiency of the system.

To avoid energy wasting, we propose Hot-N-Cold model in this work, which takes advantage of DVFS technique to select suitable frequencies to the system according to the workload predictions. Meanwhile, when some nodes are overloaded a migration strategy is introduced in the Hot-N-Cold model to balance the workloads of the system. By means of frequency selection and workload migration, the system's energy efficiency is improved. In terms of the migration, we only focus on the migration plan generation process in this work. A migration plan defines which part of the workloads should be migrated and where the workloads should be migrated. Therefore, we do not focus on the technique detail of migration execution.

The advantage of Hot-N-Cold model is that the model uses two approaches, frequency selection and workload migration, to improve the energy efficiency of the system. However, the challenge of Hot-N-Cold model is the performance of the corresponding algorithms. For example considering a cluster consists of n nodes and every node has m frequency options. Then the total amount of frequency combinations is m^n. For each frequency combination, the migration process should be applied to evaluate the frequency selection result and the migration plan. In a small case, with 3 nodes and 2 frequency options, there are 8 frequency combinations and correspondingly there are 8 migration plans, and the one that most contributes to improve the system's energy efficiency can be obtained. However, in a real database system, the solution space is much bigger. The huge solution space bring a performance challenge to the corresponding algorithms.

The rest of this paper is organized as following. After this introduction section, Section 2 introduces the Hot-N-Cold model, gives the mathematical definitions and proposes two bounded problems under Hot-N-Cold model. Section 3 proposes 2 algorithms to solve the bounded problems. Section 4 evaluates the proposed algorithms and carries out some discussions. Section 5 reviews related works. Section 6 gives some discussion about Hot-N-Cold model and Section 7 concludes and points out future work directions.

2. Hot-N-Cold model

This section introduces the mathematical definitions of the Hot-N-Cold model. Based on the model, the corresponding constraints and objectives are given.

2.1. Model definition

Cluster and nodes. A cluster $\mathbf{C}$ is a set consisting of n nodes, namely $\mathbf{C} = \{c_1, \ldots, c_n\}$, in which c_i represents the node i. To simplify the description, the nodes are taken homogeneous, in which the nodes are regarded to have the same architecture and hardware configuration. The extension of the model to heterogeneous nodes is discussed in Section 6.

There are m available frequency options for the CPU. The idle power consumption and the maximum power consumption of the node under a frequency f are denoted as c_f^{idle} and c_f^{max} respectively. If a fraction ω of the CPU is used under f, the power consumption p_f can be obtained by Eq. (1). The maximum throughput that can be reached by node c_i under a frequency f is defined as its capacity. Let the capacity measurement function be $z(c_i, f)$. Without loss of generality, the following assumptions are made. With $u, v \in [1, m]$, $u > v$ then $c_{f_u}^{idle} > c_{f_u}^{idle}$ and $c_{f_v}^{max} > c_{f_v}^{max}$. Meanwhile, with $i \in [1, n]$, $z(c_i, f_u) > z(c_i, f_v)$. Since the nodes are homogeneous, when two nodes are assigned with the same frequency f, their capabilities are same, namely $z(c_i, f) = z(c_j, f)$.

$$p_f = c_f^{idle} + \omega \times (c_f^{max} - c_f^{idle}) \tag{1}$$

Time windows. Cloud database systems can be treated as long-term servicing systems for the reason that the system should be available at all the time for users. To simplify the modeling, the whole running time of a cloud database system is taken as window series time. Let Δt be the time interval. In practice, the size of the time window should fit the users' request activities. The principles of dividing running time are:

- Within a time window, the users' activities should be stable and the frequency of accessing the cloud database should be uniform.
- Between time windows, the frequency of the users' activities could be different.

Dataset. In a cloud database system, a dataset $\mathbf{D}$ is divided into h data blocks, namely $\mathbf{D} = \{b_1, b_2, \ldots, b_h\}$, in which the size of block b_g is denoted as $|b_g|$. In order to meet data integrity and fault-tolerance requirements, the cloud database uses a replication factor r to control the number of replicas of data blocks. The dataset with replication factor r is denoted as $\mathbf{D}^r = \{b_{1,1}, \ldots, b_{1,r} \ldots b_{h,1}, \ldots, b_{h,r}\}$. At the beginning, $\mathbf{D}^r$ is evenly distributed. However, with the system running, the probability of each data block being accessed is changing. $\varphi_{g,k}$ is the predicted probability of block $b_{g,k}$ being accessed in Δt.

Workload. Workload refers to user's requests in cloud database systems. Since the system's whole running time is taken as the time window series, the workload in this paper is defined by the throughput of the system to avoid the question of the length of the time window. Let the throughput of the system be l and the workloads of c_i be denoted as ψ_i, which is shown by Eq. (2). In the equation, $w_{g,k}^i$ is a binary indicator, which equals 1 only when $b_{g,k}$ is assigned to c_i.

$$\psi_i = \sum_{g=1}^{h} \sum_{k=1}^{r} \left(w_{g,k}^i \times l \times \varphi_{g,k} \right)$$
$$w_{g,k}^i = \begin{cases} 1 \text{ If } b_{g,k} \text{ is assigned to } c_i \\ 0 \text{ If } b_{g,k} \text{ is not assigned to } c_i \end{cases} \tag{2}$$

Energy yield. The energy efficiency of cloud data systems is defined as the ratio between its requests and energy consumption [17]. For the reason that energy consumption cannot be obtained before the end of Δt, energy yield is used to evaluate the node's energy efficiency. The energy yield of c_i under a frequency f is defined as 1 only when its throughput reaches its capacity. Therefore, the energy yield value is the ratio between its workload ψ_i and its capacity $z(c_i, f)$. Energy yield y_i of c_i in Δt is defined

by Eq. (3). In the equation, x_u^i is a binary indicator, which equals 1 only when f_u is assigned to c_i.

$$y_i = \frac{\psi_i}{\sum_{u=1}^{m} x_u^i \times z(c_i, f_u)}$$
$$x_u^i = \begin{cases} 1 \text{ If } c_i \text{ is assigned with } f_u \\ 0 \text{ If } c_i \text{ is not assigned with } f_u \end{cases} \quad (3)$$

The range of y_i is [0, 1]. $y_i = 0$ indicates that there is no data being accessed in c_i during Δt, while $y_i = 1$ indicates that the node reaches its capacity. As already discussed, if the system is overloaded and attempts to exceed its capacity, the requests would take too much time due to resource competition. As a consequence, the expected throughput is unreachable. Therefore, y_i cannot exceed 1.

Power consumption. Power consumption p_f for a node under frequency f is defined by Eq. (1). However, the CPU utilization parameter ω cannot be obtained before the end of Δt. Therefore the CPU usage is defined as 1 only when its energy yield equals 1. Combined with Eq. (3), the power consumption of c_i is denoted as p_i and given by Eq. (4). The system power consumption is denoted as P, which is given by Eq. (5).

$$p_i = \sum_{u=1}^{m} \left(x_u^i \times c_{f_u}^{idle}\right) + y_i \times \sum_{u=1}^{m} \left(x_u^i \times \left(c_{f_u}^{max} - c_{f_u}^{idle}\right)\right) \quad (4)$$
$$P = \sum_{i=1}^{n} p_i \quad (5)$$

Migration cost. In order to achieve energy efficiency for each node in Δt, some data blocks need to be migrated from one node to another when the nodes are overloaded. However, the migration of data blocks costs energy. Let T denote the energy cost of data migration. In the cloud environment, according to network topology, there are mainly 3 types of migrations: migration within a rack, migration between racks and migration between data centers. This paper only focuses on the first two types. Let $\mathbf{M}_{\text{In}}$ and $\mathbf{M}_{\text{Out}}$ denote the sets of blocks migrated within a rack and the set of blocks migrated 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. Then the migration cost is defined by Eq. (6). In Eq. (6). $\sum_{b_{g,k} \in \mathbf{M}_{\text{In}}} |b_{g,k}|$ and $\sum_{b_{g,k} \in \mathbf{M}_{\text{Out}}} |b_{g,k}|$ show the total of data sizes migrated within a rack and between racks respectively.

$$T = E_{\text{In}} \times \sum_{b_{g,k} \in \mathbf{M}_{\text{In}}} |b_{g,k}| + E_{\text{Out}} \times \sum_{b_{g,k} \in \mathbf{M}_{\text{Out}}} |b_{g,k}| \quad (6)$$

2.2. Notation summary

Hot-N-Cold model is defined in Section 2.1. Some notations are used in the model's definition. Table 1 lists the specifications of the symbols used in the equations, which is used in the rest of paper as well.

2.3. Constraints

According to the descriptions in previous section, some constrains are shown as below:

$$\forall g \in [1, h], \forall k \in [1, r], \forall i \in [1, n] \quad w_{g,k}^i \in \{0, 1\} \quad (7)$$
$$\forall g \in [1, h], \forall k \in [1, r] \quad \sum_{i=1}^{n} w_{g,k}^i = 1 \quad (8)$$
$$\forall i \in [1, n], \forall g \in [1, h] \quad \sum_{k=1}^{r} w_{g,k}^i \leq 1 \quad (9)$$

Table 1
Specifications of the symbols.

Symbol	Comment
l	The predicted system throughput in Δt
$\varphi_{g,k}$	The predicted accessed probability of block $b_{g,k}$
$w_{g,k}^i$	A binary indicator which equals 1 only when $b_{g,k}$ is assigned to node c_i
x_u^i	A binary indicator, which equals 1 only when c_i is configured with frequency f_u
ψ_i	The workload (throughput) of node c_i
$\mathbf{M}_{\text{In}}$	The set of blocks migrated within a rack
$\mathbf{M}_{\text{Out}}$	The set of blocks migrated between racks
$z(c_i, f)$	The capacity (maximum throughput) of c_i under frequency f

$$\forall i \in [1, n], \forall u \in [1, m] \quad x_u^i \in \{0, 1\} \quad (10)$$
$$\forall i \in [1, n] \quad \sum_{u=1}^{m} x_u^i = 1 \quad (11)$$
$$\mathbf{M}_{\text{In}} \cap \mathbf{M}_{\text{Out}} = \emptyset \quad \sum_{b_{g,k} \in \mathbf{M}_{\text{In}}} |b_{g,k}| + \sum_{b_{g,k} \in \mathbf{M}_{\text{Out}}} |b_{g,k}| \leq \mathbb{B} \quad (12)$$
$$\forall g \in [1, h], \forall k \in [1, r] \quad \varphi_{g,k} \in [0, 1], \sum_{g=1}^{h} \sum_{k=1}^{r} \varphi_{g,k} = 1 \quad (13)$$
$$\forall i \in [1, n], \quad y_i <= 1 \quad (14)$$

Constraints (7)–(9) show the rules of block assignment. Constraints (7) and (8) indicate that each data block should be assigned to only one node. Constraint (9) means that if two data blocks are replicas of the same block, they cannot be assigned to the same node.

Constraints (10) and (11) show the rules of frequency selection for each node. Constraint (10) indicates that every node should be configured or not configured to a given frequency. Constraint (11) shows that every node should be under only one frequency.

Constraint (12) shows that a block can only be migrated within a rack or migrated between racks. $\mathbb{B}$ is the maximum data size not causing network traffic congestion.

Constraint (13) shows that the sum of the probabilities of all blocks being accessed should equal to 1, and meanwhile at least one block is accessed during Δt.

Constraint (14) shows that the throughput cannot exceed the capability of the node.

2.4. Objectives

This section focuses on the objectives of Hot-N-Cold model. At first two basic objectives are proposed. Then by means of combination basic objectives, the bounded problems are proposed.

2.4.1. Basic objectives

The energy efficiency of cloud database systems is a ratio between the workloads and the energy consumption. Reducing energy consumption is a way to improve the energy yield when the workloads are constant. As the workloads change, the nodes' frequencies are tuned correspondingly. In order to adapt to the new configuration of frequencies, a migration process is introduced. Therefore, there are two objectives in this paper. One is related to the power consumption, and the other is related to the migration cost.

The system's power consumption is defined by Eq. (5). The power consumption objective is shown by Eq. (15), in which $\mathbb{P}$ is the maximum power consumption needed to satisfy the system's need. The objective is to minimize $\mathbb{P}$.

$$P \leq \mathbb{P} \quad (15)$$

The migration cost is defined by Eq. (6). Intuitively, Eq. (16) is treated as the migration cost objective, in which $\mathbb{T}$ represents the maximum energy needed to complete the migration process. The objective is to minimize $\mathbb{T}$.

$$T \leq \mathbb{T} \tag{16}$$

2.4.2. Bounded problems

Bounded problems appear when users set constraints to objectives. In general, there are 2 types of bounded problems in Hot-N-Cold Model. One is the bounded power consumption problem, the other one is the bounded migration cost problem.

The power consumption problem. The power consumption constraint $\mathbb{P}^b$ refers to the system's maximum power consumption. With this constraint, the objective is simplified into minimizing the migration cost which is denoted as $\mathbb{P}^b\mathbb{T}^{min}$. In other words, $\mathbb{P}^b\mathbb{T}^{min}$ represents the scenario that minimizes migration cost within a system's maximum power consumption.

The migration cost problem. The migration cost constraint $\mathbb{T}^b$ refers to the maximum migration cost. With this constraint, the objective is simplified into minimizing the system's power consumption which is denoted as $\mathbb{T}^b\mathbb{P}^{min}$. In other words, $\mathbb{T}^b\mathbb{P}^{min}$ represents the lowest achievable power consumption within the migration cost requirement.

3. Algorithms

In this section, 2 algorithms are proposed to solve the bounded problems under Hot-N-Cold model.

3.1. Nonlinear programming algorithm

The bounded problems can be treated as mixed integer nonlinear programming problems. Using linear programming solver, Gurobi [12], the optimal solution can be obtained. Since the mixed integer nonlinear programming problem is a NP-hard problem, it cannot be solved in polynomial time. Therefore the nonlinear programming algorithm can only be applied to small scale problems. As described above, there are 2 bounded problems, $\mathbb{P}^b\mathbb{T}^{min}$ and $\mathbb{T}^b\mathbb{P}^{min}$. For the nonlinear programming algorithm, there are 2 types of objective function ($\mathbb{P}^{min}$, $\mathbb{T}^{min}$) and 2 different types of constraint functions, $\mathbb{P}^b$, $\mathbb{T}^b$. Due to Gurobi's properties, the objective functions need to be changed mathematically to meet the solver's requirements.

Eq. (17) gives a method to compute the power consumption P. Specially, the parameter x_u^i is moved from the denominator to the numerator inside the fraction which is different from Eq. (3). The reason for this movement is that Gurobi cannot manipulate parameters which are denominators. However, from a mathematic point of view, it is possible. The binary indicator x_u^i is to determine a frequency f_u for c_i. Therefore $\sum_{u=1}^{m} x_u^i \times z(c_i, f_u)$ is determined and constant when a certain frequency f_u is configured to c_i. The element $\psi_i/z(c_i,f_u)$ describes the node's energy yield under frequency f_u, otherwise both x_u^i and the energy yield are equal to 0.

$$st.P = \sum_{i=1}^{n}\left(\sum_{u=1}^{m}\left(x_u^i \times c_{f_u}^{idle}\right) + \sum_{u=1}^{m}\left(x_u^i \times \frac{\psi_i \times \left(c_{f_u}^{max} - c_{f_u}^{idle}\right)}{z(c_i, f_u)}\right)\right) \tag{17}$$

$$W_{g,k}{}^{-1} = \sum_{i=1}^{n}\left(i \times w_{g,k}^{i}{}^{-1}\right) \quad W_{g,k} = \sum_{i=1}^{n}\left(i \times w_{g,k}^{i}\right)$$

$$m_{g,k} = \begin{cases} 0 \text{ if } W_{g,k}{}^{-1} = W_{g,k} \\ 1 \text{ if } W_{g,k}{}^{-1} \neq W_{g,k} \end{cases}$$

$$m_{g,k}^{\text{Out}} = \begin{cases} 0 \text{ if } \left\lfloor {W_{g,k}{}^{-1}}/{\gamma} \right\rfloor = \lfloor {W_{g,k}}/{\gamma} \rfloor \\ 1 \text{ if } \left\lfloor {W_{g,k}{}^{-1}}/{\gamma} \right\rfloor \neq \lfloor {W_{g,k}}/{\gamma} \rfloor \end{cases}$$

$$st.T = \sum_{g=1}^{h}\sum_{k=1}^{r} m_{g,k} \times \left(m_{g,k}^{\text{Out}} \times E_{\text{In}} + \left(1 - m_{g,k}^{\text{Out}}\right) \times E_{\text{Out}}\right) \times \left|b_{g,k}\right| \tag{18}$$

To compute the migration cost, the comparison between two data block layouts is needed. A data block layout refers to the parameters $w_{g,k}^i$. If block $b_{g,k}$ is placed on node c_i, then $w_{g,k}^i = 1$ otherwise $w_{g,k}^i = 0$. Eq. (18) describes the method of computing the migration cost in the nonlinear programming algorithm. In Eq. (18), some new parameters are introduced. $w_{g,k}^{i}{}^{-1}$ denotes previous data block layout. $W_{g,k}$ indicates which node is assigned with which block $b_{g,k}$, and $W_{g,k}{}^{-1}$ indicates the location of the block in previous time window. $m_{g,k}$ and $m_{g,k}^{\text{Out}}$ are binary indicators as well. If block $b_{g,k}$ is moved, $m_{g,k}$ equals 1, otherwise $m_{g,k}$ equals 0. $m_{g,k}^{\text{Out}}$ indicates whether block $b_{g,k}$ is migrated between racks. If block $b_{g,k}$ is moved between racks, $m_{g,k}^{\text{Out}}$ equals 1. In Eq. (18), the amount of nodes for each rack is denoted as γ which is used to compute $m_{g,k}^{\text{Out}}$. If the amount of nodes for each rack is different, some virtual nodes can be added to cope with this problem. The corresponding constraints should be introduced to avoid the migration to the virtual nodes.

Eqs. (17) and (18) are added constraints to the constraints set for applying the nonlinear programming algorithm. In this way, the bounded problem $\mathbb{P}^b\mathbb{T}^{min}$ can be solved through adding the constraint $P \leq \mathbb{P}^b$ and using $\min T$ as the objective function. With the same approach, the bounded problem $\mathbb{T}^b\mathbb{P}^{min}$ can be solved through adding the constraint $T \leq \mathbb{T}^b$ and using $\min P$ as the objective function. By combining Eqs. (17) and (18), the bounded problems can be solved by nonlinear programming algorithm.

3.2. Multi-phases algorithm

The nonlinear programming algorithm has an outstanding advantage to obtain the optimal solution for the bounded problems. Due to the complexity of the algorithm, the bottleneck is its performance for large scale problems. In order to overcome this bottleneck, a multi-phases algorithm is proposed. The algorithm includes several phases shown as below.

1. Frequency Selection Phase
2. Frequency Assignment Phase
3. Block Selection Phase
4. Block Assignment Phase

The solution in the multi-phase algorithm includes a frequency assignment (a frequency combination and its assignment), a migration plan (a migrated block set and its migration destination). When a solution is generated, it can be evaluated according to Eqs. (5) and (6).

In first phase of the multi-phase algorithm, the frequency combinations which provided enough resources according to the workload predictions are selected. The following phases evaluate every frequency combination. Finally, the best solution (combining 4 phases) is chosen according to the bounded problem at hand.

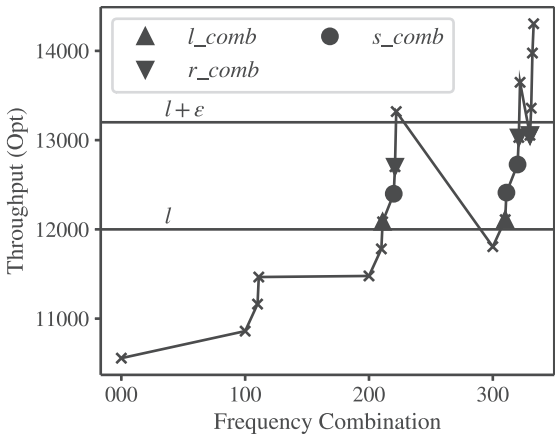


Fig. 1. Example of frequency selection plan.

This section includes 6 subsections. In first 4 subsections, each phase of the multi-phases algorithm is introduced. In Section 3.2.5, the method of computing the upper bounds of the power consumption and the migration cost is proposed, which is used in the multi-phases algorithm to solve the bounded problem. Section 3.2.6 analyses the complexity of the multi-phases algorithm.

3.2.1. Frequency selection phase

In this phase, valid frequency combinations which provide enough resources to support workloads in Δt are generated. According to the combination equation in mathematics, there are C^n_{m-1+n} combinations, in which n is the amount of nodes and m is the amount of available frequency options. To narrow down the search space, the following constraints are introduced.

1. Frequency combinations are coded as m base digital numbers. For example if there are 11 available frequencies and 3 nodes, the minimum combination is 000. The maximum combination is XXX in which $X = 10$.
2. For each coded frequency combination, a digit should not be smaller than its following digits.
3. Frequency combinations that generated throughputs within the range $[l, l + \epsilon]$ are selected.

Constraint (1) defines a code method for frequency combinations. Constraint (2) is used to eliminate redundancies within the frequency combinations. For example, 001 and 010 are same combinations because both of them select f_1 to one node and f_0 to the other two nodes. Constraint (3) defines the range of system throughputs provided by the valid frequency combinations in which ϵ is a user defined threshold.

Fig. 1 gives a traversal example of all frequency combinations (3 nodes and 4 frequency options), which shows the intuition idea of the frequency selection. The steps of the frequency selection phase are shown as follows.

- The left boundary combinations (*l_comb*) which provide the throughput closest but larger than l are located.
- The right boundary combinations (*r_comb*) which provide the throughput closest but lower than $l + \epsilon$ are located.
- Start to traverse combinations from each left boundary combination and select frequency combinations (*s_comb*) until the first combination with throughput larger than $l+\epsilon$ is met.
- Start to reversely traverse combinations from each right boundary combination and select frequency (*s_comb*) combinations until the first combination with throughput lower than l is met.
- All the combinations (*l_comb*, *r_comb* and *s_comb*) are returned

Thanks to the frequency selection phase, the frequency combinations within the range $[l, l + \epsilon]$ can be found. The key algorithm is the boundary combinations location algorithm, which is shown by Algorithm 1.

Algorithm 1 Locate Boundary Combinations Algorithm

```
1:  function FindLocation(l, ϵ, digitNum, digitMax, lower)
2:      result ← []
3:      prefixs ← []
4:      for j ∈ [1, digitMax] do
5:          prev ← [j − 1] × digitNum
6:          left ← [j] + [0] × (digitNum − 1)
7:          right ← [j] × digitNum
8:          result ← result ∪ ChooseComb(prev, left, l, ϵ, lower)
9:          if left <= l <= right then
10:             prefixs ← prefixs ∪ [[j]]
11:         end if
12:     end for
13:     for i ∈ [(digitNum − 1), ..., 2] do
14:         tempPrefix ← []
15:         for prefix ∈ prefixs do
16:             lastMax = prefix[−1]
17:             for j ∈ [1, lastMax] do
18:                 prev ← prefix + [j − 1] × i
19:                 left ← prefix + [j] + [0] × (i − 1)
20:                 right ← prefix + [j] × i
21:                 result ← result ∪
    ChooseComb(previous, left, l, ϵ, lower)
22:                 if left <= l <= right then
23:                     tempPrefix ← tempPrefix ∪ (prefix + [j])
24:                 end if
25:             end for
26:         end for
27:         prefixs ← tempPrefix
28:     end for
29:     for prefix ∈ prefixs do
30:         lastMax = item[−1]
31:         for j ∈ [1, lastMax] do
32:             left ← prefix + [j − 1]
33:             right ← prefix + [j]
34:             result ← result ∪
    ChooseComb(previous, left, l, ϵ, lower)
35:         end for
36:     end for
37:     return result
38: end function
```

Algorithm 1 shows the core function *FindLocation* in this phase. *FindLocation* locates the boundary frequency combination which meets the requirement. Parameters of the function are l, ϵ, *digitNum*, *digitMax*, *lower*. l indicates the total predicted workloads and the value should be l or $l+\epsilon$. *digitNum* indicates the number of digits in the combination and *digitMax* gives the maximum value for each digit. Namely, *digitNum* and *digitMax* refer to the number of the nodes and the number of the available frequency options respectively. *lower* is a binary parameter, which is used to indicate which boundary is required. If *lower* is *True* then the upper boundary is searched, otherwise the lower boundary is searched. At the end, *FindLocation* returns the boundary combinations according to the parameters. The main idea for *FindLocation* is to narrow the search space to find the boundaries for l. The idea is to locate combination is that the provided resource are not linear with the combination traversing order. However, in particular range, it has

a maximum of provided resources and a minimum of provided resources. For example, in Fig. 1, within the range 200–222, the combination which provided maximum throughput is 222 and the combination which provided minimum throughput is 200. The same situation occurs for the sub ranges also, for example 210–211, 221–222. However, between two ranges, there is no clearly relationship. For example, 222 and 300 belong to 2 ranges, 200–222 and 300–333, respectively, but the provided throughputs cannot be compared before the capacities are evaluated. Therefore, a check between ranges and sub-ranges is needed. Therefore, the l is tested for each range of the frequency traversing order, and meanwhile, by means of narrowing down the range, the boundary combination can be obtained.

In Algorithm 1, line 3 to line 12 check the basic ranges (only the first digital number is checked). To be noticed, the checking starts from 1 because the combination 000 — 000 is not a range. In range checking, if l is between the provided throughputs by the minimum combination of the range and the maximum combination of the range (line 9), then the range is stored (line 10) and awaits for future narrow down. Meanwhile, in each loop, the combination from the last previous range is computed (line 5) and compared with the first combination in the current range (line 8) by a function ChooseComb. If the l is between *prev* and *left* then the eligible one is put to result set. Line 13 to line 26, the ranges are narrowed down by each digit, and the possible ranges are selected. However, in each process of the range narrowing down, the combination from previous sub-range is computed (line 18) and compared within first combination in the current sub-range (line 21). To be noticed, the last digit in a combination is not checked (line 13), because with the last digit, the provided throughputs are linear. For example, the throughputs provided by 330, 331, 332, 333 are linear increasing. Line 29 to line 36, the boundaries combinations are selected. The boundary combinations are returned in the end (line 37).

3.2.2. Frequency assignment phase

In this phase, the frequencies in each frequency combination are assigned to the nodes, namely the relationship between the frequencies and the nodes are decided.

The objective of this phase is to generate a frequency assignment which minimize the total amount of migrated workloads and maximize the workloads migrated within racks. The reasons for this objective are: (1) the migration cost cannot be obtained before the whole migration process is done; (2) when a frequency combination is assigned to the nodes, the migrated workloads are determined which is related to the selection of migrated blocks and affects the migration cost.

The process of frequency assignment phase includes following steps.

1 Sort the nodes by descending their predicted workloads under the current block layout.
2 Assign the frequencies in the combination to the sorted nodes. In the frequency combination, frequencies are sorted by descending frequency value.
3 For two nodes which are from different racks, swap the assignments if they satisfy the conditions below.
   - The system throughputs are greater than the required throughputs or the system throughputs are less than the required throughputs for both nodes.
   - The total amount of migrated workloads within racks increases because of the swap.

Using step 1 and step 2, a frequency combination is assigned to the nodes sorted by descending the predicted workloads under current block layout, by which the total amount of migrated workloads is minimized (Proposition 1). Using step 3, the amount of migrated workloads within a rack can be maximized (Corollary 1).

Proposition 1. *When a frequency combination are assigned to the nodes that are sorted by descending their predicted workloads under current block layout, the total size of migrated workloads is minimum.*

Proof. Without loss of generality, we say the node set **C** is sorted by descending the predicted workloads under current block layout, namely by $\sum_{g=1}^{h}\sum_{k=1}^{r} w_{g,k}^{i} \times l \times \varphi_{g,k}$. The frequency assigned to c_i is denoted as f_i. According to the assignment method, if $i < j$, then $f_i > f_j$ and $\psi_i > \psi_j$. Furthermore, $z(c_i, f_i) > z(c_j, f_j)$. The total amount of migrated workloads is denoted as W_{Migrated}. W_{Migrated} can be obtained by Eq. (19) in which **I** is an index set indicating the nodes whose predicted workloads exceed their capabilities.

$$\mathbf{I} = \{i \mid \psi_i > z(c_i, f_i)\, i \in [1, n]\}$$
$$W_{\text{Migrated}} = \sum_{i \in \mathbf{I}} \psi_i - z(c_i, f_i) \tag{19}$$

In order to achieve an assignment which gives less total amount of migrated workloads, the frequencies are swapped between c_i and c_j. Without loss of generality, we assume $i < j$. Basically there are 4 scenarios.

Scenario 1 $z(c_i, f_i) > \psi_i$ and $z(c_j, f_j) > \psi_j$
Scenario 2 $z(c_i, f_i) < \psi_i$ and $z(c_j, f_j) < \psi_j$
Scenario 3 $z(c_i, f_i) > \psi_i$ and $z(c_j, f_j) < \psi_j$
Scenario 4 $z(c_i, f_i) < \psi_i$ and $z(c_j, f_j) > \psi_j$

Scenario 1 When f_i and f_j are swapped, $z(c_j, f_i) > \psi_j$ can be obtained because $f_i > f_j$ and $z(c_j, f_j) > \psi_j$. Since the relationship between $z(c_i, f_j)$ and ψ_i is unknown, there are 2 sub scenarios.

Scenario 1.1: $z(c_i, f_j) > \psi_i$
According to Eq. (19), the increment of W_{Migrated} under this sub scenarios is 0.

Scenario 1.2: $z(c_i, f_j) < \psi_i$
According to Eq. (19), the increment of W_{Migrated} is $\psi_i - z(c_i, f_j) > 0$.

According to the above analysis, when the swapping happens under **Scenario 1**, there is no chance for reducing the total amount of migrated workloads.

Scenario 2 When f_i and f_j are swapped, $z(c_i, f_j) < \psi_i$ is obtained because $f_i > f_j$ and $z(c_i, f_i) < \psi_i$. Since the relationship between $z(c_j, f_i)$ and ψ_j is unknown, there are 2 sub scenarios.

Scenario 2.1: $z(c_j, f_i) > \psi_j$
According to Eq. (19), the increment of W_{Migrated} under this sub scenarios is $\big(\psi_i - z(c_i, f_j)\big) - \big(\psi_i - z(c_i, f_i) + \psi_j - z(c_j, f_j)\big) = z(c_i, f_i) - \psi_j = z(c_j, f_i) - \psi_j > 0$.

Scenario 2.2: $z(c_j, f_i) < \psi_j$
According to Eq. (19), the increment of W_{Migrated} under this sub scenarios is 0.

According to the above analysis, when the swapping happens under **Scenario 2**, there is no chance for reducing the total amount of migrated workloads.

Scenario 3 When f_i and f_j are swapped, $z(c_j, f_i) > \psi_i > \psi_j > z(c_i, f_j)$ is obtained. The increment of the total amount of migrated workloads is $\big(\psi_i - z(c_i, f_j)\big) - \big(\psi_j - z(c_j, f_j)\big) = \psi_i - \psi_j > 0$. When the swapping happens under **Scenario 3**, there is no chance for reducing the total amount of migrated workloads.

Scenario 4 When f_i and f_j are swapped, $\psi_i > z(c_j, f_i) > z(c_i, f_j) > \psi_j$ can be obtained. The increment of the total amount of migrated workloads is $\big(\psi_i - z(c_i, f_j)\big) - (\psi_i - z(c_i, f_i)) = z(c_i, f_i) - z(c_i, f_j) > 0$. When the swapping happens under **Scenario 4**, there is no chance for reducing the total amount of migrated workloads.

According to the discussion above, when a frequency combination is assigned to the nodes sorted by descending the predicted workloads, the total amount of migrated workloads cannot

Table 2
1-0 knapsack details in block selection phase.

Term	Description
Item	Block whose accessed possibility is greater than 0.
Knapsack	For c_i : $z(c_i, f_i)$.
Weight	Workloads of the blocks, $\varphi_{g,k} \times l$.
Profit	Block size $\|b_{g,k}\|$.
Objective	maximize the amount of the kept block sizes.

be minimized by means of swapping 2 assignments. Therefore, the total amount of migrated workloads is minimum under this assignment. ■

Corollary 1. *The swapping cannot reduce the total amount of migrated workloads, but it can increase the amount of migrated workloads within a rack in some cases.*

Proof. In the proof of Proposition 1, two swapping scenario, **Scenario 1.1** and **Scenario 2.2**, do not increase the amount of migrated workloads. However, in some situations, they increase the amount of migrated workloads within racks.

We assume c_i and c_j are from different racks. Considering **Scenario 2.2**, the migrated workloads from the nodes before the swapping are $\psi_i - z(c_i, f_i)$ and $\psi_j - z(c_j, f_j)$ respectively. After the swapping the migrated workloads from the nodes are $\psi_i - z(c_i, f_j)$ and $\psi_j - z(c_j, f_i)$ respectively. According to the conditions of the scenario, the relationships, $\psi_i > \psi_j > z(c_i, f_i) = z(c_j, f_i) > z(c_j, f_j) = z(c_i, f_j)$ can be obtained. Using the swapping, more workloads are migrated from c_i and correspondingly, less workloads are migrated from c_j. If the rack of c_i needs some workload to migrate inside and the rack of c_j needs some workload migrated outside, then the amount of migrated workload within the rack of c_i increases, and correspondingly, the amount of migrated workload between racks of rack c_j decreases. For **Scenario 1.1**, the same case can be constructed as well. Therefore, Corollary 1 is justified. ■

3.2.3. Block selection phase

In this phase, a set of blocks for the migration are selected in order to reduce the migration cost as much as possible.

The migration cost is determined by the total size of the migration blocks as well as the migration directions (within a rack or between racks) according to Eq. (6). In this phase, attempts are done to minimize the migration cost through reducing the total size of the migrated blocks. It should be noticed here that the method of changing the migration directions is not used to reduce the migration cost in this phase.

When a frequency combination is assigned to the nodes, the amount of migrated workloads is determined in Section 3.2.2. To minimize the migration cost, the block selection problem can be interpreted in another way: Which blocks should stay in place to achieve the node's capability while the total size of the blocks is maximized? This problem can be treated as a 0/1 knapsack problem. The items are blocks, and the weight and the profit of these items are the workloads of the block and the block size respectively. The volume of knapsack is the amount of the workloads reserved by the node. The detail of the knapsack problem is shown by Table 2.

By means of dynamic programming method, the optimal kept blocks can be found. However, because of total amount of blocks, dynamic programming cannot solve this problem within acceptable time. Therefore, a greedy algorithm is used to solve the problem, in which the items are sorted by their *profit*/*weight* ratio.

Table 3
Multiple knapsacks details in block migration phase.

Term	Description
Item	Migrated Blocks which are selected in Block Selection Phase.
Knapsack	The nodes who has extra capability. $\{c_i \mid (z(c_i, f_i) - \psi_i) > 0\ i \in [1, n]\}$. Knapsack size is $z(c_i, f_i) - \psi_i$.
Weight	Workloads of the blocks, $\varphi_{g,k} \times l$.
Profit	Block size $\|b_{g,k}\|$.
Objective	maximize the amount of block sizes which are migrated within the rack.

3.2.4. Block migration phase

In this section, the migrated blocks selected by Phase 3.2.3 are redistributed. The goal of this phase is to generate a migration plan which gives the destination for each migrated block. The objective is to reduce the total migration cost. According to Eq. (6), the migration cost is related to the migrated block sizes and their destinations (i.e. within a rack or between racks). However, the migrated blocks are determined by previous phase, which means the block sizes are determined as well. Therefore, the objective of this phase is to generate a migration plan which put the migrated blocks in their own rack as much as possible. Therefore, the main process of this phase is shown as follows.

1. Migrated blocks are migrated within their own racks first.
2. If some blocks cannot be migrated within their own racks, then they are migrated within the whole cluster.

The migration problem can be interpreted in another way: Which blocks should be kept within the rack in order to minimize migration cost? Actually, the migration problem can be treated as a constrained multiple knapsack problem. The multiple knapsacks are the interspaces of the nodes whose predicted workload does not exceed their capability. The item's weight and profit are block's workload and block size respectively. The problem objective is to maximize the block sizes which are kept within the rack. The detail of the multiple knapsack problem is shown by Table 3.

In order to solve the multiple knapsack problem, a constrained exchange algorithm is introduced which is denoted as CMTHM. This algorithm is inspired by MTHM algorithm [11]. In MTHM algorithm, there are 4 steps:

1. Choose the item using a greedy algorithm for each knapsack (items are sorted by descending *profit*/*weight* and knapsacks are sorted by descending volumes).
2. Rearrange the selected items to make sure not all valuable items are stored in the same knapsack.
3. Interchange assigned items and try to insert unassigned items.
4. Exchange assigned items with unassigned items if the exchange can fit more unassigned items.

In CMTHM, the replicas of a block are regarded as conflict with each other when items are moving. If a conflict occurs, the exchange continues until the next exchange opportunity is found.

In the global migration, the remaining blocks are redistributed. In this migration, CMTHM is applied. However, compared with the migration within racks, the knapsacks are the nodes of the cluster which do not reach their capabilities.

3.2.5. Upper bounds of power consumption and migration cost

In the multi-phase algorithm, the frequency combinations generated by frequency selection phase are evaluated by following phases. If a frequency combination is obtained, the corresponding migration plan can be generated, and meanwhile the power consumption and the migration cost can be obtained which are used to evaluate the solution. However, if the migration plan is generated

for all the frequency combinations, the performance of the multi-phase algorithm is unacceptable. In this section, the methods for computing the upper bounds of power consumption and migration cost are proposed, which are used to improve the performance of the multi-phase algorithm. The usage of the upper bounds is discussed in Section 3.2.6.

Upper bound of power consumption. After the frequency assignment phase, the upper bound of power consumption can be obtained. According to Eq. (5), the power consumption P is related to the frequencies and the yields. Considering a block $b_{g,k}$ is assigned to c_i, the power consumption of c_i increases because of it, i.e. it increases with the probability access $\varphi_{g,k}$ due to that block. The increment is $\frac{\varphi_{g,k}}{\sum_{u=1}^{m} x_u^i \times z(c_i,f_u)} \times \sum_{u=1}^{m} \left(x_u^i \times \left(c_{f_u}^{max} - c_{f_u}^{idle}\right)\right)$. For a given frequency combination, x_u^i are set. Without loss of generality, we can say that node c_i is assigned with frequency f_i. Therefore the increment can be rewritten as $\frac{\varphi_{g,k}}{z(c_i,f_i)} \times \left(c_{f_i}^{max} - c_{f_i}^{idle}\right)$. In this scenario, the factor $\frac{\left(c_f^{max} - c_f^{idle}\right)}{z(c_i,f_i)}$ is constant and it is defined as the power consumption contribution factor of the node. 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 nodes with free capacity in the same rack are combined to form a big knapsack with larger free capacity. Because of the relaxation, one block can be split and put to multiple nodes. To compute the upper bound of the power consumption, following conditions are introduced.

- Let the total ordered node set be $\widetilde{\mathbf{C}} = (\mathbf{C}, \leq)$. $\forall i, j$ $\left(c_{f_i}^{max} - c_{f_i}^{idle}\right)/z(c_i,f_i) \geqslant \left(c_{f_j}^{max} - c_{f_j}^{idle}\right)/z(c_j,f_j)$, 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 $i \leqslant \widetilde{i}+1$.

With above conditions, the maximum power consumption P^{max} is shown by Eq. (20). 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 Eq. (4).

$$P^{max} = \left(z\left(c_{\widetilde{i}+1}, f_{\widetilde{i}+1}\right) - \left(\sum_{i}^{\widetilde{i}+1} z\left(c_i,f_i\right) - \sum_{i}^{n} w_i\right)\right) \times \frac{c_{f_{\widetilde{i}+1}}^{max} - c_{f_{\widetilde{i}+1}}^{idle}}{z\left(c_{\widetilde{i}+1}, f_{\widetilde{i}+1}\right)} + c_{f_{\widetilde{i}+1}}^{idle} + \sum_{i=0}^{\widetilde{i}} c_{f_i}^{max} + \sum_{i=\widetilde{i}+2}^{n} c_{f_i}^{idle} \tag{20}$$

Upper bound of migration cost. After the block selection phase, the upper bound of migration cost can be obtained by means of the same relaxation. To compute the maximum migration cost, two conditions are introduced:

1. There are U racks. For a rack Γ_u, the selected block set is denoted as $\mathbf{M}_u$. The total ordered set of $\mathbf{M}_u$ is denoted as $\widetilde{\mathbf{M}}_u = (\mathbf{M}_u, \leqslant) = \{b_1, b_2, \ldots\}$ in which the blocks are sorted by the ratio between their workloads and sizes, namely $\forall b_i, b_j \in \widetilde{\mathbf{M}}_u$, 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. $\widetilde{\mathbf{M}}_u^{\text{In}}$ and $\widetilde{\mathbf{M}}_u^{\text{Out}}$ are partitions of $\widetilde{\mathbf{M}}_u$. $\widetilde{\mathbf{M}}_u^{\text{In}}$ and $\widetilde{\mathbf{M}}_u^{\text{Out}}$ satisfy the following conditions:
   - $\widetilde{\mathbf{M}}_u^{\text{In}} = \{b_1, \ldots, b_v\}, \widetilde{\mathbf{M}}_u^{\text{Out}} = \{b_{v+1}, b_{v+2}, \ldots\}$.
   - Let the partial set of the nodes in Γ_u that have extra capacity be $\mathbf{C}_u^{\text{In}} = \{c_i | c_i \in \Gamma_u \text{ and } \psi_i < z(c_i,f_i)\}$. Then, the index v of $\widetilde{\mathbf{M}}_u$ satisfies $\sum_{j=0}^{v} l \times \varphi_j \leqslant \sum_{c_i \in \mathbf{C}_u^{\text{In}}} (z(c_i,f_i) - \psi_i) < \sum_{j=0}^{v+1} l \times \varphi_j$.

With above conditions, the maximum migration cost T^{max} can be obtained by Eq. (21), in which U is the number of racks. Since the blocks with lower workloads but larger sizes are migrated to other racks, the migration cost is the highest among all migration plans.

$$T^{max} = E_{\text{In}} \times \sum_{b_i \in \bigcup_{u=1}^{U} \widetilde{\mathbf{M}}_u^{\text{In}}} |b_i| + E_{\text{Out}} \times \sum_{b_i \in \bigcup_{u=1}^{U} \widetilde{\mathbf{M}}_u^{\text{Out}}} |b_i| \tag{21}$$

3.2.6. Complexity analysis

In previous sections, the details of the multi-phase algorithm are introduced. In this section, the multi-phase algorithm is introduced in the view of bounded problems, and the complexity of the algorithm is analyzed.

$\mathbb{P}^b\mathbb{T}^{min}$ defines the problem that minimizes the migration cost fulfilling the basic requirement of the power consumption bound. After frequency assignment phase, frequency combinations are filtered according to $\mathbb{P}^b$. For each frequency combination, if $P^{max} \leq \mathbb{P}^b$, then the combination is kept otherwise the combination is discarded. Due to the filter process, all the kept combinations can satisfy $\mathbb{P}^b$ condition. Meanwhile the frequency combinations are sorted by ascending T^{max} after block selection phase. Therefore, in block migration phase, only few solutions are evaluated and the one with minimum migration cost is returned.

$\mathbb{T}^b\mathbb{P}^{min}$ defines the problem that minimizes the achievable power consumption within the migration cost bound. In order to solve the problem, after the frequency assignment phase, frequency combinations are sorted by ascending P^{max}. Meanwhile after the block selection phase, the solutions are filtered by means of $\mathbb{T}^b$. For each solution, if $T^{max} \leq \mathbb{T}^b$, then the solution is kept otherwise the solution is discarded. Due to the filter process, all the kept solutions can satisfy $\mathbb{P}^b$ condition. Therefore, in block migration phase, only few solutions are evaluated and the one with minimum power consumption is returned.

The feasible solution is found by means of the multi-phase algorithm. In each phase, a solution is optimized with different aspects. In frequency selection phases, the power consumption of the solution is optimized. Then, the migration cost is optimized in frequency assignment phase, block selection phase and block migration phase.

Multi-phase algorithm is combined by 4 phases and 3 auxiliary operations (filtering, sorting and upper bound calculation). The complexities of each operation in the phases are listed in Table 4. In Table 4, n represents the amount of nodes and m represents

Table 4
Complexity of the operations.

Phase	Operation	Complexity	Auxiliary operation	Complexity
1	Combination locating	$O(nm^2)$	Filter combinations	$O(\kappa)$
	Combination traversing	$O(\kappa)$	Sort combinations	$O(\kappa \log \kappa)$
2	Frequency assignment	$O(n^2)$	Compute upper bound	$O(1)$
3	Block selection	$O\left(\max_{i=1}^{n}\left(\lvert\mathbf{D}_i^r\rvert log\lvert\mathbf{D}_i^r\rvert\right)\right)$		
4	Block migration	$O\left(\max\left(\max_{u=1}^{U}\left(\lvert\mathbf{M}_u^{\text{In}}\rvert^3\right), \lvert\bigcup_{u=1}^{U}\mathbf{M}^{\text{Out}}\rvert^3\right)\right)$		

the amount of frequency options. κ represents the amount of frequency combinations which are generated in frequency selection phase. The amount of blocks for each node is denoted as $|\mathbf{D}_i^r|$ in which $\forall i \in [1, n]\ \mathbf{D}_i^r = \{b_{g,k} | w_{g,k}^i == 1\}$. The amount of blocks which are migrated within their rack is denoted as $|\mathbf{M}_u^{\text{In}}|\ \forall u \in [1, U]$ in which U is amount of racks. The amount of blocks which are migrated between racks are denoted as $\sum_{u=1}^{U} |\mathbf{M}_u^{\text{Out}}|$.

- Phase 1: The operation of locating combination checks the frequency option for each node therefore the complexity is $O(nm^2)$. In traversing combinations operation, each frequency combination is checked to make sure it fits to the boundaries. Therefore the complexity for traversing combinations is $O(\kappa)$.
- Phase 2: In order to assign the frequencies to the nodes, frequencies are exchanged between nodes. The complexity of assigning frequency is $O(n^2)$.
- Phase 3: In block selection operation, the complexity is $O\left(\max_{i=1}^{n}\left(|\mathbf{D}_i^r| log |\mathbf{D}_i^r|\right)\right)$ because quick sort is applied to achieve greedy method. However, the amount of assigned blocks for each node is different, the complexity depends on the maximum amount of blocks within all the nodes.
- Phase 4: In block migration operation, the algorithm CMTHM is adopted. The complexity of CMTHM depends on the amount of items because of exchange process. However CMTHM is applied $U + 1$ times, since it is applied to every rack and the whole system respectively. Therefore the complexity of Phase 4 depends on the maximum amount of migrated blocks within a rack and the amount of migrated blocks between racks. Therefore, the complexity of CMTHM algorithm is $O\left(\max\left(\max_{u=1}^{U}\left(|\mathbf{M}_u^{\text{In}}|^3\right),\ |\bigcup_{u=1}^{U}\mathbf{M}^{\text{Out}}|^3\right)\right)$.

Note that the complexities of the operations in phase 2 to phase 4 refer to the complexity of processing one frequency combination. In worse case, the complexities of the operations in phase 2 to phase 4 need to be multiplied by $O(\kappa)$ if the all frequency combinations are processed. However, for both bounded problem $\mathbb{P}^b\mathbb{T}^{min}$ and $\mathbb{T}^b\mathbb{P}^{min}$, the feasible solution can be obtained at the beginning of the search, because of the sorting and filter process.

4. Evaluation

In this section, a series of experiments is designed and executed to evaluate the model and corresponding algorithms. There are 3 parts in this section, Setup Benchmark, Simulation Experiment, and Execution Experiment. Setup Benchmark gives the benchmark experiments to obtain the parameters in Hot-N-Cold model. The proposed algorithm are evaluated and compared in Simulation Experiment. Finally, our approach is applied to a real environment in Execution Experiment to evaluate the energy saving effect of cloud database systems in running time.

4.1. 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}.

4.1.1. Capacity benchmark

This benchmark is executed on Grid5000 [3] 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 [8] with 10 nodes belonging to the Nancy site graphene cluster is deployed. The core properties used in YCSB workload profile are shown in Table 5. To be noticed that, the cache related parameters are set to 0 to avoid the influence of cache mechanism to the experiment results.

Table 5
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 6
Core properties of YCBS workload.

Property	Value
Recordcount	3 000 000
Fieldlength	1000
Readproportion	0.95
Updateproportion	0.05
Requestdistribution	Uniform
Threadcount	500

Table 7
Node's capability under each frequency option.

Frequency	Capability	Frequency	Capability
2.53 GHz	5690	1.73 GHz	4768
2.40 GHz	5518	1.60 GHz	4440
2.13 GHz	5357	1.33 GHz	3822
2.00 GHz	5211	1.20 GHz	3520

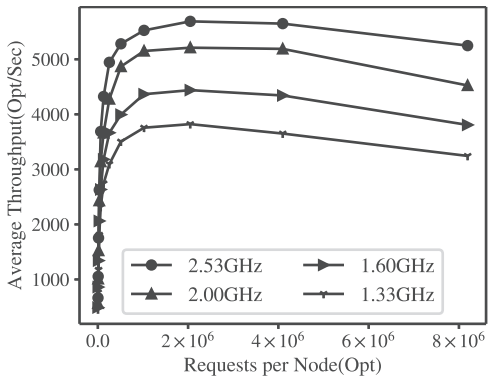
The nodes are equipped with a 4 cores Intel Xeon X3440 and 16 GB of RAM. The energy consumption values are collected by Power Distribution Units (PDU). There are 8 available frequency options: 2.53 GHz, 2.40 GHz, 2.13 GHz, 2.00 GHz, 1.73 GHz, 1.60 GHz, 1.33 GHz, 1.20 GHz. In this benchmark, the maximum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

To simulate the real workloads, Yahoo! Cloud Serving Benchmark(YCSB) framework [5] 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 6.

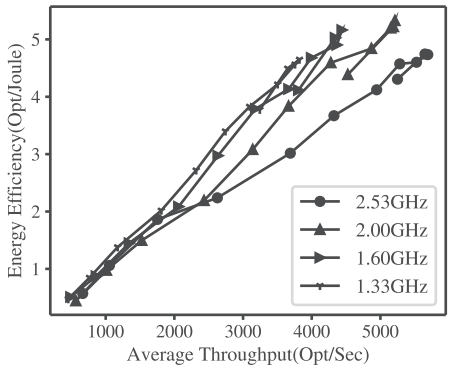
To obtain the maximum throughput of the system, more and more requests are loaded into the system. There are 12 workloads in total for each frequency option. The workload is denoted as Q_i $i \in [1, 12]$. For workload Q_i, the total amount of requests is $4000 \times 2^{i-1}$. The result is shown by Fig. 2 and Table 7.

Fig. 2(a) shows the trends of throughput along with the increasing requests for the system under frequency 2.53 GHz, 2.00 GHz, 1.60 GHz and 1.33 GHz. The trends have 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 Table 7. 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.

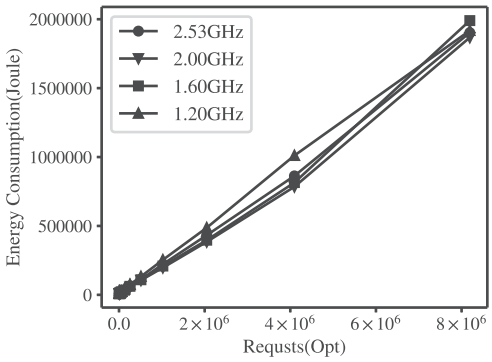
Fig. 2(b) shows the relationship between the energy efficiency and the throughputs under different frequencies. Along with the



(a) Relationship between Request Amount and Throughput



(b) Relationship between Throughput and Energy Efficiency



(c) Energy Consumption for each Workload

Fig. 2. Capability benchmark result of Cassandra system.

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.

Fig. 2(c) 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 throughputs of the system are tested, a lot of requests are

Table 8
Migration cost for unit block.

Parameter	Value
E_{In}	0.8 J/Mb
E_{Out}	1.0 J/Mb

loaded into the system. With different frequency configuration, the throughput of the system is different. For example with 2.53 GHz, 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}, $t(2.53\text{ GHz}) = 1778$ s and $t(1.33\text{ GHz}) = 2837$ s. As a consequence, the energy consumptions do not have big difference for both cases.

4.1.2. Migration cost benchmark

In Eq. (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.

$$E_{\text{In|Out}} = \frac{E_{\text{Migration}} - E_{\text{Idle}} \times \frac{t_{\text{Migration}}}{t_{\text{Idle}}}}{\sum_{b_{g,k} \in \mathbf{M}_{\text{In|Out}}} |b_{g,k}|} \tag{22}$$

E_{In} and E_{Out} are calculated by Eq. (22), 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 values of the parameters are shown in Table 8.

4.2. Simulation experiment

In this section, proposed algorithms are evaluated using simulation experiment. Nonlinear programming algorithm can find the optimal solution for both frequency selection and migration process, however it cannot be applied to larger scale problems. In contrast, multi-phases algorithm can find the sub-optimal solutions and has good scalability. In this section, we mainly focus on multi-phases algorithm. There are 4 experiments in total, which are made to evaluate multi-phases algorithm in aspect frequency selection, migrated block selection, migration plan generation and its scalability. Meanwhile a comparison experiment is made to compare nonlinear programming algorithm and multi-phases algorithm in Section 4.2.4.

To conduct the simulation experiments, some simulation test cases are generated. The method for generating test cases are shown as below.

- **System**. A system is denoted as *sn* in which *n* is the amount of nodes in the cluster. Meanwhile, it is assumed that there are 2 racks in total. For example *s*20 present a database system with 20 nodes, and there are 2 racks in which each rack contains 10 nodes.
- **Block and Block layout**. The amount of block is set to 64 per node in the experiments and the replica factor is set to 3. Therefore in *s*10, there are 1920 blocks in total. The size

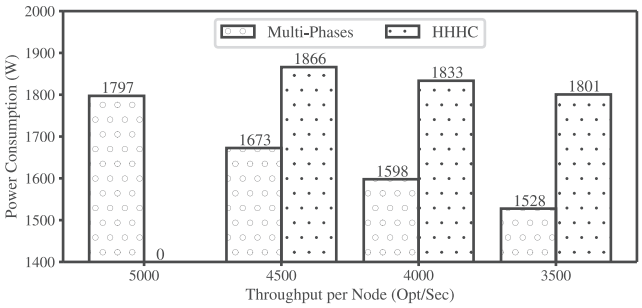


Fig. 3. Result of frequency selection comparison.

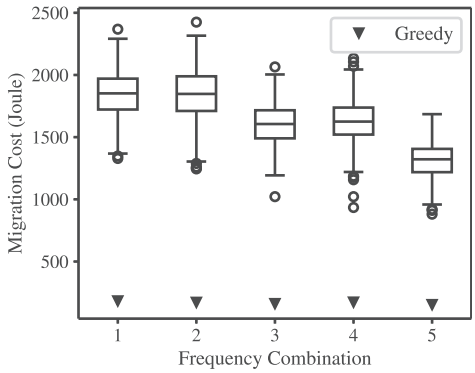


Fig. 4. Result of migrated block comparison.

of blocks are randomly generated within 10Mb to 30Mb. For the replicas, they have same block size. To be noticed this method is used to give the amount of blocks. The blocks are distributed by following rules: (1) the first replica is placed on one of the nodes by means of round-robin strategy; (2) the second replica is placed on another rack; (3) the third replica is placed on the same rack of the second, but on a different node chosen at random.

- **Block accessed possibilities**. The block accessed possibility in Hot-N-Cold model refers to the parameter $\varphi_{g,k}$. In practice, these are predicted values. In the simulation experiment, possibilities are generated by Zipf distribution [22] and the distribution factor is set to 2.5 in our experiment.
- **Workload**. The workload in this paper refers to the parameter l which is the predicted required throughput. The value of l is given by the throughput per node. For example, if we say the workload of one test case is set to 3500 Opt/Sec to $s10$, then for the whole system, the throughput is set to 35 000 Opt/Sec. However, this method is a way to set the throughput value for the whole system, the throughput for each node depends on the assigned blocks and their accessed possibilities.
- **Boundary**. There are two problem types in this paper, $\mathbb{P}^b\mathbb{T}^{min}$ and $\mathbb{T}^b\mathbb{P}^{min}$. For both problems, the boundary value of $\mathbb{P}^b$ and $\mathbb{T}^b$ should be provided. In the experiments, if not specified, then the boundary value is obtained as follows. To obtain the boundary value $\mathbb{P}^b$, the corresponding test case is solved by $\mathbb{T}^b\mathbb{P}^{min}$ in which $\mathbb{T}^b$ is set to infinite denoted as $\mathbb{T}^{Inf}$. When the value of $\mathbb{P}^{min}$ is obtained, the boundary value is set to $1.1 \times \mathbb{P}^{min}$. The boundary value $\mathbb{T}^b$ is obtained by the same way.

A test case is a combination with a system, a workload and a problem type, which is denoted as $s < n > \_ <$ workload $> \_ <$ type $>$. For example $s10\_3500\_\mathbb{P}^b\mathbb{T}^{min}$ presents a test case in which there are 10 nodes (5 for each rack), the workload throughput is set to $3500 \times 10 = 35000$ Opt/Sec. In $s10\_3500\_\mathbb{P}^b\mathbb{T}^{min}$, a power consumption boundary value is provided and the objective is to minimize migration cost.

4.2.1. Frequency selection comparison

In this comparison experiment, the frequency selection phase from the multi-phase algorithm is compared with a method from Houssem-Eddine Chihoub et al. [4]. The method is defined as HHHC (Half Hot and Half Cold) in our experiment. In HHHC, half of the nodes are set to the highest frequency(2.53 GHz) and another half of the nodes are set to the lowest frequency (1.20 GHz). In multi-phase algorithm, the frequencies are selected according to the predicted workloads.

In this experiment, 4 test cases are involved, which are denoted as $s20\_l\_\mathbb{T}^{Inf}\mathbb{P}^{min}$ $l \in [5000, 4500, 4000, 3500]$. Since there is no boundary for migration cost, minimize power consumption is the objective for both algorithms. All the cases are solved by multi-phases algorithm and HHHC and the result is shown by Fig. 3.

In Fig. 3, The power consumption given by multi-phase algorithm are lower than the corresponding result given by HHHC. The average improvement of multi-phases algorithm compared with HHHC is 12.89%. When l is set to 5000 Opt/Sec, HHHC cannot produce a valid solution. Theoretically, when HHHC applied, the system with 20 nodes can support any workloads with throughput under 92080 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 power consumption is recorded as 0. The main drawback of HHHC is its flexibility. HHHC sets the frequencies statically, while the multi-phase algorithm chooses frequencies according to the workload predictions.

4.2.2. Migrated block comparison

In the block selection phase of the multi-phases algorithm, migrated blocks are selected by a greedy algorithm to reduce the migration cost. In this section, the greedy selection is compared with the random selection.

The test case used in this experiment is $s20\_4000\_\mathbb{P}^{Inf}\mathbb{T}^{min}$. 5 frequency combinations are used in this experiment which are selected and assigned by the multi-phases algorithm. For each frequency combination, 1000 migrated block sets are generated by the random selection and the migration process is applied. The migration cost values are obtained using Eq. (6). The comparison results are shown by Fig. 4.

In Fig. 4, the migration cost values obtained by the random selected migrated blocks are shown in the form of box charts. According to the experiment result, the migration costs obtained by the greedy selected migrated blocks are much lower than the random selected migrated blocks for each frequency combination. Before the block selection phase, the total amount of migrated workloads is determined. Therefore, the objective in the block migration phase is to generate a migrated block set which satisfies the migrated workloads constraint and produces minimum migrated block sizes. In the greedy selection algorithm, the blocks are sorted by the ratio between its workload and block size, therefore the migration cost is much lower.

4.2.3. Migration comparison

In block migration phase, the migrated blocks are redistributed within the cluster. CMTHM is used to choose the destination for each migrated block. In this experiment, CMTHM is compared with the first fit algorithm. In the first fit algorithm, the nodes with extra capabilities are sorted by the descending available capability and the migrated blocks are sorted by descending the ratio between their block sizes and their workloads. In first fit, the blocks are migrated within their own rack first. if there is no space for the blocks, the remained blocks are redistributed to other racks.

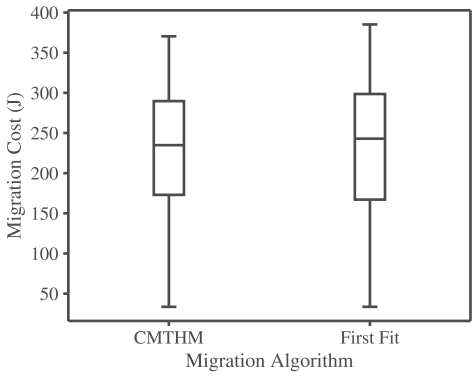


Fig. 5. Result of migration comparison.

Table 9
Properties of the test cases for nonlinear programming algorithm.

Property	Description
System	The system includes 6 nodes denoted as $s6$.
Replica	2
Block amount	3 per node(36 blocks in total)

The test case used in this experiment is $s20\_4000\_\mathbb{P}^{Inf}\mathbb{T}^{min}$. 1000 frequency combinations are used in this experiment which are selected and assigned by multi-phases algorithm. For each frequency combination, the migrated blocks are given by multi-phases algorithm as well. Then, the blocks are migrated by both CMTHM and First Fit. In the end of migration, the migration costs are collected. The result is shown by Fig. 5.

According to Fig. 5, the results given by both algorithm do not have big difference. The range of the migration cost given by CMTHM is slight lower than the results given by First Fit. The migration cost for migrating a block between racks is slight larger than the migration cost for migration a block within its rack. However, for both algorithm, they try to place the migrated blocks in its own rack first. Therefore the results given by both algorithm are similar.

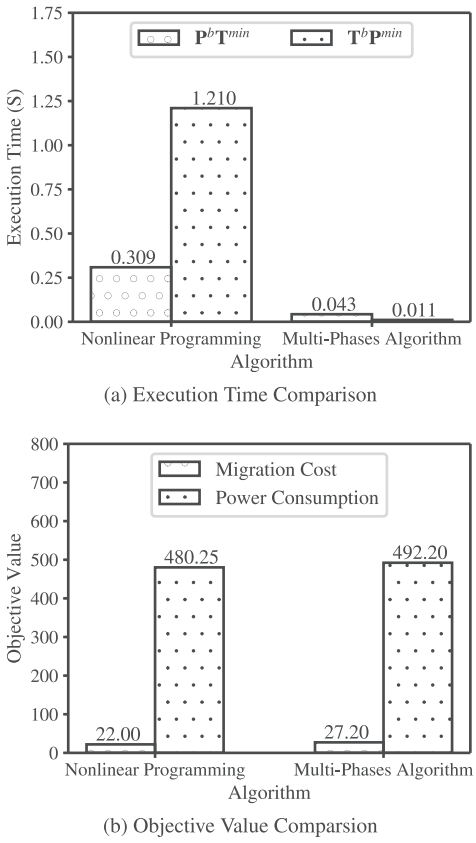
Among all the results, 81.3% of results given by CMTHM are lower than the results given by First Fit. In CMTHM, in order to improve the profit, the items are rearranged and swapped after the First Fit (first step of CMTHM). In most of the cases, these operations improve the profit of kept items, however sometimes they do not. But general, CMTHM is better than First Fit.

4.2.4. Scalability experiment

Scalability experiment includes two parts. At first a comparison experiment between the nonlinear programming algorithm and the multi-phases algorithm is executed. Then, the scalability of the multi-phases algorithm is evaluated.

Since the nonlinear programming algorithm cannot be applied to a large scale problem. Therefore, in this experiment, two special test cases, denoted as $s6\_4000\_\mathbb{P}^b\mathbb{T}^{min}$ and $s6\_4000\_\mathbb{T}^b\mathbb{P}^{min}$, are introduced. The detail about the properties of these test cases are listed in Table 9. Compared with the test case $s20\_4000\_\mathbb{P}^b\mathbb{T}^{min}$, $s6\_4000\_\mathbb{P}^b\mathbb{T}^{min}$ and $s6\_4000\_\mathbb{T}^b\mathbb{P}^{min}$ are smaller in terms of the amount of nodes and the amount of blocks. However, in our experiment, these are the biggest cases for the nonlinear programming algorithm, otherwise the algorithm cannot produce a solution within acceptable time. The comparison result between the nonlinear programming algorithm and the multi-phases algorithm is shown by Fig. 6.

In Fig. 6(a), the execution time of multi-phases algorithm is lower than the corresponding execution time of nonlinear programming algorithm for both test cases. Fig. 6(b) shows the objective value given by both algorithm for the test cases. Generally,



(a) Execution Time Comparison

(b) Objective Value Comparsion

Fig. 6. Compression between nonlinear programming and multi-phases algorithm.

the objective values given by nonlinear programming algorithm is lower than the corresponding objective values given by multi-phases algorithm. The reason of this situation is that multi-phases algorithm takes advantages of approximation algorithms in each phase to obtain a feasible solution of the problem. In contrast, nonlinear programming algorithm uses nonlinear optimization method to solve the problem and obtain the global optimal solution. Therefore, the multi-phase algorithm is more efficient than nonlinear programming algorithm in terms of performance.

After comparison of nonlinear programming algorithm and the multi-phase algorithms. Another series tests are made to evaluate the scalability of the multi-phases algorithm. In this experiment, test cases $sn\_4000\_\mathbb{P}^b\mathbb{T}^{min}$ and $sn\_4000\_\mathbb{P}^b\mathbb{T}^{min}$ are used, in which $n \in [10, 20, 30, 40, 50, 60, 70]$. The execution time for each case is shown by Fig. 7.

In Fig. 7, the execution time of the multi-phases algorithm increases with the increasing of the amount of nodes for both problem type. When the amount of nodes increases, the total frequency combinations increases as well. Therefore, the amount of frequency combinations which are evaluated in the multi-phases algorithm increases. Meanwhile, the amount of migrated blocks increases as well since the workload increases along with the node amount. Then, the execution time for both frequency selection and workload migration increases. As a consequence, the execution time of the multi-phases algorithm increases.

With the same node amount, the execution time for $\mathbb{P}^b\mathbb{T}^{min}$ is shorter than the execution time for $\mathbb{T}^b\mathbb{P}^{min}$ especially for larger problems (i.e. $s60$ and $s70$). For problem $\mathbb{P}^b\mathbb{T}^{min}$, multi-phase algorithm filters frequency combinations after the frequency assignment phase by P^{max}, and sort frequency combinations after the block selection phase by T^{max}. However, For problem $\mathbb{T}^b\mathbb{P}^{min}$,

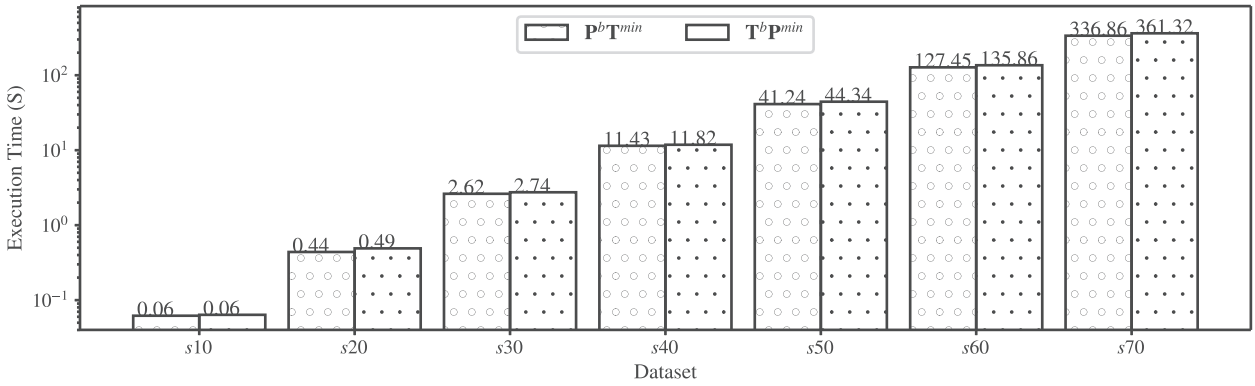


Fig. 7. Result of scalability experiment.

the multi-phases algorithm sorts frequency combinations after the frequency assignment phase by P^{max}, and filters frequency combinations after the block selection phase by T^{max}. When the same amount of frequency combinations generated in the frequency selection phase, the execution time for $\mathbb{P}^b\mathbb{T}^{min}$ is shorter because less frequency combinations are evaluated in block selection phase. When a frequency combination is filtered, then the combination is discarded. In $\mathbb{P}^b\mathbb{T}^{min}$, the filter operation happens before the block selection phase which causes less execution time.

4.3. Execution experiment

The purpose of this experiment is to evaluate how much energy can be saved by means of Hot-N-Cold model in running time. Compared with energy consumption of the running time, the migration cost is quite small, therefore in this experiment, we only focus on the energy consumption in the running time. In this experiment, a series of well designed queries are executed in a Cassandra system to simulate the workloads after the migration process. Meanwhile the corresponding energy consumptions are collected by PDU. The core configurations are shown as follows.

- **System.** The system used in this experiment is a Cassandra system with 10 nodes belonging to the Nancy site graphene cluster. The configuration of the hardware is the same with Setup Experiment.
- **Test Case** There are 4 test cases used in the experiment. $s10\_l\_\mathbb{T}^{Inf}\mathbb{P}^{min}$ in which $l \in [3500, 4000, 4500, 5000]$.
- **Frequency.** There are 2 frequency situations in this experiment.
  1. The nodes are assigned with Performance mode which is the default setting for the nodes
  2. The frequencies generated by multi-phases algorithm.
- **Query.** Queries in this experiment refer to the requests amount for each node. At first, the multi-phases algorithm is applied to each case, and corresponding results (frequency selection and migration plan) are obtained. Then, for node c_i, the throughput ψ_i are calculated. Finally the requests amount for node c_i is set to $\psi_i \times \frac{2\times10^6}{5690}$. The parameter, $\frac{2\times10^6}{5690}$, is used to make sure every node under its selected frequency option can reach its capability, and meanwhile the workloads for each node can be finished around same time. The parameter is obtained by the results shown by Fig. 2. When the request amount for a node reaches 2×10^6 Opts, it has highest energy efficiency in our environment.
- **Benchmark Framework.** The benchmark framework is YCSB and the core properties are the same with Table 6. In order to simulate the real case, another property *target* is set to the workload profile to make sure each workload for the node is executed by a designed throughput ψ_i.

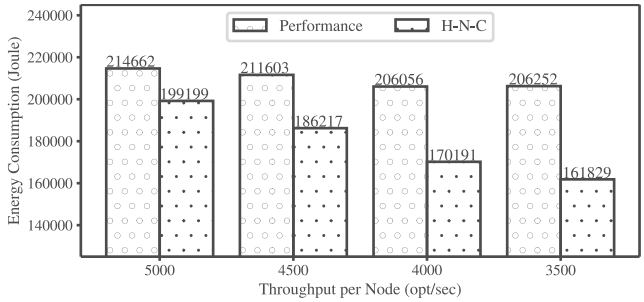


Fig. 8. Result of execution experiment.

For each test case $s10\_l\_\mathbb{T}^{Inf}\mathbb{P}^{min}$ in which $l \in [3500, 4000, 4500, 5000]$, we simulate their migration result in the testbed under two frequency configurations (Performance mode and Hot-N-Cold mode), and the energy consumptions are collected shown by Fig. 8.

In Fig. 8, the energy consumptions given by Hot-N-Cold mode is always lower than the energy consumptions given by Performance mode. In Hot-N-Cold mode, the frequencies are selected according to the predicted workloads, therefore there is no energy wasting because of resource over-provisioning. However, when the system is set to Performance mode, in some cases, there are nodes with extra capabilities which cause energy wasting.

$$O(l) = \frac{E(l, Performance) - E(l, H-N-C)}{E(l, Performance)} \tag{23}$$

For each test case, $s10\_l\_\mathbb{T}^{Inf}\mathbb{P}^{min}$, the optimization ratio is denoted as Eq. (23) in which $E(l, Performance)$ and $E(l, H-N-C)$ denote the energy consumption given by Performance mode and Hot-N-Cold mode respectively. The optimization ratio depends on the value of throughput per node. For $\forall l_1 < l_2$, $O(l_1) > O(l_2)$. With the increment of throughput per node, the optimization ratio decreases. The maximum optimization ratio is 21.5% for the case $s10\_3500\_\mathbb{T}^{Inf}\mathbb{P}^{min}$, and the minimum optimization ratio is 7.2% for the case $s10\_5000\_\mathbb{T}^{Inf}\mathbb{P}^{min}$. With lower workload amount, the energy wasting situation is more worse under Performance mode. The optimization ratio is higher with lower amount workloads.

4.4. Conclusion

In the experiment section, there are 3 experiments, Setup Experiment, Simulation Experiment and Execution Experiment.

In Setup Experiment, the node's capacities under different frequency options are obtained. Meanwhile, the energy costs per mega byte of migration within a rack and between racks are obtained. There are two conclusions that can be obtained: (1) With

higher throughput, the system achieves higher energy efficiency; (2) The cloud database system has a throughput peak under a given frequency configuration.

In Simulation Experiment, a set of simulation test cases are executed. The results show following conclusions. (1) Compared with HHHC, frequency combinations selected by multi-phases algorithm can save more energy, meanwhile the frequency selection phase of multi-phases algorithm more flexible. (2) Compared with random selection, the migrated block set generated by block selection phase can achieve lower migration cost. (3) Compared with nonlinear programming algorithm, multi-phases algorithm can obtain feasible solution but has a great scalability.

In Execution Experiment, a few test cases are executed in a 10 nodes cluster testbed. The result shows that at maximum 21.5% energy can be saved in running time by means of Hot-N-Cold model.

5. Related work

In this work, we focus on the resource provisioning problem within energy aware cloud database systems. Hot-N-Cold model and corresponding algorithms are proposed which take advantage of DVFS and workload predictions to improve energy efficiency of cloud database systems. In this section we only discuss those works that are related to energy efficiency of cloud database systems. We refer the reader to [1] for an extensive literature review of resource provisioning problem within cloud systems, and [6] for a list of possible techniques for reducing energy in large scale distributed systems.

In terms of energy efficiency in cloud database systems, there are some literatures gave the definition and influence factors. Tsirogiannis Dimitris et al. [20] analyzed the energy efficiency and its influence factor within Database Server (RDBMS). They gave the conclusion that within a single node intended for use in scale-out (shared-nothing) architectures, the most energy-efficient configuration is typically the highest performing one. Jie Song et al. [16,17] gave the definition of the energy efficiency in cloud database systems and a novel evaluation model to compute the energy efficiency. In their model, the relationship between energy efficiency and node's CPU frequency and CPU usage is studied. Meanwhile, they gave the conclusion that the maximum energy efficiency of a system occurs when CPU frequency and usage are maximum. In our work, we use the same energy efficiency definition of cloud database systems with Jie Song et al. Meanwhile, in the experiment, we concluded that the maximum energy efficiency occurs when the system has maximum throughput. Furthermore the cloud database system has a throughput peak under a given frequency configuration. This peak throughput is defined as node capability in our work which is used as a standard for frequency selection and workload migration.

There is a large literature on the topic of improving energy efficiency in cloud database systems. Willis Lang et al. [10] studied the trade-offs between the performance and the energy consumption characteristics of analytical queries and gave guiding principles for building energy-efficient cloud DBMS considering the query properties and scalability. The basic idea behind this study is to match the resource provided by the query engine and the queries properties. In our work, we mainly focus on match CPU resource with query throughput. Meanwhile a migration process is proposed to further improve energy saving. Balaji Subramaniam et al. [18] measured the power consumption and the performance of a Cassandra cluster, and used power and resource provisioning techniques to improve the energy proportionality. However, they manually reconfigured the cluster and resource providing strategies. In our work, the optimization is done by the multi-phases algorithm using the predicted workload information. Gae-Won You et al. [23] propose system Ursa which scales to a large number of storage nodes and objects and aims to minimize latency and bandwidth costs during system reconfiguration. Ursa tries to detect the hot spots in the system and re-balance these data with minimized transformation cost. Daniel Schall et al. [13–15] 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. Compared with our work, Ursa and WattDB try to optimize database itself in the storage level and query engine level to achieve energy saving. In Hot-N-Cold, the energy saving is achieved by solving resource provisioning problem and we do not optimize the database itself. Therefore, the approach can be used in other database as well. Houssem-Eddine Chihoub et al. [4] 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 extended this idea. The frequencies are set by means of frequency selection.

In Hot-N-Cold model, migration process is introduced to further improve the energy efficiency of cloud database systems. However, we only focus on the migration plan generation but the migration process. We refer the reader to Sudipto Das et al.'s work [7], in which they introduced a live database migration approach which can be used to conduct the migration process.

6. Discussion

In Section 2, the nodes are considered homogeneous. With the following extensions, the model can be applied to a 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 $z(c_i, f)$ should be specialized for different categories of nodes since the frequency options may not be the same.
3. The power consumption estimation function should be specialized for different categories of nodes.
4. When computing the migration cost, E_{In} and E_{Out} should consider the difference of node's architecture.

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

7. Conclusion and future work

At first, the energy efficiency problem in cloud database system 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 Hot-N-Cold model and corresponding algorithms. In Hot-N-Cold model, the nodes in the cluster are treated differently. According to predictions of workloads in the next time window, Hot-N-Cold model tries to assign a higher frequency to the nodes with higher predicted throughput and assign a lower frequency to the nodes with lower predicted throughput. If the workload exceeds node's capability (maximum throughput), a migration process is considered. Therefore, there are two tasks in Hot-N-Cold model: frequency selection and workload migration.

In Hot-N-Cold model, the resource-provisioning problem is considered as two bounded problems: $\mathbb{P}^b\mathbb{T}^{min}$ and $\mathbb{T}^b\mathbb{P}^{min}$. Nonlinear programming algorithm and multiphase algorithm are proposed to solve them. Nonlinear programming algorithm can obtain

a global optimal solution but has a poor scalability to large scale problems. In contrast, multiphase algorithm has a good scalability for large scale problems by means of approximation method. According to the experiment result in execution experiment, 21.5% energy can be saved maximum by means of Hot-N-Cold model

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

Acknowledgments

This paper is supported by the National Natural Science Foundation of China under Grant No. 61672143, U1435216. This research also is supported by the scholarship from China Scholarship Council (CSC, see www.campuschina.org) under the Grant CSC N° 201506080029. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see www.grid5000.fr).

References

[1] M. Amiri, L. Mohammad-Khanli, Survey on prediction models of applications for resources provisioning in cloud, J. Netw. Comput. Appl. 82 (2017) 93–113, http://dx.doi.org/10.1016/j.jnca.2017.01.016.

[2] V. Anand, C.M. Rao, MongoDB and Oracle NoSQL: A technical critique for design decisions, in: 2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS), IEEE, Pudukkottai, India, 2016, pp. 1–4, http://dx.doi.org/10.1109/ICETETS.2016.7602984.

[3] 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, O. Richard, Grid'5000: a large scale and highly reconfigurable grid experimental testbed, IEEE, Seattle, USA, 2005, p. 8 pp, http://dx.doi.org/10.1109/GRID.2005.1542730.

[4] H.-E. Chihoub, S. Ibrahim, Y. Li, G. Antoniu, M. Perez, L. Bouge, Exploring Energy-Consistency Trade-Offs in Cassandra Cloud Storage System, IEEE, Florianópolis-SC, Brazil, 2015, pp. 146–153, http://dx.doi.org/10.1109/SBAC-PAD.2015.28.

[5] B. Cooper, Yahoo! cloud Serving Benchmark. ULR https://github.com/brianfrankcooper/YCSB, 2010.

[6] G. Da Costa, D. Careglio, R. I. Kat, A. Mendelson, J.-M. Pierson, Y. Sazeides, Hardware leverages for energy reduction in large scale distributed systems, Technical Report IRIT/RT2010-2FR, IRIT, 2010.

[7] S. Das, S. Nishimura, D. Agrawal, A.E. Abbadi, Live database Migration for Elasticity in a Multitenant Database for Cloud Platforms, Tech. Rep. 2010–09, UCSB CS, 2010.

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

[9] S. Ibrahim, T.-D. Phan, A. Carpen -Amarie, H.-E. Chihoub, D. Moise, G. Antoniu, Governing energy consumption in Hadoop through {CPU} frequency scaling: An analysis, Future Gener. Comput. Syst. 54 (2016) 219–232, http://dx.doi.org/10.1016/j.future.2015.01.005.

[10] W. Lang, S. Harizopoulos, J.M. Patel, M.A. Shah, D. Tsirogiannis, Towards energy-efficient database cluster design, Proc. VLDB Endowment 5 (2012) 1684–1695, http://dx.doi.org/10.14778/2350229.2350280.

[11] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, 1990.

[12] B. Meindl, M. Templ, Analysis of commercial and free and open source solvers for the cell suppression problem, Trans. Data Privacy 6 (2) (2013) 147–159.

[13] D. Schall, T. Härder, Approximating an energy-proportional DBMS by a dynamic cluster of nodes, in: S.S. Bhowmick, C.E. Dyreson, C.S. Jensen, M.L. Lee, A. Muliantara, B. Thalheim (Eds.), Database Systems for Advanced Applications, Springer International Publishing, 2014, pp. 297–311.

[14] D. Schall, T. Härder, WattDB - a journey towards energy efficiency, Datenbank-Spektrum 14 (3) (2014) 183–198, http://dx.doi.org/10.1007/s13222-014-0168-8.

[15] D. Schall, T. Härder, Energy and performance - Can Wimpy-Node Cluster Challenge a Brawny Server? Hambury, 2015, pp. 197–216.

[16] J. Song, T. Li, X. Liu, Z. Zhu, Comparing and analyzing the energy efficiency of cloud database and parallel database, in: D.C. Wyld, J. Zizka, D. Nagamalai (Eds.), Advances in Computer Science, Engineering & Applications: Proceedings of the Second International Conference on Computer Science, Engineering & Applications (ICCSEA 2012), May 25-27, 2012, New Delhi, India. Volume 2, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 989–997.

[17] J. Song, T. Li, Z. Wang, Z. Zhu, Study on energy-consumption regularities of cloud computing systems by a novel evaluation model, Computing 95 (4) (2013) 269–287, http://dx.doi.org/10.1007/s00607-012-0218-8.

[18] B. Subramaniam, W. Feng, On the Energy Proportionality of Distributed NoSQL Data Stores, Vol. 8966, Springer, New Orleans, LA, USA, 2014, pp. 264–274, http://dx.doi.org/10.1007/978-3-319-17248-4+14.

[19] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Antony, H. Liu, R. Murthy, Hive - a petabyte scale data warehouse using Hadoop, in: 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), IEEE, Long Beach, California, United State, 2010, pp. 996–1005, http://dx.doi.org/10.1109/ICDE.2010.5447738.

[20] D. Tsirogiannis, S. Harizopoulos, M.A. Shah, Analyzing the energy efficiency of a database server, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, in: SIGMOD '10, ACM, Indianapolis, Indiana, USA, 2010, pp. 231–242, http://dx.doi.org/10.1145/1807167.1807194, URL http://doi.acm.org/10.1145/1807167.1807194.

[21] M.N. Vora, Hadoop-HBase for large-scale data, in: Proceedings of 2011 International Conference on Computer Science and Network Technology, Vol. 1, IEEE, Harbin, China, 2011, pp. 601–605, http://dx.doi.org/10.1109/ICCSNT.2011.6182030.

[22] Wikipedia, Zipf's law, URL https://en.wikipedia.org/wiki/Zipf'slaw, 2017.

[23] G.-W. You, S.-W. Hwang, N. Jain, Ursa: scalable load and power management in cloud storage systems, ACM Trans. Storage 9 (1) (2013) 1–29, http://dx.doi.org/10.1145/2435204.2435205.

[24] M. Zakarya, L. Gillam, Energy efficient computing, clusters, grids and clouds: A taxonomy and survey, Sustainable Comput.: Inf. Syst. 14 (2017) 13–33, http://dx.doi.org/10.1016/j.suscom.2017.03.002.

[25] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, J. Internet Serv. Appl. 1 (1) (2010) 7–18, http://dx.doi.org/10.1007/s13174-010-0007-6.

Chaopeng Guo received the B.S. degree in software engineering from Northeastern University of China in 2013. Currently, he is pursuing the doctoral degree at University Paul Sabatier. His current research focuses on energy efficiency, energy aware cloud database.

Jean-Marc Pierson received his Ph.D. in Computer Science from École Normale Supérieure de Lyon, LIP laboratory. He has a full professorship position at the University Paul Sabatier, Toulouse 3 since 2006. His research interesting includes Distributed Systems and High Performance Computing, Distributed Data management, Security Grid, Cloud Computing, and Energy Aware Distributed Computing.

Jie Song received the Ph.D. degree in computer science from Northeastern University in 2008. He is an associated professor of software college of Northeastern University of China. His main research interests include big data, data intensive computing, and energy efficient computing.

Christina Herzog obtained her Ph.D in Computer Science in 2015 from the University of Toulouse, France, where she worked at IRIT laboratory on multi-agent systems for Green IT. She is now associated with the LIUPPA, laboratory of computer science at the Université de Pau et des Pays de l'Adour. She founded the Efficit company in 2014, an IT service company. Besides, she focuses her research on energy efficiency, sustainability in clouds and software developments.