

# An Empirical Investigation on the Simulation of Priority and Shortest-Job-First Scheduling for Cloud-based Software Systems

Jia Ru

Department of Computing  
The Hong Kong Polytechnic University  
Hong Kong SAR  
Email:csrjia@comp.polyu.edu.hk

Jacky Keung

Department of Computer Science  
City University of Hong Kong  
Hong Kong SAR  
Email:Jacky.Keung@cityu.edu.hk

## Abstract—

**Background:** Given the dynamics in resource allocation schemes offered by cloud computing, effective scheduling algorithms are important to utilize these benefits.

**Aim:** In this paper, we propose a scheduling algorithm integrated with task grouping, priority-aware and SJF (shortest-job-first) to reduce the waiting time and makespan, as well as to maximize resource utilization.

**Method:** Scheduling is responsible for allocating the tasks to the best suitable resources with consideration of some dynamic parameters, restrictions and demands, such as network restriction and resource processing capability as well as waiting time. The proposed scheduling algorithm is integrated with task grouping, prioritization of bandwidth awareness and SJF algorithm, which aims at reducing processing time, waiting time and overhead. In the experiment, tasks are generated using Gaussian distribution and resources are created using Random distribution as well as CloudSim framework is used to simulate the proposed algorithm under various conditions. Results are then compared with existing algorithms for evaluation.

**Results:** In comparison with existing task grouping algorithms, results show that the proposed algorithm waiting time and processing time decreased significantly (over 30%).

**Conclusion:** The proposed method effectively minimizes waiting time and processing time and reduces processing cost to achieve optimum resources utilization and minimum overhead, as well as to reduce influence of bandwidth bottleneck in communication.

**Index Terms**—Cloud Computing, Task Grouping, Scheduling, SJF, Software Metrics

## I. INTRODUCTION

Cloud computing is a new software system technology, which allows dynamic resource allocation on consolidated resources using a combination of techniques from parallel computing, distributed computing, as well as platform virtualization technologies [1]. Cloud computing has been a primary focus in both the research community and the industry over recent years because of its flexibility in software deployments, and of its elasticity capability on resource consolidation.

Software engineering for cloud platform systems is a new domain of research, requiring careful considerations on its characteristic with respect to traditional software development

paradigms. More importantly the area of effective scheduling run time tasks become one of the research focuses. The aim of cloud computing is to realize cooperation work and resource sharing, but different kinds of resources reflect isomerism, dynamic nature and a diversity of user demands. This makes resource management very complex. Therefore, scheduling problem is an important research area in this regard. The utilization rate of huge resources in data centre is related to scheduling mechanisms applied. Adopting which scheduling mechanism to improve utilization rate of resource is a significant challenge due to a number of factors which we will explore in this study.

The fundamental mechanism of this new kind software system is to schedule the applications to the resources pool which is consisted of hugely distributed computers [2]. Scheduling is a decision process, and its content is deploying resources to applications of different clients at a suitable time, or during a specific period of time. The target of optimizing scheduling considers one or two factors that include cost, task completion time, task priority, profit and so on. In the premise of guaranteed resource utilization rate, scheduling policies mainly focus on allocation management of resources and satisfy the resource demands of users. Eventually, scheduling policies should effectively improve the number of completed applications, increase profit of service party, reduce cost which is undertaken by service party when accepting applications and guarantee QoS (Quality of Service) demand of clients.

In cloud computing system, there exists some applications with a great deal of light-weight tasks. Dispatching these fine-grained tasks to a pool of resources that provide high processing capability is not economical and consumes extra waiting time and turnaround time by comparing a coarse-grained task allocation to the resource [3]. Since that the overall turnaround time includes each task scheduling time, execution time and transmission time. A large amount of fine-grained tasks will spend a lot of time on scheduling and transmission. It is rather impractical to consume the resource processing capability, and lowers resource utilization rate when a fine-grained task is allocated and executed to a

resource with high processing capability. The total turnaround time of fine-grained tasks can be further reduced by grouping these fine-grained tasks as coarse-grained tasks in the entire scheduling process.

This paper mainly focuses on evaluating and improving such type of deployment policy, and conducts an empirical experiment to examine various different scheduling algorithms which are important to the development of software for the cloud platforms. CloudSim simulation platform has been employed in the experiments. We have extended the CloudSim simulator for our experimentation that enables useful parameters and data to be varied easily such as scheduling algorithms, task characteristics as well as file characteristics.

The rest of this paper is organized as follows: Section II discusses background. Section III presents the proposed algorithm and its strategy. Section IV provides simulations and experiments on the proposed scheduling algorithm using CloudSim. Section V provides results and discussion on the experiments in comparison with existing algorithms. Section VI concludes the paper and proposes future research directions.

## II. BACKGROUND

Cloud computing consists of a cluster of computing resources that are delivered over a network, which is accomplished by utilizing virtualization technologies to consolidate and allocate resources suitable for various different software applications. It provides a platform for solutions requiring different configurations, emulating physical hardware combinations in a virtualized cloud environment managed by cloud platform software to deliver enhanced services. The strategies used in the cloud platform software become important, which directly influence the runtime performance of software applications running on its platform. Therefore the effective scheduling policies to maximize the utilization of the virtualized resources are the primary focus of this study. This section provides a summary of related scheduling approaches applied in cloud computing.

### A. Scheduling models in Cloud Computing

In traditional distributed environment, the aim of optimizing scheduling is mainly focusing on system performance, such as system throughput, CPU utilization rate and almost never considering QoS. In cloud computing environment, we are not only emphasizing resource utilization rate and system performance, but also requiring a guaranteed QoS of users based on different demands. Users can choose the resource in the cloud by themselves according to their own requirements.

1) **Cloud computing scheduling model:** Cloud computing scheduling model is mainly constructed by Client, Broker, Resources, Resources supporter and Information Service. Fig.1 shows the scheduling model structure [4]. The tasks that users need to implement usually can be divided into serial application, parallel application, parameter scan application, cooperation application and so on. System allows users to set up resource demand and parameter preference. Different

clients use resources at different prices, which may vary from time to time. Broker is a middle interface between clients and resources as well as used to find resources, choose resources, accept tasks, return scheduling results, and exchange information between clients and resources. Broker supports different scheduling policies, which can allocate resources and schedule tasks in accordance to the demands of clients. Broker is constituted by Job Control Agent, Schedule Advisor, Explorer, Trade Manager and Deployment Agent.

- **Job Control Agent:** It is responsible for monitoring jobs in the software system, such as schedule generation, jobs creation, status of jobs and communicating with clients and schedule advisor.
- **Schedule Advisor:** It is used to determine resources, allocate available resources which satisfy the demands of clients such as deadline and cost, as well as to allocate jobs.
- **Cloud Explorer:** It is a tool that communicates with cloud information service to find resources and identifies the list of authorized machines as well as records resources status information.
- **Trade Manager:** It determines resources access cost and tries to communicate with resources at a low cost under the guidance of schedule advisor.
- **Deployment Agent:** It uses scheduler instruction to activate the execution of tasks as well as to update the status of execution sending back to Job Control Agent in regular intervals.

During the transaction between clients and service providers, service providers register resource information at first. After clients submit tasks to broker, broker searches resources in information service and deploys tasks to appropriate resources in accordance to the corresponding scheduling algorithms. Before execution of tasks, broker evaluates completion time and cost of tasks. If the time exceeds deadline or the cost is higher than budget of clients, the broker will deny tasks. If the execution of tasks is accomplished, broker will return the deployment results to clients and gain relevant profits, otherwise, send error message back to clients.

2) **Basic scheduling methods:** Scheduling methods always consider two aspects: one is characteristics of tasks, and the other is characteristics of datacenter resources [5] [6]. Tasks submit on the resources which are free and where the input data is available or on the other hand, tasks submit on some specific resources based on some criteria [7].

3) **Resource allocation:** The resources in the cloud computing can be allocated in many different ways. Traditional and simple method of task scheduling in cloud environment uses the client tasks as the overhead application base [1]. The allocation of resources that need to consider maximum utilization rate of resources are FCFS (First-Come-First-Service), SJF (Shortest-Job-First) scheduling, priority scheduling, RR (Round-Robin) scheduling, random, greedy, Genetic Algorithm [8]. The scheduling of tasks can also be FCFS, SJF, priority-based, RR, job grouping and so on [9]. Scheduling algorithms choose a task to be performed and corresponding

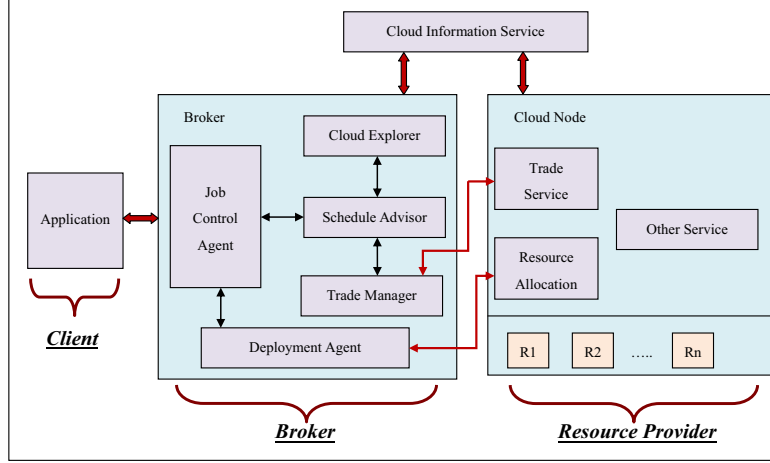


Fig. 1. Chart of scheduling model structure

resource in which the task will be executed, according to different characters of resource such as bandwidth, processing capabilities, cost, load balancing, and so on, as well as base on clients requirements of deployment.

In this paper, we both focus on resource allocation and task scheduling, as well as take into account some specific criteria or priorities of tasks and resources, such as resource bandwidth and processing capability, task granularity (fine-grained and coarse-grained) and deadline.

#### B. Task grouping scheduling algorithm

Task grouping implies that similar type or characteristic tasks can be grouped together in a group and scheduled corporately [10]. The dynamic grouping strategy mainly pays attention to utilization of resources. Clients submit tasks to the scheduler and the scheduler gains the correlative characters of resources. Then the scheduler selects a specific resource according to some priorities and multiplies resource MIPS (processing capability) with granularity size. The calculated result implies a resource total MI (processing requirement) during granularity size. Next the scheduler groups tasks of clients by accumulating each of task MI. By comparing the accumulation result (total tasks MI) and a resource total MI, when the accumulation result is over a resource total MI, grouping will be stopped. The last task added to the group should be removed and then the accumulation result subtracts the last added task MI. The final accumulation result is used for a new task MI. Sequentially, a new grouped task is created, whose MI is the final accumulation result with a unique ID. Finally, the scheduler allocates the new grouped task to the corresponding resource. The grouping process will continue until all the tasks are performed in new groups and allocated to the cloud resources. Then cloud resources execute all these new grouped tasks and send the executed grouped tasks back to the clients after completion of processing tasks.

A basic factor that influences job grouping is granularity size. Granularity size is the time when a job is processed at the resource. It is used to account for the number of jobs executed

on a specific resource during a particular time. The total number of tasks, processing requirements of tasks, the total number of resources and processing capabilities of resources need to be considered. Therefore, it is very important to determine the value of granularity size to minimize the overhead, cost and waiting time and to maximize the utilization of these resources [11].

#### C. Prioritization with bandwidth awareness

The basic concept of bandwidth-aware scheduling is used at the Stream Control Transmission Protocol (SCTP) layer [12]. A general estimation of bandwidth available on each round-trip path is operated through transmitting pairs of SCTP heartbeats on each path. The corresponding heartbeat-acks are sent back to the receiver and evaluated by using the Packet-Pair Bandwidth Estimation (PPBE) technology [13]. Due to current network low bandwidth and high latency, reasonable distribution of jobs to resources can reduce the delays and overheads. Before grouping, the scheduler receives the information of resources such as processing capabilities and network bandwidth. The scheduler sorts the resources based on descending order of network bandwidth to reduce communication latency between tasks and resources and minimize the waiting time to a certain extent. Firstly, the scheduler selects the resource with highest communication and transmission rate and groups independent fine-grained tasks based on the selected resource processing capability and after grouping sends the grouped coarse-grained task to the chosen resource. Secondly, the resource with second highest communication and transmission rate is chosen to operate tasks using the above method. This process is repeated until all the tasks are grouped and executed.

#### D. Waiting time

Waiting time is the sum of the periods spent waiting in the ready queue [14]. Minimizing waiting time is an optimized target in job scheduling, which is a famous scheduling problem and significant in providing Quality of Service (QoS) in industries [15]. For a batch of jobs, a job executing on a

resource, minimizing waiting time can reduce the completion time and meet task priority-deadline demands.

### III. THE PROPOSED ALGORITHM

The proposed algorithm integrates with the three algorithms mentioned in Section II, and they are task grouping, prioritization (bandwidth-aware) and SJF. The proposed algorithm maximizes the utilization of cloud resources, and aims to reduce task waiting time. Fig.2 shows main procedures of proposed algorithm.

#### A. Improved task scheduling algorithm

The traditional grouping-based scheduling algorithm does not consider the network bandwidth and tasks file size [3]. The traditional grouping-based scheduling algorithm only takes into account the fine-grained jobs which processing requirements are small and almost equal [1] [3] [11] [12]. However, in practice, tasks are uncertain and very random, and therefore the size of tasks cannot be very lightweight or processing requirements are not pre-established. In this paper, proposed grouping-based scheduling algorithm is suitable for both very lightweight jobs and the tasks which processing requirements are random and unpredictable.

**Improvement:** Traditional task grouping scheduling does not take into account in situations such that a task processing requirement (MI) is over each resource total MI occasionally. Under this circumstance, grouping will not continue. In the proposed algorithm, once a task MI is over each resource total MI, the task is grouped itself as a new grouped task with its own MI and a unique ID. This grouped task is allocated to the sequential resource.

#### B. Prioritization with processing capability awareness

Before grouping, the scheduler receives the information of each cloud resource processing capability. The scheduler sorts the resources based on descending order of processing capabilities (MIPS) to reduce processing time of grouped tasks and minimize the waiting time to a certain extent. Firstly, the scheduler selects the resource with largest MIPS and groups independent fine-grained tasks based on the selected resource processing capability as well as after grouping submits the grouped coarse-grained task to the chosen resource. Secondly, the resource with second largest MIPS is chosen to group the rest tasks according to the selected resource processing capability as well as similarly after grouping submits the grouped coarse-grained task to the chosen resource. This process is repeated until all the tasks are grouped and executed.

#### C. Proposed scheduling algorithm (pseudo code)

TABLE I illustrates the definitions of the proposed algorithm. The proposed algorithm can be divided into 4 phases.

1) **Phase 1:** Initialize input data and available resources. Gaussian distribution function is used to create tasks (cloudlets). These tasks processing requirements (MI) follows Normal Distribution, since actually all the tasks are generated randomly and cannot be predicted. Random function is used

to set up tasks (cloudlets) file size and output file size, which belong to Uniform Distribution. Similarly, random function is used to create resources which MIPS and bandwidth follow Uniform Distribution.

2) **Phase 2:** Prioritize tasks with different demands and factors. For bandwidth awareness, the resource which network has highest communication and transmission rate is selected firstly to reduce the transmission latency. The scheduler sorts resources based on bandwidth in a descending order. For processing capability awareness, the resource with largest processing capability is chosen firstly to decrease the execution time and minimize the overhead time. The scheduler sorts resources based on processing capabilities in a descending order. The following shows the prioritization method using the sort function.

---

#### Algorithm 1 prioritization method

---

```
1: sort (ResList, bandwidth in descending order)
2: sort (ResList, MIPS in descending order)
```

---

3) **Phase 3:** In the improved task grouping algorithm, coarse-grained task is considered and can be executed as an individual grouped task. Then the grouped task is assigned to the sequential resource. The data file size after execution is collected to calculate processing cost. Algorithm 2 describes the improved task scheduling algorithm pseudo code. Firstly, justify whether a task MI is less or more than cloud resources MIPS\*granularity\_size. If a task MI is larger than total\_resMI<sub>j</sub>, the task is grouped itself as one group. Secondly, otherwise a group is constituted by some fine-grained tasks, and maximum task file size and output file size of each group is selected as grouped task file size and grouped task output file size, correspondingly.

4) **Phase 4:** SJF scheduling algorithm is adopted to reduce tasks waiting time. In order to decrease waiting time of tasks, the grouped tasks are sorted based on grouped tasks processing requirements in ascending order. After sorting, the grouped task with shortest processing requirement is executed first on the corresponding resource, and similarly the grouped task with second shortest processing requirement is operated next on its commensurate resource. The rest can be done in the same approach. Algorithm 3 describes the pseudo code of SJF scheduling algorithm and allocation of grouped tasks to resources.

### IV. EXPERIMENT SETUP

CloudSim is a new generation and extensible simulation platform which enables seamless modeling, simulation, and experimentation of emerging Cloud computing infrastructures and management services to be accomplished [16] [17] [18]. CloudSim is used to verify the correctness of the proposed algorithm. CloudSim toolkit is used to simulate heterogeneous resource environment and the communication environment. The layered CloudSim architecture mainly contains 4 layers: SimJava, GridSim, CloudSim and User code [17].

Cloud resources are depicted as some parameters, such as Resource ID, processing elements, processing capability (MIPS), bandwidth, RAM, cost and so on in TABLE II.

In this experiment, 10 time-shared cloud resources are created to simulate and each resource is allocated in one virtual machine with different characteristics as shown in TABLE II. The bandwidth, processing capability of resources are main factors that we consider in this experiment. The

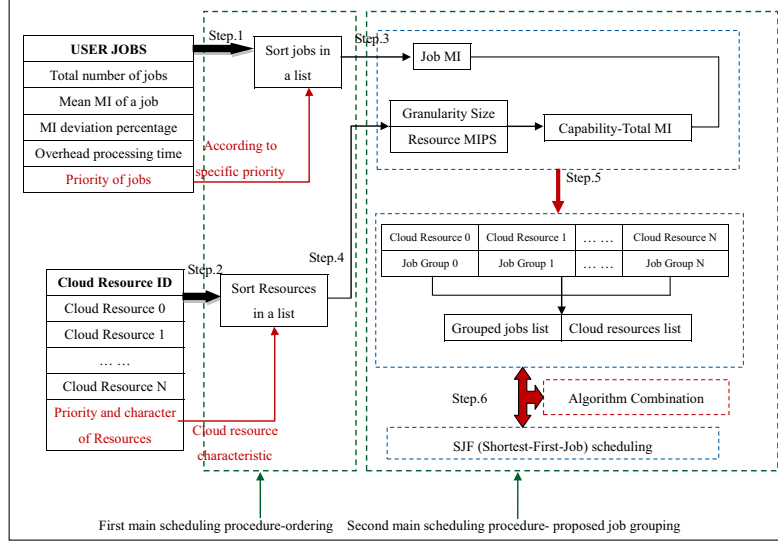


Fig. 2. Flow chart of proposed algorithm

TABLE I  
THE DEFINITIONS IN THE PROPOSED ALGORITHM

Definitions of the proposed algorithm	
MI:	Million instruction or processing requirements of a task
MIPS:	Million instruction per second or processing capability of a resource
FileSize:	The input file size of the task before submitting to a cloud Resource (Unit: Bytes)
OutputFileSize:	The output size of the task after submitting and executing to a cloud Resource (Unit: Bytes)
cloudletList <sub>i</sub> _FileSize:	The $i^{th}$ task input file size before execution
cloudletList <sub>i</sub> _OutputFileSize:	The $i^{th}$ task file size after submitting and executing to a resource
cloudletList:	List of the tasks submitted to the broker
ResList:	List of resources available in datacenter
cloudletList_size:	The total number of tasks
ResList_size:	The total number of resources
granularity_size:	Granularity size (unit: second) for task grouping
total_jobMI:	The sum of tasks processing requirement in one group
total_resMI <sub>j</sub> :	Total processing capabilities during a time of the $j^{th}$ resource
ResList <sub>j</sub> _MIPS:	The $j^{th}$ resource MIPS
max_cloudletFileSize:	The maximum of task input file size in one group
max_cloudletOutputFileSize:	The maximum of task file size after submitting and executing to a resource in one group
cloudletList <sub>i</sub> _MI:	The $i^{th}$ task processing requirement (MI)
groupedcloudletList:	List of tasks after grouping scheduling
groupedvmList:	List of resources for each task group
groupedcloudletList_size:	The total number of grouped tasks after grouping

TABLE II  
CHARACTERISTIC OF CLOUD RESOURCES FOR THE SIMULATION

Resource ID	Processing capability (MIPS)	Band-width	Mem-ory	Cost per bandwidth
R1	165	885	2GB	0.005
R2	257	971	2GB	0.005
R3	183	1100	2GB	0.005
R4	250	1264	2GB	0.005
R5	252	700	2GB	0.005
R6	201	802	2GB	0.005
R7	179	736	2GB	0.005
R8	254	1097	2GB	0.005
R9	239	771	2GB	0.005
R10	212	1023	2GB	0.005

cost of resources plays a significant part in this experiment to evaluate the total processing cost and verifies the superiority

of proposed algorithm. All the tasks submit to the datacenter broker. The processing capability (MIPS) of resources is defined as 200MI with a random variation of -30% to 30%. The bandwidth of resources is defined as 1,000 with a random variation of -30% to 30%. In CloudSim, tasks are packaged as cloudlets, which contain jobs requirements (MI), size of job input and output data (in bytes) and other various parameters related with execution when tasks are deployed to corresponding resources by broker. These cloudlets simulate Cloud-based application services, such as content transfer, social networking, and so on. Each application complexity is described by computational requirements. Therefore, each application has a pre-defined processing requirements component which is inherited from cloudlets and amount of data transfer which is related with input file size and output file

---

**Algorithm 3 SJF scheduling algorithm and allocation of grouped tasks to resources**

---

```
1: sort (groupedcloudletList, MI in ascending order)
2: for k := 0 to groupedcloudletList_size-1 do
3:   assign the groupedcloudletListk to the groupedvmListk for executing tasks
4: end for
5: for k := 0 to groupedcloudletList_size-1 do
6:   receive accomplished grouped tasks-groupedcloudletListk from the groupedvmListk
7: end for
```

---

---

**Algorithm 2 the improved task scheduling algorithm**

---

```
1: groupid = 0
2: for i := 0 to cloudletList_size-1 do
3:   m = i
4:   for j := 0 to ResList_size-1 do
5:     total_jobMI = 0
6:     max_cloudletFileSize = 0
7:     max_cloudletOutputSize = 0
8:     pre_cloudletFileSize = 0
9:     pre_cloudletOutputSize = 0
10:    total_resMIj = ResListj_MIPS*granularity_size
11:    while total_jobMI ≤ total_resMIj and
12:      i ≤ cloudletList_size-1 do
13:      total_jobMI = total_jobMI + cloudletListi_MI
14:      if
15:        max_cloudletFileSize < cloudletListi_FileSize then
16:        max_cloudletFileSize =
17:        cloudletListi_FileSize
18:      end if
19:      if
20:        max_cloudletOutputSize < cloudletListi_OutputFileSize then
21:        max_cloudletOutputSize =
22:        cloudletListi_OutputFileSize
23:      end if
24:      i++
25:    end while
26:    i--
27:    if total_jobMI > total_resMIj then
28:      total_jobMI = total_jobMI - cloudletListi_MI
29:      max_cloudletFileSize = pre_cloudletFileSize
30:      max_cloudletOutputSize =
31:      pre_cloudletOutputSize
32:      i--
33:    end if
34:    if (m-1) == i then
35:      i++
36:      total_jobMI = cloudletListi_MI
37:    end if
38:    Create a new task whose MI equals total_jobMI
39:    Set a unique ID (groupid) to the created task
40:    Insert this task into groupedcloudletList
41:    Put the task on the groupedcloudletListgroupid
42:    Insert the corresponding resource ResListj into
43:    groupedvmList
44:    Put the corresponding resource ResListj on
45:    groupedvmListgroupid
46:  end for
47: end for
```

---

size [17]. To ensure tasks are similar to real jobs in practice, we use Gaussian distribution function to generate simulated tasks and ensure their processing requirements larger than 1 MI, as well as adopt Random function to assign tasks file size and output file size. The file size of tasks is defined as

100 bytes with a random variation of -30% to 30%. Similarly, the Output file size of tasks is defined as 100 bytes with a random variation of -30% to 30%. The number of tasks (cloudlets) varies 1,000 to 7,000 with 1,000 step size. The granularity size changes 10 seconds to 30 seconds with 5 seconds step size. The selection of granularity size influences experimental evaluation. Therefore, it is an important factor of task grouping algorithm to achieve minimum tasks turnaround time and maximum cloud resources utilization.

## V. RESULTS

### A. Experimental Results

For the simulation, we focus on average waiting time, the total processing time, and processing cost.

(1) The waiting time and total finishing time is computed in seconds.

The execution time of a grouped task is given as follow:

$$T_{exec}^i = T_{comp}^i + T_{comm}^i \quad (1)$$

Where,

$T_{exec}^i$  implies the  $i^{th}$  grouped task execution time;

$T_{comp}^i$  implies the computation time of  $i^{th}$  grouped task;

$T_{comm}^i$  implies the communication time of  $i^{th}$  grouped task.

The turnaround time formula is given as follow:

Turnaround time = resource waiting time + tasks processing requirement / resource processing capability

$$T_{proc} = T_{group} + T_{sub} + T_{exec} + T_{rece} \quad (2)$$

Where,

$T_{proc}$  implies total processing time (turnaround time);

$T_{group}$  implies tasks grouping time;

$T_{sub}$  implies time when all the groups submit to resources;

$T_{exec}$  implies tasks execution time;

$T_{rece}$  implies time when all the executed tasks are received by users.

Simulations on different number of cloudlets are performed to analyze and compare with different scheduling algorithms. TABLE III shows different scheduling algorithms in this experiment. Fig.3 shows different number of tasks average waiting time with different granularity size and TABLE IV represents the improvement ratio of waiting time.

In the case of granularity size being 10 seconds, waiting time of task grouping integrated with SJF and bandwidth awareness is the smallest regardless the number of cloudlets applied. The waiting time of task grouping with SJF is the second smallest. According to TABLE IV, compared with pure

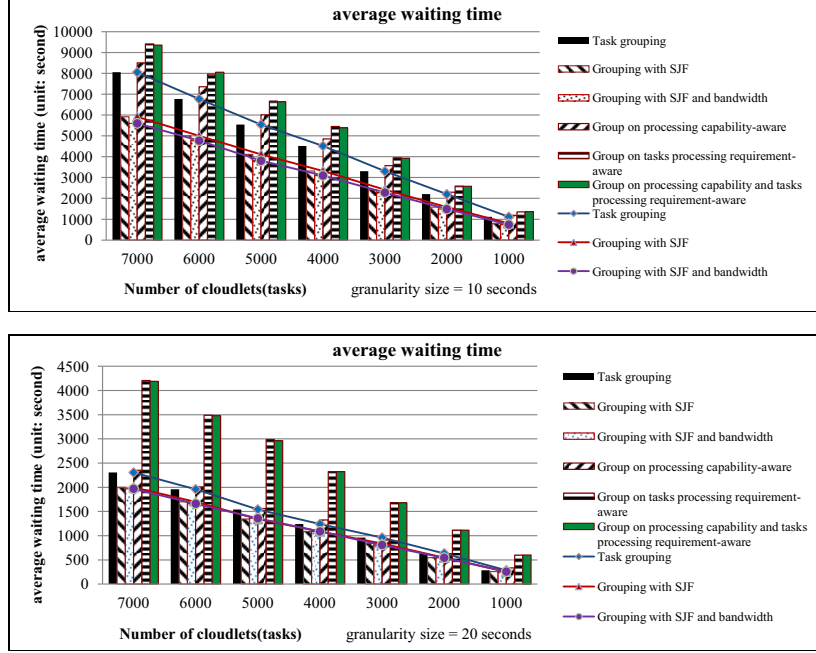


Fig. 3. Average waiting time with different number of tasks

TABLE III  
THE DEFINITIONS IN DIFFERENT SCHEDULING ALGORITHMS

definitions in different scheduling algorithms	
Task grouping	task grouping scheduling algorithm without considering any other constraint conditions on resources processing capabilities and tasks processing requirements awareness
Grouping with SJF	task grouping scheduling algorithm integrated SJF (shortest-job-first) scheduling algorithm
Grouping with SJF and bandwidth	task grouping scheduling algorithm integrated SJF (shortest-job-first) scheduling algorithm and regarding on bandwidth awareness
group on res.	task grouping scheduling algorithm regarding on resources processing capabilities awareness
group on len.	task grouping scheduling algorithm regarding on tasks processing requirements awareness
group on res. and len.	task grouping scheduling algorithm regarding on resources processing capabilities and tasks processing requirements awareness

task grouping algorithm, task grouping integrated with SJF and bandwidth awareness waiting time decreases by 31.38% while task grouping with bandwidth awareness reduced by 26.75%. The difference of these two algorithms waiting time is about 4.63%. With the reduction of number of cloudlets, the curves of these two algorithms are nearly overlapped. It indicates that when the number of tasks is the smallest, the processing requirements of tasks and processing capabilities of resources are significant factors to determine the turnaround time. When

TABLE IV  
IMPROVEMENT RATIO OF WAITING TIME

Improvement ratio of waiting time compared with task grouping ( granularity size = 10 seconds)					
No. of cloudlets (tasks)	Grouping with SJF	Grouping with SJF and bandwidth	Group on res.	Group on len.	Group on res. and len.
7000	26.56%	30.48%	-5.60%	-16.78%	-16.12%
6000	26.07%	29.54%	-8.64%	-17.60%	-18.94%
5000	25.84%	31.34%	-8.41%	-20.38%	-19.72%
4000	26.51%	31.41%	-7.45%	-20.73%	-19.38%
3000	26.79%	30.91%	-8.30%	-19.61%	-19.12%
2000	28.25%	32.48%	-4.45%	-17.64%	-17.41%
1000	27.29%	33.52%	-6.76%	-20.86%	-22.41%
Improvement ratio of waiting time compared with task grouping ( granularity size = 20 seconds)					
No. of cloudlets (tasks)	Grouping with SJF	Grouping with SJF and bandwidth	Group on res.	Group on len.	Group on res. and len.
7000	13.29%	14.62%	-1.96%	-82.41%	-81.78%
6000	13.2%	15.32%	-2.39%	-77.91%	-77.47%
5000	12.85%	11.95%	-1.34%	-93.53%	-92.02%
4000	12.19%	12.49%	-6.63%	-87.29%	-87.36%
3000	12.65%	15.85%	5.25%	-75.32%	-74.73%
2000	13.72%	14.41%	0.78%	-75.75%	-76.06%
1000	10.07%	10.25%	-16.29%	-109.13%	-111.25%

the number of tasks is very large, network bandwidth plays an important role in task scheduling, because of the network delays and communication overheads. The waiting time of traditional task grouping is the third smallest and much larger than that of above mentioned two algorithms. It is able to show that task scheduling integrated with SJF can effectively reduce waiting time. Waiting time influences deadline, as the



TABLE V  
IMPROVEMENT RATIO OF PROCESSING TIME

Improvement ratio of processing time compared with task grouping ( granularity size = 10 seconds)		
No. of cloudlets (tasks)	Grouping with SJF	Grouping with SJF and bandwidth
7000	2.97%	6.13%
6000	2.65%	5.31%
5000	3.70%	8.23%
4000	3.80%	8.22%
3000	2.19%	5.71%
2000	4.64%	7.76%
1000	4.78%	9.36%

waiting time is smaller, the tasks can be much more influential to meet the deadline constraint. However, task grouping with resources processing capabilities awareness or tasks processing requirements awareness takes much more time of execution. Compared with pure task grouping algorithm, grouping with processing capabilities awareness algorithm waiting time is increased by 7.08% and grouping with processing requirements awareness algorithm waiting time is increased by about 19.08%. We can observe that when tasks processing requirements are considered redundant, task grouping method consumes a hefty amount of time in grouping. Since tasks are sorted based on processing requirements in a descending order, the tasks with large processing requirements are difficult to be grouped together with others. Therefore, each of these kind of tasks can only be grouped themselves in an individual group.

In the case of granularity size being 20 seconds, waiting time of task grouping with SJF and network bandwidth awareness and task grouping with SJF are almost the same. Increasing granularity size will increase the processing capabilities during a period of time and under this circumstance, network latency hardly plays a part in processing tasks.

Fig.4 shows the total processing time of different number of tasks. TABLE V depicts the improvement ratio of processing time compared with task grouping. Obviously, when the number of tasks reduces, the turnaround time decreases. Compared with pure task grouping scheduling algorithm, task grouping with SJF, task grouping with SJF and bandwidth awareness, task grouping with SJF and bandwidth awareness turnaround time is smallest and decreased by 7.25% on average. Grouping with SJF processing time is decreased by 3.53% compared with pure task grouping scheduling algorithm. When the number of tasks decreases, the improvement ratio of turnaround time increases smoothly in general, and simultaneously when the number of tasks equals 5,000 or 4,000, the improvement ratio rises. In the case of granularity size being 10 seconds, communication rate-bandwidth significantly influences tasks execution performance. Therefore bottleneck bandwidth factor should be taken into account when granularity size is not very large.

Fig.5 presents average waiting time with different granularity. When granularity size is less than 35 seconds, average waiting time of all the experimental scheduling algorithms are different. Whenever the number of tasks is 5,000 or 7,000,

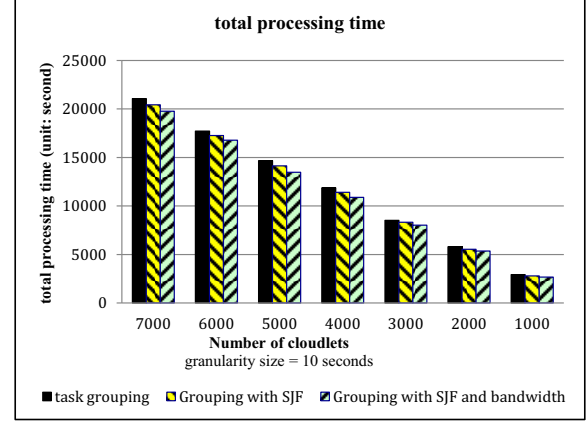


Fig. 4. Total processing time with different number of tasks

average waiting time of task grouping with task processing requirements awareness is largest, but grouping integrated with SJF and bandwidth awareness waiting time is smallest. Simultaneously, grouping with task processing requirements awareness and grouping with both processing capabilities and tasks processing requirement awareness curves are nearly overlapped. That implies when task processing requirements are considered as primary constraint condition, sorting resources based on processing capabilities almost does not affect scheduling and processing. Especially when granularity size is very large, these two curves are much more closely. These two kind algorithms average waiting time are both smaller than that of pure task grouping scheduling algorithm. Thus, we can see that even if granularity size changes, scheduling algorithms of grouping with SJF and grouping with SJF and bandwidth awareness perform better to achieve minimum turnaround time.

(2) The processing cost is calculated based on the actual CPU time when the tasks are accomplished to execute on the cloud resources and the cost of resources per second.

The processing cost formula is given as follow:

Processing Cost = input data transfer cost + processing cost + output transfer cost

$$C_{proc} = \sum_{i=0}^n T_{exec}^i * C_{reso} \quad (3)$$

Where,

$C_{proc}^i$  implies processing cost;

$\sum_{i=0}^n T_{exec}^i$  implies the total execution time of tasks (the number of tasks: n);

$C_{reso}$  implies the cost of resources per second.

Fig.6 describes processing cost with different granularity size. Fig.7 presents processing cost with different number of tasks. From Fig.7, whenever the number of tasks changes, the variance trend and difference of all the experimental scheduling algorithms are similar. When the number of tasks is between 1,000 and 7,000, the processing cost of task



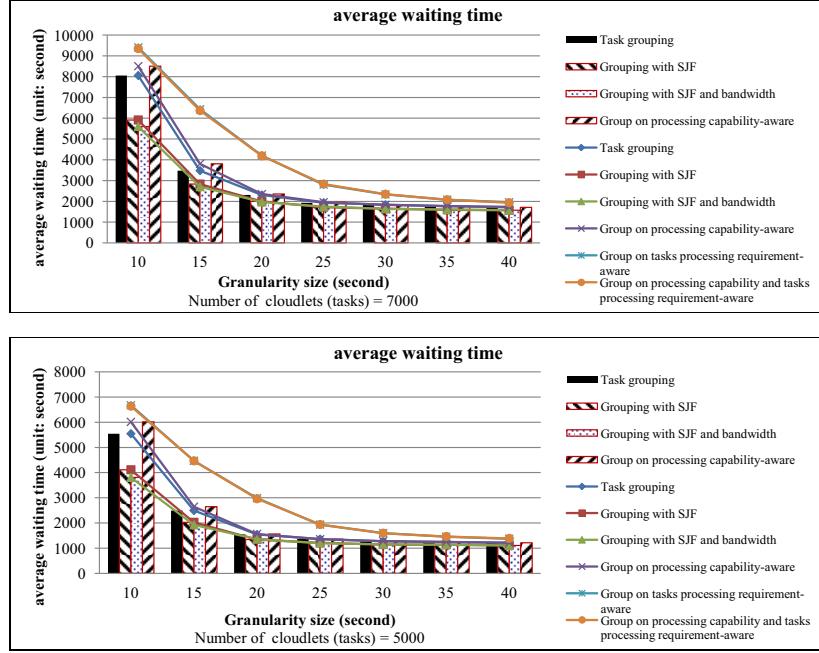


Fig. 5. Average waiting time with different granularity size

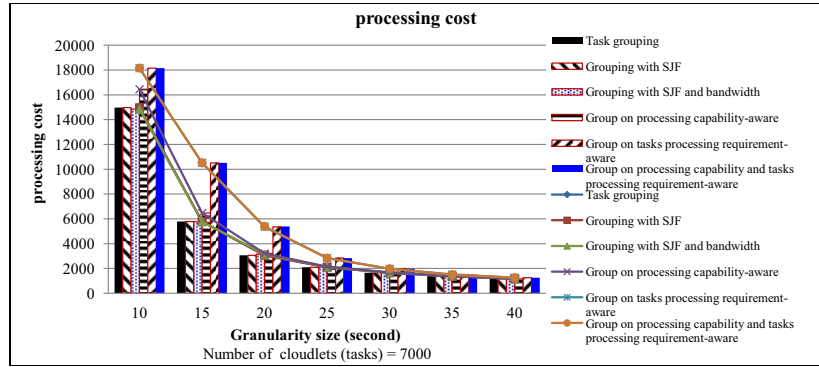


Fig. 6. Processing cost with different granularity size

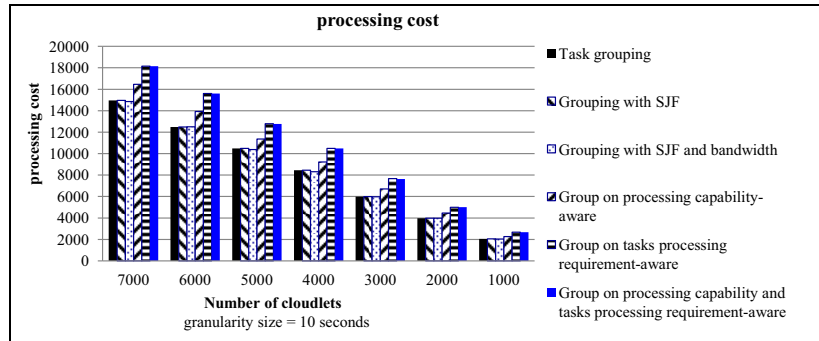


Fig. 7. Processing cost with different number of tasks

grouping, task grouping combined with SJF and task grouping integrated with SJF and bandwidth awareness are nearly the same and are smaller than that of other three algorithms.

Only when the number of tasks is over 4,000, processing cost of task grouping integrated with SJF and bandwidth awareness is smallest inconspicuously. Processing cost determines

scheduling cost. We can distinctly observe that when the constraint factor-processing capability is prioritized, processing cost is increased compared with pure task grouping. While the constraint factor-tasks processing requirements is prioritized, processing cost is largest in this experiment. Since if these constraint factors are prioritized first, then task grouping algorithms will take much time on grouping and executing, as well as the processing cost will be larger. Because our tasks adopt Normal Distribution, when some tasks processing requirements are relatively larger, grouping will cost much more time. But adopting this model enables experiments to be like practical cloud environment. In Fig.6, when granularity size is smaller than 30 seconds, processing cost decreases significantly with the improvement of granularity size. While granularity size is over 30 seconds, the curves of experimental scheduling algorithms are almost unchanged. Even if granularity size is configured larger, the processing cost does not change. Therefore, granularity size is a factor affecting processing cost and choosing an appropriate granularity size is crucial.

From the experiments, it shows the proposed task grouping algorithm integrated with SJF and bandwidth awareness can effectively minimize waiting time and processing time and reduce processing cost to achieve maximum of resources utilization and minimum overhead.

## B. Discussion

In this work, we focus on simulated experiments, and therefore all the experiments are operated in a simulated framework. The results from the experiments have been simulated for 100 times and the figures show the average results given in those simulations. Although all the hypotheses are close to real, the resources and tasks are still simulative and not real data. In the future, to gain more scientific and realistic results, we will collect real data sets and adopt the real cloud-based environment. Further experiments will be carried out on other heuristic scheduling methods such as Genetic algorithm [8] and Ant algorithm [19]. It is also important to evaluate the proposed solution in a physical real run-time environment to observe its true efficacy for cloud-based systems.

## VI. CONCLUSION

To enhance the scheduling capability on cloud computing based software systems, simulations are used to facilitate the evaluations on different approaches under various run-time scenarios in a cloud environment. The study proposes a task grouping scheduling algorithm combined with shortest-job first and bandwidth awareness algorithms in an attempt to reduce the waiting time and its associated processing costs.

Experimental results show improved utilization of resources and minimized turnaround time, as well as reduced influence on the bottleneck of bandwidth usage. Compared with existing task grouping algorithms, the proposed algorithm waiting time has been reduced by 31.38%. Even if in comparison with the task grouping with bandwidth awareness algorithm, the waiting time has also been reduced by 4.63%. The proposed

scheduling algorithm achieves a minimum waiting time which is also able to determine deadline constraints. Better yet, the processing time of proposed algorithm has also been reduced to satisfy the real-time demand of clients. The empirical results illustrate that the proposed scheduling policies are a significant improvement and serve an important baseline for benchmarking, and for the future development of scheduling algorithms for cloud-based software applications and systems.

## REFERENCES

- [1] S. Selvarani and G. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," in *Computational Intelligence and Computing Research (ICCIC)*, IEEE International Conference on, 2010.
- [2] B. Xu, C. Zhao, E. Hu, and B. Hu, "Job scheduling algorithm based on berger model in cloud environment," *Advances in Engineering Software*, vol. 42, no. 7, pp. 419–425, 2011.
- [3] Q. Liu and Y. Liao, "Grouping-based fine-grained job scheduling in grid computing," in *Education Technology and Computer Science. First International Workshop on*, 2009.
- [4] R. Buyya, D. Abramson, J. Giddy *et al.*, "An economy driven resource management architecture for global computational power grids," in *Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*, 2000, pp. 26–29.
- [5] V. Korkhov, J. Moscicki, and V. Krzhizhanovskaya, "Dynamic workload balancing of parallel applications with user-level scheduling on the grid," *Future Generation Computer Systems*, vol. 25, no. 1, pp. 28–34, 2009.
- [6] S. Gomathi and D. Manimegalai, "An adaptive grouping based job scheduling in grid computing," in *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN)*, 2011 International Conference on. IEEE, 2011, pp. 192–196.
- [7] S. Singh and K. Kant, "Greedy grid scheduling algorithm in dynamic job submission environment," in *Emerging Trends in Electrical and Computer Technology (ICETECT)*, 2011 International Conference on. IEEE, 2011, pp. 933–936.
- [8] S. Sivanandam and S. Deepa, *Introduction to genetic algorithms*. Springer Publishing Company, Incorporated, 2007.
- [9] M. Choudhary and S. Peddoju, "A dynamic optimization algorithm for task scheduling in cloud environment," *International Journal of Engineering Research and Applications*, vol. 2, pp. 2564–2568, 2012.
- [10] P. Wild, P. Johnson, and H. Johnson, "Understanding task grouping strategies," *PEOPLE AND COMPUTERS*, pp. 3–20, 2004.
- [11] N. Muthuvelu, J. Liu, N. Soe, V. Srikumar, A. Sulistio, and R. Buyya, "A dynamic job grouping-based scheduling for deploying applications with fine-grained tasks on global grids," in *Conferences in Research and Practice in Information Technology Series*, vol. 108, 2005, pp. 41–48.
- [12] T. Ang, W. Ng, T. Ling, L. Por, and C. Liew, "A bandwidth-aware job grouping-based scheduling on grid environment," *Information Technology Journal*, vol. 8, no. 3, pp. 372–377, 2009.
- [13] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks," *Performance evaluation*, vol. 27, pp. 297–318, 1996.
- [14] A. Silberschatz, P. Galvin, G. Gagne, and A. Silberschatz, *Operating system concepts*. Addison-Wesley, 1998, vol. 4.
- [15] X. Li, N. Ye, T. Liu, and Y. Sun, "Job scheduling to minimize the weighted waiting time variance of jobs," *Computers & Industrial Engineering*, vol. 52, no. 1, pp. 41–56, 2007.
- [16] R. Buyya, C. 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," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.
- [17] R. Calheiros, R. Ranjan, C. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *arXiv preprint arXiv:0903.2525*, 2009.
- [18] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
- [19] R.-S. Chang, J.-S. Chang, and P.-S. Lin, "An ant algorithm for balanced job scheduling in grids," *Future Generation computer systems*, vol. 25, no. 1, pp. 20–27, 2009.