ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Cost-based Competitive Analysis for Cloud Platform Service Using Online Dynamic Bin Packing

[I]Seunghwan Yoo, [II]Sungchun Kim
[I, II]Computer Science & Engineering, Dept. of Sogang University, Seoul, South Korea

Abstract
Recently, the issue of scalability and cost due to the rapid increase of user requests in the cloud platform environment has been highlighted. In this paper, we propose a provisioning method that distributes and allocates user request to system in real time in order to maximize resource utilization and minimize cost. The concept of FaaS (Function as a Service), which is a concept of cloud platform service, is attracting new attention, and serverless micro service for this is attracting attention. We consider cost-efficient resource allocation algorithm for serverless micro services for FaaS, which is one of cloud platform service concept, considering characteristics of users request task is very irregular and unpredictable in cloud platform environment. To do this, we propose a method that focuses on reducing system operation cost by applying on - line dynamic bin packing, which is a variant of well - known bin packing method among existing job scheduling algorithms. And we analyze the lower and upper bounds of the competitive ratio for the proposed technique through the analysis of the approximation algorithm, and finally try to evaluate the competitive ratio of the proposed scheme.

Keywords
Cloud platform Service, Provisioning, On-line dynamic bin packing, Competitive analysis, Function as a Service

I. Introduction

Recently, FaaS (Function as a Service), a new type of cloud platform service, has been introduced to solve the problem of increase in maintenance cost of cloud platform, which is becoming an increasingly important research issue, which has been caused by the existing cloud platform service. For example, popular service providers (eg, Amazon, Google, and MS) now offer FaaS, named Serverless Micro-Service, to users for more flexible scalability and cost savings. The service provider, which currently offers a variety of public cloud platform services, is providing users with services based on FaaS starting in 2014. For example, Amazon supports Lambda, and Google and Microsoft provide Google Function and Azure Function.[1]

In other words, it's actually possible to pay only what users are using, which is an ideal goal when cloud computing is first proposed. In the past, we had set up a virtual machine environment somewhat more commonly than the actual required system resources. But with serverless micro-service, users can reduce the need for unnecessary resources and service providers can lower the cost of maintaining the cloud platform service.

In this paper, we propose a new approach to solve the NP-hard problem in order to deal with numerous micro-operations in the FaaS cloud platform service environment. We propose a task allocation scheme using online dynamic bin packing algorithm. In addition, existing studies that apply the modified packing technique focus on cost reduction through detailed cost-based modeling while processing micro-operation requests in real-time, compared to resource utilization.

In the cloud computing environment, cloud service providers(SP) and users enter into a kind of agreement called service level agreement (SLA). The goal of resource management is to dynamically allocate resources that satisfy the negotiated SLAs to tasks that the user requests. However, since the cloud provider's available resources are virtualized, they can be changed dynamically, and the user's needs to use the resources are very diverse. Therefore, there is a need for a task allocation algorithm that enables automatic and intelligent resource allocation based on the SLA determined through negotiation between the service provider and the user.

So there are various practical difficulties in predicting and preparing users' demands in a cloud environment in which a service corresponding to the user needs to be provided in real time. Providing resources according to the needs of these users is called as automatic provisioning.

The proposed scheme can be classified into 'Resource Management Aware Provisioning' based on 'Autonomic Architectures'. In addition, load balancing can be achieved through virtual resource migration in a cloud computing environment. However, a migration method that does not consider the cost of migration or searching for a resource suitable for a service type may reduce the availability, In this case, delays or errors in cloud platform service processing may occur, so care must be taken.[2]

As such, Automatic provisioning, which can save costs and improve the user's convenience than existing computing environments. In particular, this paper is a study on how to efficiently process requests of users with high fluctuation characteristics through cloud platform service. Existing researches related to this have focused mainly on how to efficiently provide various applications on the cloud platform in general terms. On the other hand, we propose provisioning based on an online bin packing algorithm to allocate a large number of user requests with a goal to obtain minimum system maintenance cost with a minimum system processing time in real time

In other word, we propose a method to reduce the number of empty bottlenecks for packing. And we analyze our online dynamic bin packing algorithm to solve the problem of having a minimal processing cost by analyzing cost in consideration of a relatively small size user request operation in a more resource - oriented manner.

In order to do so, we use the system modeling and the competitive analysis to find the upper bound of the competitive ratio, which is the ratio of the cost of the proposed algorithm to the cost of the proposed algorithm and a lower bound. Through the system modeling, we conduct the experiment with the theoretical value which is competitive ratio.

II. Related Work

A. Online bin packing

The classical packing problem is a matter of putting several items in a bin under certain constraints. One of the very classic optimization problems is that there are a lot of variations depending on the specific application. In the most basic packing problem, each item is represented by a single scalar value, and all bins have a fixed capacity. The sum of the items accommodated in each bin can't exceed the capacity of the bin, and the goal is to determine the minimum number of bins needed to accommodate all the items. In the field of computer system, it is used as an algorithm to solve problems such as parallel task scheduling and resource assignment problem.

In addition, the packing problem is known to be an NP-hard problem. Only heuristic methods are the way to get accurate answers to these problems. In theory, it is possible to know the number of correct bins by a single execution of finite size and number of items and bins. However, it is a theoretically possible, and practically impossible, method that requires a great deal of time. For this reason, research related to the packing problem has been directed toward finding an approximation algorithm that can find a value close to the optimal solution in a reasonable amount of time.

A typical offline bin packing scheme is (First Fit)FF, (Next Fit) NF, (Best Fit)BF, and (Worst Fit)WF. it refers to a technique of packing all the items to be packed into a minimum number of bins in a finite time while being sequentially inputted in a given state. On the other hand, the online version of the bin packing problem reflects more realistic scenarios rather than being given one at a time.

Since the input is determined very unpredictably and arbitrarily, not the fixed input, it is necessary to determine the placement of the item at each point in time. Because of this nature, decisions that have already been processed can't be undone, which affects the overall cost of the overall algorithm. Also, the dynamic version of the online version, which will be mainly discussed in this paper, additionally considers the departure of items at any time. This is a more realistic model and is suitable for solving dynamic allocation problems of computer resources (memory, storage).

FF, NF, BF and WF, which are offline bin packing algorithms, are used as the existing algorithms for solving the online bin packing problem. However, since the environment for inputting items is different, FFD (First Fir Decreasing) applied to FF can't be used among offline blank packing methods. This is because the FFD is required to sort the input items in the reverse order of size, because in the online bin packing environment, information on all the items inputted can't be known.

In addition, a number of studies have been conducted on online bin packing algorithms, and a study published in 2012 assumes that any on-line bin packing algorithm has a general precondition, the competitive ratio can't be less than 1.54037.[3]

To sum up, online dynamic bin packing model exists assuming that a bin of unit size is provided infinitely, an item arrives at an arbitrary time, is allocated to a bin for a certain period of time, and can start at an arbitrary time. The item size and arrival time can be known only when the item arrives. Also, the sum of the sizes of the allocated items of the bin should not always exceed the capacity of the bin. The goal of solving the online dynamic bin packing problem is to minimize the maximum number of bins used all the time while satisfying all of the above conditions. As with the online static bin packing model, item movement between bins is not allowed, and the difference is that item rearrangement within the bin is allowed.

B. Provisioning

In a cloud environment, service providers need to manage virtual machines (VMs) to maximize their benefits and provide quality of service (QoS) to their users. In this paper, we propose an optimal resource allocation method considering various criteria such as SLA, resource utilization, and types of user work requests to be processed.

First, based on price, there is an advanced cost-based scheduling [4] to efficiently map work on available resources. This research measures the resource cost and computational performance of the system and improves computation / communication ratios and assigns them to the system through grouping of users according to the processing capability of a specific cloud resource. In addition, the framework FlexPrice [5] provides users with a variety of pricing in terms of price and execution time, giving users the flexibility to meet their needs by scheduling work according to a user-selected quote. These studies have been conducted only in terms of economic transactions and benefits between users and service providers, so further research on the maintenance costs of actual cloud platform services is needed.

The existing research on resource management has an adaptive resource allocation scheme considering Burst-Workload in order to maintain performance balance satisfying the SLA proposed by Tai, J. Zhang, J. Li, J. Meleis, and W. Mi. [6] The proposed scheme uses two states of Markovian Modulated Poisson Process(MMPP) and allocates the task to the system according to the load without burstiness, strong burst and normal, level.

There is also research on a combination auction-based mechanism [7] for consumption, operating costs and carbon savings. This technique is an algorithm for winner (user) determination and payment calculation based on a linear relaxation based randomized algorithm and a greedy algorithm. Likewise, the above studies basically considered only the total number of operating cloud data centers and resource utilization, so more detailed system maintenance costs need to research

III. Proposed Scheme

A. Minimun Usage Dynamic Packing

As the cloud platform environment becomes more popular, the need to expedite applications to meet exponentially growing user demands will become increasingly important. It is also necessary to reduce the costs associated with service-related tasks other than immediate service-related work among service provider (SP) service users, so a more detailed analysis of costs is necessary.

As mentioned, we propose a new method to handle user task requests of a certain size or less in the cloud platform system in terms of cost, unlike the conventional method that focuses on finding the minimum number of bins based on the concept of online bin packing. (Modified First Fit) algorithm, MUDP (Minimum Usage Dynamic Packing (MUDP)), which is a modified First Fit (MFF) algorithm

Also. Approximation algorithm for the minimum usage cost obtained by the proposed method. Approximate solution to the optimal solution is obtained through the problem solving method, and the upper limit and the lower limit for the optimal solution are mathematically proved.

Table. 1 : MUDP Algorithm

Minimum Usage Dynamic Bin Packing Algorithm

```
1: for All itemArray[i] = 0.1, 0.2, .... , m do
2: for All binArray[j] = 1, 2, .... , n do
3:    if ( itemsize + bin_level <= 1 )
4:        Pack item i in bin j.
5:        Break the loop and pack the next item.
6: end if
7: end for
8: if ( itemsize+bin_level > 1 )
9:        Open new bin(n+1 index) and pack item i.
10: end if
11: for All binArray[j] = 1, 2, .... , n do
12:    if ( bin_level == 0 )
13:        Close j index bin
14:    end if
15: end for
16: end for
```

Generally, in the existing bin packing theory, it is assumed that all bins are open, that is, a certain number of bins are open without assuming a standby state to receive an item. Item insertion decision method is the same as existing First Fit. However, there are some modification.

•Modified First Fit : one or more open bins that can accommodate the current item, places the item in the lowest-indexed open bin that has room for it. and the bin used for item allocation is empty, all items deleted at the specific time, the bin is not idle to wait for the item to be allocated just close the bin as so saving the cost.

B. Preliminaries

According to the notations, each component has the following relationship. The system resource requirement (maintenance cost) to process the user request operation is defined as follows.

Table. 2 : Key Notations

Notation	Definition
r	Individual user request operations
R	Complete List of User Requests
$t_a(r)$	Time of end of user request
$t_d(r)$	Time of end of user request
$s(r)$	Size of user request operation
$I(r)$	The time at which the request operation is processed (Interval)
$dist(I(r))$	The length of time
$d(r)$	If a user request (Demand) / required resource
$d(R)$	Resources required to handle all user request operations
$AI(R)$	When the system is processing at least one user request operation (Active Interval)
P	Packing to process all user request operations.
$Cost_{total}(P_{A,R})$	The cost required to process all user request operations through the algorithm A
$OPT_{total}(R)$	Optimized cost (Worst-Case Ratio criterion) for handling all user request jobs through offline algorithm

$$d(r) = s(r) \cdot dist(I(r))$$

$$d(R) = \sum_{r \in R} d(r)$$

Also, satisfying the following condition is expressed by the time length of the total union of the time when the user requested job is processed.

$$t_a(r) < t_d(r)\ ,\ dist(I(r)) = t_d(r) - t_a(r)$$

$$AI(R) = dist(\cup_{r \in R} I(r))$$

it is aimed to minimize the total system maintenance cost by allocating the entire user request work to the system over time. Below is the formula for total system maintenance cost.

$$Cost_{total}(P) = \int_{\min_{r \in R} t_a(r)}^{\max_{r \in R} t_d(r)} P(t)dt$$

The performance of the online bin packing algorithm is generally measured by the competitive ratio, which is considered as an important criterion for evaluating the performance of the approximation algorithm. The approximate algorithm is an algorithm for solving NP-Complete or NP-Hard optimization problems during polynomial time. It is an offline-based algorithm that is often used to obtain an approximate solution in terms of problem cost and time complexity. The competitive ratio, which is the most important criterion in the approximation algorithm, is the ratio of the value of the approximate solution to the value of the optimal solution. This is expressed by the following equation.

$$A(R) \le c \cdot OPT(R) + b$$

Competitive Ratio was introduced in the concept of competitive analysis. Competitor analysis was introduced by Sleator and Tarjan [8] to compare and measure the performance of the worst case of online algorithm compared to the existing offline algorithm. That is, the offline algorithm, knowing the entire input in advance, can calculate the optimal solution for a given problem. The computational complexity to find the optimal solution can be very high and the optimal algorithm can't be explicitly known, but the minimum cost of the solution can be shown through a comparison of performance

If there is a constant such that the online algorithm is for arbitrary input for cost minimization problem, it is said that it is competitive with the optimal off-line algorithm. Where and represents the cost of the solution to the input and is a constant. In order to define the MUDP algorithm competition ratio, the optimal cost that can be processed by the offline algorithm considering Worst-Case is defined as follows at a specific time t for the entire list of user request work.

$$OPT_{total}(R) = \int_{\min_{r \in R} t_a(r)}^{\max_{r \in R} t_d(r)} OPT(R, t)\, dt$$

Based on the above, we define the competitive ratio of the MUDP algorithm as follows. The total cost(time) to process the entire user request operation through the algorithm is less than or equal to the optimal cost(time) to process all user requested operations multiplied by a constant value. Here it is a constant and the algorithm is said to be bounded.

$$Cost_{total}(P_{A,R}) \le \alpha \cdot OPT_{total}(R)$$

IV. Competitive Analysis of MUDP

A. Lower Bound

$$\mu = \frac{\max_{r \in R} dist(I(r))}{\min_{r \in R} dist(I(r))}$$

The above is the ratio of the length of the user requested job processing time that requires the longest processing time compared with the length of the user requested job processing time which requires the shortest processing time among all the user requests.

Theorem 1. Algorithm (proposed Algorithm in this paper Modified First Fit Packing criterion) has at least lower bound is $\mu + 1 (\mu > 1)$

To prove the above theorem 1, first Δ define the minimum processing time length of the user request operation. Also, according to the definition above, the maximum processing time length of the user request operation is $\mu\Delta$. It is also assumed that the following integer has the following condition.

$$\frac{1}{k} < (\mu - 1)$$

Proof. Fig 1, it is assumed that At time 0, let k^2 items of size 1/k arrive. it needs to open k bins to pack these items. Then, let one item depart from each open bin at time Δ. Next time period, let k items of size 1/k arrive. It is easy to see that each of the k new items will be assigned to a different bin. After that, let all the "old" items (i.e., the items arrived at time 0) leave the system at next period(third period). At last period, all the remaining items leave the system

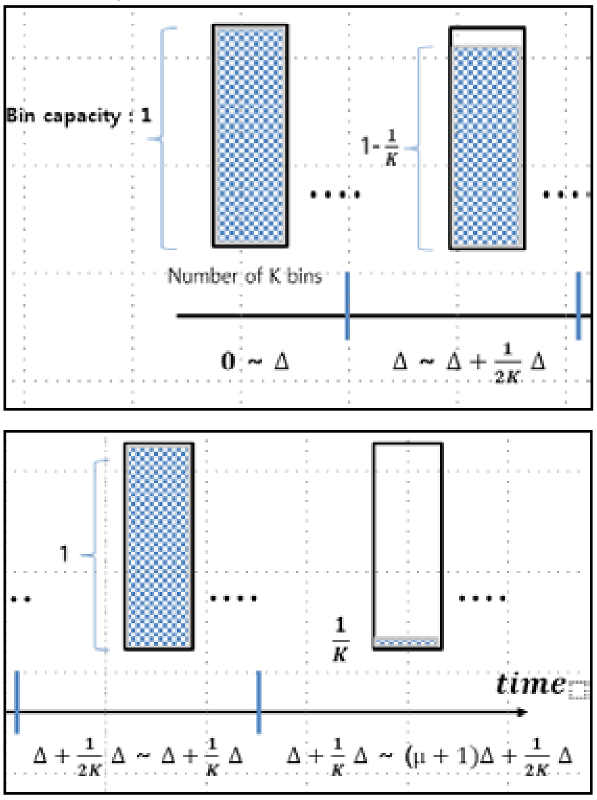

Fig 1 : levels by a Modified algorithm.

As can be seen in Fig 1, since the user task request comes in, it always knows that the k bins is used to process the user request task. Since the total cost of processing is the sum of the total used bin size multiply user request processing time, the total cost of the algorithm is:

$$Cost_{total}(P_{A,R}) = k(\mu + 1 + \frac{1}{2k})\Delta$$

We can see that the optimal cost of the algorithm can be allocated to one bin as shown in Fig 1. In addition, since it has a very small item size, it is summarized as follows.

$$OPT_{total}(R) \le k(\Delta + \frac{1}{k}\Delta) + (\mu\Delta + \frac{1}{2k}\Delta - \frac{1}{k}\Delta)$$

$$OPT_{total}(R) \le k(\Delta + \frac{1}{k}\Delta) + \mu\Delta - \frac{1}{2k}\Delta$$

The competition ratio α of the algorithm can be obtained by dividing the total cost as defined above by the optimized cost. The competitive ratio can be seen from the above formula as follows.

$$\frac{Cost_{total}(P_{A,R})}{OPT_{total}(R)} \ge \frac{k(\mu+1)\Delta + \frac{1}{2}\Delta}{k(\Delta + \frac{1}{k}\Delta) + (\mu\Delta - \frac{1}{2k}\Delta)}$$

$$\ge \frac{k(\mu+1) + \frac{1}{2}}{k(1 + \frac{1}{k}) + (\mu - \frac{1}{2k})}$$

The right side of the above formula is summarized as follows.

$$\frac{k(\mu+1) + \frac{1}{2}}{k + 1 + \mu - \frac{1}{2k}} = \frac{\mu + 1 + \frac{1}{2k}}{1 + \frac{1}{k} + \frac{\mu}{k} - \frac{1}{2k^2}}$$

$$\lim_{k \to \infty} \frac{\mu + 1 + \frac{1}{2k}}{1 + \frac{1}{k} + \frac{\mu}{k} - \frac{1}{2k^2}} = \mu + 1$$

It is shown that the proposed algorithm has a minimum value. That is, the lower bound value of the competition ratio of the MUDP algorithm is. $\mu + 1$

B. Upper Bound

First, it is assumed that only a small job having a size of a user work request is smaller than a certain size. The following is an additional definition of user request size and processing time for the analysis of the upper bound on the competitive ratio.$(k > 1)$

$$s(r) < \frac{1}{k}$$

I_i : The resources allocated to each m bins are assigned to the user task request R in the first-fit manner, and the processing time (Usage period of bin)

I_i^-, I_i^+ : The start and end of processing time when a user task request utilizes resources in each bin

$$dist(I_i) = I_i^+ - I_i^-$$

For the total cost of processing user work requests $Cost_{total}(P_{FF',R})$, define the following for user request work processing time:

R_i : User requests inserted by Modified First Fit
In this paper, the size of the bin is assumed to be 1.

$$Cost_{total}(P_{FF',R}) = c \cdot \sum_{i=1}^{m} dist(I_i)$$

In order to get the total system maintenance cost, it is first necessary to analyze in detail the time that the user request operation is processed. To do this, add the following definition.

E_i : Before using new bin, the time point, which all of the available capacity has been filled and the use of the bin has been closed

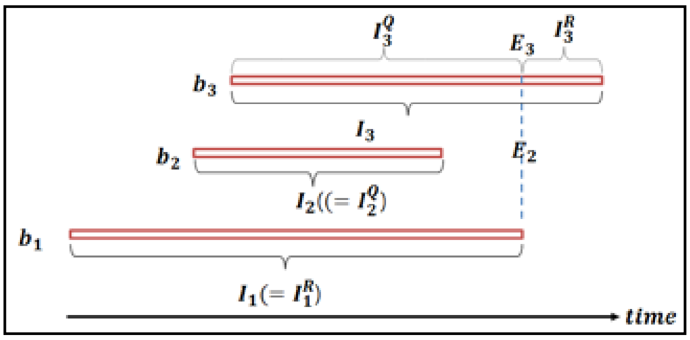

Fig 2 : example of usage period

I_i^Q, I_i^R : If you divide by E_i it becomes a quotient I_i^Q and a remainder I_i^R according to it.

The above definitions are summarized in the following equations

$$Cost_{total}(P_{FF',R}) = c \cdot \sum_{i=1}^{m} (dist(I_i^Q) + dist(I_i^R))$$

$$Cost(P_{FF',R}) = \sum_{i=1}^{m} dist(I_i^Q) + AI(R) \qquad (1)$$

For the detailed analysis, we summarize as follows. , We divide each of them into unit time of the following constant size $(\mu.+1)$ Δ

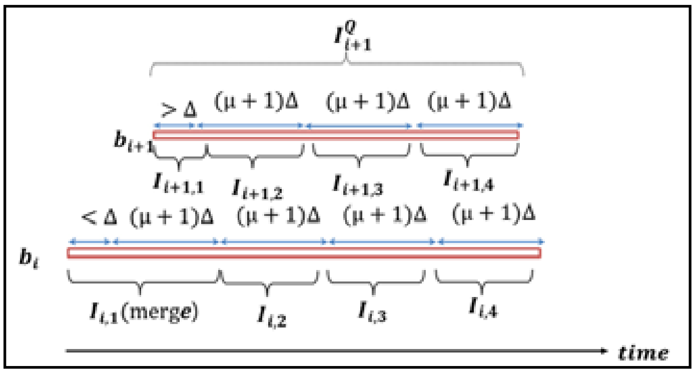

Fig 3 : Example of division and combination of unit time

As theorem can be seen from the above Fig. 3, we can see that this holds true

Theorem 2. It is always satisfied with $dist(I_{i,j}) < (\mu+2)\Delta$ for all *i, j*.
Theorem 3. It is always satisfied with $dist(I_{i,j}) = (\mu+1)\Delta$ for all *i, j*≥ 2.

Theorem 4. If $I_{i,2}$ exists for all i, $dist(I_{i,j}) > \Delta$ is always satisfies.

Based on the definition above, it can be seen that at least one user request is processed in each bin.
In addition, since this research is based on Online Dynamic Bin Packing, we consider the start time and end time of the user request operation according to the definition in order to examine the total cost. Likewise, for more sophisticated system modeling, there are start and end times for actual user request tasks arriving at the system and assigned to them. Therefore, we want to add the definition as follows and draw additional theorem through it.
$t^*_{i,j}$: The time at which the first new user request is assigned to b_i during $I_{i,j}$ is defined as a reference time point.

Theorem 5. It is $t^*_{i,j} = I^-_{i,1}$ in each $I_{i,1}$ depending on how the First Fit algorithm works.

Theorem 6. It is $I^-_{i,j} \le t^*_{i,j} \le I^-_{i,j} + \mu\Delta$ in each $I_{i,1}$.

Theorems 5. and 6. can be used to infer the following lemma 1.

Lemma 1. if i > 1, each $t^*_{i,j}$ has at least one bin(b_h) that satisfies h < i and $t^*_{i,j} < I^+_h$.

Proof. Assuming that all $b_h(h < i)$ are $t^*_{i,j} > I^+_h$ in the reverse of lemma 1, then the case $E_i \le t^*_{i,j}$ occurs. That is, the case of $t^*_{i,j} \in I_i^R$ occurs, which is contradictory to the fact that it was defined earlier, so Lemma 2 is true.

We will define in more detail the virtual processing time for the virtual bin and its internal processing time to calculate the related work processing cost through modeling the work assigned to the corresponding a upon arrival at the system.
$b^*_{i,j}$: Defines a virtual bin that has allocated the most recent user request operation during $I_{i,j}$.

$p^*_{i,j}$: Defines a virtual time period during $t^*_{i,j} - {}^1/_2\,\Delta < t < t^*_{i,j} + {}^1/_2\,\Delta$ which a user request operation is processed during $I_{i,j}$ to $b^*_{i,j}$.

In order to make it easier to understand what is defined above, we have shown in detail Fig. 4.

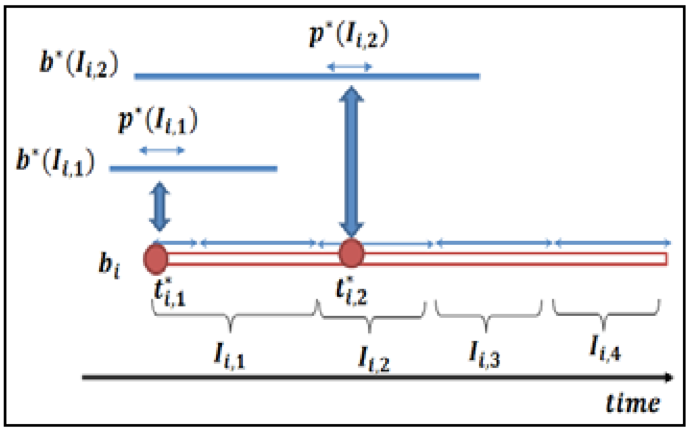

Fig. 4 : Examples of reference time, and reference bins

In addition, since the processing time must be at least Δ, the processing cost (resource usage) of the user work request assigned to is defined as follows.

$$d(p^*_{i,j}) \geq \left(1 - \frac{1}{k}\right) \bullet \triangle \tag{2}$$

If the p*, which is the time to allocate each item to the bin, overlaps, that is, when the allocation occurs in the same time zone, it is processed simultaneously on the system. The conditions under which this occurs are as shown in Fig 4. The two overlapping conditions belonging to each bin are $b^*_{i,j} = b^*_{k,g}$ and $\left| t^*_{i,j} - t^*_{k,g} \right| < \Delta$ at the same time. Based on this, the following lemma 2 can be deduced.

Lemma 2. If $I_{i,j}$. and $I_{k,g}$ are completely separated in time, but there is overlapping time(p^*) between $I_{i,j}$ and the p^* associated with $I_{k,g}$, the following conditions are always satisfied.

$$i < k \quad t^*_{k,g} \geq I^+_i \quad j = 1 \quad g = 1$$

$$dist(I_{i,1}) - dist(I_{k,1}) < \triangle$$

Proof. It is assumed that $t^*_{i,j} = I^-_{i,j}$ and $t^*_{k,g} = I^-_{k,g}$ are known by theorem 4, let's suppose the opposite case of $\left| t^*_{i,j} - t^*_{k,g} \right| < \Delta$. and assume that $p^*_{i,k}$, $p^*_{k,g}$ overlap each other. This indicates that $t^*_{k,g} \geq I^-_{k,g} \geq I^+_{i,j}$ is conflict with $b^*(I_{i,j}) = b^*(I_{k,g})$. So we knows $\left| t^*_{i,j} - t^*_{k,g} \right| > \Delta$ Since this is a non-overlapping condition, it must be true.

Lemma 3. In all bi to $I^-_i \leq t < I^+_i$ hours at a particular time. There are at most two P^*s.

Proof. Suppose that there are $t^*_{g,1}, t^*_{k,1}, t^*_{j,1}$ where overlap occurs where g > k > j > i is formed. In addition, it can be seen that all user requests require the minimum processing time, and that $t^*_{k,1} \geq I^+_j \geq I^-_j + \Delta$ and $t^*_{g,1} \geq I^+_k \geq I^-_k + \Delta$ are established by the above-mentioned $t^*_{k,1} \geq I^+_j$. It can be seen that $t^*_{g,1} \geq I^-_j + \Delta$ is defined by $I^-_k = t^*_{k,1}$. Therefore, it proves that the above theorems are true because there are differences over Δ.

If we rearrange equation (1) based on theorem 2., we can deduce the following equation.

$$\sum_{i=1}^{m} dist(I^Q_i) = Cost_{total}(P_{FF',R}) - AI(R)$$

$$\frac{\sum_{i=1}^{m} dist(I^Q_i)}{(\mu+2)\triangle} \geq \frac{Cost_{total}(P_{FF',R}) - OPT_{total}(R)}{(\mu+2)\triangle}$$

As mentioned above in Equation (2), if each of the user request unit processing time minimum processing time is recalled, there is a maximum of two p* by applying the lemma 3. just mentioned to the condition that the processing cost is at least $(1 - {}^{1}/_{k})\Delta$ Assuming that the overlap occurs, the system processing cost used in the overlaying is considered as one sum cost, and is summarized in the following equation.

$$OPT_{total}(R) = \sum_{r \in R} d(r)$$

$$\geq \frac{1}{2} \bullet \left(1 - \frac{1}{k}\right)\triangle \bullet \frac{Cost_{total}(P_{FF',R}) - OPT_{total}(R)}{(\mu+2)\triangle}$$

$$\geq \frac{1 - \frac{1}{k}}{2\mu+4} \bullet \left(Cost_{total}(P_{FF',R}) - OPT_{total}(R)\right)$$

$$\geq \left(\frac{1 - \frac{1}{k}}{2\mu+4}\right) \bullet \left(\frac{2\mu+4}{2\mu+4+1-\frac{1}{k}}\right) \bullet Cost_{total}(P_{FF',R})$$

$$\geq \left(\frac{1 - \frac{1}{k}}{2\mu+5}\right) \bullet Cost_{total}(P_{FF',R})$$

We can divide both sides of the above equation by $\boldsymbol{OPT_{total}}$ (R) and summarize as follows.

$$\alpha \approx \frac{Cost_{total}(P_{FF',R})}{OPT_{total}(R)} \leq \left(\frac{k}{k-1}\right) \bullet (2\mu+5)$$

Thus, the upper bound of the MUDP algorithm competition ratio (α) is $({}^{k}/_{k-1}) * (2\mu + 5)$ when the size of the user task request is smaller than ${}^{1}/_{k}$ $(k > 1)$.

V. Experiment

Through experiments, we analyze the performance of the algorithm by comparing the experimental results and the results that were introduced in the previous research. In this paper, We will focus on whether it shows performance.

A. Experimental Setup

Table. 3 : Simulation of H/W Environment

H/W	
Element	Spec
CPU	Intel XEON 2.4 Ghz
MEM	DDR3 128Gb
HDD	3TB
LAN	1000M

Table. 4 : Simulation of S/W Environment

S/W	
Element	Spec
Host OS	Citrix XenServer 7.0
Guest OS	Ubuntu 14.04(Kernel 3.2.0)
Language	C++

Similarly, for various user task sizes allocated to the system, an arbitrary integer is generated by the following equation, and then the rule is normalized and inserted into the bin (the values Upper and Lower are integers between 1 and 100 Value.)

$$Random\ Nmber = (lower + rand()\%(upper - lower + 1))$$

In addition, although the sizes of the items are different, the total sum of the sizes of the items must be the same. This is because it is possible to objectively compare the performance measurement results of each algorithm by inputting items of various sizes by setting the total sum of the sizes to be the same.
In order to fit the online dynamic bin packing model, an arbitrary item is removed from each bin randomly for each *2k+1(k > 1)* th Loop searching the entire array. For the modified operating

condition of the MFF proposed in this paper If all the items that existed in the bin used for the allocation in the corresponding loop are left after the operation, the corresponding bin is removed without consideration for packing the new item. (Array index exception processing)

B. Analysis of existing techniques and MUDP

We attempted to obtain the average of the results of the competition ratio by performing 1000 runs under the assumption that 50, 100, and 1000 requests of the job size are sequentially received. In addition, when the size of the action request is smaller than a specific size, we divided into 4 types. Also, we tried to find the worst case among the experimental values in each case.

The competitive ratio value of the online dynamic bin packing model obtained from this experiment is the average value of (current bin index value × bin capacity / sum of item size) after 1000 runs.

Table 5,6,7 show the difference between the existing first blank packing first fit according to the item size and the modified first fit (modified) focused on cost efficiency Comparisons of competitive ratio and their outcomes.

Table. 5 : Experimental results on First Fit(Item number: 50)

Algorithms / ItemSize(S)	Competitive Ratio		Worst Value	
	FF	MFF	FF	MFF
0 ~ 0.1	1.1550	1.1632	1.1550	1.1632
0 ~ 0.2	1.1607	1.1677	1.2505	1.1688
0 ~ 0.5	1.8933	1.9183	1.9686	2.1830
0 ~ 1.0	2.4239	2.4319	2.4805	2.4830

Table. 6 : Experimental results on First Fit(Item number: 100)

Algorithms / ItemSize(S)	Competitive Ratio		Worst Value	
	FF	MFF	FF	MFF
0 ~ 0.1	1.1003	1.1233	1.1119	1.2002
0 ~ 0.2	1.2455	1.2450	1.3601	1.3988
0 ~ 0.5	1.7331	1.8188	1.8009	2.0119
0 ~ 1.0	2.2912	2.3999	2.4015	2.4829

Table. 7 : Experimental results on First Fit(Item number: 100)

Algorithms / ItemSize(S)	Competitive Ratio		Worst Value	
	FF	MFF	FF	MFF
0 ~ 0.1	1.1010	1.1330	1.1010	1.1400
0 ~ 0.2	1.1413	1.1500	1.1663	1.1698
0 ~ 0.5	1.6003	1.8838	1.8190	2.0093
0 ~ 1.0	2.1030	2.4004	2.4201	2.4195

The tendency of the experimental result value according to the size and type of the input item has a similar flow to the online dynamic empty packing method which has been tested. In addition, it was found that the result of the competition ratio varies according to the number of items. According to the modified MFF condition, there is a slight increase in the competition ratio and the worst value compared with the existing technique, but it is not a big difference. That is, it can be seen that the modified condition does not have a great influence on the optimization.

Also, Worst-case tends to appear as the item size is small and the number of inputs is small. Through the above experiment, it can be seen that the number of users requesting work in the cloud platform environment assumed by this paper is lower than the competition rate proved by the existing theory.

VI. Conclusions

In addition to Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), the concept of Function as a Service (FaaS) has recently received renewed attention. This refers to a service that processes individual user request operations in a small unit of function rather than in a virtual machine unit as in the conventional method. In other words, it is necessary to study how to efficiently allocate many different very small units of work for such serverless micro services.

In this paper, we study the characteristics of users request task in the cloud platform environment is very irregular and unpredictable in time, and a cost - effective resource allocation algorithm for serverless micro services for FaaS, one of the concepts of cloud platform service. We propose a technique to reduce system operation cost by applying on - line dynamic bin packing, which is a variant of well - known bin packing method among existing job scheduling algorithms. In other words, if the existing bin packing method aims to utilize the minimum bin, the proposed method is a provisioning method that considers a more costly aspect by minimizing the time of using the system infrastructure. In addition, we analyze the competitive ratio of the proposed method through the detailed system modeling and the analysis method of the approximate algorithm for the user requested work time.

And we perform a basic experiment to measure the performance of MUDP and the existing on-line bin packing algorithm. The performance of the MUDP algorithm is analyzed through comparative analysis of the competition ratio values that have been proven in previous studies. It can be seen that, as in the actual cloud platform service environment, as the number of user work requests (items) is more than a certain number and the range of the item size is smaller, it converges to the ideal competition value 1.

As a further of this research, we propose a homogeneous system based on the assumption that all bins have the same size and capacity. Therefore, we propose a more general algorithm by modifying the proposed algorithm for a heterogeneous system environment. It is necessary to develop. We also investigate the upper bound and the lower bound of the proposed MUDP algorithm through a more detailed analysis.

References

[1] Amazon lambda Pricing, accessed on OCT. 30, 2017 [Online].Available: https://aws.amazon.com/lambda/pricing/

[2] Shreenath Acharya and Demian Antony D'Mello, "Cloud Computing Architectures and Dynamic Provisioning Mechanisms", In proceeding of International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), 2013 pp.798-804, 2013.12

[3] J. Balogh, J Békési, and G. Galambos. New lower bounds for certain classes of bin packing algorithms. Theoretical

Computer Science, pp.440–441:1–13,8 2012.

[4] *Zhu, Q., and Agrawal, G. "Resource provisioning with budget constraints for adaptive applications in cloud environments." Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing HPDC '10, pp.304-307. 2010*

[5] *Henzinger, T. a., Singh, A. V., Singh, V., Wies, T., and Zufferey, D. "FlexPRICE: Flexible Provisioning of Resources in a Cloud Environment." 2010 IEEE 3rd International Conference on Cloud Computing, pp83–90. 2010*

[6] *Tai, J., Zhang, J., Li, J., Meleis, W., and Mi, N. . ArA: "Adaptive resource allocation for cloud computing environments under bursty workloads." 30th IEEE International Performance Computing and Communications Conference, pp. 1–8. 2011*

[7] *Huu, T. T., and Tham, C.-K. "An Auction-Based Resource Allocation Model for Green Cloud Computing." IEEE International Conference on Cloud Engineering (IC2E), pp. 269–278. 2013*

[8] *D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. Commun. ACM, 28(2) pp. 202–208, 1985. 02*

[9] *M. Flammini, G. Monaco, L. Moscardelli, H. Shachnai, M. Shalom, T. Tamir, and S. Zaks. "Minimizing total busy time in parallel scheduling with application to optical networks." In Proceedings of the 23th IEEE International Symposium on Parallel and Distributed Processing, pp.1–12. 2009.*

[10] *G. B. Mertzios, M. Shalom, A. Voloshin, P. W. Wong, and S. Zaks. "Optimizing busy time on parallel machines." In Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium, pp.238–248. 2012.*

[11] *R. van Stee. SIGACT news online algorithms column 20: The power of harmony. SIGACT News, 43(2) pp.127–136, 2012.06*

[12] *X. Han, C. Peng, D. Ye, D. Zhang, and Y. Lan. "Dynamic bin packing with unit fraction items revisited." Inf. Process. Lett., 110(23) pp.1049–1054, 2010.11*

[13] *Balogh, J. Bekesi, G. Galambos, and G. Reinelt, "Lower bound for the online bin packing problem with restricted repacking," SIAM J. Comput. 38 (1) pp.398–410. 2008*

[14] *W.-T. Chan, T.-W. Lam, and P.W.H. Wong, Dynamic bin packing of unit fractions items, Theoret. Comput. Sci. 409 pp.521–529. 2008*

[15] *Song, W., Xiao, Z., Chen and Q., Luo, H.: "Adaptive resource provisioning for the cloud using online bin packing." IEEE Trans. Computers 63(11), pp.2647–2660 2014*

[16] *Sunilkumar S.Manvi a and GopalKrishnaShyamb "Resource management for Infrastructure as a Service (IaaS) in cloud*

[17] *Computing:A survey",Journal of Network and Computer Applications Volume 41 pp.424-440 2014.05*

[18] *Nguyen Quang-Hung and Nam Thoai, "Minimizing Total Busy Time with Application to Energy-efficient Scheduling of Virtual Machines," International Conference on Advanced Computing and Applications pp. 141-148 2016*

[19] *T.P. Shabeera, S.D. Madhu Kumar, Sameera M. Salam and K. Murali Krishnan, "Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO meta –heuristic algorithm," Engineering Science and Technology, an International Journal 20 pp.616–628 2017*

Author's Profile

Seunghwan Yoo *received the B.S. and M.S. degrees in the Depart-ment of Computer Science and Engineering at Sogang University in 2007 and 2009, respectively. He currently researches cloud plat-form service and provisioning as part of research towards a Ph.D. degree at the same university.*

Sungchun Kim *received the B.S. degree at Seoul National Uni-versity, Korea in 1975, and recei-ved M.S and Ph. D. degrees in the Department of Computer Enginee-ring at Wayne State University in 1979 and 1982, respectively. During 1982-1984, he was an assistant professor at the Depart-ment of Computer Science, Cal-ifornia State University, Fullerton, USA. 1984-1985, he worked at Gold Star Semi-conductor, Korea, as a researcher from 1984-1985. He is now a professor at the Department of Computer Science and Engineering, Sogang Uni-versity, Korea, since 1985 to the present day*