Novel algorithms and equivalence optimisation for resource allocation in cloud computing

Weiwei Lin* and Chaoyue Zhu
School of Computer Engineering and Science,
South China University of Technology,
Panyu, Guangzhou 510006, China
Email: linww@scut.edu.cn
Email: 1216282689@qq.com
*Corresponding author

Jin Li
School of Computer Science,
Guangzhou University,
Panyu, Guangzhou 510006, China
Email: jinli71@gmail.com

Bo Liu
School of Computer,
South China Normal University,
Guangzhou 510631, China
Email: liugubin530@126.com

Huiqiong Lian
School of Computer Engineering and Science,
South China University of Technology,
Panyu, Guangzhou 510006, China
Email: 920587784@qq.com

Abstract: In this paper, we model the optimisation of the resource allocation in cloud computing as a constraint satisfaction problem considering three types of resources (CPU, RAM and bandwidth) and design a Choco-Based algorithm (CB) for VM resource allocation in virtualised cloud data centres. We also propose an Improved First-Fit Decreasing Algorithm (IFFD) and an Improved Best-Fit Decreasing Algorithm (IBFD) and conduct performance evaluation experiments using Choco. The experimental results show that CB has better results, whereas its solution time is longer than IFFD and IBFD in resource allocation. Moreover, to reduce the complexity of solving the problem of CSP-based resource allocation, we propose an equivalence optimisation which can greatly reduce the search space for resource allocation by making tree pruning with resource equivalence. Then, a resource allocation algorithm based on Equivalent Optimisation (EO) is designed. Experimental results also show that compared with CB, EO greatly reduces the time of allocating resource of cloud computing.
1 Introduction

Cloud computing refers to both the applications delivered as services over the internet and the hardware and systems software in the data centres that provide those services (Foster et al., 2008; Vaquero et al., 2009; Taniar et al., 2008). Cloud computing is characterised as making services available on demand (Armbrust et al., 2009) unlike traditional database systems (Goel et al., 2005; Taniar and Wenny, 2008). The emergence of cloud computing (Armbrust et al., 2009; Buyya et al., 2009; Li et al., 2010; Flahive et al., 2013a; Flahive et al., 2013b) offers the flexibility of managing the computing resources in a more dynamic manner using the virtualisation technology to abstract, encapsulate and partition computing resources. Corporate data centres are in the process of adopting a cloud computing architecture where computing resources are provisioned
Novel algorithms and equivalence optimisation for resource allocation

based on demand to handle peak loads instead of being statically allocated. Such cloud infrastructures should improve the average utilisation rates of IT resources, which are currently in the range of 15–20%. However, this new computing paradigm has also raised new challenges to efficient resource management.

Different solutions (Abdelsalam et al., 2009; Beloglazov and Buyya, 2010; Bichler et al., 2006; Chaisiri et al., 2009; Chang et al., 2010; Lin et al., 2014; Hermenier et al., 2009; Ferreto et al., 2011; Garg et al., 2011; Lee and Zomaya, 2012; Rodero et al., 2010; Van et al., 2009a; Van et al., 2009b; Van et al., 2010; Wei et al., 2010; You et al., 2009; Younge et al., 2010) have been proposed to address the resource allocation problem in cloud computing over the past few years. Some resource scheduling methods of improving resource utilisation in cloud computing of data centre are presented in some research papers (Younge et al., 2010; Chang et al., 2010; Wei et al., 2010; Van et al., 2009a; Van et al., 2010; Chaisiri et al., 2009; Hermenier et al., 2009; Bichler et al., 2006; Khanna et al., 2006; Van et al., 2009b; Lin et al., 2011; Lin et al., 2014). The market-oriented resource allocation schemes based on market economy theory are proposed in the work of Beloglazov and Buyya (2010) and Van et al. (2009b). Some scheduling techniques of reducing energy consumption are discussed in the work of Beloglazov and Buyya (2010), Rodero et al. (2010), Abdelsalam et al. (2009), Lee and Zomaya (2012), Beloglazov et al. (2012) and Lee and Zomaya et al. (2012). Nonetheless, these works in cloud computing resource management did not consider the impact of network bandwidth. Resource allocation based only on the availability of CPU; memory resources may be delayed due to insufficient network bandwidth, resulting in the waste of CPU and memory resources. With the increasing trend towards more communication intensive applications in data centres, the bandwidth usage among virtual machines (VMs) is rapidly growing (Govindan et al., 2009; Meng et al., 2010). It becomes very important for data centre to allocate suitable bandwidth for VMs according to application’s requirements and environment conditions to balance traffic load. Recent studies (Van et al., 2010; Hermenier et al., 2009) have shown that, even only considering CPU and memory resources in resource allocation of cloud computing, the problem is still NP complete, and adding network bandwidth into the mix will further increase the complexity. In this paper, we use a constraint programming approach to formulate and solve the optimisation problem of resource allocation in cloud computing. The difference between our approach (Lin et al., 2013) and that used by other researchers (Van et al., 2010; Hermenier et al., 2009; Ferreto et al., 2011) is that our resource allocation considers network bandwidth constraint. By solving this constraint satisfaction problem using the CHOCO constraint programming solver (Jussien et al., 2008), we can get the optimised number of physical machines (PMs) that host virtual machines (VMs). However, when there are too many resources to be allocated, the CSP approach fails in finding an optimal solution in a short time. So, we present an equivalence optimisation that uses resource equivalence to make tree pruning which can greatly reduce the search space for resource allocation. Then we design a Choco-based algorithm (CB) for the resource allocation and a resource allocation algorithm based on Equivalent Optimisation (EO). Moreover, heuristics algorithms such as First-Fit Decreasing (FFD) (Ferreto et al., 2011) and Best-Fit Decreasing (BFD) (Ferreto et al., 2011) have been used popularly for VM allocation in virtualised data centres. However, FFD and BFD are conducted on single constraint variable. The idea of the algorithm is keeping a dynamic sorted list according to a single constraint variable. In this paper, we take three types of resources (RAM, CPU and bandwidth) into account and we propose an improved First-Fit Decreasing Algorithm (IFFD) algorithm and an improved Best-Fit Decreasing Algorithm
(IBFD) algorithm. To evaluate the effectiveness of the equivalence optimisation in resource allocation and the proposed algorithms, we conduct performance evaluation using Choco and Java. The experimental results show that IFFD and IBFD algorithms are effective in finding a feasible solution in a short time. However, CB and EO algorithms can get better solutions for the resource allocation problem than IFFD and IBFD.

The rest of the paper is organised as follows. Section 2 overviews the related work. In Section 3, we propose a resource allocation optimisation model for cloud computing which considers CPU power, RAM size and network bandwidth. We provide detailed analysis and performance studies of our algorithms and the proposed equivalence optimisation in Section 4. In Section 5, we conduct performance studies using simulation, and analyse the performance of the proposed resource allocation model under different resource usage scenarios compared with IFFD and IBFD. Finally, we conclude our concluding remarks and discuss future studies in Section 6.

2 Related work

Recently, there are some researches which focus on resource management in cloud computing. Beloglazov and Buyya (2010) proposed a market-oriented resource allocation scheme that integrated both customer-driven service management and computational risk management to sustain Service Level Agreement (SLA)-oriented resource allocation. Another market-based resource allocation strategy, RAS-M, was proposed by You et al. (2009). This scheme is based on market economy theory, where the problem of resource allocation is transformed into finding the equilibrium price vector and corresponding equilibrium solution, and a GA-based price adjusted algorithm is introduced to deal with the problem. But RAS-M scheme is proposed for resource allocation at the physical level of the cloud computing, and it only manages the CPU resource. Beloglazov and Buyya (2010) proposed an energy efficient resource management system for virtualised cloud data centres that reduces operational costs and provides required Quality of Service (QoS). Energy saving is achieved by continuous consolidation of VMs according to current utilisation of resources, virtual network topologies established between VMs and thermal state of computing nodes. Rodero et al. (2010) presented an energy-aware online provisioning approach for HPC applications on consolidated and virtualised computing platforms. Energy efficiency is achieved using a workload-aware, just-right dynamic provisioning mechanism and the ability to power down subsystems of a host system that are not required by the VMs mapped to it. Abdelsalam et al. (2009) created a mathematical model for power management in a cloud computing environment that primarily serves clients with interactive applications such as web services. The mathematical model computes the optimal number of servers and the frequencies at which they should run. Young et al. (2010) presented a new framework that provides efficient green enhancements under a scalable cloud computing architecture. Using power-aware scheduling techniques, variable resource management, live migration, and a minimal virtual machine design, overall system efficiency is improved in data centres based on cloud with minimal overhead. Chang et al. (2010) studied optimal resource allocation in clouds and formulated demand of computing power and other resources as a resource allocation problem with multiplicity. They presented an approximation algorithm and a proof of its approximation bound that can yield closely to optimum solutions in polynomial time. Wei et al. (2010) used game theory to handle the resource allocation in cloud computing. In their approach, a binary integer programming method
Novel algorithms and equivalence optimisation for resource allocation

is proposed to solve the parallel tasks allocation problem on remote machines connected across the internet. Their algorithms take both optimisation and fairness into account and provide a relatively good compromise resource allocation.

Van et al. (2009b) proposed an autonomic resource manager to control the virtualised environment, which decouples the provisioning of resources from the dynamic placement of virtual machines. The manager aims to optimise a global utility function which integrates both the degree of SLA fulfilment and the operating costs. Van et al. (2010) proposed a resource management framework that combines a utility-based dynamic virtual machine provisioning manager and a dynamic VM placement manager. Both problems are modelled as constraint satisfaction problems. The VM provisioning process aims at maximising a global utility capturing the performance of the hosted applications. Chaisiri et al. (2009) proposed an Optimal Virtual Machine Placement (OVMP) algorithm. The algorithm can minimise the cost spending in each plan for hosting virtual machines in a multiple cloud provider environment under future demand and price uncertainty. Hermenier et al. (2009) proposed a new approach, entropy, in a homogeneous cluster environment, which takes into account both the problem of allocating the VMs to available nodes (physical hosts) and the problem of migrating the VMs to these nodes. The performance overhead is determined by the time required to choose a new configuration and to migrate VMs according to the configuration. The entropy resource manager can choose migrations that can be implemented efficiently, incurring a low performance overhead. Several optimisation approaches, based on the bin packing problem, for configuring virtualised servers are described in the literature (such as Bichler et al., 2006; Khanna et al., 2006). Garg et al. (2011) proposed near-optimal scheduling policies that exploit heterogeneity across multiple data centres for a cloud provider. They consider a number of energy efficiency factors (such as energy cost, carbon emission rate, workload, and CPU power efficiency) which vary across different data centres depending on their location, architectural design and management system. Additionally, most of these approaches only consider the processor as the unique resource, leaving out other important resources such as memory and storage. The massive aggregation of workloads in the same node might represent bottleneck, which reduces the performance and increases the energy consumption as mentioned in the work of Lee and Zomaya (2012).

Unlike the existing methods, we model the optimisation of the resource allocation of cloud computing as a constraint satisfaction problem considering three types of resources (RAM, CPU and bandwidth) and use constraint programming to solve the CSP-based resource allocation model. The constraint programming approach can find the optimal solution of the resource allocation problem. In addition, we propose an equivalence optimisation that uses resource equivalence to make tree pruning which can greatly reduce the search space for the resource allocation. As a result, the equivalence optimisation approach is very effective in finding optimal solutions of the resource allocation problem in cloud computing.

3 Resource allocation model and equivalence optimisation

3.1 Resource allocation model

A key enabling technology of cloud systems is server virtualisation which allows developers to decouple applications and services from the physical server infrastructure.
Server virtualisation makes it possible for several virtual machines (VMs) to execute concurrently in a single physical machine (PM), where each VM hosts a complete software stack (operating system, middleware, applications) and is given a partition of the underlying resource capacity (CPU power, RAM size and network bandwidth). Virtualisation provides a means for server consolidation and allows for migration and dynamic allocation of these VMs on demand. The dynamic allocation problem describes the decision of how many servers are required overall and how VMs are allocated to servers in the individual time steps (e.g. time slot is 3 h). The optimisation of resource allocation is to minimise the number of PMs needed to host all VMs.

To express the optimisation of resource allocation as a CSP, we consider a cloud computing data centre (Figure 1), which consists of a set of PMs and VMs needed to be assigned. The goal of the optimisation is to find a resource allocation strategy that minimises the number of PMs. Let \( P = (p_1, p_2, \ldots, p_j, \ldots, p_q) \) denote the set of PMs in the data centre \( C = (c_1, c_2, \ldots, c_j, \ldots, c_q) \) be the resource capacity (CPU, RAM, BW) of all PMs, where \( c_j = (c_j^{cpu}, c_j^{ram}, c_j^{bw}) \). Let \( V = (vm_1, vm_2, \ldots, vm_l, \ldots, vm_v) \) denote the set of VMs needed to be assigned to PM and \( R = (r_1, r_2, \ldots, r_l, \ldots, r_r) \) denote the resource capacity (CPU, RAM, BW) of all VMs, where \( r_i = (r_i^{cpu}, r_i^{ram}, r_i^{bw}) \). For each PM \( p_j \in P \), the bit vector \( H_j = (h_{j1}, h_{j2}, \ldots, h_{jl}, \ldots, h_{jv}) \) denotes the set of VMs assigned to \( p_j \) (i.e. \( h_{jl} = 1 \) if \( p_j \) is hosting \( vm_l \)). For each PM \( p_j \in P \), its remaining resource capacity (CPU, RAM, BW) must satisfy all the VMs assigned to it (its CPU capacity cannot be lower than the total CPU capacity of VM assigned, the same to its RAM and BW). All we want is to get the minimum \( X \) of PMs needed to satisfy the VMs resource requirement. The status of PMs is expressed by an array \( \{u_j\} \), where \( u_j = 1 \) (\( \exists vm_l \in V \ | \ h_{jl} = 1 \)), and 0 otherwise. We express the physical resource constraints as follows:

\[
\sum_{i=1}^{l} r_{ij}^{cpu} h_{jl} \leq c_j^{cpu} \quad 1 \leq j \leq q
\]
(1)

\[
\sum_{i=1}^{l} r_{ij}^{ram} h_{jl} \leq c_j^{ram} \quad 1 \leq j \leq q
\]
(2)

\[
\sum_{i=1}^{l} r_{ij}^{bw} h_{jl} \leq c_j^{bw} \quad 1 \leq j \leq q
\]
(3)

The goal is to minimise the number of active PMs, \( X \) as:

\[
X = \sum_{j=1}^{q} u_j
\]
(4)

where \( u_j = 1 (\exists vm_l \in V \ | \ h_{jl} = 1) \) and 0 otherwise.

The solution of the VM packing CSP produces the VM placement vectors \( H_j \) which are used to place VMs on PMs.
Novel algorithms and equivalence optimisation for resource allocation

Figure 1 Resource allocation in cloud computing data centre

3.2 Equivalence optimisation of resource allocation

In essence, the above-mentioned resource allocation problem is a combinatorial optimisation problem. That is, it is an NP-hard problem that selects the best one from a variety of all the possible resource allocation combinations. When the scale of the problem is small, we can use the branch and bound method or dynamic programming methods to solve it. However, when the scale of the problem is large (e.g. the number of physical servers is 1000 and the number of virtual machines needed to pack is 10,000), the solution space is large and it is very difficult to obtain the optimal solution within a reasonable time. But we can use constraint programming-based backtracking to find a feasible solution of the problem. Moreover, we found that many combinations of physical machines and virtual machines are equivalent. For example, when the resources (CPU, RAM, BW) capacities of physical machine $p_i$ and $p_j$ are equal, the virtual machine $v_x$ can be placed on the physical machine $p_i$ if the virtual machine $v_x$ cannot be placed on the physical machine $p_j$. On the contrary, the virtual machine $v_x$ should not be placed on the physical machine $p_i$ if the virtual machine $v_x$ cannot be placed on the physical machine $p_j$. So, we can use the equivalences to optimise the resource allocation in cloud computing. These equivalences are defined as follows:

For physical machines $p_i$ and $p_j$, if their resources (CPU, RAM, BW) are all equal in the capacity, then they are equivalent. The equivalence of physical machine is defined as follows:

$$\forall p_i, p_j \in P \mid p_i = p_j \Leftrightarrow C_p(p_i) = C_p(p_j) \land C_m(p_i) = C_m(p_j) \land C_b(p_i) = C_b(p_j)$$

(5)

An equivalence class of physical machines consists of all equivalent physical machines.

For virtual machines $v_x$ and $v_y$, if their resources (CPU, RAM, BW) are all equal in the capacity, then they are equivalent. The equivalence of virtual machine is defined as follows:

$$\forall v_x, v_y \in V \mid v_x = v_y \Leftrightarrow C_p(v_x) = C_p(v_y) \land C_m(v_x) = C_m(v_y) \land C_b(v_x) = C_b(v_y)$$

(6)

An equivalence class of virtual machines consists of all equivalent virtual machines. Based on the idea of the equivalence optimisation, we design the constraint rules as follows:
The physical machine exclusion rule states that the virtual machine \( v_x \) should not be placed on the physical machine \( p_j \), which is equivalent with the physical machine \( p_i \) if the virtual machine \( v_x \) cannot be placed the physical machine \( p_i \):

\[
\exists v_x \in V, \forall p_i, p_j \in P | p_i = p_j, h_{ij} = 0 \Rightarrow h_{ji} = 0
\]

(7)

The virtual machine exclusion rule states that the physical machine \( p_i \) should not host the virtual machine \( v_x \), which is equivalent with the virtual machine \( v_y \) if the physical machine \( p_i \) cannot host the virtual machine \( v_y \):

\[
\exists p_i \in P, \forall v_x, v_y \in V | v_x = v_y, h_{xi} = 0 \Rightarrow h_{yi} = 0
\]

(8)

These constraint rules are used in the CSP-based resource allocation model to greatly reduce the search space for the resource allocation and optimise the solving of resource scheduling problem.

4 The proposed algorithms for VM resource allocation

4.1 Implementation of CB

The basic idea of Choco-Based algorithm (CB) for VM resource allocation is to find the minimum number of PMs that can host all VMs and output the VM allocation in this case. The implementation of CB is described as follows.

Input: an array, the resource vector \( C \) (CPU, RAM, BW) of PMs.
Output: the allocation matrix \( H \) (\( h_{jl} = 1 \) if \( p_j \) is hosting \( v_{ml} \)).

Step 1: initialise the physical machine resource vector \( C \) (CPU, RAM, BW);
Step 2: initialise the virtual machine resource vector \( R \) (CPU, RAM, BW);
Step 3: initialise the allocation matrix \( H \) (\( \forall j,l | h_{jl} = 0 \));
Step 4: set the constraint conditions ensuring every VM can distribute to one and only one PM;
Step 5: read the arrays of VM and PM, and add the constraint condition formulas (1)–(3) above;
Step 6: define the goal function minimizes (X) according to formula (4) above;
Step 7: launch the solving program and get all the optimisation results;
Step 8: return the allocation matrix \( H \).

In the above steps of the implementation of CB, Step 7 is the most important. We use the Choco constraint solver to solve the resource allocation optimisation model described in step 7. When using Choco to model and solve the constraint problem, we have to describe the problem through some interfaces and functions provided in Choco, e.g. the function addConstraint() in class CPModel which is used to add constraints to the constraint satisfaction model. Our goal is to allocate all the VMs under the constraint conditions to minimise \( SUM \) as an optimisation solution, so we need to define a variable \( SUM \) which expresses the number of active PMs. And the function in Choco is minimise \( (s.getVar(SUM), true) \). To reduce the time of the solution, we use the result of BFD algorithm as the lower bound of SUM. The main Java code we use to model and solve the resource allocation model based on constraint programming is depicted as follows:
public static void solve()
{
    ... 
    /* ensuring every VM can distribute to one and only one PM */
    for(int i=0;i<VMn;i++)
        m.addConstraint(Choco.eq(Choco.sum(pos[i]),1));
    ...
    /* ensuring each PM resource can satisfy all the requirement of VM assigned to it*/
    for(int i=0;i<PMn;i++){
        m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMCPU),PMCPU[i]));
        m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMRAM),PMRAM[i]));
        m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMBW),PMBW[i]));
    }
    ...
    /* set the optimisation goal of minimising the number of PMs */
    Solver s = new CPSolver();
    s.read(m);
    s.minimise(s.getVar(SUM), true);
    s.setObjective(s.getVar(SUM));
    s.solveAll();
    s.launch();
}

4.2 Implementation of IFFD

The purpose of the improved algorithm is to find as few PM as possible to satisfy all the VMs’ requirements. We keep a dynamic sorted list in the whole running process of FFD. The process is shown as follows:

Input: hostList and vmList with the resource capacity (CPU, RAM, BW).
Output: the allocation of VMs.

Step 1: Clear the usedPMlist and copy all the PMs to unusedPMlist;

Step 2: For each PM, calculate its \( x_j = r_{j}^{\text{cpu}} * p_{j}^{\text{cpu}} + r_{j}^{\text{ram}} * p_{j}^{\text{ram}} + r_{j}^{\text{bw}} * p_{j}^{\text{bw}} \);

Step 3: For each VM, check the first machine of usedPMlist. If the PM can satisfy VM’s requirement, assign the VM to it, and reinsert the PM to usedPMlist;

Step 4: If cannot, find one PM which can satisfy the VM’s requirement in all the PMs on unusedPMlist and assign the VM to it. And remove the PM from unusedPMlist to usedPMlist;

Step 5: If there is not such PM in unusedPMlist, the allocation fails and exits;

Step 6: return the allocation of VMs.
4.3 Implementation of IBFD

IBFD algorithm is similar to IFFD, and the difference between IBFD and IFFD is that in step 3 IBFD checks all the PMs of usedPMlist, while IFFD only checks the first PM. The detail is listed as following:

Input: hostList and vmList with the resource capacity (CPU, RAM, BW).
Output: the allocation of VMs.

Step 1. Clear the usedPMlist and copy all the PMs to unusedPMlist;
Step 2. For each PM, calculate its $x$: $x_j = r_{j,\text{cpu}} \cdot p_{\text{cpu}} + r_{j,\text{ram}} \cdot p_{\text{ram}} + r_{j,\text{bw}} \cdot p_{\text{bw}}$;
Step 3. For each VM, check all machines of usedPMlist and find the first PM that can satisfy VM’s requirement, assign the VM to it, and reinsert the PM to usedPMlist;
Step 4. If cannot, find one PM which can satisfy the VM’s requirement in all the PMs on unusedPMlist and assign the VM to it. And remove the PM from unusedPMlist to usedPMlist;
Step 5. If there is not such PM in unusedPMlist, the allocation fails and exits;
Step 6: return the allocation of VMs.

4.4 Implementation of EO

The resource allocation algorithm based on Equivalent Optimisation (EO) is similar to CB. The differences between EO and CB are the resource allocation model and its solution. In EO, we add equivalent resource constraints to the resource allocation model and reduce the time required to solve the resource allocation model by using the equivalence optimisation. The main Java codes we use to model and solve the resource allocation model based on constraint programming are depicted as follows:

```java
public static void solve(){
    ...
    /* ensuring every VM can distribute to one and only one PM */
    for(int i=0;i<VMn;i++)
        m.addConstraint(Choco.eq(Choco.sum(pos[i]),1));
    ...
    /* ensuring each PM resource can satisfy all the requirement of VM assigned to it */
    for(int i=0;i<PMn;i++)
        { // ensurin location[i]=j <> pos[j]=1
            m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMCPU),PMCPU[i]));
            m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMRAM),PMRAM[i]));
            m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMBW),PMBW[i]));
        }
    // ensures all PM resource can satisfy all the requirement of VM assigned to it
    for(int i=0;i<PMn;i++)
        { // ensuring location[i]=j <> pos[j]=1
            m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMCPU),PMCPU[i]));
            m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMRAM),PMRAM[i]));
            m.addConstraint(Choco.leq(Choco.scalar(dualpos[i], VMBW),PMBW[i]));
        }
}
```
for(int i=0; i<VMn; i++)
    m.addConstraint(Choco.domainChanneling(location[i], pos[i]));

// add the VM equivalent constraint to the model
m.addConstraint(new ComponentConstraint(new ConstraintManager.class, tttt, location));

/* set the optimisation goal of minimising the number of PMs*/
Solver s = new CPSolver();
s.read(m);
s.minimise(s.getVar(SUM), true);
s.setObjective(s.getVar(SUM));
s.solveAll();
s.launch();
}

5 Experiments and results

5.1 The comparison of CB, FFD, IFFD and IBFD

The complexity of CB, IBFD or IFFD depends on the number of PMs and VMs. In the experiments, we randomly generate different scale of the experimental data (i.e. different number of PM and VM) to measure four different algorithms including CB, FFD, IFFD and IBFD. Since the physical machine and application are often heterogeneous, we set the range of the resource size of PMs and VMs in the experiments. We set the resource of physical machines and virtual machines as shown in Table 1.

<table>
<thead>
<tr>
<th>Physical machine</th>
<th>CPU (Hz)</th>
<th>RAM (MB)</th>
<th>BW (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper</td>
<td>8000</td>
<td>8000</td>
<td>8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtual machine</th>
<th>CPU (Hz)</th>
<th>RAM (MB)</th>
<th>BW (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper</td>
<td>4000</td>
<td>4000</td>
<td>1000</td>
</tr>
<tr>
<td>Lower</td>
<td>500</td>
<td>500</td>
<td>100</td>
</tr>
</tbody>
</table>

5.1.1 The results of the resource allocation

After implementing the proposed algorithms, we use different sets of resources to measure CB, FFD, IFFD and IBFD. In the experiments, we set the number of PM from 4 to 10 and set the number of VM from 4 to 16. The results of the resource allocation are presented in Figures 2a and 2b where the number of VMs is small and large, respectively, and we assume the unused physical machines are always enough.
Figure 2  Number of used PMs (see online version for colours)

Figure 2a shows the number of used PMs by four algorithms (FFD, IFFD, IBFD and CB) when the number of VMs is small. And Figure 2b shows that the number of used PMs by three algorithms (FFD, IFFD and IBFD) when the number of VMs is large. We can conclude from the experimental results: (a) the number of used PMs by CB is less than others; (b) the number of PMs used by IFFD or IBFD is less than that of FFD, especially when the numbers of VMs and PMs are large. But FFD, IFFD and IBFD cannot ensure the solution is optimal. CB is always guaranteed to find optimisation (i.e. using the least PMs to satisfy all VMs requirement); however, CB cannot obtain the solution within the specified time when the numbers of VMs and PMs are large.
### 5.1.2 Running time comparison among FFD, IBFD, IFFD and CB

The running time of four algorithms is presented as follows. The running time of CB is shown in Figure 3. The running time of FFD, IFFD and IBFD is shown in Figure 4a and 4b where the number of VMs is small and large, respectively, and we assume the unused physical machines are always enough.

**Figure 3** Running time of CB (see online version for colours)

![Figure 3](image-url)

It can be seen from the experiment result (Figure 3) that when the number of VMs and PMs increases, the running time of Choco increases fast. For example, with 10 PMs and 16 VMs, it costs more than 400 s to finish the resource allocation. But Choco provides time limit to return optimise solution ever found by using setTimeLimit(), so we can avoid getting no solution during long time. It can be seen from the experimental result in Figure 4a and 4b that the running time of IFFD or IBFD is more than that of FFD. But the running time of FFD, IFFD and IBFD is relatively small.

**Figure 4** Running time (see online version for colours)

![Figure 4](image-url)

(a) Running time of FFD, IFFD and IBFD
Figure 4  Running time (see online version for colours) (continued)

5.2 Performance evaluation of EO

To evaluate the effectiveness of the proposed equivalence optimisation of resource allocation, we conduct experiments with CB and EO algorithms under the same condition and compare the performance of these two algorithms. We generate five sets of configurations, each with four equivalence classes of VMs. All the sets use 100 PMs, each with an 8 GHz CPU, 8 GB of RAM and 8 Mbps. But they differ in the number of VMs. The numbers of five sets are, respectively, 25, 50, 100, 150 and 200. The number of PMs used by EO is the same as CB, but the running time of CB and EO is different. Figure 5 shows their running time. We can see that the running time of CB is significantly larger than EO, especially when the number of VMs needed to be assigned is large. We can conclude from the experimental results that the proposed equivalence optimisation of resource allocation can greatly reduce the time of the allocating resource in cloud computing.

Figure 5  Running time of CB and EO (see online version for colours)
In addition, to observe the impact of the classes of equivalence on the duration of solving process, we have generated four sets of configurations composed of 200 VMs and 100 PMs, which differ in the number of classes of equivalence. The first set has two classes of equivalence, where all VMs consist of two capacity types of virtual machines. The second set and the third set have four and six, respectively, classes of equivalence. The last set has eight classes of equivalence, where all VMs consist of eight capacity types of virtual machines. Figure 6 shows the running time of EO with four different sets of configurations. We observe that the running time for EO is affected by the number of equivalence classes. About 8 s is necessary to solve the first set of configuration with two classes of equivalence, about 40 s for the second set of configuration and 140 s for the last. It can be seen from the experimental result (Figure 6) that when the number of equivalence classes increases, the running time of EO increases significantly.

Figure 6  Running time of EO with different equivalence classes (see online version for colours)

6 Conclusions and future work

Resource scheduling is one of the most important problems in cloud computing. However, it is a big challenge to design and implement an efficient resource scheduling algorithm, since general scheduling problem is NP-complete. To address the problems in existing resource allocation schemes for cloud data centres, we model the optimisation of the resource allocation as a CSP considering three types of resources (RAM, CPU and bandwidth) and design a Choco-Based algorithm (CB) for VM resource allocation. We also propose an improved FFD (IFFD) and an improved BFD (IBFD), and conduct performance evaluation experiments using Choco and Java. The experimental results show that IFFD and IBFD algorithms are effective in finding a feasible solution in a short time. However, CB and EO algorithms can get better solutions for the resource allocation problem of cloud computing than IFFD and IBFD. To reduce the complexity of the solution of the CSP-based resource allocation model, we present an equivalence optimisation that uses resource equivalence to make tree pruning which can greatly
reduce the search space for the resource allocation. We also design a resource allocation algorithm based on Equivalent Optimisation (EO) and conduct a performance evaluation. The experimental results show that the proposed equivalence optimisation of resource allocation can greatly reduce the time of allocating resource of cloud computing.

With the development of cloud computing theory and technology, cloud applications are becoming increasingly popular. Therefore, more and more cloud data centres are being built. The efficient resource scheduling is one of the most important issues in cloud data centres. The proposed novel scheduling algorithms can be applied to improve resource utilisation and reduce energy consumption in cloud data centres. Especially, the equivalence optimisation can reduce the complexity of large-scale cloud resource scheduling, so it will have a very broad application prospects in green cloud data centres. Currently, we are trying to improve cloud data centre energy efficiency using the proposed heuristic algorithms (IFFD and IBFD) and EO. Next steps will be focused on implementing our algorithm in the Xen virtual machine monitor.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that have improved the presentation, quality and correctness of this paper. This work is financially supported by the National Natural Science Foundation of China (grant nos. 61402183, 61272382 and 61100224), Guangdong Natural Science Foundation (grant no. S201203006242), Guangdong Provincial Science and Technology Projects (grant nos. 01051860218297035 and 00269240218968034), the Fundamental Research Funds for the Central Universities, SCUT (2013ZZ0044), GuangZhou City Science and Technology Projects (grant no. 1517000373), and the Guangzhou Zhujiang Science and Technology Future Fellow Fund (grant no. 2012J2200094).

References


Novel algorithms and equivalence optimisation for resource allocation


Novel algorithms and equivalence optimisation for resource allocation


