International Journal of Grid and Distributed Computing
Vol. 5, No. 3, September, 2012

Alleviating Dynamic Resource Allocation for Bag of Tasks Applications in Cloud Computing

Masoud Salehpour and Asadollah Shahbahrami

Department of Computer Engineering, Faculty of Engineering
University of Guilan, Rasht, Iran

salehpour@msc.guilan.ac.ir, shahbahrami@guilan.ac.ir

Abstract

Cloud computing can provide facilities such as infrastructure and applications to clients on demand. Successful servicing of cloud paradigm necessitates accurate resource allocation. Whereas there are different workload types with different characteristics that should be supported by cloud computing, there is no any single solution that can allocate resources to all imaginable demands optimally. Therefore, there is a need to design specific solutions to allocate resources for each workload type. Based on that, in this paper, we focus on bag of tasks workload type and propose an idea to facilitate dynamic resource allocation for it. Technically, the proposed approach exploits users' service level agreement parameters and classifies them. It controls utilization of servers to response users in a reasonable time. To validate the proposed approach, we evaluate it using the Monte Carlo simulation. Then, the experimental results are compared with two reference models, namely First Fit and Proportional Share. The proposed approach outperforms the reference models in terms of the total cost of resource allocation and response time of clients.

Keywords: *Cloud Computing, Dynamic Resource Allocation, Bag of Tasks (BoT) Applications*

1. Introduction

Cloud computing provides resources range from computing infrastructure to applications and delivers them to clients as services on demand [5, 30, 25, 35]. Depending on type of clients' request, there are three delivery models, namely Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [30]. The SaaS provides applications which are hosted in cloud systems, and it presents an alternative to run applications locally such as on-line word processors [24]. The PaaS can present software platforms where applications run on them such as Google's application engine [10], and the IaaS is involved in hardware level management such as Amazon EC2 [2].

In cloud computing, different requests should be sent to datacenters to get servicing, and after that related results depart the center. Because of diversity of requests and time dependency of them, datacenters should meet user's expected Quality of Service (QoS). To guarantee the QoS, a contracted Service Level Agreement (SLA), including various descriptor parameters such as cost of operation, availability, and response time, describes different aspects of servicing for both clients and datacenters [37]. Therefore, cloud providers provisions resources for customers in regard to the SLAs.

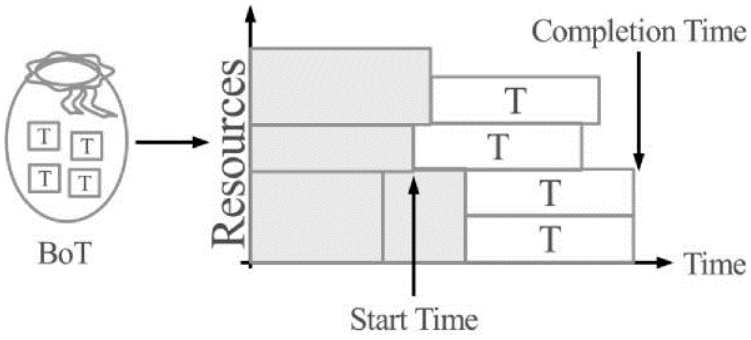

Figure 1. Tasks of a BoT Gravitates to be Completed at Different Times

However, resource allocation in cloud computing is still a challenging issue [11, 21]. Due to existence of different workload types with various requirements that should be supported by cloud computing, no any single hardware or software solution can allocate resources to all imaginable types efficiently [22, 20, 36]. Also, each type has its specific nature properties and a single solution cannot deal with in that regard optimally. Thus, it is a need to provide specific solutions for different workload types such as Bag of Tasks (BoT) and message passing applications, and provision resources in such a way that customers be able to just concentrate on demanded requests' results [4, 8, 7, 31].

Specifically, since an enormous fraction of jobs in the large distributed cloud systems is requested in the form of BoT, and the BoT applications are currently exploiting in several fields including data mining and image processing [29, 33, 32, 16], this paper focuses on allocating resources to the BoT applications. Technically, the BoT as a type of high performance computing application incorporates loosely coupled and compute intensive tasks demanding minimal intertask where the final results of all tasks prepare answer of a single problem [1, 32, 17, 26]. Moreover, tasks of a BoT in a single server are not postulated to be executed at the same time simultaneously, and tasks of the same BoT gravitate to be completed at different times (see Figure 1).

Although the BoT applications have received significant attention recently, the existing approaches to allocate resources to it are still in need of improvement specially in terms of the total response time [27, 32, 5]. The objective of this paper is proposing an idea to facilitate dynamic resource allocation for the BoT applications. Our approach uses the desired response time parameter of clients' SLA. The collected parameters are classified. It prevents tasks of a BoT from dispersion and distinguishes between tasks arrival, execution, and BoTs finalizations period. Technically, our approach pays much attention to a period of time during which all tasks of a BoT must be served and agreed response time of users' SLA. Briefly, main contributions of this paper are as follows:

- Proposing an idea to allocate resources to the BoT applications using a classification technique. The proposed technique omits idle time of servers and decreases waiting time of clients.
- Our approach monitors allocation process to avoid the BoTs long delay.
- Analyzing performance of different aspects of the proposed approach, and also simulating it based on the characteristics of the BoT applications and cloud providers.

- Comparing the obtained results with two reference models, namely First Fit and Proportional Share. The results show the benefit of our approach compared to the reference models in terms of the total response time.

This paper is organized as follows. Section 2 addresses and reviews some related works. Section 3 discusses some background information and presents our assumptions. Section 4 presents the proposed approach in detail. Performance analysis of our approach is presented in Section 5. To validate our approach, Section 6 is contained our evaluation process. Finally, Section 7 concludes this paper.

2.Related Work

Some previous works are addressed in the following. For example, Lee, et. al., [19] focused on allocating resources for a specific type of workload. Precisely, they worked on workflow model, but their strategy in regard with change of requests have not covered QoS parameters of users. Fu, et. al., [9] focused on a specific workload and the SLA for allocating resources, they worked on streaming in a distributed environment. They proposed an algorithm namely squeeze, to find resources for users based on their budget and expectations. However, they assumed in their contribution that resources are infinite in a distributed network where it is obviously unrealistic.

In [23], Liu, et. al., proposed a solution named Proportional Share (PS) for resource allocation. Originally, it distributed tasks between all servers where this caused to increase the total response time. Goudarzi, et. al., in [11] evaluated a modified version of the original PS to decrease the number of allocated servers that each customer is assigned to. They solved the problem of searching and selecting in the PS by using the assumption of having only one server with the processing capacity equal to the sum of all servers' processing capacity. In [38], Yarmolenko et al. provided various solutions based on the users' SLA, and they tried to decrease response time of servers by using parallel processing techniques, but their contributions have not focused on large scale infrastructure.

Juedes, et. al., [18] presented First Fit (FF) model. It contained a straightforward greedy algorithm. It placed the received tasks in the first server that can support the task's requirement. The FF could provide a fair load balancing alongside of resource allocation process. Ardagna, et. al., [3] extended previous work of Zhang, et. al., in [39] and considered different classes for servers and clients. They proposed a heuristic approach and used a discrete utility function with iterative attempts to find under utilization servers in a distributed environment to optimize resource allocation. However, the iterative attempts caused a time consuming process.

Song, et. al., [34] presented a software approach based on genetic algorithm. It defined some priorities and used a complicated software scheme to manage computational resources, but due to its complexity, it had a very long execution time. In addition, this paper's considerations (advantages) compared to the mentioned related work are as follows. The proposed approach manages resources of large scale datacenters with enormous number of users. It works on resources' utilization to omit idle time of physical servers of a datacenter. It deals with the clients' QoS requirements to prepare final results in an acceptable time for users.

3.Background and Preliminary

This section presents some information on bag of tasks applications, service level agreement, and the paper's assumptions.

3.1. Bag of Tasks

The BoT applications are composed of independent tasks without any communication with each other, where these tasks can be executed in any order [32]. Examples of the BoT applications include many topics and fields such as massive searches (such as key breaking), image manipulation, data mining algorithms, astronomy, bioinformatics, and physics [32]. For instance, to detect spams or to optimize advertising, Facebook exploits advantages of the BoT applications. Technically, a large percent of workloads perceived in real distributed systems such as workloads of LCG [12], Grid'5000 [13], DAS [6], and NorduGrid [28], is submitted in form of the BoT applications [16].

Whereas the BoT applications consist of independent and heterogeneous tasks, the final results produced by all tasks provide the solution of a single problem, where clients who were sent the tasks mostly are in need of some post process and post analysis based on the whole set of executed tasks' result. Therefore, allocating resources to the BoT needs much attention to a period of time during which all tasks of a BoT must be served. Also, the BoT does not necessarily require all tasks to be started at the same time, so, tasks of the same BoT tend to be completed at different times.

3.2. Service Level Agreement

An SLA should be contained the most important parameters such as contract length, bearable response time, cost of violation (penalty), and cost of operation. The length shows how long a provider is legally obligated to service. The response time determines a permitted period of time during which clients' request must be serviced. It is a time between sending tasks and receiving results. It should not be longer than predefined parameters of an SLA. Providers should pay penalty in case of degradation of the SLA's properties. Based on predefined cost of operation a provider charges a client. Shorter response time causes to more expensive service for customers. Therefore, clients who are not concerned about time commonly offer cheaper services.

3.3. Assumptions

This paper considers a datacenter which is contained a number of physical servers as our IaaS environment. In this datacenter, each server is identified by a distinctive id, represented by index *k*. As well as servers, each client should be recognized by a unique id, determined by index *n*. Each server is also modeled by its single core CPU capacity and memory capacity. Therefore, different tasks of a single client can be represented by Φ_{nj} where it shows the j^{th} task of the n^{th} client. Moreover, the datacenter has some master nodes to collect information of resources' utilization and available capacity.

4. The Proposed Approach

Allocating resources to the BoT applications and providing their responding process need a network based structure for working. This is why the amount of workload is unknown in advance. Therefore, a network based solution can cope with different amount of workloads and facilitate dynamic resource allocation. So, the proposed approach uses a network based approach by exploiting two main parts, namely classifier and Resource Allocator (RA). Users' requests as the preliminary actor receive by the classifier. It checks clients' requests. After that, the RA provides initializing process for the BoTs to run them based on available resources.

More specifically, to serve clients, first, they send their BoTs through networks and the classifier captures their requests. The classifier can be placed in a SaaS environment. Second, to explore each BoT and allocate resources to serve it, all requests of clients should be assigned to the RA where it can be positioned in a PaaS environment. Then, the RA based on the available facilities of the datacenter allocate required resources to run the BoTs of users. As mentioned in previous section, the datacenter has some master nodes to collect information of resources' capacity and utilization. These nodes provide a basis for the RA part's monitoring activity. Nonetheless, to highlight functionalities of the classifier and the RA, information of operations of each one is discussed in the following.

Since the SLA's properties can be different for dissimilar clients, the classifier's objective is to analyze the SLA's predefined parameters entirely. Hence, the classifier exploits clients' desired properties of their SLAs to classify them. Precisely, the classifier divides clients in terms of their contracted response time into three classes. Thresholds of classification for this parameter can be predicted based on the users' history. It means that when the classifier receives a new BoT, determines the sender's class and puts the BoT and sender's class information into service by delivering the request to the RA. Besides, clients can send their BoTs through different ways such as point to point or web based connections, so, the classifier provides a gate of entrance and deals with different connections.

Based on the received information about clients' classes from the classifier, the RA can take advantage of the received information to line up requests in different queues exploiting Generalized Processor Sharing (GPS) discipline. Hence, each class of clients belongs to a specific queue. The classes vary in size, so, the queues vary in size as well. Then, the RA can manage these queues like three independent lines. Obviously, clients of a queue are in the same class and their BoTs are not depended on the other classes' BoTs. Therefore, since there is no communication intertask between tasks of a BoT, it is possible for the RA to manage these queues like three independent lines. Lastly, the RA assigns resources to each task of a BoT based on its arrival time.

Technically, based on the information about class of a client, the RA can control the received traffic due for support different desired level of response time. It is proven that when the service time for user's requests is not too large to handle, the GPS can be implemented by Weighted Fair Queue (WFQ) which it can be estimated that processing times of tasks of a BoT can be finished in a reasonable period of time. Therefore, the RA particularly exploits the WFQ and it uses clients' classes as the WFQ's priorities. Consequently, the GPS by using the WFQ can replace different clients' BoT by three single class queues.

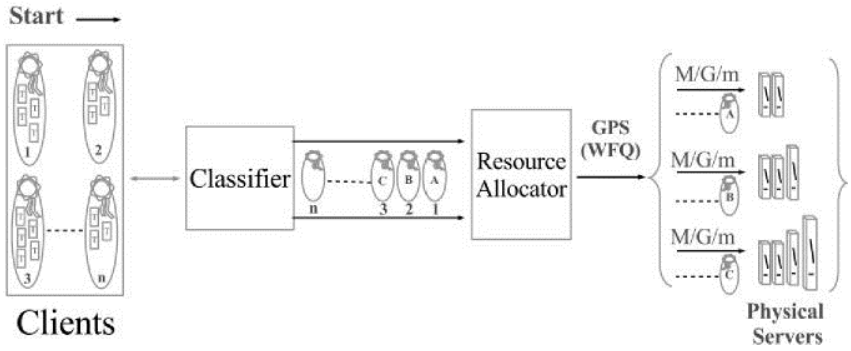

Figure 2. Functionalities of different parts of the proposed approach in detail. As it can be seen, the process starts by user's requests.

Ultimately, The RA considers each of the three single queues as an M/G/m queue where the arrival BoTs must wait to be serviced after allocating resources. Technically, an M/G/m queue contains the BoT of clients from the same class. The included tasks' arrival times have an exponential distribution while service times follow a general distribution, and our datacenter contains *m* servers which render service in order of task request arrivals (FCFS). The RA allots resources to a BoT one at a time from the front of the queue. When the allocation process is completed the BoT leaves the queue and the number of clients in the system reduces by one. Technical details of different parts of the proposed approach are shown in Figure 2.

5. Performance Analysis

This section presents an analytical model to provide a basis for discussing about performance of the proposed approach.

5.1. Explanation

Since supposedly arrival of the BoTs in our M/G/m queues is followed Poisson distribution and the service time for different requests is independent and exponential, so, because of the memoryless property of the exponential distribution we are able to use the Markov process [14] and its embedded Markov chain technique [15] to analyze performance issues of our approach where these characteristics are proven in the following. To increase paper readability, Table 1 presents key symbols used throughout this section along with their definitions.

Table 1. Notation and Definitions

Symbol	Definition
S_n	The n^{th} BoT's service time
C_n	The exact number of BoTs after the n^{th} BoT's departure
X_n	The number of BoTs arriving during S_n
k_j	Probability of arriving j customers during S_n
λ	Mean of the Poisson distribution
$B(t)$	The CDF of BoTs' service time when $(t < S_n \leq t + dt)$
$S = \{0, 1, 2, ...\}$	A state space to show the number of BoTs of a queue
P_{ij}	The one step transition of BoTs after departure time $(i \rightarrow j)$
P	Transition matrix for the Markov chain
$K(z)$	The number of BoTs arriving during a service time
$\psi(.), \vartheta(.)$	These are uses to show LST
π_i	Limiting vector for different transitions
$\varrho(z)$	Limiting distribution to calculate performance metrics
$E(.)$	To show expected value (mean)
$V(.)$	To show variance
mnq	The mean number of BoTs in the queues
T	The total time during which a BoT waits
$F(.)$	A function which can distribute T
R	Clients' mean response time
W	Clients' mean waiting time
Se	Mean service time of the proposed approach

Assume the n^{th} BoTs independent and identically distributed (i.i.d.) service time is equal to S_n, and C_n is the exact number of BoTs in the line after the n^{th} BoTs departure. In the following, it can be shown that $\{C_n , n = 1, 2, 3, ...\}$ is a Markov chain. Assume X_n is the number of BoTs arriving during S_n , then we have:

$$k_j = P(X_n = j) = \int_0^\infty \frac{(\lambda t)^j e^{-\lambda t}}{j!} dB(t) \qquad j = 0, 1, 2, ...$$

Where (λt) is the BoTs' arrival parameter and B(t) is the Cumulative Distribution Function (CDF) of the BoTs service time when $(t < S_n \leq t + dt)$. Here, to mix discrete and continuous distributions, Stieltjes notation is exploited. Besides, when we analyze the number of BoTs in departure times, it is realized that there is a relationship between C_n and C_{n+1}:

$$C_{n+1} = \begin{cases} C_n + X_{n+1} - 1 & C_n > 0 \\ X_{n+1} & C_n = 0 \end{cases}$$

Here, C_{n+1} can be represented by C_n and X_{n+1}. Specifically, X_n is i.i.d. and it is unrelated to both any BoTs arrival before and C_n. Then, because of only one step dependence of the number of BoTs in departure times, $\{C_n, n = 0, 1, 2, ...\}$ is a Markov chain. Moreover, since the departure points of BoTs after resource allocation and servicing can be considered as Markov points, in which the state of the M/G/m queue is captured, it is possible to represent the number of BoTs of the queue with a state space like S where S : {0, 1, 2, ...}. Due to embedded nature of this parameter space, it can be addressed as an embedded Markov chain. Therefore, we have:

$$P_{ij}^n = P(C_n = j \mid C_0 = i), \quad i, j \in S$$

and

$$P_{ij}^{(1)} \equiv P_{ij}$$

Based on relationships on (2) and (1), we have:

$$P_{ij} = \begin{cases} k_{j-i+1} & i > 0 \\ k_j & i = 0 \end{cases}$$

Then, based on (4), we can form matrix P:

$$P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ \vdots \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & \ldots \end{array} \\ \begin{pmatrix} k_0 & k_1 & k_2 & \ldots \\ k_0 & k_1 & k_2 & \ldots \\ & k_0 & k_1 & \ldots \\ & & k_0 & \ldots \\ & & & \vdots \end{pmatrix} \end{array}$$

Classification of the BoTs states must be followed based on (5), and it is depended on the expect number of BoT arriving during servicing time after resource allocation. Where we can show the number of BoTs arriving during a service time with:

$$K(z) = \sum_{j=0}^{\infty} k_j z^j \quad |z| \leq 1$$

Then, getting from (1), we have:

$$K(z) = \int_0^{\infty} \sum_{j=0}^{\infty} \frac{(\lambda t z)^j e^{-\lambda t}}{j!} dB(t)$$

With (7) at hand, we can use Laplace Stieltjes Transform (LST) to reach:

$$K(z) = \Psi(\lambda - \lambda z)$$

Based on (8), we get:

$$K'(z) = -\lambda \psi'(\lambda - \lambda z)$$

It can be calculated that:

$$K'(1) = \lambda b$$

The (9) equation determines an important state for our approach. Performance of the proposed approach can be analyzed only when $K'(1) < 1$, because of the Markov chain's steady state in this equilibrium. Altogether, based on the presented matrix in (5), as *n* in P_n increases the values of each column tend toward an identical value. Therefore, this characteristic can be exploited to calculate a limiting distribution to analyze different aspects of the proposed approach.

5.2. Limiting Distribution

We should calculate a limiting distribution to overcome complexity of necessary computation and provide a basis to analyze performance of our approach. To this end, it is possible to represent the number of BoTs of an M/G/m queue in our approach with a state space S : {0, 1, 2, ...}. Assume (π_0, π_1, π_2, ...) is a limiting vector where:

$$\pi_i = \lim_{n \to \infty} P_{ji}^{(n)}, \quad i, j \in S \tag{10}$$

Also, assume ϱ is a matrix with identical rows like $\pi = (\pi_0, \pi_1, \pi_2, ...)$, so, we can use Chapman Kolmorov relations to write:

$$\varrho = \varrho P \quad or \quad P^{(n)} = P^{n-1} P \tag{11}$$

Now we can apply (10) to (11) and write:

$$\pi = \pi P \tag{12}$$

To provide a limiting distribution for further analysis we should solve this equation:

$$\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij} \quad j = 0, 1, 2, ... \tag{13}$$

and

$$\sum_{j=0}^{\infty} \pi_j = 1$$

To solve it we get P from (5) and assume it = (it_0, it_1, it_2, ...), then:

$$\begin{array}{l} k_0\pi_0 + k_0\pi_1 = \pi_0 \\ k_1\pi_0 + k_1\pi_1 + k_0\pi_2 = \pi_1 \\ \quad\vdots \qquad\qquad\quad \vdots \end{array} \tag{14}$$

Now with solving (14), we can get our limiting distribution. To this end, we use (6) and also define:

$$\varrho(z) = \sum_{j=0}^{\infty} \pi_j z^j \qquad |\, z \,| \leq 1 \tag{15}$$

After multiplication of (14), we reach to:

$$\varrho(z) = \pi_0 K(z) + \frac{K(z)}{z}\left[\varrho(z) - \pi_0\right] \tag{16}$$

By using (8) and (9), we finally reach to a limiting distribution:

$$\varrho(z) = \frac{(1 - K'(1))(z-1)K(z)}{z - K(z)} \tag{17}$$

Lastly, we can determine the performance related aspects by expanding (17) in the following.

5.3. Mean Number of BoTs in the Queues

The limiting distribution that we calculated in (17) can be used by BoTs queues in an arrival epoch, departure times, and an arbitrary time point. Precisely, this is feasible only due to property of BoTs arrival distribution which it is called Poisson Arrivals See Time Averages (PASTA) property. It can be described by the fact that BoTs' arrival is completely unknown in advance. Let L(t) be the number of BoTs in an M/G/m queue at time *t* and *L* be the mean number of BoTs in the queues. First, the mean and variance of C_n when $\lim_{n\to\infty} C_n$ can be calculated from (17). Assume:

$$C = \lim_{n\to\infty} C_n$$

Then, these values are equal to:

$$E(C) = \varrho'(1)$$

$$V(C) = \varrho''(1) + \varrho'(1) - [\varrho'(1)]^2$$

And differentiating of $\varrho(z)$ with regard to z by using L'Hopital's rules for several times $(z \to 1)$ and considering (9) is:

$$E(C) = K'(1) + \frac{\lambda^2 E(S^2)}{2(1 - K'(1))} \tag{18}$$

Where $K'(1)$ is the expected number of BoTs after resource allocation and in servicing time, ultimately, the mean number of BoTs in the queues for different requests based on (18) can be analyzed by having:

$$mnq = \frac{\lambda^2 E(S^2)}{2(1 - K'(1))} \tag{19}$$

5.4. Clients' Mean Response Time

Response time refers to the time that a BoT spent in the system. This time starts from receiving a request till returning results. Now assume T is the total time during which a BoT waits in an M/G/m queue and after resource allocation and servicing departs the

system. Also, assume $R = E(T)$ and there is a function like F which can distribute T, now by using LST, we have:

$$P(C = n) = \int_0^\infty e^{-\lambda t} \frac{(\lambda t)^n}{n!} dF(t) \qquad n = 0, 1, 2, \ldots \tag{20}$$

Now like the calculation we did in (7) and the equation in (8), we can use (20) to calculate that:

$$\varrho(z) = \int_0^\infty e^{-\lambda t} \sum_{n=0}^{\infty} \frac{(\lambda t z)^n}{n!} dF(t)$$

Then using LST:

$$\varrho(z) = \vartheta(\lambda - \lambda z) \tag{21}$$

With having (17) and (21) at hand, we can write:

$$\vartheta(\lambda - \lambda z) = \frac{(1 - K'(1))(z - 1)K(z)}{z - K(z)} \tag{22}$$

Finally, by using (8) and employing α instead of $\lambda - \lambda z$, we have:

$$\vartheta(\alpha) = (1 - K'(1)) \frac{\alpha \psi(\alpha)}{\alpha - \lambda(1 - \psi(\alpha))} \tag{23}$$

Altogether, clients' mean response time in the proposed approach can be analyzed based on (23) and is equal to:

$$R = \vartheta'(0) \tag{24}$$

5.5. Mean Waiting and Service Time of Clients

We can exploit Little's lemma to analyze mean waiting and service time of the proposed approach. Based on this lemma, result of multiplication of BoTs arrival parameter by R is equal to mnq, which (24) and (19) represent these equations respectively, and this is:

$$mnq = \lambda R$$

Now based on the mentioned equations, we can analyze mean waiting time (W) of the proposed approach by using:

$$W = \frac{\lambda E(S^2)}{2(1 - K'(1))} \tag{25}$$

Since the response time of a BoT can be calculate by adding waiting and service time of that, so, to analyze mean service time (Se) of the proposed approach to serving clients, we can use:

$$Se = R - W \tag{26}$$

6. Evaluation

To evaluate our approach and validate the analytical model presented above, we have simulated the proposed approach. Information on metrics, methodology, and preliminary results of the simulation process is discussed in the following.

6.1. Assumptions and Metrics

As mentioned, whereas the number of client's requests is unknown in advance, this paper assumes that users' requests follow a Poisson distribution with mean of X where it is a positive real number equal to the expected number of occurrences during a given interval. In this paper, receiving a client's BoT by the RA is considered as an event

occurrence. Thus, the exact arrival time for the j^{th} task of a particular BoT as an independent portion of the n^{th} client's request during X time can be represented by result of Φ_{nj} X.

As we analyzed in previous section, the main metric that must be measured is users' response time. This time calculates from the point in which the classifier receives a BoT till the point that results referred to the client. Besides, this paper considers metrics such as the number of BoTs in the system and mean service time of users' BoTs to present. Moreover, based on the response time and the service time, it is possible to calculate clients' waiting time to be served after resource allocation, because response time is broken down into service time and waiting time.

6.2. Methodology

There is no existing complete framework for evaluating resource allocation strategies, so this paper exploits the Monte Carlo method to simulate all components of the system, from clients' BoTs to servers. Based on the Monte Carlo method, our evaluation process defines a domain of possible inputs, then generate inputs randomly from a probability distribution over the domain and finally, it performs a deterministic computation on the received inputs to aggregate the results. To this end, the number of servers in our IaaS environment, the number of clients, and the number of the SLA classes were set to 15, 3000, and 3, respectively.

More specifically, 17% of clients were belonged to class A of the SLA. Class B and C also were contained 33% and 50% of clients, respectively. The total CPU capacity of servers was equal to 14000.0 MIPS where the total RAM capacity was equal to 32768 MB for servers. The value of X was set with a random variable between 0.1 and 0.9. The number of tasks of each BoT was also set with a random variable between 30 and 150. The required CPU and RAM of each task in a BoT were set with random variables. Dispersion of BoTs CPU requirement (CV) was assigned values of 0.12 and 0.35.

In addition, to compare the obtained results of the proposed approach with known resource allocation methods, two reference models were used. These were namely the modified PS and the FF. Details of the approaches were discussed earlier in Section 2. We selected these solutions because these received significant attention of researchers presently and these are using in distributed environments.

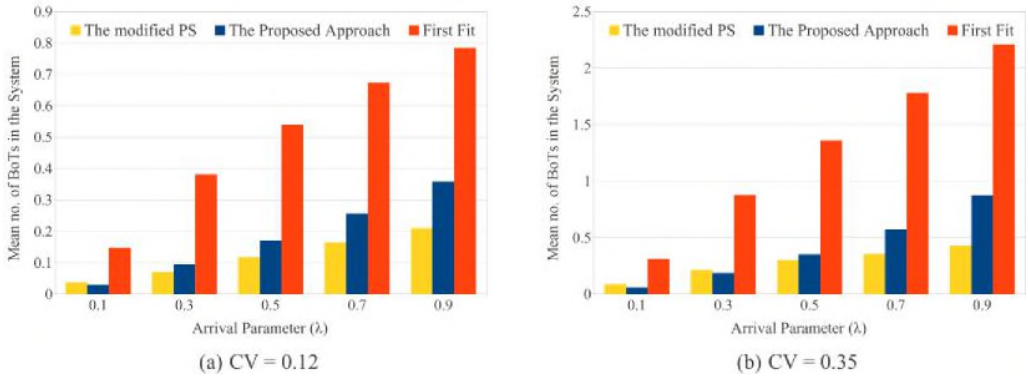

Figure 3. Mean number of BoTs in the Systems

6.3. Results

Based on the discussions from previous sections, in order to measure the mentioned metrics and validate our results, the experiments were executed and repeated hundreds of times. Mean number of BoTs in the system is shown in Figure 3. As we analyzed in Section 2, BoTs arrival is completely unknown in advance, and it increase in accordance with clients' requests' arrival rate. In fact, this metric can show effect of servers' utilization on the number of tasks in the system at the time of a new task arrival. As it can be seen, the effectiveness of our approach in terms of this metric puts it in the second place between the two reference models. Two different values of CV for each metric give technical insight into the behavior of approaches in accordance with different dispersion of BoTs tasks.

Mean response time of users is shown in Figure 4. For example, when X = 0.1 and CV = 0.12, mean response time for our approach is equal to 0.2954 seconds while this value is equal to 1.4695 and 0.3672 for the FF and the modified PS, respectively. Due to omitting idle time of servers in our approach, when the X increases our approach turns from the first place into the second. Finally, mean service time of BoTs is shown in Figure 5. Whereas the response time of users was broken down into service time and waiting time, this figure completes the presented information on mean response time of servers. Compare to the reference models, Figure 5 shows at any arbitrary time our approach can be in a steady state.

In addition, idle time of servers for the FF and the modified PS ranges from 0.061 to 1.237 and from 0.844 to 8.833 respectively. However, the proposed approach could omit idle time of servers. This is why the RA continuously checks servers and exploits a deallocation process to switch unnecessary servers off. Hence, the two reference models are not comparable to our approach in terms of the total idle time of servers. Besides, since the total cost of resource allocation must be calculated based on both the solutions' service time and idle time, our approach can provide a cheaper resource allocation compared to the reference models.

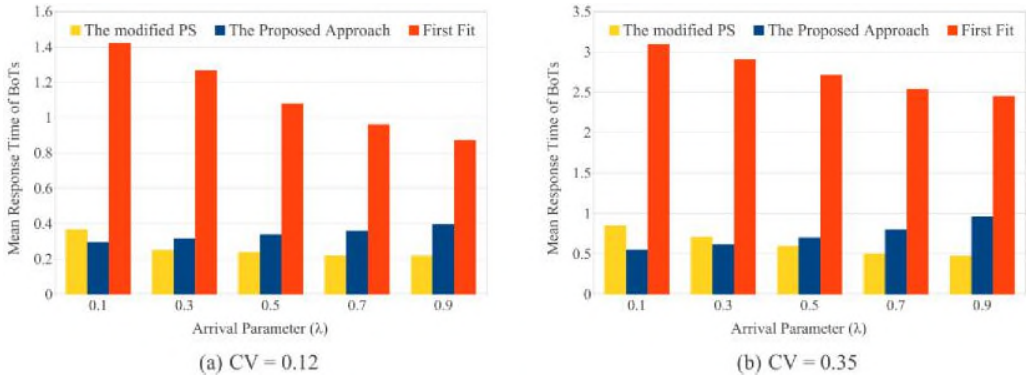

Figure 4. Mean response time of clients' BoTs in different approaches

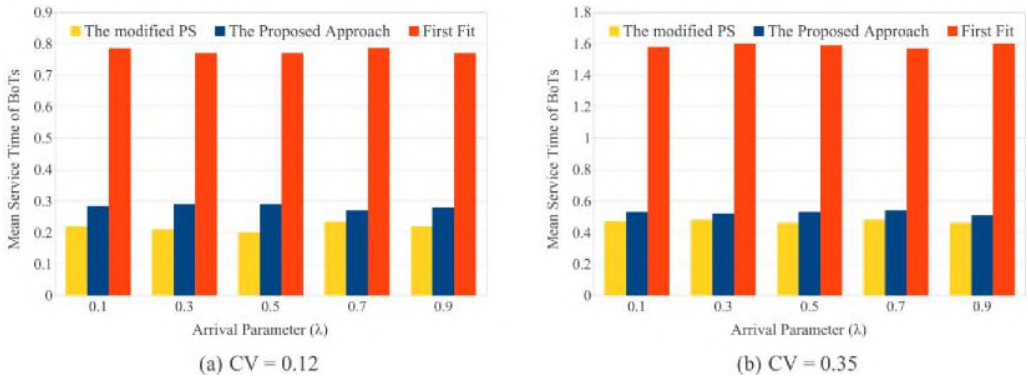

Figure 5. Mean Service Time of BoTs in Different Approaches

7. Conclusions

There are different workload types with different characteristics that should be supported by cloud computing, but there is no any single solution can allocate resources to all imaginable demands optimally. Consequently, it is necessary to design specific solutions to allocate resources for each workload type. To this end, this paper has focused on Bag of Tasks applications. It has proposed an idea to facilitate dynamic resource allocation this workload type. Our approach has monitored server's traffic intensity to response users based on an appropriate resource selection and allocation in a reasonable time.

Technically, we have exploited users' bearable response time of their service level agreement and a classification technique based on that. Also, we have analyzed performance of our approach and evaluated it using the Monte Carlo simulation. The obtained results have showed that our approach can omit idle of servers. To validate our results, we have compared it with two reference models, First Fit and Proportional Share. The proposed approach could outperform the reference models in terms of the total cost of resource allocation and response time of clients.

References

[1] D. Abramson, B. Bethwaite, C. Enticott, S. Garic and T. Peachey, "Parameter exploration in science and engineering using many-task computing", IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 6, **(2011)**, pp. 960–973.
[2] Amazon-EC2, https://awa.amazon.com/ec2, **(2012)**.
[3] D. Ardagna, M. Trubian and L. Zhang, "Sla based resource allocation policies in autonomic environments", Journal of Parallel and Distributed Computing, vol. 67, no. 3, **(2007)**, pp. 259–270.
[4] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A view of cloud computing", Communications of the ACM, vol. 53, no. 4, **(2010)**, pp. 50–58.
[5] S. Chaisiri, B. Lee and D. Niyato, "Optimization of resource provisioning cost in cloud computing", IEEE Trans. on Services Computing, vol. 4, no. 1, **(2011)**.
[6] DAS-4, http://www.cs.vu.nl/das4/, **(2012)**.
[7] J. Ekanayake, T. Gunarathne and J. Qiu, "Cloud technologies for bioinformatics applications", IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 6, **(2011)**, pp. 998–1011.
[8] P. T. Endo, A. V. de Almeida Palhares, N. N. Pereira, G. E. Goncalves, D. Sadok, J. Kelner, B. Melander and J. E. Mangs, "Resource allocation for distributed cloud: Concepts and research challenges", IEEE Network, vol. 25, no. 4, **(2011)**, pp. 42–46.

[9] Y. Fu and A. Vahdat, "Sla based distributed resource allocation for streaming hosting systems", In: http://issg.cs.duke.edu, **(2002)**.
[10] Google-App-Engine, https://appengine.google.com, **(2012)**.
[11]H. Goudarzi and M. Pedram, "Maximizing profit in cloud computing system via resource allocation", In: Proc. of Int. Conf. on Distributed Computing System, **(2011)**, pp. 1–6.
[12] W. L. C. Grid, http://lcg.web.cern.ch/lcg/, **(2012)**.
[13] Grid5000, http://www.Grid5000.fr, **(2012)**.
[14]G. Grimmett and D. Stirzaker, "Probability and Random Processes", Oxford University, third edition, **(2010)**.
[15] D. P. Heyman and M. J. Sobel, "Stochastic Models in Operations Research", vol. 1, Dover, **(2004)**.
[16] A. Iosup, O. Sonmez, S. Anoep and D. Epema, "The performance of bags-of-tasks in large-scale distributed systems", In: Proc. of Int. Conf. on High Performance Distributed Computing, **(2008)**, pp. 97–108.
[17] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer and D. H. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing", IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 6, **(2011)**, pp. 931–945.
[18] D. Juedes, F. Drews, L. Welch and D. Fleeman, "Heuristic resource allocation algorithms for maximizing allowable workload in dynamic, distributed real-time systems", In: Proc. of the IEEE Int. Symp. on Parallel and Distributed Processing, **(2004)**, pp. 117–130.
[19] Y. Lee, C. Wang, A. Zomaya and B. Zhou, "Profit-driven service request scheduling in clouds", In: Proc. of the IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing, **(2010)**.
[20] F. Leymann, C. Fehling, R. Mietzner, A. Nowak and S. Dustar, "Moving applications to the cloud: An approach based on application model enrichment", Int. Journal of Cooperative Information Systems, vol. 20, no. 3, **(2011)**, pp. 307–356.
[21] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin and Z. Gu, "Online optimization for scheduling preemptable tasks on iaas cloud systems", Journal of Parallel and Distributed Computing, **(2012)**.
[22] X. Liu, C. Qiao, D. Yu and T. Jiang, "Application-specific resource provisioning for wide-area distributed computing", IEEE Network, vol. 24, no. 4, **(2010)**, pp. 25–34.
[23] Z. Liu, M. Squillante and J. Wolf, "On maximizing service-level-agreement profits", In: Proc. of the ACM Int. Conf. on Electronic Commerce, **(2006)**.
[24] I. Llorente, R. Moreno-Vozmediano and R. Montero, "Cloud computing for on-demand grid resource provisioning", Advances in Parallel Computing, vol. 18, **(2009)**, pp. 177–191.
[25] R. Montero, R. Moreno-Vozmediano and I. M. Llorente, "An elasticity model for high throughput computing clusters", Journal of Parallel and Distributed Computing, vol. 71, no. 6, **(2012)**, pp. 750–757.
[26] R. Moreno-Vozmediano, R. S. Montero and I. M. Llorente, "Multicloud deployment of computing clusters for loosely coupled mtc applications", IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 6, **(2011)**, pp. 924–930.
[27] M. Netto and R. Buyya, "Offer-based scheduling of deadline constrained bag-of-tasks applications for utility computing systems", In: Proc. of the IEEE Int. Symp. on Parallel and Distributed Processing, **(2009)**, pp. 1–11.
[28] NORDUGRID, http://www.nordugrid.org/, **(2012)**.
[29] V. Pande, I. Baker, J. Chapman, S. Elmer, S. Larson, Y. Rhee, M. S. C. Snow, E. Sorin and B. Zagrovic, "Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing", Biopolymers, vol. 68, no. 1, **(2003)**, pp. 91–109.
[30] B. Rajkumar, C. Yeo, S. Venugopal and S. Malpani, "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, **(2009)**, pp. 599–616.
[31] M. Salehpour and A. Shahbahrami, "Alienable services for cloud computing", In: Proc. of Int. Symp. On Intelligent Distributed Computing, **(2011)**, pp. 195–200.
[32] F. Silva and H. Senger, "Scalability limits of bag-of-tasks applications running on hierarchical platforms", Journal of Parallel and Distributed Computing, vol. 71, no. 6, **(2012)**, pp. 788–801.
[33] S. Smallen, H. Casanova and F. Berman, "Applying scheduling and tuning to on-line parallel tomography", In: Proc. of the ACM/IEEE Int. Conf. on Supercomputing, **(2001)**, pp. 46–56.
[34] Y. Song, Y. Li, H. Wang, Y. Zhang, B. Feng, H. Zang and Y. Sun, "A service-oriented priority-based resource scheduling scheme for virtualized utility computing", In: Proc. of Int. Conf. on High Performance Computing, **(2008)**, pp. 220–231.
[35] L. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A break in the clouds: Towards a cloud definition", ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, **(2008)**, pp. 50–55.

[36] D. Warneke and O. Kao, "Exploiting dynamic resource allocation for efficient parallel data processing in the cloud", IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 6, **(2011)**, pp. 985–997.
[37] L. Wu, S. K. Garg and R. Buyya, "Sla-based resource allocation for software as a service provider (saas) in cloud computing environments", In: Proc. of the IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Computing, **(2011)**, pp. 195–204.
[38] V. Yarmolenko and R. Sakellariou, "An evaluation of heuristics for sla based parallel job scheduling", In: Proc. of Int. Symp. on Parallel and Distributed Processing, **(2006)**.
[39]L. Zhang and D. Ardagna, "Sla based profit optimization in autonomic computing systems", In: Proc. of Int. Conf. on Service Oriented Computing, **(2004)**.

Authors

Masoud Salehpour received the BSc and MSc degrees in computer science from the University of Guilan in 2009 and 2012, respectively. His research interests are in the areas of resource management, scheduling, design and performance analysis of large scale distributed systems (in particular grids and clouds), and workload characterization. He may be reached at pouyasalehpour@gmail.com.

Asadollah Shahbahrami is an assistant professor in Department of Computer Engineering at the University of Guilan, Rasht, Iran. He got the PhD degree from the Delft University of Technology, the Netherlands in 2008. His research interests include advanced computer architecture, reconfigurable computing, SIMD programming, and distributed systems performance analysis. He may be reached at asadshahbahrami@gmail.com.