Journal of Software Engineering and Applications, **2013, 6, 1-13**
http://dx.doi.org/10.4236/jsea.2013.61001 Published Online January 2013 (http://www.scirp.org/journal/jsea)

Heuristic Scheduling Algorithms for Allocation of Virtualized Network and Computing Resources

Yichao Yang, Yanbo Zhou, Zhili Sun, Haitham Cruickshank

The Centre for Communication Systems Research (CCSR), Faculty of Engineering and Physical Sciences, University of Surrey, Surrey, UK.
Email: y.yang@surrey.ac.uk, y.zhou@surrey.ac.uk, z.sun@surrey.ac.uk, h.cruickshank@surrey.ac.uk

Received September 16th, 2012; revised October 15th, 2012; accepted October 24th, 2012

ABSTRACT

Cloud computing technology facilitates computing-intensive applications by providing virtualized resources which can be dynamically provisioned. However, user's requests are varied according to different applications' computation ability needs. These applications can be presented as meta-job of user's demand. The total processing time of these jobs may need data transmission time over the Internet as well as the completed time of jobs to execute on the virtual machine must be taken into account. In this paper, we presented V-*heuristics scheduling algorithm* for allocation of virtualized network and computing resources under user's constraint which applied into a service-oriented resource broker for jobs scheduling. This scheduling algorithm takes into account both data transmission time and computation time that related to virtualized network and virtual machine. The simulation results are compared with three different types of heuristic algorithms under conventional network or virtual network conditions such as MCT, Min-Min and Max-Min. e evaluate these algorithms within a simulated cloud environment via an abilene network topology which is real physical core network topology. These experimental results show that V-*heuristic scheduling algorithm* achieved significant performance gain for a variety of applications in terms of load balance, Makespan, average resource utilization and total processing time.

Keywords: Cloud Computing; Meta-Job Scheduling; Heuristic Algorithm; Load Balance; Network Virtualization

1. Introduction

Cloud computing technology is a new paradigm for utility virtualized resources, designed for end users in a dynamic computing environment to provide reliable and guaranteed services [1-4]. In this environment, on-demand services of computing-intensive applications can be increased by user requests. According to the number of users rapidly increasing, the virtualization is one of main techniques to improve the utilization of physical resources in cloud environments [5]. It allows abstraction and isolation of underlying physical resource and reduces the number of hardware equipment. These virtualization techniques include network virtualization and allow the operator to create several Virtual Machines (VMs) on a single physical server. Specifically, VMs can be designed by increasing or decreasing the CPU power and/or the number of CPUs [6].

In order to efficiently utilize the virtualized resources to execute computing-intensive application, the effective meta-job scheduling algorithms are needed. The traditional job scheduling problem schedule meta-job of applications across computation resources in order to reduce the jobs completed time while ignoring the specific shared nature of the network resource [6,7]. In a super computer data center, the scheduling problem is enhanced by scheduling a set of applications from different users to the set of computation resources while maximizing system utilization. Previous researchers have been conducted in this area leading to an extensive study of the major theoretical and practical results [8]. However, with the emergence of the virtualization techniques in computational cloud systems, new scheduling algorithms are required to deal with concerns originating from the cloud infrastructure. In cloud computing, the objective of job scheduling algorithm is to achieve high system throughput, improve the load balance and minimizing the meta-job total processing time while matching the meta-job requirements with available virtualized resources.

Job scheduling is a general problem of mapping a set of jobs to a set of VMs to fulfill the user's requests within cloud environment. The objective of job scheduleing is to achieve high system throughput and minimizing the meta-job total processing time while matching the meta-job requirements with available virtualized resources. The scheduler in this environment needs to con-

sider virtualized resources and user's required constraint to get better match between applications and resources. When users consider a variety of network resources related to the quality of service, job scheduling becomes more complicated in cloud computing.

Many traditional heuristics scheduling algorithms such as Minimum Completion Time (MCT), Minimum-Minimum Completion Time (Min-Min), Maximum-Minimum Completion Time (Max-Min) do not consider the network impact of cloud applications. We present a V-heuristic scheduling algorithm to address the match of jobs from the application and the status provided by the diverse virtualized resources in cloud computing. Consequently, the scheduling algorithm improved the efficiency and the utilization of a cloud system. To provide this service, cloud service provider (CSP) deploys virtual network from infrastructure provider (InP) and one or multiple VMs on single physical machine that are coordinated together. Within service provider, heterogeneous physiccal networks are abstracted by using virtualization technologies which can provide efficient data transmission among of VMs in cloud computing. We can provide heuristic scheduling algorithm for optimal virtualized resources utilization, maximum throughput, minimize the job processing time, load balancing over VMs and satisfy on-demand services of user requests.

In this paper, we are experiment with different computation applications by varying its computation power need and file size of data. The simulation results are compared with 3 different types of heuristic scheduling algorithms under conventional physical network and computation resource or virtualized network and VMs conditions such as MCT, Min-Min and Max-Min. We evaluate these algorithms within a simulated cloud environment via an abilene network topology [9]. These experimental results show that heuristic scheduling algorithm achieved significant performance gain for a variety of applications in terms of load balance, makespan, average resource utilization and total processing time.

2. Related Work

Currently, there are some existing heuristics scheduling algorithms which schedules meta-job to a set of resources [6]. A meta-job can be defined as a collection of independent jobs with no inter-job data dependency. Previous research has been related to grid job scheduling, to solve the problem of mapping a set of jobs to a set of computing resources. These existing heuristic algorithms are applied in a grid environment for utilizing a distributed different high-performance computing resources with high-speed network to perform computationally intensive applications. However, the network will surely effect these applications according to the diverse user's request. Therefore, the virtualized network is important parameter for the scheduling algorithm in cloud environments, which aim to efficient utilized potential computation resources. The problem of matching jobs to computation resources by considering a network resource in cloud system has been proven is an NP-complete problem [8].

The definitions of the static meta-job mapping heuristics are provided. Some preliminary terms must be defined. Machine availability time t_i is the earliest time a machine j can complete the execution of all the jobs that have previously been assigned to do it. Completion time $CT[i, j]$ is the machine availability time plus the execution time of job u_i on machine, *i.e.* $CT[i, j] = t_i + ET[i, j]$ for $0 \le i \le w$ and $0 \le j \le y$, (where: w is the number of jobs; y is the number of machines). The performance criterion used to compare the results of the heuristics is the maximum value of $CT[i, j]$ for each mapping, it is also known as the makespan. Therefore, each heuristic is attempting to minimize the makespan (*i.e.*, finish execution of the meta-job as soon as possible).

Existing matching heuristics can be separated into two modes that include on-line and batch mode [6]. The online mode heuristic is represented for mapping a job to a resource while it arrives to the scheduler. Each job is considered for matching and scheduling at once, such as the MCT and the MET heuristics. This mode of matching is efficient when the job arrival rate is low. In batch mode heuristic that is collected into a set of job, after that mapping jobs are performed at prescheduled times called mapping events. Each job matching to resource is performed at every mapping event, until it begins its execution on resource.

2.1. MCT

The main ideal of the MCT heuristic algorithm is that allocates every job in arbitrary order to the resource, and that resource has minimum completion time to that assigned job [6]. The completion time is computed by adding the expected execution time of a job on that resource with the ready time of the resource. This causes some jobs to be allocated to resources without have minimum execution time for them.

2.2. Min-Min

The Min-Min heuristic is one of scheduling algorithms which starts with computing the minimum expected completion time for every unallocated job in a set of meta-jobs [6]. The next phase, found the job with a minimum of overall minimum expected completion time from a set of minimum expected completion time in that set of meta-jobs. Final phase, the job with the overall minimum expected completion time is chosen and allocated to the

corresponding resource. The ready time of the resource is updated. After that, the job is removed from meta-job and repeats the process until all unallocated jobs are all allocated. Compare with MCT, this algorithm all jobs set at a time after finish matched. The purpose of the Min-Min is trying to allocate many more jobs to the resource which has minimum expected completion time and power capability, eventually minimized the total completion time of all the jobs. In fact, this heuristic algorithm begins with the set U of all unmapped jobs. Then, the set of minimum completion time

$$SCT = \left\{ m_j : m_j = \min_{0 \le j < y} \left(CT[i, j] \right), \text{for } i \in w \right\}$$

is found for each unmapped job. Next, the job i with the overall minimum completion time from set of SCT is selected and assigned to the corresponding machine. Hence it is named Min-min. Then the workload of the selected machine will be updated and finally the newly mapped job is removed from U. This process repeats until all jobs are mapped (*i.e.* U is empty).

2.3. Max-Min

The Max-Min heuristic algorithm is much similar to Min-Min, unless the second phase of the procedure [6]. It allocates the job with a maximum of overall minimum expected completion time to the corresponding resource in this phase. The general ideal of the Max-Min algorithm is first compute the minimum expected completion time for every job in set of the meta-jobs; again find the job with maximum of overall minimum expected completion time and the corresponding resource; finally allocate the job with the maximum of the overall minimum expected completion time to the corresponding resource. The Max-Min algorithm may give a matching with more load balance over the resource in cloud environments. In fact, this heuristic algorithm is similar to the min-min heuristic algorithm. It also begins with the set U of all unmapped jobs. Then, the set of minimum completion time

$$SCT = \left\{ m_j : m_j = \min_{0 \le j < y} \left(CT[i, j] \right), \text{for } i \in w \right\}$$

is found for each unmapped job. Next, the job i with the overall maximum completion time in set of SCT is selected and assigned to the corresponding machine (hence the name Max-min). Then the workload of the selected machine will be updated and finally the newly mapped job is removed from U. This process repeats until all jobs are mapped (*i.e.* U is empty).

3. Problem Description

In this section, we employ a service-oriented resource broker that enables the CSP to make the best combination of the access network and computation resources to serve users in different locations. The service provider provides on-demand services with sufficient information, such as, user's requirements, network status and computation resource, they are expected to be able to perform complex computing-intensive applications via the optimal allocation of resources to jobs.

As mentioned before, we have studied the resource virtualization technology where the physical resource can be abstract to virtual resources, which mean that the resource utilization will be efficiently improved by virtual resources for the purpose of the delivery on-demand services. In this context, we extend the traditional job allocation problem associated with computation resource, to the combination resource optimization problems associated with network resource, to the job allocation problem for user's request in cloud environment. The solution to the optimizing job allocation problem is to determine the best combination of access networks and computation resources to serve the required computational application, so as to maximize the system throughput and load balance whilst at the same time satisfying the user's QoS requirements.

In the following, the job allocation problem is described in a high level and math formulation manner. The virtual network guided heuristic static algorithm is proposed and presented to solve the job allocation problem.

3.1. High-Level Problem Formation

Figure 1 illustrates an example of maximizing the job allocation problem from a high level point of view. This high level view of the optimization problem consists of users, service-oriented resource broker, cloud service providers and infrastructure service providers. This system requires two types of inputs, the users request and the resource information. A user can create one or many of jobs. Each job has its own requirements. These requirements consist of the data size of job and job length that is mean job process power.

Cloud computing environments may consist of the diverse virtualized resources provided by cloud service provider. Service provider control and manage own virtualized network and VMs. It is responsible for composing the diverse virtual network and VMs from infrastructure provider and offers a service to cloud users.

Network virtualization is an abstraction concept of the network to run multiple virtual networks on the same physical network without interfering each other in cloud computing [10]. Actually, it is the technology that allows the simultaneous operation of multiple logical networks on a single physical platform [11,12]. It allows service provider reserve resources from multiple InPs to create virtual networks and deploy resources to offer service to the users. The cloud computing operator deploys the on-demand service over the virtualized network without

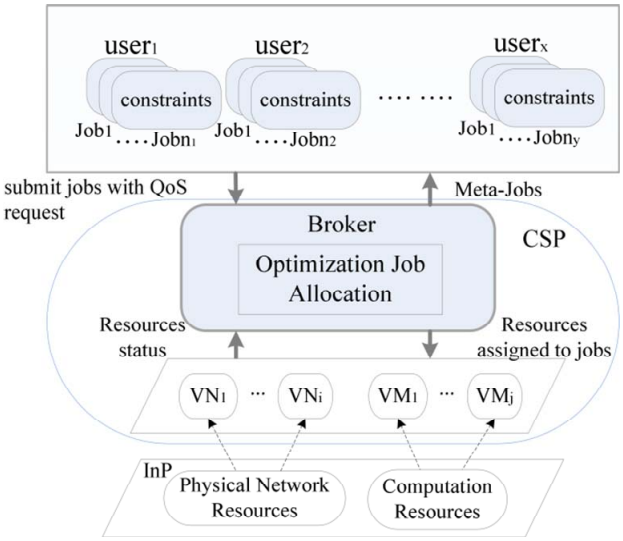

Figure 1. High-level system overview.

concerning for the physical networks. Each service provider can create their own virtual network with application-specific mechanism to meet the needs of users. Thus, the requirements of network QoS can be satisfied. In virtual network scenario, the virtualization path accelerates the data transmission process and reduces the network traffic since it can combine multiple channels into a routing path. Finally, the service provider via virtual network selects candidate physical network paths and decides the allocated path.

Plenty of the VMs to run computing-intensive applications are growing with each passing day [13]. The VM is the abstraction of computing resource such as multiple VMs resides on the single physical machine. These VMs can enhance the computation capability by employing the proportional share policy from computing resource. These cloud applications need share many of VMs through scalable networks. When the network traffic between the VMs is congested, the cloud applications performance will be degraded. Therefore, in such application environments, the role of the cloud networking resources is a key technique to satisfy user demands with efficiently utilized physical resources. Cloud computing operators have to provide capabilities of controlling bandwidth and latency from different network providers that aim for guaranteed service between the VMs which are residing in different geographic locations. In order to provision the customized environment for a given user requests or applications, both for computing and network resource requirements are all need to be satisfied. However, it is difficult to satisfy these requirements under diverse network performance and reliabilities.

The infrastructure providers are responsible for development, operation, and management of the underlying physical infrastructure. This physical infrastructure consists of various networks, such as the access network, metropolitan area network and a core network. It is offered a service directly to the SPs, does not relate to the users. The physical machine is represented as data center which provides multiple computation resources to meet demand of services. This architecture provisions virtualized network resources and VMs to utilize the physical resources, and scheduling the VM with cloud network conditions to satisfy the user's requirements.

The optimization job allocation is a core component in the broker, providing efficiently utilize physical resources over multiple service providers. It aggregates virtualized resources to handle end-to-end service provisioning. When broker receive the users request, it start to discover the available virtualized resources from service provider. After that, the broker schedules and reserves these candidate resources for optimizing resource utilization and guarantee the service performance for user's requirements. The function of optimization job allocation is job shop scheduling problem, including many different heuristic scheduling algorithms that have been proved NP-complete problem [8]. The goal of the job scheduling algorithm is load balancing in the different VM and satisfies user's requirements. In order to seek the complete matching between meta-jobs and the virtual resources, the heuristic meta-job scheduling algorithm is enabling to solve this problem.

3.2. Interaction of Broker

This heuristic scheduling algorithm in broker architecture (in **Figure 2**) will consider both computation and network resources information form the index service. According to the reply from the index services, the scheduling algorithm is able to estimate meta-job execution time and data transmission time, by taking the available computational and network resource into account. As meta-job send to broker, the objective of scheduling algorithm is minimized the expected completion time of meta-job. This value is determined by the available processing power for that job on the VM and the job's data size (and job length) associated with residual bandwidth on the observed virtual path from VM to end-users. After that, the broker will select the optimal VM with network to perform the computational applications.

The scheduling algorithm interaction procedure can be descripted as follows:

- Before the meta-job send to the broker, the virtual network and VM with their resources information such as bandwidth and computation ability have been automatically registered to index service (IS). At this stage, the proposed mapping of virtualized resource consists of virtual network and VM. Allocation of the multiple VM from data center depends on the proportional provisioning policy. Mapping of virtual link

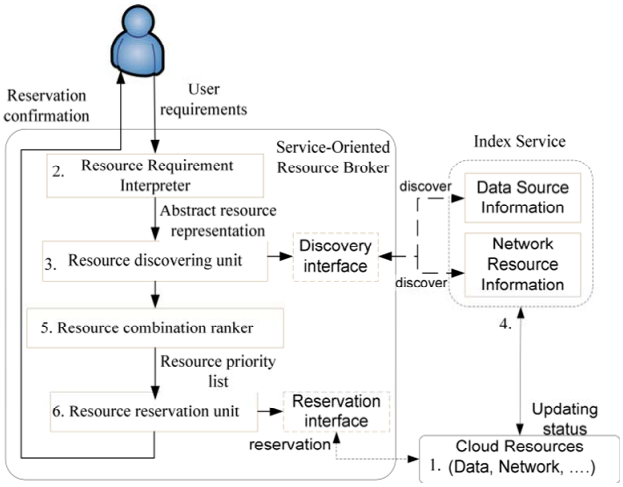

Figure 2. Service-oriented resource broker.

according to the best path selection which follow open shortest path first (OSPF) routing protocol from physical network. The establishments of virtual network topology are connecting with the type of service to users and monitor the capacity of physical network.

- Initially computation application's requirement from client side pass messages to broker. User will send request to process meta-jobs. Details about the job's requirements such as the total number of jobs, size of each data file and computation ability (length of job) needed of each job from the job's request.
- The broker discovers a set of potential virtualized resources. It requests virtual network and virtual machine information suitable for the proposed meta-job. These virtual resources information is obtained from index service (IS). These all possible combined virtual resources are listed in broker for optimizing the meta-job allocation.
- The IS maintained by the CSP needs to gather information about dynamic virtual network conditions and VM in order to obtain optimized job scheduling.
- Resource combination ranker starts to calculate the relevant parameter to scheduler the jobs after receiving the information from both users request and virtual resources. The heuristic scheduling algorithm for meta-job scheduling has been proposed in broker architecture. These meta-job scheduling algorithms such as MCT, Min-Min and Max-Min provide best match of jobs to virtual resources. Therefore, the cloud system performance will be improved.
- Finally, the selected combined virtual resources have reserved to provide the reliable and efficient service to users. After that, these meta-jobs will be sent to the selected virtual resource for data transmission over physical network and job execution on VM. After that, the meta-job execution results will be sent back to the users.

3.3. Mathematics Description

In this section, we first formulate the network resource, VN mapping and user's requirement. We also formulate the heuristic scheduling algorithm for which optimizing the virtualized resource utilization.

1) *Network Resource Definitions*

We form the substrate network as a unidirectional graph $G=(N,E)$, where N is set of substrate nodes and E is set of substrate links. Each substrate node $x \in N$ has an associated geographical location $loc(x)$ [14]. A substrate link $l \in E$ has a bandwidth capacity *b*.

Lemma 1. *Substrate Network.* Given P is set of simple substrate path of G. Consider a path P $(p \in P)$ of e physical links $l_1, l_2, \cdots, l_e$ $(l_e \in p)$ with base bandwidths $b_1, b_2, \cdots, b_e$ respectively. The estimate the bottleneck bandwidth of containing e links along with path p, *i.e.* estimate $\min_{1\le k\le e} b_k$, in here, b_k is used to denote the bottleneck bandwidth in the interval between e links of path p.

The virtual network topology is setting by CSPs that enable the service to serve user. As substrate network, we form the virtual network (VN) as a unidirectional graph $G'=(N',E')$. Each virtual node or access point $x' \in N'$ has a geographical location $loc(x')$ which associates to storage resources. A virtual link $l'(l' \in E')$ is characterized by a bandwidth capacity b'.

Lemma 2. *Virtual Network.* Given P' represents a set of virtual path of G'. A virtual path $p'(p' \in P')$ contains d virtual links $l'_1, l'_2, \cdots, l'_d$ $(l'_d \in p')$ with characterized bandwidth capacity requirement $b'_1, b'_2, \cdots, b'_d$, respectively. The bottleneck bandwidth of *d* links along with path p' is estimated as $\min_{1\le k\le d} b'_k$. The bottleneck bandwidth in the interval between d links of path p' is denoted b'_k.

Lemma 3. *VN Mapping.* It is defined by the mapping of the virtual (logical) network topology reside on the substrate (physical) network topology subject to certain constraints. This process can be separated into two stages:

Node mapping: From the CSPs, each storage resource is associated with one of the backbone nodes in the physical network topology. Therefore, node mapping is based on the geographical location of the storage resource. Each virtual node from a VN is mapped to a single different substrate node by mapping, *i.e.*

$M_N : N' \leftarrow N$

Such that

$$M_N(x) = M_N(x'), \text{iff } x = x' \quad \forall x, x' \in N$$

subject to the location constraints $loc(x) = loc(x')$.

Link mapping: Each virtual link is mapped to a flow based substrate paths between the substrate nodes corresponding to the end virtual nodes of that virtual link. It is defined by a mapping $M_E : E' \leftarrow p$. Such that

$\forall l' = (x_1', x_2') \in E'$, and p is the set of simple paths of G.

Proof. From Lemmas 1, 2 and 3, we have the virtual network properties in Equation (1).

$$M_E(l') \subseteq p\left(M_N(x_1'), M_N(x_2')\right) \tag{1}$$

Subject to the bandwidth capacity constraints:

$$b' \le b_{(x_1,x_2)}, l' : \exists p \in M_E(l')$$

2) *Description of Jobs and End Resource*

In cloud computing environment, the users sent the jobs with different requirements to cloud service provider. These requirements include job data size and processing power. After that, the service-oriented broker is responsible for matching the jobs to available virtual resources which provided from index service.

Lemma 4. Let Assume that we have H VMs where each VM h_j has its own processing power. It can be defined by Equation (2).

$$H = \left\{h_j(c_j) \middle| j = 1, 2, \cdots, y\right\} \tag{2}$$

where

h_j : denotes the VM j

c_j : expresses the computing ability of VM j

y : represents the number of VMs

Lemma 5. Assume the set jobs U of i number where each job u_i has its own data size and processing power. It is expressed in Equation (3).

$$U = \left\{u_i(s_i, q_i) \middle| i = 1, 2, \cdots, w\right\} \tag{3}$$

where

u_i : indicates a given job i

s_i : denotes the data file size of a given job u_i

q_i : expresses the processing power (MI) of a job u_i

w : represents the number of jobs

Proof. From Lemmas 4 and 5, we have Γ that describes the matching relationships between jobs and VMs. The value scope of $\Gamma_{j,i}$ is either 1 or 0, which means alternatively. See Equation (4)

$$\Gamma = \left\{\Gamma_{i,j} \middle| i = 1, 2, \cdots, w; j = 1, 2, \cdots, y\right\}$$

$$\Gamma_{i,j} = \begin{cases} 1 & \text{if assign job } u_i \text{ on VM } h_j \\ 0 & \text{if do not assign job } u_i \text{ on VM } h_j \end{cases} \tag{4}$$

3) Formulation of V-Heuristic Scheduling Algorithm

According to diverse user requests, broker allocates the independent jobs to the VM by selecting the best match of virtual resources. The selection strategy can be based on the predication of the computing power of the VM and availability of virtual network resource. This proposed scheduling algorithm will focus on reducing the computational time, total processing time and at the same time to balance the entire VM available incloud environment.

The description of the general meta-jobs mapping heuristics are provided below. The estimation total processing time $ETPT[i, j]$ of meta-job includes estimation transmission time $ETT[i]$ and complete time in VMs $CT[i, j]$.

We define some preliminary terms about meta-job complete time in VMs. The expected time to compute (ETC) matrix is accurate estimate execution time for job u_i on VM h_j, jobs U and VMs H can obtain a $R \times V$ of ECT matrix. Each row R of the ETC matrix represents a given job's estimated execution time on different VMs H. Each column V in ECT matrix represents a given VM's estimated execution time on different Jobs that in meta-job. The expected execution time $ET[i, j]$ of job u_i on VM , it can be express as the time taken by h_j to execute the job u_i when there is no load with h_j. The expected completion time $CT[i, j]$ of job u_i on VM h_j, it can be express as the wall-clock time when h_j completes the execution of job u_i after finishing any previously assigned jobs. Let t_i denote to the beginning time of the execution of job u_i, the other word, it is the ready time of VM h_j. From the above expressions: $CT[i, j] = t_i + ET[i, j]$. Let $C[i]$ be the completion time of job u_i, and it is equal to $CT[i, j]$ when VM h_j is assigned to execute job u_i.

Lemma 6. The makspan for the completed time of meta-job is defined in Equation (5).

$$\max_{(i \in w, j \in y)}\left(CT[i, j]\right) \tag{5}$$

It is used for evaluating the heuristic algorithm for scheduling set of jobs U, Makespan is a measure metric of the throughput of the heterogeneous computing system.

Lemma 7. The estimation of transfer time of each job in set of u_i is defined as $ETT[i]$ in Equation (6). It represents the transmission time for the job transfer to VM that is based on the actual file size of the job and the available bandwidth of VN between path of job to VM. *i.e.*

$$ETT[i] = \frac{s_i}{b_k'} \tag{6}$$

where: b_k' is expressed available bandwidth of the virtual path which mapped from physical network resources. The virtual path bandwidth is updated between the users and VMs.

Proof. From Lemmas 6 and 7, estimated total processing time: $ETPT[i, j] =$ estimation transmission time $ETT[i]$ + completion time $CT[i, j]$. Then, the formula is defined by Equation (7).

$$ETPT[i, j] = \sum_{i=1}^{w} \frac{s_i}{b_k'} + \max_{(i \in w, j \in y)}\left(CT[i, j]\right) \tag{7}$$

3.4. Heuristics Algorithm Codes

Due to cloud's dynamic nature complicates the planning of the meta-job scheduling activity for minimizing the applications total processing time. Therefore, a job scheduling strategy is essential to optimize the utilization of cloud resource processing capabilities, and reduces total process time taken to process user job.

The service-oriented resource broker should reduce the total transmission time of the user jobs to/from the VMs, and the completion time of the jobs on the VMs. Several simple heuristic algorithms for static independent jobs are proposed in: MCT, Min-Min, Max-Min. The general algorithm is shown in **Figure 3**. The metric denotes the ECT matrix as mentioned before. These Scheduling algorithms are heuristic iteratively assign jobs to VMs with VN affect by considering their expected Minimum Completion Time (MCTs). Each job is in an arbitrary order of meta-job for job submission (line 2). For each job that is computed by tentatively scheduling it to each VM (line 5) and estimate the job's completion time on each VM (line 6). The job transmission time from user to VM is computed by (line 7). Also for each job, a metric function "F_1" is computed over all the VMs with VN (line 9). After wards, the job/VM pair with the best metric match (i, j) is selected using selection function "F_2" (line 11). We compute the minimum completion of this job/VM pair (line 12) and minimum job transmission time (line 13). After that, we then compute the minimum total processing time of meta-job (line 14) and assign the job n to the VM m (line 15). The process is repeated until all jobs have been allocated (line 1 and line 16).

[1] **While** schedule U is not empty
[2] **For** each job u_i in meta-job U (in an arbitrary order)
[3] **For** meta-job U to schedule
[4] **For** each VM $h_j\left(h_j \in H\right)$
[5] **For** all VM H
[6] **Compute** $CT[i,j] = t_i + ET[i,j]$
[7] **Compute** $ETT[i] = s_i / b_k'$
[8] **Endfor**
[9] **Compute**
metric $= F\left\{\left(\left(CT[i,1], ETT[i,1]\right), \left(CT[i,2], ETT[i,2]\right), \cdots\right)\right\}$
[10] **Endfor**
[11] **End For**
[12] Select best metric match $(i, j) = F_2\left\{metric1, metric2, \cdots\right\}$
[13] **Compute** minimum $CT[i,j]$
[14] **Compute** minimum $ETT[i]$
[15] **Compute** minimum $ETPT[i,j] = ETT[i] + CT[i,j]$
[16] Allocate jobs U on VM H with corresponding VN resource
[17] **End For**
Endwhile

Figure 3. Pseudo code of the general static heuristic algorithm.

Based on this general VN guided heuristic algorithm, the four heuristics have been applied to it. These algorithms are including MCT, Min-Min, Max-Min which defined by the different definitions of "F_1" and the best metric match selection function "F_2".

3.4.1. V-MCT Heuristic Algorithm

The core procedure of the V-MCT heuristic algorithm is depicted in **Figure 4**. From beginning of the meta-job submission, the V-MCT scheduling algorithm allocates each job to VM until all jobs have been matched (line 1-14). In the "for" loop, the scheduling algorithm computes the minimum earliest completion time for each job in VMs (line 2-4). Secondly, the algorithm computes the minimum estimated transmission time from user to VM via virtual network path for each job (line 5). Thirdly, the algorithm computes the each job minimum total processing time is obtained by completion time and transmission time (line 6).

For each job in meta-job, the algorithm finds minimum earliest completion time and the minimum total processing time (line 8). In final stage, first, assigns the each job to the VM that gives it the minimum earliest total processing time (line 9). Secondly, the algorithm removes the matched the jobs from meta-job (line 10). Thirdly, update complete time for all VM and update virtual network status which mapped from physical networks (line 12). In final line 14, the scheduling algorithm finishes each job allocation on VM and repeats the process next job until all unmatched job are matched.

3.4.2. Min-Min Heuristic Algorithm

From beginning of the meta-job submission, the V-Min-Min scheduling algorithm computes the time of all the jobs on the VMs (line 2-8). After that, we map the whole meta-job to the VMs. In the first "for" loop, initially, for each job with data size request in the meta-job, the algo-

[1] **While** schedule U is not empty (in an arbitrary order)
[2] **For** each job u_i in meta-job U (in an arbitrary order)
[3] **For** each VM $h_j\left(h_j \in H\right)$ (in a fixed arbitrary order)
[4] **Compute** earliest completion time $CT[i,j]$ on VM h_j
[5] **Compute** $ETT[i]$ on VM h_j
[6] **Compute** minimum $ETPT[i,j] = CT[i,j] + ETT[i]$
[7] **Endfor**
[8] Find VM h_j with minimum CT and minimum ETT
[9] Assign job u_i to the VM h_j that gives earliest $ETPT$
[10] Delete job u_i to from job set of U
[11] Update CT for all VM H
[12] Update virtual network status
[13] **End For**
[14] **Endwhile**

Figure 4. Pseudo code of V-MCT heuristic algorithm.

rithm computes the earliest completion time and the resource that obtains it (lines 2-4). Secondly, the algorithm computes the minimum estimated transmission time from user to VM via virtual network path (line 5).

Thirdly, the algorithm computes the meta-job minimum total processing time is obtained by completion time and transmission time (line 6). The initial matrix is created after (line 8) as shown in **Figure 5**. In the second "for" loop, for each job in meta-job, the algorithm finds minimum earliest completion time and the minimum total processing time (lines 9-10) for job allocation. In final stage, first, assigns the job to the VM that gives it the minimum earliest total processing time (line 12). Secondly, the algorithm finalizes the loop by removing the matched the jobs from meta-job and repeat the process until all unmatched job are matched (line 13). Thirdly, update complete time for all VM and update virtual network status which mapped from physical networks (line 14 and line 15). After finishing the mapping all the jobs, we send the job to VM for executing computation processing.

3.4.3. V-Max-Min Heuristic Algorithm

Figure 6 shows the core procedures of the Max-Min heuristic algorithm. From beginning of the meta-job submission, the Max-Min scheduling algorithm computes the time of all the jobs on the VMs (lines 2-8). After that, the met-job map to the VMs. In the first "for" loop, the processing of compute minimum total processing time is same as the Min-Min algorithm. After line 8, the initial matrix of executing and transmission time will be created.

The main different is in second "for" loop compared with Min-Min scheduling algorithm. For each job in meta-job, the algorithm finds the maximum earliest completion time and total processing time (lines 9-10). In final stage, first, the algorithm allocates the job to the corresponding VM that gives it the maximum earliest total processing time (line 12). Secondly, the algorithm finalizes the loop by removing the matched the jobs from meta-job and repeat the process until all unmatched job are matched (line 13). Thirdly, update complete time for all VM and update virtual network status which mapped from physical networks (lines 14 and line 15). After finishing the mapping all the jobs, we send the job to VM for executing computation processing.

[1] **While** schedule U is not empty (in an arbitrary order)
[2] **For** each job u_i in meta-job U (in an arbitrary order)
[3] **For** each VM $h_j \left(h_j \in H\right)$ (in a fixed arbitrary order)
[4] **Compute** earliest completion time $CT[i,j]$ on VM h_j
[5] **Compute** $ETT[i]$ on VM h_j
[6] **Compute** minimum $ETPT[i,j] = CT[i,j] + ETT[i]$
[7] **Endfor**
[8] **For** each job u_i in meta-job set U
[9] Find job u_i with minimum earliest $ETPT$
[10] **End for**
[11] Assign job u_i to the corresponding VM h_j with the minimum earliest $ETPT$
[12] Delete job u_i to from job set of U
[13] Update CT for all VM H
[14] Update virtual network status
[15] **End For**
[16] **Endwhile**

Figure 5. Pseudo code of V-Min-Min heuristic algorithm.

[1] **Whileschedule** U **is not empty (in an arbitrary order)**
[2] **For** each job u_i in meta-job U (in an arbitrary order)
[3] **For** each VM $h_j \left(h_j \in H\right)$ (in a fixed arbitrary order)
[4] **Compute** earliest completion time $CT[i,j]$ on VM h_j
[5] **Compute** $ETT[i]$ on VM h_j
[6] **Compute** minimum $ETPT[i,j] = CT[i,j] + ETT[i]$
[7] **Endfor**
[8] **For** each job u_i in meta-job set U (in arbitrary order)
[9] Find job u_i with maximum earliest $ETPT$
[10] **End for**
[11] Assign job u_i to the corresponding VM h_j with the maximum earliest $ETPT$
[12] Delete job u_i to from job set of U
[13] Update CT for all VM H
[14] Update virtual network status
[15] **End For**
[16] **Endwhile**

Figure 6. Pseudo code of V-Max-Min heuristic algorithm.

4. Performance Evaluation

In this experiment, we fixed the parameter for the computation resources and used three different job submission scenarios. The heuristic scheduling algorithm and the conventional heuristic scheduling with physical network connection are compared within three type of scheduling algorithms, such as MCT, Min-Min and Max-Min. At the same time, we compare the total processing time and average resource utilization rate in these scheduling algorithms on the same set of jobs. In addition, we also investigate how the load balance affects the system performance. In this section, we describe performance metrics, the experiment setup and the simulation results.

4.1. Performance Metrics

There are different performance criteria that can be used to describe resource scheduling systems. Depending on what scheduling performance is desired in cloud there exists different performance metrics for evaluating different job scheduling algorithms [15]. The simulation

results are evaluated job scheduling algorithm on the basis of following performance metrics.

4.1.1. The Estimation Total Processing Time

The estimation total processing time $ETPT[i,j]$ of meta-job include estimation transfer time $ETT[i]$ and makespan $CT[i,j]$. Makespan is measure of the throughput of the cloud computing system. The aim of this work is to minimize the meta-job total processing time, it is defined in Equation (8).

$$ETPT[i,j] = \min\left\{\sum_{i=1}^{w}\frac{s_i}{b_k'} + \max_{i\in w, j\in y}\left(CT[i,j]\right)\right\} \quad (8)$$

4.1.2. Average Resource Utilization Rate

The resource utilization rate η_j of each VM h_j can be calculated by using Formula (9):

$$\eta_j = \frac{\sum_{\forall i,\Gamma_{i,j}=1}\left(te_i - ts_i\right)}{T}\times 100\% \quad (9)$$

te_i: The function of the broker is to find the earliest possible time for each job to complete.

ts_i: The earliest possible start time for the jobon a selection of VM is the latest free time of all the selected VM if there are still jobs running on the selected VMs.

T: The time T is defined the total job execution time when a set of jobs are allocated onto VM H. It can be calculated by using Equation (10)

$$T = \max_{1\le i\le w}\{te_i\} - \min_{1\le i\le w}\{ts_i\} \quad (10)$$

Average resource utilization rate η of total resources is calculated through Equation (11)

$$\eta = \frac{\sum_{j=1}^{y}\eta_j}{y},\ 0<\eta<1. \quad (11)$$

4.1.3. Load Balance Level

Resource load balancing is one of the most difficult problems that must be handled in cloud computing system. The mean square deviation of η_j is defined as Equation (12).

$$d = \sqrt{\frac{\sum_{j=1}^{y}\left(\eta-\eta_j\right)^2}{y}} \quad (12)$$

And the relative deviation of d over η that described the load balancing level of the system is β as Equation (13).

$$\beta = \left|1-\frac{d}{\eta}\right|\times 100\% \quad (13)$$

If d equals to 0 and then β equals 100%, that mean the cloud system provide efficient load balancing for virtual machines. Actually, the objective of load balance is equally spread the load on each VM, in order to get optimal VM utilization, maximize throughput, minimizing the total job execution time and avoid overload of VM. Aims to achieve this objective, the different between the heaviest-loaded VM and the lightest VM should minimize.

4.2. Experiments Setup

To evaluate and compare to the proposed V-heuristic scheduling algorithm with three basic heuristic algorithms, V-MCT, V-Min-Min and V-Max-Min in cloud environments. The CloudSim simulator with the abilene network topology has been used to create the simulation environment [16,17]. This simulator allows modeling and simulation of cloud system components such as users, data center, VMs and resource provisioning policies. In this simulator, the computing resource processing ability can be measured in the form of MIPS (Million Instruction Per Second) as per SPEC (Standard Performance Evaluation Corporation) benchmarks [18]. The summary characteristics of the computation resources are shown in **Table 1**.

An adequate real network topology is based on abilene network topology which is a high performance Internet 2 backbone network for serving to educational and research proposes. The Abilene network consists of 11 nodes with link capacity and connections are described in Abilene backbone specifications. The actual bandwidth of all the links in the Abilene topology is based on 10 Gbps links between nodes [9]. It represents backbone network to support cloud applications and evaluation of their performance. The configuration files of Abilene nodes are publicly available. Therefore, the topology setting of this network can be easily used for simulation and testing purposes. In addition, all links share the same characteristics such as Maximum Transmission Unit (MTU) size of 1500 bytes and latency of 10 msec.

In this computation application scenario, the VN is composed with 4 virtual nodes in full mesh connection that are abstracted from the Abilene network nodes. The virtual link is abstracted from physical network path

Table 1. Characteristic of the resources.

Name	Resource type and characteristics	No. of CPU	A SPEC rating (MIPS)
R1	PC with Intel Pentium 2.0 Ghz, 512 MB RAM	1	377
R2	IBM eServer with dual Intel Xeon 2.6 Ghz, 2 GB RAM	2	525
R3	IBM eServer with dual Intel Xeon 2.6 Ghz, 2 GB RAM	2	525
R4	IBM eServer with dual Intel Xeon 2.6 Ghz, 2 GB RAM	4	1050
R5	IBM eServer with dual Intel Xeon 2.6 Ghz, 2 GB RAM	4	1050

which adopted Open Shortest Path First (OSPF) with weight of physical distance as the routing protocol to select widest-shortest paths. Five data centers are created in different locations which are connected by the Abilene network. Each data center is allocated VMs by time-sharing allocation policy.

In our simulation experiment The VMs share the processor capacity in equal proportion to cloud service provider. For instance, in computation resource R2 that provides two equal proportions VMs. To create a sufficiently functioning cloud environment, 60 jobs are created by different users. These jobs are sharing the same characterizes:

- Total number of jobs: 60 jobs are generated randomly
- Job data size: each job data size is uniformly distributed (10 - 500 MByte).
- Job processing power: each job processing requirement equals to uniformly distribute in (10 - 20,000 MI).
- Job submission: The jobs arrivals are modeled by aPoisson random process among 5 computation resources.

The resource virtualization technology has a big impact on the performance of the job scheduling. According to real cloud applications or users request, the experimental testing of our heuristic is performed in three scenarios:

1) Scenario 1: Length of jobs are random determined.

2) Scenario 2: 15% long length of jobs along with 85% short length of jobs.

3) Scenario 3: 15% short length of job along with 85% long length of jobs.

For each of the scenario, we compare the performance of the virtual network guided heuristic scheduling algorithm and the conventional heuristic scheduling algorithm. Number of resource is chosen to be 5 and the number of jobs is 60. Both of their characteristics are fixed through three scenarios.

4.3. Experimental Results and Discussion

The total process time results of three scenarios are shown in **Figure 7**. It shows that the proposed heuristic scheduling algorithm with virtualized resource condition is outperforms conventional heuristic scheduling algorithm with physical network condition. Therefore, the concept of virtualization for cloud computing has been proved in cloud environment. In other worlds, the virtual network and virtual machine can efficiently improve the performance over the cloud computation applications. From these simulation results, we can see the V-Max-Min scheduling has provided minimum total processing time compared with other two algorithms in these three different scenarios. Therefore, the V-Max-Min scheduling algorithm is the best way to optimize the meta-job

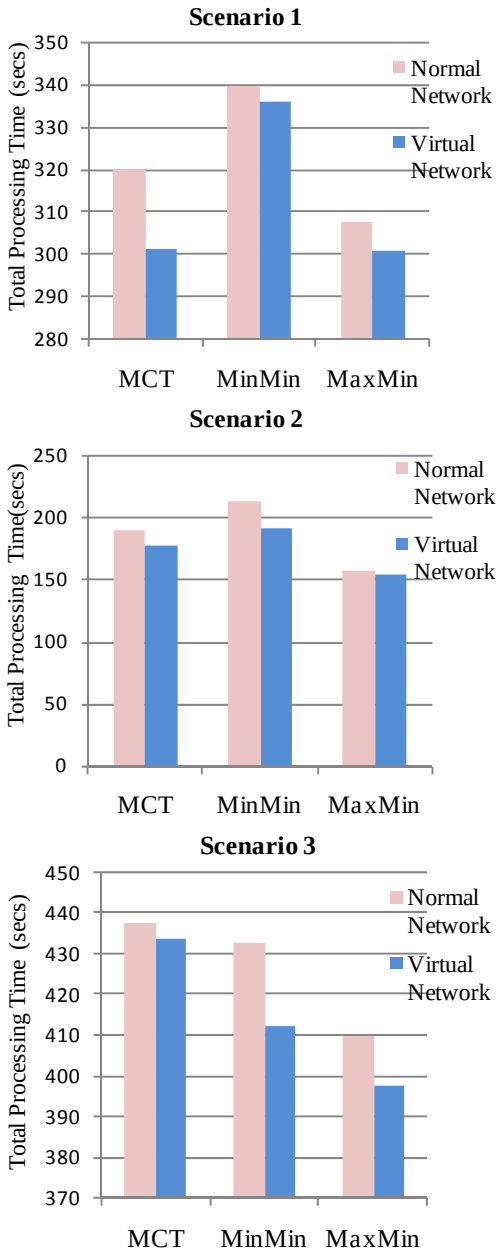

Figure 7. Total processing time results in three scenarios.

scheduling.

In scenario 1, the meta-job's total processing time using MCT can be as much as 5.84% shorter than using the conventional MCT. In same condition, the Min-Min can be as much as 1.06% shorter than using the conventional Min-Min. In case of Max-Min, the total processing time is improved by 2.23%.

In scenario 2, where a few long length of jobs and many short lengths of jobs are required. From the results, we can see the MCT can improve 6.93%, Min-Min improves 10.74% and Max-Min improves 2.27%. Compared these three scheduling algorithm, the Min-Min have the best efficient improvement within this scenario.

In scenario 3, the **Table 2** shows the MCT can improve 0.88%, Min-Min improves 4.73% and Max-Min improves 2.99%. This efficient improvement of meta-job's total processing time in cloud system due to two main reasons: the first one is virtual network that can improve the data transmission time over the network. And the other one is the VM provisioning which can reduce the Makespan of meta-jobs processing.

As shown in **Table 2**, the total processing time are compared for all three scenarios in three types of heuristics scheduling algorithms.

Figure 8 shows the average resource utilization rate in each scenario with three different scheduling algorithms.

The comparison of average utilization rate in different scenarios can be shown in **Table 3**. In scenario 1, the MCT can provide best average resource utilization rate compared with other two algorithms. The average resource utilization rate has improved by heuristic scheduling algorithm. The MCT has improved 4.49%, Min-Min improves 2.05% and Max-Min improves 1.81%. In scenario 2, a few long length of jobs have been submitted to broker. The Max-Min provide best average resource utilization rate. The MCT has improved 10.57%, Min-Min improves 6.69% and Max-Min improves 8.92%. In scenario 3, many long length of jobs have been submitted for scheduling. In this case, the Min-Min provides best average resource utilization rate compared with others. The MCT improves 0.52%, Min-Min improves 5.72% and Max-Min improves 3.34%. Therefore, the Min-Min algorithm has more efficient improvement in these cloud applications.

Figure 9 shows the comparison of the load balancing level with three different algorithms among three scenarios. To maximum the performance of a computation cloud system, it is essential to distribute the load among the cloud resources. In other words, it is desirable to

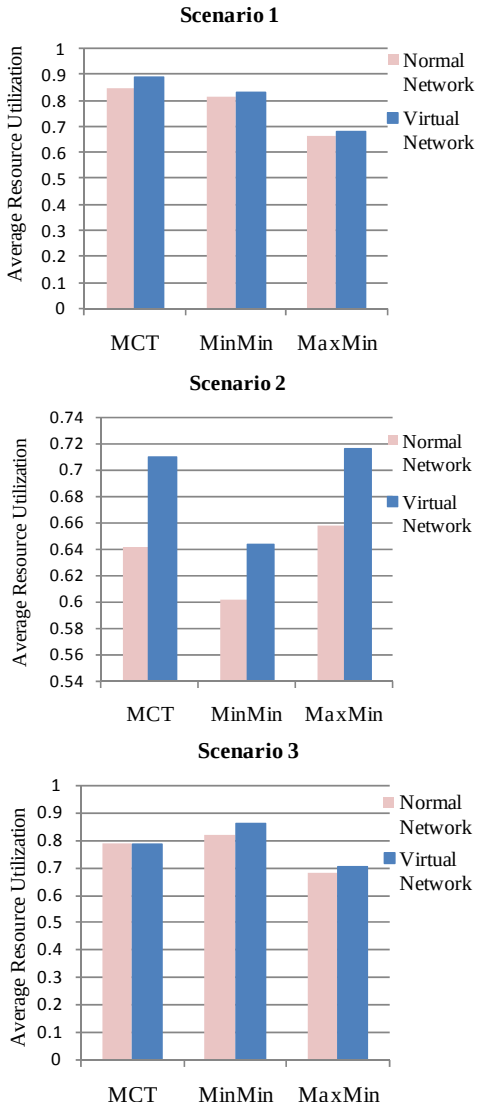

Figure 8. Average resource utilization in three scenarios.

Table 2. Summary of total processing time results.

Total Process Time				
Scenario	Heuristics scheduling algorithm	Conventional physical network (msec)	Virtual network guided (msec)	Improvement (%)
1	MCT	320.02961	301.33901	5.84%
	Min-Min	339.6709	336.06801	1.06%
	Max-Min	307.68842	300.81337	2.23%
2	MCT	189.85591	176.69453	6.93%
	Min-Min	213.42023	190.50065	10.74%
	Max-Min	157.26812	153.70261	2.27%
3	MCT	437.90426	434.06973	0.88%
	Min-Min	432.7449	412.26696	4.73%
	Max-Min	409.97551	397.71294	2.99%

Table 3. Summary of average resource utilization in three scenarions.

Average Resource Utilization				
Scenario	Heuristics scheduling algorithm	Conventional physical network (msec)	Virtual network guided (msec)	Improvement (%)
1	MCT	0.852066	0.890287	4.49%
	Min-Min	0.814373	0.831033	2.05%
	Max-Min	0.664729	0.676744	1.81%
2	MCT	0.641634	0.709434	10.57%
	Min-Min	0.601818	0.643699	6.96%
	Max-Min	0.65742	0.71605	8.92%
3	MCT	0.783836	0.78793	0.52%
	Min-Min	0.814998	0.861646	5.72%
	Max-Min	0.680609	0.703355	3.34%

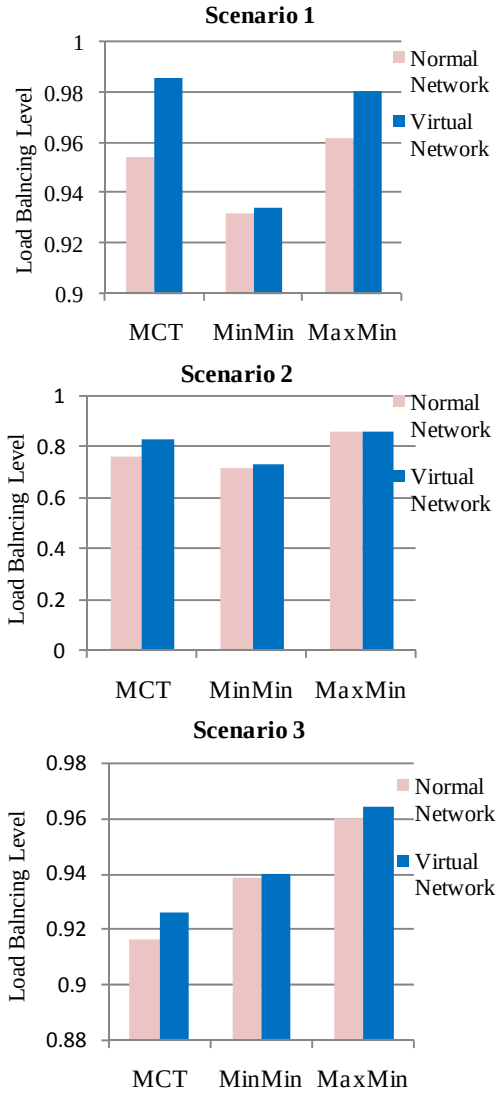

Figure 9. Load balancing level in three scenarios.

prevent the condition where one cloud computation resource is overloaded with a backlog of jobs while another resource is lightly or idle. The load balancing level metric is closely related to cloud job scheduling.

In scenario 1 of **Table 4**, the load balancing level of MCT improve 3.33%, for the Min-Min improves 0.23% and Max-Min improves 1.94%. In scenario 2, the MCT improve 8.91%, the Min-Min improves 2.23% and Max-Min improves 0.13%. In scenario 3, the MCT improve 1.03%, the Min-Min improves 0.16% and Max-Min improves 0.42%.

In scenario 2 and scenario 3, the Max-Min acts like the best scheduling algorithm which has better load balancing level of allocating job to resources. The reason of that is the Min-Min assigns the job with the earliest completion time in each phase, results in some resources becoming busy all the time and others becoming idle most of the time. Therefore, it has less load balancing level

Table 4. Summary of Load balancing level in three scenarions.

	Load Balancing Level			
Scenario	Heuristics scheduling algorithm	Conventional physical network (msec)	Virtual network guided (msec)	Improvement (%)
1	MCT	0.954343	0.986149	3.33%
	Min-Min	0.932312	0.934456	0.23%
	Max-Min	0.962239	0.980931	1.94%
2	MCT	0.765872	0.834127	8.91%
	Min-Min	0.720648	0.736743	2.23%
	Max-Min	0.864311	0.865454	0.13%
3	MCT	0.916186	0.925633	1.03%
	Min-Min	0.938521	0.940023	0.16%
	Max-Min	0.959674	0.963685	0.42%

than Max-Min where it assigns the task with maximum completion time and lets other tasks executes along on the other resources, therefore have better load balancing level in overall cloud system.

Through exploit the merit of the MCT, Min-Min and Max-Min for each scenario, it shows the heuristic scheduling algorithm not only has the smaller total processing time that conventional heuristic scheduling algorithm with physical network condition, but also has better average resource utilization rate and load balancing ability.

5. Conclusion

In this paper, we present V-heuristic scheduling algorithm in service-oriented resource broker for improving computation applications performance. This scheduling algorithm leverages network virtualization and virtual machine techniques to provide efficiently data transmission and job execution services for user requests. In this scheduling algorithm, we used three different heuristic scheduling algorithms to perform meta-job execution such as MCT, Min-Min and Max-Min. We compared the meta-job total processing time, average resource utilization rate and load balancing produced by our proposed scheduling algorithm against that produced by tradition heuristic scheduling algorithm with physical network supported. The simulation results obtained from the simulator consistently showed that performance metrics of computation applications can be improved significantly when using V-heuristic scheduling algorithm.

6. Acknowledgements

The authors would like to acknowledge the support from CATT-Surrey collaboration project and the EU FP7 MONET and EVANS projects.

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *Proceedings of ACM SIGCOMM Computer Communication Review*, Vol. 39, ACM, New York, 2009, pp. 50-55.

[2] Q. Zhang, L. Cheng and R. Boutaba, "Cloud Computing: State-of-the-Art and Research Challenges," *Journal of Internet Services and Applications*, Vol. 1, No. 1, 2010, pp. 7-18. doi:10.1007/s13174-010-0007-6

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Journal on Future Generation Computer Systems*, Vol. 25, No. 6, 2009, pp. 599-616. doi:10.1016/j.future.2008.12.001

[4] T. Dillon, C. Wu and E. Chang, "Cloud Computing: Issues and Challenges," *Proceedings of the IEEE* 24*th International Conference Advanced Information Networking and Applications*, Perth, 20-23 April 2010, pp. 27-33.

[5] F. Baroncelli, B. Martini and P. Castoldi, "Network Virtualization for Cloud Computing," *Journal of Annals of Telecommunications*, Vol. 65, No. 1-12, 2010, pp. 713-721.

[6] T. D. Braun, H. J. Siegal, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, Y. Bin, D. Hensgen and R. F. Freund, "A Comparison Study of Static Mapping Heuristics for a Class of Meta-Tasks on Heterogeneous Computing Systems," *Proceedings ofthe* 8*th Heterogeneous Computing Workshop*, San Juan, 12 April 1999, pp. 15-29.

[7] M. Maheswaran, S. Ali, H. J. Siegal, D. Hensgen and R. F. Freund, "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems," *Journal of Parallel and Distributed Computing*: *Special Issue on Software Support for Distributed Computing*, Vol. 59, No. 2, 1999, pp. 107-131. doi:10.1006/jpdc.1999.1581

[8] J. D. Ullman, "NP-Complete Scheduling Problems," *Journal of Computer System Sciences*, Vol. 10, No. 3, 1975, pp. 384-393. doi:10.1016/S0022-0000(75)80008-0

[9] The Abilene Network Topology, 2007. http://abilene.internet2.edu

[10] J. Carapinha and J. Jimenez, "Network Virtualization: A View from the Bottom," *Proceedings of the* 1*st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, Barcelona, 16-21 August 2009, pp. 73-80. doi:10.1145/1592648.1592660

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, "Xen and the Art of Virtualization," *Proceedings of the* 19*th ACM Symposium on Operating Systems Principles*, Bolton Landing, 19-22 October 2003, pp. 164-177.

[12] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Journal of Software Practice & Experience*, Vol. 41, No. 1, 2011, pp. 23-50. doi:10.1002/spe.995

[13] J. Cao, D. P. Spooner, S. A. Jarvis and G. R. Nudd, "Grid Load Balancing Using Intelligent Agents," *Journal of Future Generation Computer Systems*, Vol. 21, No. 1, 2005, pp. 135-149. doi:10.1016/j.future.2004.09.032

[14] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," 19*th Annual Joint Conference of the IEEE Computer and Communications Societies*, Tel Aviv, March 2000, pp. 519-528.

[15] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Journal of Software Practice & Experience*, Vol. 41, No. 1, 2011, pp. 23-50. doi:10.1002/spe.995

[16] M. H. Jamal, A. Qadeer, W. Mahmood, A. Waheed and J. J. Ding, "Virtual Machine Scalability on Multi-Core Processors Based Servers for Cloud Computing Workloads," *Proceedings of the* 2009 *IEEE International Conference on Networking, Architecture, and Storage*, Zhangjiajie, 9-11 July 2009, pp. 90-97.

[17] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. D. Lara, M. Brudno and M. Satyanarayanan, "SnowFlock: Rapid Virtual Machine Cloning for Cloud Computing," *Proceedings of the* 4*th ACM European Conference on Computer Systems*, Nuremberg, 1-3 April 2009, pp. 1-12. doi:10.1145/1519065.1519067

[18] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities," *Proceedings of the International Conference on High Performance Computing & Simulation*, Leipzig, 21-24 June 2009, pp. 1-11.

