# QoS Optimization of Service Clouds Serving Pleasingly Parallel Jobs

Xiulin Li[1], Li Pan[1(✉)], Shijun Liu[1(✉)], Yuliang Shi[1,2(✉)], and Xiangxu Meng[1]

[1] School of Software, Shandong University, Jinan 250101, China
lixiulind@163.com, {panli,lsj,shiyuliang,mxx}@sdu.edu.cn
[2] Dareway Software Co., Ltd., Jinan 250101, China

**Abstract.** A service cloud could improve its QoS (Quality of Service) by partitioning jobs into multiple tasks and processing those tasks in parallel. In contrast to processing all jobs with the same degree of parallelism (DOP), dividing jobs into different groups and processing them with varying DOPs may achieve better performance results, especially focusing on those jobs which have a greater impact on performance of service clouds. In this paper, we describe a novel differentiated DOP policy, which divides jobs into several groups identified by jobs' service time and sets proper DOPs for different groups of jobs. Then, we propose a parallel multi-queue and multi-station analytical model for service clouds with our differentiated DOP policy, to predict important performance metrics. Thus this model can guide cloud providers to determine optimal DOPs and resource allocation schemes for different groups to improve the total QoS of a service cloud. We also present a new metric, called Optimized Performance of Groups (OPG), to quantify the level of performance optimization of every group. The objective is to maximize the minimum OPG to ensure OPG within a certain range, thereby enforcing a fair trade-off between all groups. Through extensive experiments, we validate the effectiveness of the proposed differentiated DOP policy and analytical model.

**Keywords:** QoS optimization · Service cloud
Pleasingly parallel jobs · Degree of parallelism

## 1 Introduction

Because of economies of scale and other factors, service clouds usually operate a large number of resources to provide a wide range of professional services to users at low prices [5,15]. Thus more and more users are beginning to migrate their jobs to service clouds. In general, in order to meet requirements of users, service cloud providers need to provide effective job execution strategies and resource allocation schemes to accelerate jobs' execution and thus to improve QoS (Quality of Service), which is a measurement of the overall performance of a service cloud and includes response time, waiting time, probability of immediate


C. Pahl et al. (Eds.): ICSOC 2018, LNCS 11236, pp. 560–575, 2018.
https://doi.org/10.1007/978-3-030-03596-9_41

service, and so on [3,9]. Taking a rendering service cloud as an example, after accepting a batch of rendering jobs submitted by users, it could divide those jobs into different groups and process them with proper resources and degrees of parallelism (DOPs) to speed up the processing of the jobs.

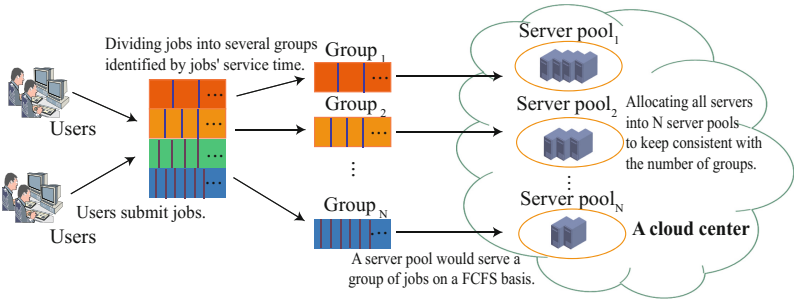


**Fig. 1.** A motivating example for a service cloud with differentiated DOP policy

A special kind of service cloud which serves pleasingly parallel jobs is considered in this paper, such as rendering service cloud [17], translating service cloud [16], designing and manufacturing service cloud [4], and so on [13]. A main characteristic of a pleasingly parallel job is that it can be partitioned into multiple tasks and these tasks are independent of one another and thus can be served separately by different servers in parallel without any overhead [6]. In contrast to processing all jobs with the same DOP, dividing jobs into different groups and processing them with varying DOPs may achieve better QoS, especially focusing on jobs that have a greater impact on performance of service clouds. However, for a service cloud provider, determining proper DOPs and resource allocation schemes for different groups of jobs to achieve optimal QoS is particulary complicated. This is mainly because of the following three reasons:

- The DOP of jobs can make a big difference in a service cloud's performance. A higher DOP could accelerate the processing of jobs, and thus improve the performance of a service cloud. However, it could cause more servers to be occupied which leads newly arrived tasks to be blocked.
- When the number of servers is fixed, allocating servers to serve different jobs can also make a big difference in a service cloud's performance. For example, if we allocate more servers for one group of jobs, there may not be enough severs available for other jobs. It could cause more jobs to wait in the buffer and increase mean response time of the jobs so that the number of completed jobs may drop in duration.
- When the optimal performance metrics are pursued, such as mean response time or mean waiting time of service clouds, the level of performance optimization of some groups of jobs may become ignored. This will lead to an unfair trade-off between different groups.

For achieving optimal QoS of service clouds, performance analytical models are usually established for service clouds, which could guide cloud providers to determine proper DOPs and resource allocation schemes. For dealing with the problem of QoS optimization for parallel jobs in service clouds, several studies have used queuing theories to establish analytical models [7,14,19]. These models could predict the important performance metrics of service clouds to help cloud providers improve the QoS of parallel jobs with the same DOP, but they cannot interpret more complicated situations where jobs have varying DOPs.

In this paper, for dealing with the problems of QoS optimization of service clouds serving jobs with varying DOPs, based on queuing theory, we propose a parallel multi-queue and multi-station analytical model to evaluate service clouds' performance. Based on homogeneous Markov chain model, we solve our analytical model to obtain an approximate estimation of important performance metrics such as mean service response time, mean waiting time, and so on. As different sizes of jobs have diverse impacts on performance of service clouds, we propose a differentiated DOP policy which divides jobs into several groups identified by jobs' service time and sets proper DOPs for different groups of jobs, as shown in Fig. 1. Then, we divide all servers into multiple server pools to remain consistent with the number of groups and one server pool serves one group of jobs on a FCFS (First Come First Serve) basis. Thus, based on our analytical model, we can determine optimal DOPs and resource allocation schemes for different groups of jobs, and thus improve the QoS of service clouds. We also propose a metric, called Optimized Performance of Groups (OPG), to quantify the level of performance optimization of each group, which is the ratio between the response time a group of jobs has spent in a service cloud without differentiated DOP policy and the response time this group would have spent in that service cloud with differentiated DOP policy. The objective is to maximize the minimum OPG to ensure OPG within a certain range, thereby enforcing a fair trade-off between all groups. Through extensive simulations based on synthetic data, we validate the effectiveness of our analytical model and differentiated DOP policy.

The rest of the paper is organized as follows. Section 2 describes our differentiated DOP policy. Section 3 describes our parallel multi-queue and multi-station model and solves our model to obtain performance metrics for a service cloud. Section 4 describes our proposed performance metric. Section 5 presents and discusses analytical results as well as simulation results. We discuss the related works in Sect. 6. Finally, we state concluding remarks and future work in Sect. 7.

## 2 Differentiated DOP Policy

Dividing jobs into different groups and processing them with varying DOPs may achieve better QoS, especially focusing on jobs that have a greater impact on performance of service clouds. In this paper, we describe a differentiated DOP policy in order to set proper DOPs for different jobs. Since different sizes of jobs have a diverse impact on performance, it could improve the QoS of service clouds obviously if we set a higher DOP for jobs which require longer service

time. In this work, we divide jobs into $N$ groups which are $\{group_1, group_2, \ldots, group_N\}$ identified by jobs' service time. There are $N+1$ points of service time: $\{t_0, t_1, \ldots, t_N\}$ and $t_0 < t_1, \ldots, < t_N$. If the service time of a job is within the range of $t_{i-1}$ to $t_i$, this job will be divided into $group_i$, so that service time of jobs in $group_i$ is within the range of $t_{i-1}$ to $t_i$, for $1 \leq i \leq N$. Then, we could set a proper DOP for each group of jobs and use $d_i$ to denote the DOP of $group_i$.

We only consider the case of identical servers and allocate $c$ identical servers contained in a service cloud into $N$ server pools to remain consistent with the number of groups, which are $\{pool_1, pool_2, \ldots, pool_N\}$, as shown in Fig. 1. Servers in a server pool would serve a group of jobs on a FCFS basis, e.g., $pool_i$ contains $c_i$ servers that serve jobs from $group_i$. Therefore, we have:

$$\sum_{i=1}^{N} c_i = c \tag{1}$$

In order to achieve specific QoS and save costs, we need to determine optimal settings, such as resource allocation schemes or DOP settings. Thus, a proper analytical model is needed to predict the performance metrics with different resource allocation schemes and DOP settings.

## 3 Analytical Model Formulation

In this work, based on queuing theory, we propose a parallel multi-queue and multi-station model to evaluate the performance of service clouds serving jobs with varying DOPs, as shown in Fig. 2. $N$ queues can be constructed for a service cloud with differentiated DOP policy, which are $\{queue_1, queue_2, \ldots, queue_N\}$. In $queue_i$ $(1 \leq i \leq N)$, $c_i$ servers in server $pool_i$ serve jobs from $group_i$.

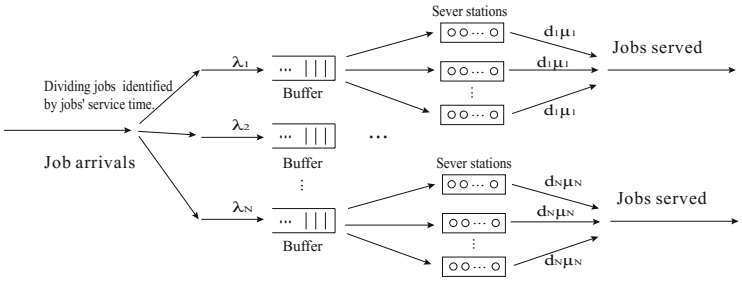


**Fig. 2.** A parallel multi-queue and multi-station model for jobs with varying DOPs

In order to obtain performance metrics of a service cloud, we need to predict performance metrics of all queues in that service cloud. For $queue_i$ that is any one of all queues, before jobs are partitioned, we assume that job arrivals follow an approximate Poisson process with a rate of $\lambda_i$ and service time is an approximate

exponentially distributed with a rate of $\mu_i$, as is widely assumed in previous works [20]. This is because if job arrivals follow a Poisson process and service time is exponentially distributed, analytical results of these jobs can supply approximate lower bounds in performance prediction for other distributions of job arrivals, such as random distributions or normal distributions [14].

As discussed above, before jobs are partitioned, it can be concluded that $queue_i$ can be modeled as an $M/M/c_i$ queuing system. However, after jobs are partitioned into tasks, the actual probability distribution of service time would not follow the original distribution. As tasks will occupy more servers, the number of busy servers is different from the number of jobs in service clouds. Thus, we need to carefully establish a parallel multi-station analytical model for $queue_i$ serving the tasks partitioned from the original jobs, which is an important part of the parallel multi-queue and multi-station model.

### 3.1 Multi-station Model for $Queue_i$

Based on $M/M/c_i$ queuing model of $queue_i$, we describe a parallel multi-station model for $queue_i$, as shown in Fig. 2. In this work, a partition method is designed to help us establish a multi-station model for $queue_i$. All $c_i$ servers contained in $queue_i$ are divided into $d_i$ stations of servers, which are $\{C_{i1}, C_{i2}, \ldots, C_{id}\}$. We use $c_{ij}$ $(j \in \{1, \ldots, d_i\})$ to denote the number of servers contained in the station $C_{ij}$. Here when dividing servers into multiple stations, a main principle we use is to *equally* divide all servers into $d_i$ stations. Therefore, each server station consists of the same number of servers, which means $c_{i1} = c_{i2} = \ldots = c_{id} = \lfloor c_i/d_i \rfloor$. At each station, tasks will be served on a FCFS basis by one of the servers in that station. If at least one server is idle in each server station at the time when a job arrives, tasks of this job will get into service immediately. If all servers are busy at the time when a job arrives, the job will wait in the buffer. We assume that the buffer size of a service cloud is unlimited, so that jobs won't be lost. Based on our partition method, we describe how to obtain the relevant parameters of tasks in every station.

If a system could be modeled as an $M/M/c$ queuing system and jobs in this system could be equally partitioned into $k$ $(k > 0)$ tasks, as the property of exponential distribution, the service time of tasks is also exponentially distributed [2]. However, its base has changed, and the mean service time of the tasks is $1/k$ of the mean job service time. In our analytical model, before entering into servers, each job is *equally* partitioned into $d_i$ tasks which have the same service time. Then by dividing all servers equally into $d_i$ stations, after an arriving job is equally split into $d_i$ tasks, each task could be assigned to one of the $d_i$ stations individually and no two stations could accept a same task. Thus all the stations have the same workload to serve and since they have the same number of servers with the same processing capability, they would achieve the same performance metrics in processing these tasks. Based on the above description, we can conclude that the service time in each server station is an independent identically distributed (IID) random variable having negative exponential distributions, with mean service time of $1/(d_i\mu_i)$. The arrival of each job is the

same as its tasks. So task arrivals follow a Poisson process, which means the task interarrival time is exponentially distributed with a rate of $\lambda_i$, the same as the primitive job. The traffic intensity $\rho_i$ denotes the average proportion of time for which each of the servers is occupied. Thus, the traffic intensity of one server station could be defined as $\rho_i = \lambda_i/(c_{i1}d_i\mu_i)$, where for practical reasons, it is known that $\rho_i < 1$.

As discussed above, we can establish an $M/M/c_{i1}$ queuing system for each station. All $d_i$ stations have the same processing capability and the same workload, so that the performance metric of one station can represent the performance of other stations as well as $queue_i$. Thus we can analyze the server station $C_{i1}$ to obtain its performance metrics through the proposed analytical model and then conclude performance metrics of the whole $queue_i$ from that.

In order to construct the analytical functions to calculate the important performance metrics, we need to calculate the steady-state probabilities of the number of tasks in the $M/M/c_{i1}$ queuing system of server station $C_{i1}$, which can be figured out by using a homogeneous Markov chain.

**Underlying Markov Model.** In this work, we establish a homogeneous Markov chain for server station $C_{i1}$ to calculate the important performance metrics. We model the number of tasks in server station $C_{i1}$ (both those in service and those queued but not yet served) at the moments immediately before a new task arrival as a Markov point. Then by enumerating these instances as $\{0, 1, 2, \ldots, n-1, n\}$, we can obtain a homogeneous Markov chain. Therefore, we can calculate the steady-state probabilities of the number of tasks in the queuing system to obtain the performance metrics of this server station.

When the steady-state system stays at state $n$, we use $P_n$ $(0 \leq n)$ to denote the probability of having $n$ tasks in the system immediately before a new task arrives. The balance equations can be established as:

$$d_i\mu_i P_1 = \lambda_i P_0 \tag{2}$$

$$(n+1)d_i\mu_i P_{n+1} + \lambda_i P_{n-1} = (\lambda_i + nd_i\mu_i)P_n \quad (1 \leq n \leq c_{i1}) \tag{3}$$

$$c_{i1}d_i\mu_i P_{n+1} + \lambda_i P_{n-1} = (\lambda_i + c_{i1}d_i\mu_i)P_n \quad (c_{i1} < n) \tag{4}$$

The normalization equation of the transition probability matrix $P$ is:

$$\sum_{j=0}^{\infty} P_j = 1 \tag{5}$$

As we have defined above, the traffic intensity for $C_{i1}$ server station is:

$$\rho_i = \lambda_i/(c_{i1}d_i\mu_i) \qquad (\rho_i < 1) \tag{6}$$

Using recurrence relations to solve differential equation (2) (3) (4) via (5) (6), in which $P_n$ can be deduced from $P_{n-1}$ that we have known, we can obtain the steady-state probability as:

$$P_0 = [\sum_{k=0}^{c_{i1}-1} \frac{1}{k!}(\frac{\lambda_i}{d_i\mu_i})^k + \frac{1}{c_{i1}! \times (1-\rho_i)}(\frac{\lambda_i}{d_i\mu_i})^{c_{i1}}]^{-1} \tag{7}$$

$$P_n = \begin{cases} \frac{1}{n!}(\frac{\lambda_i}{d_i\mu_i})^n P_0 & (n \leq c_{i1}) \\ \frac{1}{c_{i1}! \times {c_{i1}}^{n-c_{i1}}}(\frac{\lambda_i}{d_i\mu_i})^n P_0 & (c_{i1} < n) \end{cases} \tag{8}$$

**Calculating Performance Metrics for *Queue*$_i$.** Once we have obtained the steady-state probabilities, which are the basis of steady-state queuing service system, we can use them to construct analytical functions to calculate the important performance metrics concerning $C_{i1}$ server station. As previously discussed, the performance metrics of one server station can be taken as the performance metrics of $queue_i$. Now we illustrate how to calculate the following five important performance metrics, which are the mean number of tasks queued in the buffer, the mean number of tasks in queuing system, mean response time, mean waiting time, and the probability of immediate service.

We use $Lq_i$ to denote the expectations of the mean number of jobs queued in the buffer of $queue_i$, which is the waiting queue length. Therefore:

$$Lq_i = \sum_{n=c_{i1}+1}^{\infty} (n - c_{i1})P_n \tag{9}$$

We use $Ls_i$ to denote the expectations of the mean number of all jobs currently in $queue_i$ (including both jobs being served in servers and jobs waiting in the buffer). Therefore:

$$Ls_i = \frac{\lambda_i}{d_i\mu_i} + Lq_i \tag{10}$$

We use $Ws_i$ to denote mean response time of $queue_i$, which is the sum of jobs' service time and jobs' waiting time. We have discussed that the average number of all jobs in the system is $Ls_i$, thus according to Little's Law [8]:

$$Ws_i = \frac{Ls_i}{\lambda_i} \tag{11}$$

As is known, when a job arrives, if all servers are busy, the job will wait in the buffer until it can be served by an idle server. Thus, according to Little's Law, the average waiting time can be calculated as:

$$Wq_i = \frac{Lq_i}{\lambda_i} \tag{12}$$

At the time when a new job arrives, if there is no job waiting in the buffer and at least one server out of a station is idle, the new job will be served immediately.

Thus the probability of immediate service $PI_i$ can be calculated as:

$$PI_i = \sum_{n=0}^{c_{i1}-1} P_n \tag{13}$$

### 3.2 Calculating the Performance Metrics for a Service Cloud

Once the performance metrics of $queue_i$ are obtained, we can use them to calculate the performance metrics concerning a service cloud, which are the mean of all jobs' performance value. Using $Ws$ to denote mean response time of a service cloud and $Q_i$ ($i \in \{1, \ldots, N\}$) to denote the number of jobs contained in $queue_i$. Mean response time of a service cloud can be calculated as:

$$Ws = \frac{\sum_{i=1}^{N} Q_i \cdot Ws_i}{\sum_{i=1}^{N} Q_i} \tag{14}$$

Using $Wq$ to denote mean waiting time of a service cloud and $Wq_i$ to denote mean waiting time of $queue_i$, mean waiting time of a service cloud can be calculated as:

$$Wq = \frac{\sum_{i=1}^{N} Q_i \cdot Wq_i}{\sum_{i=1}^{N} Q_i} \tag{15}$$

Using $PI$ to denote the probability of immediate service of a service cloud and $PI_i$ to denote the probability of immediate service of $queue_i$, the probability of immediate service of a service cloud can be calculated as:

$$PI = \frac{\sum_{i=1}^{N} Q_i \cdot PI_i}{\sum_{i=1}^{N} Q_i} \tag{16}$$

Based on the analytical results, we can determine proper DOP settings and resource allocation schemes to achieve the optimal mean response time, mean waiting time and the probability of immediate service, and thus to improve the QoS of service clouds.

## 4 The Proposed Performance Metric

However, when optimal performance metrics are pursued, such as mean response time or mean waiting time of service clouds, the level of performance optimization of some groups may be ignored. This will lead to an unfair trade-off between different groups. Therefore, we propose a new metric, called Optimized Performance of Groups (OPG), which is the ratio between the response time a group of jobs has spent in a service cloud without differentiated DOP policy and the response time this group would have spent in that service cloud with differentiated DOP policy. Using $Wsp_i$ to denote the response time that $group_i$ has spent in a service cloud without differentiated DOP policy and $Wsd_i$ to denote the

response time that $group_i$ of the jobs would have spent in a service cloud with differentiated DOP policy. Hence, $OPG_i$ can be formulated as follows:

$$OPG_i = \frac{Wsp_i}{Wsd_i} \tag{17}$$

The metric $OPG_i$ is used to evaluate the level of performance optimization of $group_i$. $OPG_i$ is within the range of $(0, +\infty)$, and the lower it is, the worse level of performance optimization the $group_i$ has. If the minimum OPG of all groups is less than a certain value which is a lower bound of the level of performance optimization, we need to improve the OPG of the group which has the minimum OPG of all groups by determining a higher DOP or allocating more servers to this group. Our final objective is to maximize the minimum $OPG_i$ to ensure $OPG_i$ within a certain range, thereby enforcing a fair trade-off between all groups when mean response time of all jobs are optimized.

## 5 Experimental Evaluation

In order to evaluate the effectiveness of our proposed differentiated DOP policy and analytical model, we have built a discrete event simulator of the cloud server system and conducted extensive experiments based on synthetic data extracted from real-world rendering jobs.

### 5.1 Simulation Methodology and Parameter Settings

There are four main purposes of our experimental simulations. First, the effectiveness of our proposed differentiated DOP policy is validated by comparing it to the performance of a service cloud without the policy. Second, the effectiveness of our proposed analytical model is validated by investigating whether the performance metrics predicted by the analytical model are close to the simulation results. Third, we examine how the parameter settings, such as different combinations of DOP settings or resource allocation schemes, can improve the QoS of a service cloud. Finally, we investigate whether the minimum $OPG$ of all experiments is higher than the lower bound of the level of performance optimization and if not, we had better maximize the minimum $OPG_i$ of that experiment by increasing the DOP or allocating more servers to $group_i$ to enforce a fair trade-off between all groups. For these aims, we built a simulator, in which we can change the number of servers, resource allocation schemes and DOPs.

In all of the simulations, the number of jobs used in each simulation is above 10000, and the testing time is more than 168 h (7 days). The service time of all jobs is collected from our working rendering service platform, which is within the range of [70 min, 130 min). Jobs are assigned into three groups identified by jobs' service time. Jobs' service time of $group_1$ is within the range of [70 min, 90 min) and mean service time is 80 min. Jobs' service time of $group_2$ is within the range of [90 min, 110 min) and mean service time is 100 min. Jobs' service time of $group_3$ is within the range of [110 min, 130 min] and mean service time is 120 min.

The jobs' service time of all groups is approximate exponentially distributed. Considering the characteristics of real-world rendering job arrivals, we assume that job arrivals of all groups follow an approximate Poisson process and the mean job interarrival time of $group_1$, $group_2$ and $group_3$ are 1.5 min, 1.4 min and 1.35 min respectively. Then, we could set a DOP for each group. Besides, we assume that the lower bound of the level of performance optimization is 0.9. Jobs in service clouds won't be lost as buffer size without limit. These settings are in reasonable sizes considering the real-world rendering service cloud and they are kept the same for all of the simulations. For other parameters concerning each simulation, we will explain them in the respective subsections.

### 5.2 Effect of Resource Allocation on Performance

**Table 1.** Resource allocation schemes

| Experiment number | NoS in $group_1$ | NoS in $group_2$ | NoS in $group_3$ | All servers |
|---|---|---|---|---|
| Experiment 1 | 58 | 76 | 106 | 240 |
| Experiment 2 | 58 | 84 | 98 | 240 |
| Experiment 3 | 62 | 80 | 98 | 240 |
| Experiment 4 | 62 | 84 | 94 | 240 |
| Experiment 5 | 66 | 76 | 98 | 240 |
| Experiment 6 | 66 | 80 | 94 | 240 |

In this subsection, we investigate the influence of different resource allocation schemes on a service cloud's performance. Cloud providers need to determine how to allocate servers to serve different groups to achieve an optimal performance of a service cloud. For this aim, we randomly select six experiments, which have different resource allocation schemes for three groups, and investigate which scheme has better mean response time, mean waiting time or OPG. Experiment settings are introduced in Table 1. Besides, we use 'NoS' to denote the number of servers.

Response time, waiting time and the minimum OPG are shown in Table 2. It shows that the resulting mean response time and mean waiting time of all jobs are different with the change of resource allocation schemes. A service cloud would get an optimal response time and waiting time when 62 servers are allocated to serve $group_1$, 80 servers are allocated to serve $group_2$ and 98 servers are allocated to serve $group_3$. We find that compared with other experiments, groups in experiment 3 don't have higher traffic intensity. We could observe that allocating fewer servers to serve the group in which jobs require shorter service time would lead to a poor result, as the change of NoS has a greater influence in traffic intensity of that group.

Table 2 also shows that the simulation results are less than the analytical results when the traffic intensity is bigger. This is mainly because the following

**Table 2.** Effect of resource allocation on performance

| Experiments | Method | Response time | Waiting time | The minimum OPG |
|---|---|---|---|---|
| 1 | Simulation | 55.41 | 5.25 | 0.87 |
| | Analytical | 57.36 | 7.35 | |
| 2 | Simulation | 53.54 | 3.37 | 0.87 |
| | Analytical | 55.50 | 5.50 | |
| 3 | Simulation | 52.50 | 2.32 | 0.96 |
| | Analytical | 54.05 | 4.05 | |
| 4 | Simulation | 54.35 | 4.18 | 0.91 |
| | Analytical | 56.29 | 6.29 | |
| 5 | Simulation | 53.22 | 3.06 | 0.91 |
| | Analytical | 55.91 | 5.91 | |
| 6 | Simulation | 53.30 | 3.14 | 0.91 |
| | Analytical | 56.45 | 6.45 | |

two reasons. First, queuing theory has assumed that the jobs' service time is within the range of $[0, +\infty)$. In reality, the jobs' service time of one group is within the certain range, e.g., jobs' service time of $group_1$ is within the range of [70 min, 90 min). Obviously, the jobs in each group don't contain the jobs with a service time that exceeds the upper limit or that is under the lower limit, which has a significant impact on performance. Second, as the service time of all jobs is collected from real-world working rendering service platform, jobs' service time in each group is not completely following exponential distribution. So analytical results and simulation results show little difference, which is less than 10 min. More importantly, our analytical model can accurately indicate the trend of performance change and its results can supply approximate lower bounds in all situations. Thus, we can conclude that the analytical model could help us to determine proper resource allocation schemes to achieve optimal QoS of service clouds.

The resulting minimum OPG of all experiments is also detailed in Table 2. OPG of experiments 1 and 2 are less than 0.9. In this situation, we had better allocate more servers to the group which has the minimum OPG to ensure a fair trade-off between all groups.

### 5.3 Effect of DOP on Performance

In this subsection, we investigate the influence of different combinations of DOP settings on response time and the immediate service rate. We also investigate the minimum OPG of every experiment to ensure a fair trade-off between all groups. 240 servers contained in a service cloud are allocated to three groups. Considering the same traffic intensity, we allocate 60 servers to serve $group_1$, 80 servers to serve $group_2$ and 100 servers to serve $group_3$. In all cases, the

traffic intensity is reasonably set as $\rho = 0.88$. We then select 8 experiments, that contain different combinations of DOP settings. Experiment settings are introduced in Table 3. Experiment 1 and 5 are comparative experiments which won't use our differentiated DOP policy. Then we compare the performance metrics of experiment 2, 3, and 4 with experiment 1. We also compare the performance metrics of experiment 6, 7, and 8 with experiment 5.

**Table 3.** Experiment settings with varying DOPs and their minimum OPG

| Experiments | DOP of $group_1$ | DOP of $group_2$ | DOP of $group_3$ | The minimum OPG |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | |
| 2 | 2 | 2 | 5 | 0.93 |
| 3 | 2 | 5 | 2 | 0.93 |
| 4 | 5 | 2 | 2 | 0.97 |
| 5 | 4 | 4 | 4 | |
| 6 | 2 | 4 | 10 | 0.47 |
| 7 | 4 | 4 | 10 | 0.85 |
| 8 | 4 | 5 | 10 | 0.85 |

The minimum OPG of all experiments is shown in Table 3. The OPG of experiment 6 is 0.47, which means at least one group' performance is seriously worse than before. Thus, even if mean response time of all jobs is better than before, this parameter setting is not recommended. In this situation, we had better set a higher DOP to the group, which has the minimum OPG in all groups, to ensure a fair trade-off between all groups.

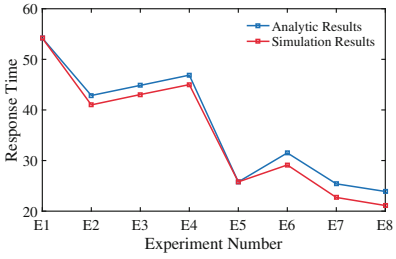


**Fig. 3.** Mean response time

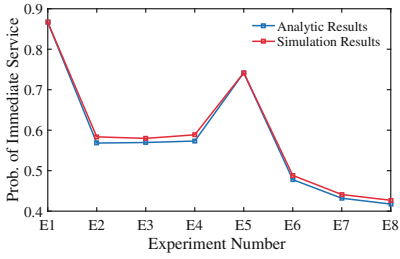


**Fig. 4.** Immediate service rate

The resulting response time is shown in Fig. 3. It can be noted that in contrast to processing all jobs with the same DOP, dividing jobs into different groups and processing them with varying DOPs has diverse service response time. As

different groups of jobs have different influences on performance, we find that setting a higher DOP to a group that requires longer service time could achieve optimal performance, which proves the validity of our differentiated DOP policy.

Comparing with experiment 5, experiment 6 has poor performance results which include mean response time and the minimum OPG. Thus, we need to set a higher DOP for the group that has the minimum OPG of all groups to optimize mean response time, as shown in experiment 7 or 8. Therefore, analytical model could be helpful in setting DOPs for different groups to achieve optimal performance.

The probability of immediate service is shown in Fig. 4. In contrast to treating all jobs with the same DOP, treating jobs with varying DOPs produces an inferior immediate service rate. With the increasing of DOP, immediate service rate of all jobs decrease rapidly. Thus, when pursuing optimal immediate service rate, DOP should be limited.

Figures 3 and 4 also show that analysis results given by our proposed analytical model are close to simulation results and the results can supply approximate lower bounds in all situations. Thus, the analytical functions are valid for estimating response time and the probability of immediate service. Finally, we can conclude that the analytical model could help us to determine proper DOPs for different group to achieve optimal QoS of service clouds.

## 6 Related Work

With the rapid development of cloud computing technologies, there are more and more researchers focusing on optimizing QoS of a cloud center by establishing analytical models. Performance analysis generally focuses on evaluating a system's key performance metrics such as response time, throughput, and so on [18].

In [9,20], the cloud center is modeled as a queuing system with single task arrival and finite buffer capacity. For evaluating the performance of cloud computing, these analytical models obtain accurate estimations of the complete probability distribution of response time and other important performance indicators. Based on these accurate estimations, cloud providers could determine proper buffer space for different classes of tasks to avoid sudden long delay.

[11] and [10] propose analytical performance models that address the complexity of cloud centers. It has been assumed that a cloud center has a number of servers and each server has been configured as a number of virtual machines (VMs). They obtain a detailed assessment of cloud center performance. Several performance metrics are defined and evaluated to analyze the behavior of a cloud data center: utilization, availability, waiting time, and responsiveness [1]. A resiliency analysis is also provided to take into account load bursts. However, these analytical models cannot work well with parallel jobs.

In [12], the analysis model has been extended to a system where jobs consist of multiple tasks that have general independent service time distributions. The model could account for the deterioration of performance due to the workload

at each node. In [2], the authors propose a mathematical model to predict the performance of parallel jobs based on a queuing system. This model has a good accuracy in predicting execution time and waiting time. In [19], the researchers focuses on the jobs that may consist of multiple tasks with each task requiring a VM for its execution. It has derived job blocking probabilities and distribution of the utilization of resources as a function of the traffic load under various scenarios for systems with both homogenous and heterogeneous VMs. In a previous study, we have proposed an approximate analytical model for cloud computing centers serving parallelizable jobs using M/M/c/r queuing systems [14]. This model has a good accuracy in predicting metrics of parallelizable jobs with the same DOP. The above models concern parallel jobs, but they didn't consider the problem of differentiated processing of jobs.

As discussed above, current performance analysis models cannot deal with the situations of service clouds serving parallel jobs with varying DOPs. Our approach can establish an analytical model for jobs with varying DOPs. Using this model, we can obtain an accurate estimation of the complete probability distribution of response time, waiting time and probability of immediate service, which could guide cloud providers to determine optimal DOPs and resource allocation schemes to achieve specific QoS of service clouds.

## 7 Conclusions and Future Work

Dividing jobs into different groups and processing them with varying DOPs could achieve better performance results, especially focusing on jobs that have a greater impact on performance of service clouds. In this paper, for a service cloud that serves pleasingly parallel jobs, we propose a differentiated DOP policy, which divides jobs into several groups identified by jobs' service time and sets proper DOPs for different groups of jobs. Then, we propose a parallel multi-queue and multi-station analytical model for a service cloud with our proposed policy to predict performance metrics. This model could help us make optimized decisions in determining DOPs and resource allocation schemes for different groups of jobs, and thus to improve the QoS of a service cloud. Besides, we present a metric named OPG, which quantifies the level of performance optimization of every group. The objective is to maximize the minimum OPG, thereby enforcing a fair trade-off between all groups. Through extensive experiments based on synthetic data extracted from real-world rendering jobs, we validate the effectiveness of our differentiated DOP policy and analytical model.

For the future work, we plan to extend our research to QoS optimization of service clouds which contain composite parallelizable services. Besides, we also plan to further validate our approach by collecting workload traces from more real-world cloud systems.

**Acknowledgments.** The authors would like to acknowledge the support provided by the National Key Research and Development Program of China (2017YFA0700601, 2018YFB1003800), the Key Research and Development Program of Shandong Province (2017CXGC0605, 2017CXGC0604, 2018GGX101019, 2016GGX106001,

2016GGX101008, 2016ZDJS01A09), the Natural Science Foundation of Shandong Province for Major Basic Research Projects (No. ZR2017ZB0419), the Young Scholars Program of Shandong University, and the TaiShan Industrial Experts Program of Shandong Province (tscy20150305).

## References

1. Bruneo, D.: A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. IEEE Trans. Parallel Distrib. Syst. (TPDS 2014) **25**(3), 560–569 (2014)
2. Chao, S., et al.: Predicting the performance of parallel computing models using queuing system. In: Proceedings of International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2015), pp. 757–760 (2015)
3. Cordeiro, T.D., et al.: Open source cloud computing platforms. In: Proceedings of International Conference on Grid and Cooperative Computing (GCC 2010), Nanjing, China, pp. 366–371 (2010)
4. Dazhong, W., et al.: Cloud-based design and manufacturing: a new paradigm in digital manufacturing and design innovation. Comput.-Aided Des. **59**, 1–4 (2015)
5. Feng, G., Buyya, R.: Maximum revenue-oriented resource allocation in cloud. Int. J. Grid Util. Comput. **7**(1), 12–21 (2016)
6. Gunarathne, T., et al.: Cloud computing paradigms for pleasingly parallel biomedical applications. In: Proceedings of ACM International Symposium on High Performance Distributed Computing (HPDC 2010), Chicago, Illinois, pp. 460–469 (2010)
7. Haghighi, A.M., Mishev, D.P.: A parallel priority queueing system with finite buffers. J. Parallel Distrib. Comput. **66**(3), 379–392 (2006)
8. Keilson, J., Servi, L.D.: A distributional form of little's law. Oper. Res. Lett. **7**(5), 223–227 (1988)
9. Khazaei, H., et al.: Performance analysis of cloud computing centers using m/g/m/m+r queuing systems. IEEE Trans. Parallel Distrib. Syst. (TPDS 2012) **23**(5), 936–943 (2012)
10. Khazaei, H., et al.: Analysis of a pool management scheme for cloud computing centers. IEEE Trans. Parallel Distrib. Syst. (TPDS 2013) **24**(5), 849–861 (2013)
11. Khazaei, H., et al.: A fine-grained performance model of cloud computing centers. IEEE Trans. Parallel Distrib. Syst. (TPDS 2013) **24**(11), 2138–2147 (2013)
12. Khazaei, H., et al.: Performance of cloud centers with high degree of virtualization under batch task arrivals. IEEE Trans. Parallel Distrib. Syst. (TPDS 2013) **24**(12), 2429–2438 (2013)
13. Li, S., et al.: Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud. IEEE Trans. Parallel Distrib. Syst. (TPDS 2017) **28**(6), 1607–1620 (2017)
14. Li, X., et al.: Performance analysis of cloud computing centers serving parallelizable rendering jobs using m/m/c/r queuing systems. In: Proceedings of International Conference on Distributed Computing Systems (ICDCS 2017), Atlanta, GA, USA, pp. 1378–1388 (2017)
15. Liu, C., Shang, Y., Duan, L., Chen, S., Liu, C., Chen, J.: Optimizing workload category for adaptive workload prediction in service clouds. In: Barros, A., Grigori, D., Narendra, N.C., Dam, H.K. (eds.) ICSOC 2015. LNCS, vol. 9435, pp. 87–104. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48616-0_6

16. Mai, X.T., et al.: Policy-aware optimization of parallel execution of composite services. In: Proceedings of International Conference on Services Computing (SCC 2015), New York, USA, pp. 106–113 (2015)
17. Pan, L., An, B., et al.: Nash equilibrium and decentralized pricing for qos aware service composition in cloud computing environments. In: Proceedings of International Conference on Web Services (ICWS 2017), Honolulu, HI, USA, pp. 154–163 (2017)
18. Seneviratne, S., Levy, D., Buyya, R.: A taxonomy of performance prediction systems in the parallel and distributed computing grids. Eprint Arxiv arXiv:1307.2380 (2013)
19. Vakilinia, S.: Modeling of the resource allocation in cloud computing centers. Int. J. Comput. Telecommun. Netw. **91**, 453–470 (2015)
20. Yang, B., Tan, F., Dai, Y.-S., Guo, S.: Performance evaluation of cloud service considering fault recovery. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 571–576. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_54