Article

Resource Demand Forecasting Model Based on Dynamic Cloud Workload

Chunyan An [1], Jiantao Zhou [1,*]

[1] College of Computer Science, Inner Mongolia University, Hohhot, China; ann@imu.edu.cn
* Correspondence: cszjtao@imu.edu.cn;

Abstract: The primary attraction of IaaS is providing elastic resources on demand. It becomes imperative that IaaS-users have an effective methodology for learning what resources they require, how many resources and for how long they need. However, the heterogeneity of resources, the diversity resource demands of different cloud applications and the variation of application-user behaviors pose IaaS-users big challenge. In this paper, we purpose a unified resource demand forecasting model suiting for different applications, various resources and diverse time-varying workload patterns. With the model, taking input from parameterized applications, resources and workload scenarios, the corresponding resources demands during any time interval can be deduced as output. The experiments configure concrete functions and parameters to help understanding the above model.

Keywords: cloud computing; workload model; workload-aware resource forecasting model

1. Introduction

At present, ever-increasing enterprises, SMBs (Small-and-Medium-sized Businesses) and organizations are running their applications on IaaS (Infrastructure as a Service) [1]. To take true advantages of IaaS, IaaS-users need to be able to rent the resources really on demand. For that, IaaS-users need to compare alternatives IaaS resources when building new systems, to assess capacity of selected resources when setting up systems, to adjust the use of cloud resources dynamically during running of the system.

However, with the development of cloud computing, IaaS-users face a bigger challenge. The challenges are mainly from the following reasons:

(1) The heterogeneity of IaaS resources

Different IaaS providers usually have different system design and system implementation. Besides, each IaaS provider provides a number of different optional IaaS resources. Moreover, different cloud environments are the different combinations of cloud resources across multiple hybrid clouds. To evaluate and compare different cloud environments is a big challenge.

(2) The diversity of applications

With the adoption of clouds in more and more areas, even including traditional areas, the type of cloud applications is more diverse [2]. Different applications, even if belong to the same area, the demand for cloud resources may have very big difference.

(3) The complexity of user behaviors

In reality, multiple applications are often deployed in one cloud environment. At the same time, end users are from around the world, they request the applications according to their own habits and requirements. This makes the user behaviors more complex. Thus the patterns of collective user behaviors, i.e. the workload patterns in this paper, have more possibilities.

Benchmarks[3] in cloud computing solve the first challenge partly. Benchmarks generate a select set of workloads that can be ported to different cloud environments and used as the basis for comparison and evaluation. However, most Benchmarks are either domain specific, like Rubis[4] for e-business, or programming-model specific, like MapReduce[5] etc. Even if the applications of IaaS-users belong to the similar domain or similar programming-model of the respective

benchmarks, the result from the benchmarks can be very different from the actual running result. Another shortcoming of benchmarks is the workload patterns defined in a benchmark are too limited to cover the possible workload scenarios.

Workload modeling is another option to address the above challenges. Workload modeling is to set up general statistical model, which can be used to generate synthetic workloads. In [6], workload models are classified into two types: descriptive models and generative models. Descriptive models mimic realistic workload traces mostly by fitting distributions. Generative models simulate user behaviors at the first place. This paper falls into basically generative models with an attempt of higher abstraction degree.

This workload modeling provides a general definition for cloud applications, which is suitable to specify most of cloud applications. Moreover, this paper offers a general statistical model, which can be used to generate diverse workload patterns. The generated workload here actually is a collection of end user requests to the applications. The statistical model specifies how the collective user requests change over time. Then, based on the workload modeling, the numbers of each basic service units in every certain period are computed out. Since the mean execution times of each basic service units running on each resource type can be measured, thereby the demands to a cloud environment in every certain period are to be estimated. In the previous work [7], a hierarchical workload modeling was proposed. In this work, the workload modeling is improved and based on the workload modeling a transformation methodology between the workload and resource demand is presented.

The remainder of this paper is organized as follows: Section 2 presents the existing related works. Section 3 specifies the workload modeling approach and resource demand estimation approach. Section 4 illustrates the process of generating workload based on workload model with an example. The last, Section 5 is a conclusion and discussion of the future work.

2. Related Works

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation as well as the experimental conclusions that can be drawn. The challenges on cloud application workload modeling and workload-aware cloud infrastructure resource prediction have attracted many researchers. The related works will be specified in four aspects: workload population modeling/cloud application modeling, workload variation modeling, cloud infrastructure resource modeling and workload-aware resource prediction.

- Cloud application modeling

Cloud Application Model defines the basic components (e.g., a web URL, a function as a service, mapper/reducer) and the dependencies between basic components. Most of existed cloud application models are application-specific or programming-model-specific. For example,[8] presented workload modeling techniques only for web applications like Rubis[4], cloud application model is defined by a PFSM(Probabilistic Finite State Machine) tuple M = (I, O, S, T, P), which specified the transitions probabilities between web-pages and data dependencies (input/output) between web-pages. Similarly, [9], a workload generator developed by Berkley University, specified the request transition probabilities in a web application with Markov Matrix. And the works from MaoMing[10,11] aimed at workflow application. An application consists of a set of service units, and a job class is defined by a DAG (Directed Acyclic Graph) with deadline. And three types of workflows are studied in the paper: pipeline, parallel and Hybrid. For Bag-of-Tasks (BoT) applications, Alexandru Iosup[12] defined workload which consists of the BoT jobs submitted by different users, the users are ranked by the number of their submitted jobs. The tasks in a bag usually are assumed independent of each other[13] or a set of sequential tasks (possibly only one)[12]. For MapReduce applications, Yanpei Chen[14,15] offered a general MapReduce application definition, in which the execution of each MapReduce job is divided into three stages: map(input)/shuffle/reduce(output), a job is specified by input data-size, input/shuffle/output data ratio and data format. All above models for specific application types are not general enough to

support multi-application-types workloads in reality. At present, we see the efforts in general cloud application model from cloud service brokers (e.g. Cloud Application Template from RightScale[16] and international standards organizations(e.g. SPEC OSG Cloud Working Group of SPEC. OSG[2]). Our cloud application model attempts to build a consistent definition which can cover most of cloud application types.

- Workload variation modeling

Representativeness, reality and generalization are the requirements of most workload modeling. Existed workload models for cloud applications cannot meet all the three requirements.

(1) Benchmarks:

Many researchers and organizations have been devoting themselves to developing representative workload models as benchmarks, for examples [3], [17, 21]. However, the workload patterns in benchmarks are usually very limited, cannot span the spectrum of possible workloads. Moreover, a benchmark aimed at a specific application, cannot reflect real usage of applications of cloud users, even if they are in similar domain.

(2) Workload modeling based on trace:

In order to build a more realistic workload model, many researchers fitted out the mathematical workload models using trace logs from some cloud providers [22, 25]. Though a fitting model is closer to the reality than a model based on mathematical assumption, it can reflect only some specific workload patterns of specific cloud providers during some specific time periods.

(3) Abstract mathematic workload modeling:

Abstract mathematic workload modeling tries to capture the main characteristics of workloads or some workload patterns. Through the parameterization of abstract workload model, diverse workload models or specific workload models conformed to some workload patterns can be produced. For example, [26] gave a methodology of building a burst workload model. The idea that generated a spike workload by superposition of normal workload and increasing workload inspired our work. [27] used periodicity and burstiness to specify and workloads in order to classify the workloads for elastic resource provision. [28] proposed a Hierarchical Bundling Model (HIBM) to model inter-arrival process. Our workload model falls into abstract mathematic workload modeling. We constructed a dynamic hierarchical workload model. On each layer two main variations are captured: number and mix. Then the superposition the variations of each layer produced the final workload model. Our model is so general that diverse workload patterns can be generated.

- Cloud infrastructure resource modeling

Cloud Infrastructure Resource consists of mainly CPU, memory, storage, disk I/O and network. Most Mainstream IaaS providers, like Amazon, RackSpace, provide different types of VMs (Virtual Machines). Each type of VM is configured with a set of resources. VM instances are the basic units of resource allocation. Therefore, many researchers modeled infrastructure resources as VM_s [29, 31]. Some researchers defined an infrastructure resources as a set $\{R_1,\ldots\ldots R_m\}$, in which R_k (k=1..m) is a given level capacity (e.g. CPU, memory, storage, disk I/O and network)[32]. Some researchers [24] concerned about the detailed consumptions of individual hardware (e.g. CPU, memory, disk). Since the objectivity of this paper is to help a cloud user to manage their rented infrastructure resources, the resource model is based on VM.

- Workload-aware resource prediction

The research of workload-aware resource prediction includes two parts: (1) at some time point, the relationship of the amount of workload and the needed resources. (2) in a certain workload scenario (e.g. burst or diurnal pattern), the correspondence between workload fluctuation and the variation of resource requirements. For example, [14] used KCCA (Kernel Canonical Correlation Analysis) method to predict the execution time of MapReduce jobs. And [33] studied the optimized resources scaling options based on workload variation under the premise of ensuring SLA. The work in this paper has done the first part transforming the workload into the amount of basic execution units, which is the base of prediction of resource requirements.

3. Resource Demand Forecasting Model Based on Cloud Workload

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

Workload model includes two parts: a standard cloud application model and a workload variation model. Cloud application model includes a group of service unit specifications and the dependencies between the service units. A right stochastic matrix defines the dependencies between the service units. The stochastic matrix specifies the probabilities of users moving from service unit i to service unit j. This model is suitable for describing the cloud applications in which the service units are loosely-coupled or independent (e.g. web applications) or the service units are sequentially executed (e.g. MapReduce). And the model is not suitable for specifying the complex workflow application which includes control flow dependencies like Synchronize. Considering that loosely-coupled web applications and map-reduce applications covered most of cloud applications, we choose a stochastic matrix to specify the dependencies between service units rather than DAG (Directed Acyclic Graph) [7].The workload variation model abstracts main characteristics of workload variations. Here workload variation is abstracted a compound arrival process, which specify arrival process, popularity of service units, data-size or computation-scale distribution of each service unit. Workloads which are generated based on the workload model are a set of mixed requests to different service units. Then the requests to different service units are normalized into basic execution units. Last, the requests during any time interval are transformed into the numbers of needed resources. The models are detailed in the following four sub-sections.

3.1. Application Model

An Application is specified by $A_i = (S(A_i), D(A_i), C(A_i))$, where $S(A_i) = \{s_1 \dots s_n\}$ is a finite set of service unit types, $D(A_i)$ is a $n \times n$ matrix to specify the probability links between the service units, $C(A_i)$ is a constant to specify the input unit data size.

About $S(A_i) = \{s_1 \dots s_n\}$, $s_i\ (i = 1, \dots n)$ means a service unit type in the application A_i, n is the number of service units types in the application. Here s_n is additional service unit, which means exit state.

And in $D(A_i)$, $p(s_i, s_j)$ denotes the probability that service unit s_i links next to s_j .

$$D(A_i) = \begin{bmatrix} p(s_1,s_1) & p(s_1,s_2) & \cdots & p(s_1,s_n) \\ p(s_2,s_1) & \ddots & & \vdots \\ \cdots & & p(s_i,s_j) & \\ p(s_n,s_1) & \cdots & & p(s_n,s_n) \end{bmatrix} \quad (1)$$

Where

1. $p(s_i, s_j) \geq 0, \quad i, j = 1,2, \dots n$
2. $\sum_{j=1}^{n} p(s_i, s_j) = 1$
3. $\begin{cases} p(s_n, s_i) = 0, & \text{if } i \neq n \\ p(s_n, s_n) = 1 \end{cases}$

We assume that user can stop the request with some probability after performing any service unit. $p(s_i, s_n)$ is the probability, after performing service unit s_i, that the user will stop the request.

We choose Markov matrix to specify the processing dependencies between service units are based on the following belief:

(1) Most web-based Applications are loosely coupled. That means the service units in an application have no complex processing dependencies like join/split. The Markov matrix can specify all the possible processing orders and user preferences.

(2) Map-Reduce Applications usually include map->shuffle->reduce three service units, and the three units execute sequentially. So the Markov matrix specifying a map-reduce application will be like this:

i) $A_i = (S(A_i), D(A_i), C(A_i))$

ii) $S(A_i) = \{map, shuffle, reduce, end\}$

iii) $$D(A_i) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

3.2. Infrastructure Resource Model and Service Unit Execution Time Definition

Firstly, a VM (Virtual Machine) type VM_p is defined by a capacity consisting of a group of optional infrastructure hardware ((CPU, CPU number), (Memory, Memory Size), (Disk, Storage Size), (Network, Bandwidth)).

Secondly, on a VM(Virtual Machine) type VM_p, the basic execution times of service units of application A_i is defined as a $T(VM_p, A_i) = (t(VM_p, s_1), \dots t(VM_p, s_{n-1}), 0)$, in which $t(VM_p, s_j)$ represents the mean execution time of service unit s_j with input unit data size $C(A_i)$ on a VM_p. $t(VM_p, s_n)$=0 because s_n is the exit service unit, which executes null. The other $t(VM_p, s_j)(j = 1,2, \dots n-1)$ satisfy with $t(VM_p, s_j) > 0$. The basic execution time of Application A_i on VM_p is defined as $\text{BasicTime}(VM_p, A_i) = \min\{t(VM_p, s_j), (j = 1,2, \dots n-1)\}$.

Thirdly, for each service unit s_j, there exists a scaling function $f_j(x_j, VM_p)$, x_j is the input data size, $f_j(x_j, VM_p)$ is the execution time of service unit s_j on VM_p with the input data size x_j. $f_j(x_j, VM_p)$ describes the relationship between input data size and execution time on VM_p for service unit s_j.

Lastly, a basic service unit here means a service unit which has a basic input data size or a basic computational scale. A set of basic service units of application A_i is defined as $n \times 1$ matrix $\text{Basic}(VM_p, A_i) = (b(VM_p, s_1), \dots b(VM_p, s_{n-1}), 0)$. $b(VM_p, s_j) > 0, j \neq n$ represents the basic input unit of service unit s_j on VM_p. And $b(VM_p, s_n) = 0$ because s_n is the exit service unit, whose input is null.

For normalization, each $b(VM_p, s_j)$, $j \neq n$ should choose appropriate value to make $f_j(b(VM_p, s_j), VM_p) = \text{BasicTime}(VM_p, A_i)$. That means, the execution times of all basic service units except s_n are equal to $\text{BasicTime}(VM_p, A_i)$.

3.3.Workload Variation Model

For simplifying the problem, the workload consists of one application type. In a time interval new requests consist of two parts: new arrival requests and transform requests. New arrival requests are the requests to the service units which are belong to the set of start service units of an application; Transform requests are determined by the dependencies between service units in an application.

Firstly, the workload variation during a short interval$(t, t+\tau]$, $0 < \tau < \text{BasicTime}(VM_p, A_i)$is computed. There are two properties of the interval$(t, t+\tau]$.

(1) In this interval, the new arrived requests for A_i will not transform to next service unit.

(2) In this interval, the currently existing service units arrived before t can only transform to next one step.

All requests during an interval $(t_1, t_2]$ are defined as follows:

i. Inter-arrival process is defined as follows:

(1) Mean arriving rate

If start time is 0, until time t the mean of the number of arrived requests for A_i is $N(A_i, t)$, then $[N(A_i, t+\tau) - N(A_i, t)]/\tau$ is called the mean arriving rate of requests for A_i during an interval$(t, t+\tau]$. The arriving rate at some time point t is

$$\alpha(A_i, t) = \lim_{\tau \to 0}[N(A_i, t+\tau) - N(A_i, t)]/\tau = d(N(A_i, t))/dt \quad (3)$$

So the mean arriving rate of requests for A_i during the interval is $(t, t+\tau]$

$$a(A_i, t, t+\tau) = \int_t^{t+\tau} \alpha(A_i, t)dt \Big/ \tau \quad (4)$$

And the mean of the number of arrived requests for A_i is during an interval $(t, t+\tau]$

$$N(A_i, t, t+\tau) = \int_t^{t+\tau} \alpha(A_i, t)dt \quad (5)$$

(2) Compound arriving process considering the popularity of different service units in A_i

Each new arrived request for A_i could be one of types of service units in $S(A_i)$. The probability distribution PR^{A_i} represents the popularity of service units in new arrived requests. PR^{A_i} is defined by a $n \times 1$ matrix

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

$$PR^{A_i} = \begin{pmatrix} PR^{A_i}(s_1) \\ \cdots \\ PR^{A_i}(s_n) \end{pmatrix} \text{ and satisfied } \sum_{j=1}^{n} PR^{A_i}(s_j) = 1 \quad (6)$$

Let $S(A_i^{start})$ denote the set of all possible service units to which new arrived A_i requests refer. If $s_j \notin S(A_i^{start})$, then $PR^{A_i}(s_j) = 0$; Obviously, s_n is exit service unit, so $s_n \notin S(A_i^{start})$, $PR^{A_i}(s_n) = 0$.

Assume during an interval $(t, t+\tau]$, if the arriving rate at some time point t of requests for s_j is $\alpha(A_i: s_j, t)$, and the mean of the number of arrived requests for s_j is $N(A_i: s_j, t, t+\tau)$, then

$$N(A_i: s_j, t, t+\tau) = \int_t^{t+\tau} \alpha(A_i: s_j, t)dt \quad (7)$$

in which

$$\alpha(A_i: s_j, t) = PR^{A_i}(s_j)\alpha(A_i, t) \quad (j = 1,2 \ldots n) \quad (8)$$

So

$$N(A_i: s_j, t, t+\tau) = \int_t^{t+\tau} PR^{A_i}(s_j)\alpha(A_i, t)dt \quad (9)$$

And

$$N(A_i, t, t+\tau) = \sum_{j=1}^{n} N(A_i: s_j, t, t+\tau) \quad (10)$$

Define a $1 \times n$ matrix $N^R(A_i, t, t+\tau)$ as the numbers of arrived requests for each s_j of Application A_i during an interval $(t, t+\tau]$,

$$N^R(A_i, t, t+\tau) = \left(N(A_i: s_1, t, t+\tau), \quad \cdots N(A_i: s_n, t, t+\tau)\right) \quad (11)$$

(3) Compound arriving process considering bulk arrival.

If each arrived request for each s_j $(j = 1,2 \ldots n)$ of Application A_i could include multiple basic service unit s_j, then the arrival process is called bulk arrival process. Let random variable Z_{S_j} represent the number of basic service unit s_j in each arrived request for s_j , its probability mass function is $P(z_j) = P\left(Z_{S_j} = z_j\right), z_j \geq 0$ and the mean of Z_{S_j} is $E[Z_{S_j}]$. Let us define $Z_{S_j,\ 1}, Z_{S_j,\ 2}, \ldots$ as the iid sequential numbers of basic service units$_j$, the number of basic service unit s_j during an interval $(t, t+\tau]$, $N^S(A_i: s_j, t, t+\tau)$ is given by

$$N^S(A_i: s_j, t, t+\tau) = \sum_{m=1}^{N(A_i: s_j, t, t+\tau)} Z_{S_j,\ m} \quad (12)$$

Define a $1 \times n$ matrix $N^S(A_i, t, t+\tau)$ as the arrived numbers of each service unit of Application A_i during an interval $(t, t+\tau]$

$$N^S(A_i, t, t+\tau) = \left(N^S(A_i: s_1, t, t+\tau), \quad \cdots N^S(A_i: s_n, t, t+\tau)\right) \quad (13)$$

ii. Transform requests during the interval $(t, t+\tau]$ are defined as follows:

Assume the numbers of currently existing service units of Application A_i at time t is defined as a $1 \times n$ matrix $\mathcal{N}^S(A_i, t)$

$$\mathcal{N}^S(A_i, t) = \left(\mathcal{N}^S(A_i: s_1, t), \quad \cdots \quad \mathcal{N}^S(A_i: s_{n-1}, t), \mathcal{N}^S(A_i: s_n, t)\right) \quad (14)$$

Then after one step transform, the numbers of transformed service units during the interval $(t, t+\tau]$ is defined as $\mathcal{N}(A_i, t, t+\tau)$

$$\mathcal{N}(A_i, t, t+\tau) = \mathcal{N}^S(A_i, t) * D(A_i) \quad (15)$$

$$= \left(\mathcal{N}^S(A_i: s_1, t) \quad \cdots \mathcal{N}^S(A_i: s_n, t)\right) \begin{bmatrix} p(s_1, s_1) & p(s_1, s_2) & \cdots & p(s_1, s_n) \\ p(s_2, s_1) & \ddots & & \vdots \\ \cdots & & p(s_i, s_j) & \\ p(s_n, s_1) & \cdots & & p(s_n, s_n) \end{bmatrix}$$

From the equitation 15, the each $\mathcal{N}(A_i: s_j, t, t+\tau)$ can be computed out,

$$\mathcal{N}(A_i: s_j, t, t+\tau) = \sum_{m=1}^{n} \mathcal{N}^S(A_i: s_m, t) * p(s_m, s_j) \quad (16)$$

iii. All requests during the interval$(t, t+\tau]$are defined as follows:

$\mathcal{N}^S(A_i, t+\tau)$ is the sum of the new arrived service units during $(t, t+\tau]$ and the one-step transformed service units during $(t, t+\tau]$.

$$\mathcal{N}^S(A_i, t+\tau) = N^s(A_i, t, t+\tau) + \mathcal{N}(A_i, t, t+\tau) \quad (17)$$

$$= N^s(A_i, t, t+\tau) + \mathcal{N}^S(A_i, t) * D(A_i) \quad (18)$$

iv. All requests during an interval $(t_1, t_2]$ are defined as follows:

Next, we consider the workload variation during an interval $(t_1, t_2]$ $t_2 > t_1 \geq 0$ following the steps.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

(1) $\left[\frac{t_2-t_1}{\tau}\right] = \min\{k \in \mathbb{Z} | k \geq \frac{t_2-t_1}{\tau}\}$, k is the numbers of transforms during$(t_1, t_2]$, its value is the smallest integer not less than $\frac{t_2-t_1}{\tau}$. (19)

(2)Based on Eq.(18), the numbers of each service units of Application A_i at time $t + j\tau$ can be deduced by

$$\mathcal{N}^S(A_i, t + j\tau) = N^S(A_i, t_1 + (j-1)\tau, t_1 + j\tau) + \mathcal{N}^S(A_i, t_1 + (j-1)\tau) * D(A_i) \quad (20)$$

(3)Finally, after k loop iterations, the numbers of each service units of Application A_i at time t_2 are computed out:

$$\mathcal{N}^S(A_i, t_2) = N^S(A_i, t_1 + (k-1)\tau, t_2) + \sum_{j=1}^{k-1}[N^S(A_i, t_1 + (j-1)\tau, t_1 + j\tau) * D(A_i)^{k-j}] + \mathcal{N}^S(A_i, t_1) * D(A_i)^k \quad (21)$$

3.4. Resource Forecasting

To forecast the demands of VM_p during an interval$(t_1, t_2]$, firstly, define a workload $W_j(A_i, t_1, t_2)$ is a $1 \times k$ matrix, k is as same as Eq.(19).

$$W_j(A_i, t_1, t_2) = (\mathcal{N}^S(A_i, t_1), \mathcal{N}^S(A_i, t_1 + \tau), \ldots, \mathcal{N}^S(A_i, t_2)) \quad (22)$$

Then, define a $1 \times k$ matrix $N(VM_p, W_j(A_i, t_1, t_2))$ as the numbers of needed VM_p for Application A_i during an interval $(t_1, t_2]$.

$$N(VM_p, W_j(A_i, t_1, t_2)) = (\mathcal{N}^S(A_i, t_1), \mathcal{N}^S(A_i, t_1 + \tau), \ldots, \mathcal{N}^S(A_i, t_2))/Capacity(VM_p, A_i) \quad (23)$$

Where $Capacity(VM_p, A_i)$ means the maximal number of parallel basic service units on one VM_p with an acceptable performance or satisfied with some SLA (Service Level Agreement). Table 1 lists the important annotations in the model.

Table 1. The Annotations in the Model

Annotations	Type	Description
$A_i = (S(A_i), D(A_i), C(A_i))$	tuple	application definition
$S(A_i)$	set	a finite set of service unit types
$D(A_i)$	$n \times n$ matrix	specify the probability links between the service units
$C(A_i)$	int	specify the input unit data size
VM_p	tuple	VM(Virtual Machine) type definition
$T(VM_p, A_i)$	1×(n-1) matrix	the basic execution times of service units of application A_i is defined as a $T(VM_p, A_i) = (t(VM_p, s_1), \ldots t(VM_p, s_{n-1}),0),$, in which $t(VM_p, s_j)$ represents the mean execution time of service unit s_j with input unit data size $C(A_i)$ on a VM_p
$BasicTime(VM_p, A_i)$	float	The basic execution time of Application A_i on VM_p is defined as $BasicTime(VM_p, A_i) = \min\{t(VM_p, s_j), (j = 1,2, \ldots n-1)\}$.
$f_j(x_j, VM_p)$	function	$f_j(x_j, VM_p))$ describes the relationship between input data size and execution time on VM_p for service unit s_j.
$Basic(VM_p, A_i)$	$1 \times n$ matrix	A set of basic service units of application A_i is defined as $Basic(VM_p, A_i) = (b(VM_p, s_1), \ldots b(VM_p, s_{n-1}), 0)$. $b(VM_p, s_j) > 0, j \neq n$ represents the basic input unit of service

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

		unit s_j on VM_p
$\alpha(A_i, t)$	float	describes the arriving rate at some time point t
$a(A_i, t, t+\tau)$	float	the mean arriving rate of requests for A_i during an interval $(t, t+\tau]$
$N(A_i, t)$	int	If start time is 0, until time t the mean number of arrived requests for A_i
$N(A_i: s_j, t, t+\tau)$	int	The mean number of arrived requests for s_j during an interval $(t, t+\tau]$
$N^R(A_i, t, t+\tau)$	1×n matrix	the numbers of arrived requests for each s_j of Application A_i during an interval $(t, t+\tau]$
Z_{s_j}	int	represent the number of basic service unit s_j in each arrived request for s_j
$N^S(A_i: s_j, t, t+\tau)$	int	the number of basic service unit s_j during an interval $(t, t+\tau]$

4. Experiments

In this section, a simple example of workload generation process is given to show how to generate a workload based on a workload model.

i. Workload model definition

(1) application definition

$A_i = (S(A_i), D(A_i), C(A_i))$

$S(A_i) = \{S1, S2, S3, end\}$

$$D(A_i) = \begin{bmatrix} 0.28 & 0.21 & 0.28 & 0.23 \\ 0.40 & 0.18 & 0.27 & 0.15 \\ 0.28 & 0.27 & 0.29 & 0.16 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(2) workload variation definition

time interval $\tau = 1$

$$PR^{A_i} = \begin{pmatrix} 0.16 \\ 0.48 \\ 0.36 \\ 0 \end{pmatrix}$$

Arriving process is possion process with parameter λ=3, abbreviated to PP(λ=3).

ii. Workload generation process

(1) Generate 40 service unit requests: arriving times are generated conformed to PP(λ=3), arriving service units are generated randomly according to PR^{A_i}. The detail is shown in Table2.

Table 2. The Annotations in the model.

Arriving Time	Service Unit	Arriving Time	Service Unit	Arriving Time	Service Unit
0.0173	2	0.3518	2	0.5995	3
0.0466	2	0.3869	1	0.6090	3
0.0688	2	0.4829	2	0.6216	3
0.0826	3	0.5010	2	0.6922	2
0.1019	1	0.5227	2	0.6941	1
0.2249	3	0.5487	1	0.7017	2
0.3075	3	0.5540	2	0.7271	2
0.3203	1	0.5978	2	0.7277	1

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

Arriving Time	Service Unit	Arriving Time	Service Unit
0.7518	2	1.1117	2
0.8386	3	1.1188	3
0.8727	3	1.2830	2
0.8903	2	1.3949	2
0.9359	1	1.4153	1
0.9987	2	1.6881	2
1.0007	3	2.2034	2
1.0435	2	2.2652	1

(2) As time interval $\tau = 1$, count up the numbers of each service units arrived during each time interval. The detail is shown in Table 3.

Table 3. New arrival service unit requests per time interval.

Time Interval	S1	S2	S3
1	7	15	8
2	1	5	2
3	1	1	0

(3) Count up the numbers of randomly transformed service units during each time interval. The detail is shown in Table 4.

Table 4. Transformed service unit requests per time interval.

First time interval	S1	S2	S3	S4
From S1 to	0	2	4	1
From S2 to	4	1	9	1
From S3 to	1	3	3	1
Second time interval	**S1**	**S2**	**S3**	**S4**
From S1 to	3	1	1	1
From S2 to	3	3	3	2
From S3 to	6	3	6	3

(4) Count up the total numbers of service units during each time interval.

Table 5. Total service unit requests per time interval.

Time Interval	S1	S2	S3
1	7	15	8
2	6	11	18
3	13	8	10

The figures of four steps are shown in corresponding figure1-4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 October 2017 doi:10.20944/preprints201710.0051.v1

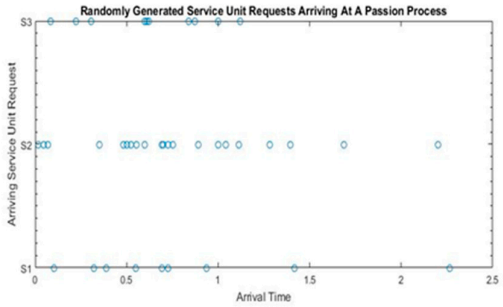

Figure 1. Randomly generated requests arriving at passion process with λ=3.

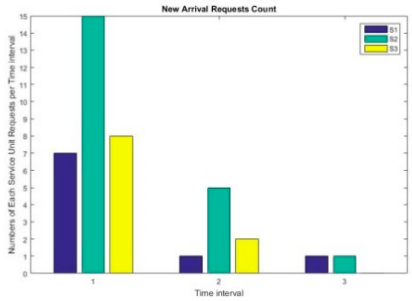

Figure 2. New arrival requests.

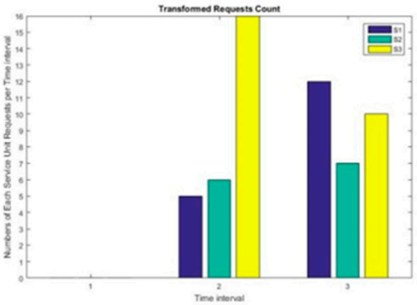

Figure 3. Transformed requests.

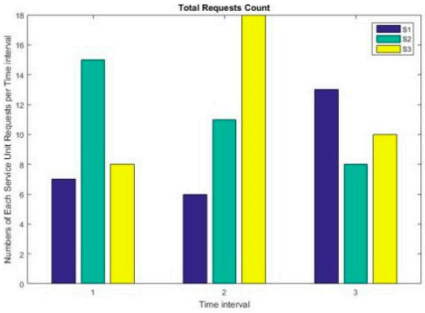

Figure 4. Total requests.

5. Conclusions

In this paper, we proposed a projection methodology from a hierarchical workload model to the numbers of each basic service units in any time periods. When the execution-time and the scaling functions of each service unit on a VM are known, the needed infrastructure resources can be deduced by the numbers of each service units. The resource prediction model is general and flexible because the workload model has the advantages:

(1) Generalization:

The workload model provides a generalization of both applications and workload variations. Different types of cloud applications are supported, for example, web applications, MapReduce applications. Since two main characteristics of workload variations are captured: the numbers and the mix, diverse workloads can be specified by parameterizing the workload model.

(2) Flexibility:

Application definitions are departed from workload variations. That means, a cloud user can pick up any possible workload patterns to test any their applications. Moreover, the workload model enables cloud users to modify the values of parameters in order to generate a workload fitting a certain situation. Furthermore, the multiple workloads generated by the workload model can be superposed. For example, a diurnal pattern workload can superpose a burst pattern workload.

In future, we put emphasis on refining the model of the relationship between the numbers of requests and the needed infrastructure resources. For that, we will design a set of experiments to measure different applications on different VMs.

Acknowledgement: This work is supported by National Natural Science Foundation of China [No.61662054 and No.61262082], and Inner Mongolia Technological Innovation Team Project "Cloud Computing and Software Engineering". Thanks the students Jiajia Xu, Ruiqing Yan and Jing Yang for the help with the experiments.

References

1. RightScale. Cloud Computing Trends: 2016 State of the Cloud Survey from RightScale, **2016**.
2. Milenkoski, a, Iosup, a, Kounev, S., Sachs, K., Ding, J., & Rosenberg, F.; Cloud Usage Patterns®: A Formalism for Description of Cloud Usage Scenarios. *Tech Report SPEC-RG-2013-001 v1.0.1, SPEC Research Group*, **2013**, 12–13.
3. Alexandru Iosup, Radu Prodan; IaaS Cloud Benchmarking®: Approaches, Challenges, and Experience. *MTAGS*, **2012**.
4. RUBiS. **2013**.
5. Huang, S., Huang, J., Dai, J., Xie, T., & Huang; The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. *2010 IEEE 26th International Conference on Data Engineering Workshops, ICDEW*, **2010**, 41–51. https://doi.org/10.1109/ICDEW.2010.5452747.
6. Feitelson, D. G.; Workload modeling for computer systems performance evaluation, **2010**.
7. An, C., Zhou, J., Liu, S., & Geihs, K.; A multi-tenant hierarchical modeling for cloud computing workload. Intelligent Automation & Soft Computing, (May), 1–8. https://doi.org/10.1080/10798587.2016.1152774.
8. Bahga, A.; Synthetic Workload Generation for Cloud Computing Applications. Journal of Software Engineering and Applications, **2011**, 04(07), 396–410. https://doi.org/10.4236/jsea.2011.47046.
9. Beitch, A., Liu, B., & Yung, T.; Rain:A workload generation toolkit for cloud computing applicationsand Computer,**2010.**
10. Mao, M., & Humphrey, M.; Auto-scaling to minimize cost and meet application deadlines in cloud workflows. *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, **2011**, 1–12. https://doi.org/10.1145/2063384.2063449.
11. Mao, M., & Humphrey, M; Scaling and scheduling to maximize application performance within budget constraints in cloud workflows. *Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS*, **2013**, 67–78, https://doi.org/10.1109/IPDPS.2013.61.
12. Iosup, A., Sonmez, O., Anoep, S., & Epema, D.; The performance of bags-of-tasks in large-scale distributed systems. *In Proceedings of the 17th international symposium on High performance distributed computing - HPDC*, **2008**, (p. 97). New York, USA: ACM Press. https://doi.org/10.1145/1383422.1383435.
13. Oprescu, A., & Kielmann, T.; Bag-of-tasks scheduling under budget constraints. *Cloud Computing Technology and Science. IEEE Computer Society*, **2010**:351-359.
14. Ganapathi, A., Chen, Y., Fox, A., Katz, R., & Patterson, D.; Statistics-driven workload modeling for the cloud. *Proceedings - International Conference on Data Engineering*, **2010**, 87–92. https://doi.org/10.1109/ICDEW.2010.5452742.
15. Chen, Y., Ganapathi, A., Griffith, R., & Katz, R.; The case for evaluating mapreduce performance using workload suites. *IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems - Proceedings*, **2011**, 390–399. https://doi.org/10.1109/MASCOTS.2011.12.

16. RightScale. IT AS A CLOUD SERVICES BROKER®: Provide Self-Service Access to Cloud, **2014**.
17. Gillam, L., Li, B., O¿Loughlin, J., & Tomar, A. P.; Fair Benchmarking for Cloud Computing systems. *Journal of Cloud Computing: Advances, Systems and Applications*, **2013**, 2(1). https://doi.org/10.1186/2192-113X-2-6.
18. Huang, S., Huang, J., Liu, Y., Yi, L., & Dai, J.; HiBench: A Representative and Comprehensive Hadoop Benchmark Suite. *ICDE Workshops*, **2010.** 192.198.165.138. Retrieved from http://192.198.165.138/sites/default/files/blog/329037/hibench-wbdb2012-updated.pdf.
19. Moschetta, J., & Casale, G; OFBench.: An enterprise application benchmark for cloud resource management studies. *Proceedings - 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC*, **2012**, 393–399. https://doi.org/10.1109/SYNASC.2012.39.
20. TPC-W Web Commerce Benchmark, **2006**.
21. Turner, A., Fox, A., Payne, J., & Kim, H. S.; C-MART: Benchmarking the cloud. *IEEE Transactions on Parallel and Distributed Systems*, **2013**, 24(6), 1256–1266. https://doi.org/10.1109/TPDS.2012.335.
22. Chen, Y., Alspaugh, S., & Katz, R.; Interactive analytical processing in big data systems: a cross-industry study of MapReduce workloads. *Proceedings of the VLDB Endowment*, **2012**, 5(12), 1802–1813.https://doi.org/10.14778/2367502.2367519.
23. Gong, Z., & Gu, X.; PAC: Pattern-driven application consolidation for efficient cloud computing. *Proceedings - 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, **2010**, 24–33. https://doi.org/10.1109/MASCOTS.2010.12.
24. Moreno, I. S., Garraghan, P., Townend, P., & Xu, J.; Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud. *IEEE Transactions on Cloud Computing*, **2014**, 2(2), 208–221. https://doi.org/10.1109/TCC.2014.2314661.
25. Li, H.; Realistic workload modeling and its performance impacts in large-scale escience grids. *IEEE Transactions on Parallel and Distributed Systems*, **2009**, 21(4), 480–493. https://doi.org/10.1109/TPDS.2009.99.
26. Bodik, P., Fox, A., Franklin, M. J., Jordan, M. I., & Patterson, D. a.; Characterizing, modeling, and generating workload spikes for stateful services. *Proceedings of the 1st ACM Symposium on Cloud Computing - SoCC*, **2010**, d, 241. https://doi.org/10.1145/1807128.1807166.
27. Ali-eldin, A., Tordsson, J., Elmroth, E., & Kihl, M.; Workload Classification for Efficient Auto-Scaling of Cloud Resources, *Computer Systems*, **2013**, 1–36.
28. Juan, D. C., Li, L., Peng, H. K., Marculescu, D., & Faloutsos.; Beyond poisson: Modeling inter-arrival time of requests in a datacenter. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), **2014**, 8444 LNAI (PART 2), 198–209. https://doi.org/10.1007/978-3-319-06605-9_17.
29. Vakilinia, S., Ali, M. M., & Qiu, D.; Modeling of the resource allocation in cloud computing centers. *Computer Networks*, **2015**, 91, 453–470.,https://doi.org/10.1016/j.comnet.2015.08.030.
30. Calheiros, R. N., Ranjan, R., De Rose, C. A. F., & Buyya, R.; CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. **2009**, arXiv Preprint arXiv: 0903.2525, 9. Retrieved from http://arxiv.org/abs/0903.2525.
31. Wu, L: SLA-based Resource Provisioning for Management of Cloud-based Software-as-a-Service Applications. PhD Thesis, University of Melbourne, **2014**, March.
32. Tsai, J.-T., Fang, J.-C., & Chou, J.-H.; Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research*, **2013**, 40(12), 3045–3055. https://doi.org/10.1016/j.cor.2013.06.012.
33. Gandhi, A., Dube, P., Karve, A., Kochut, A., & Zhang, L.; Modeling the impact of workload on cloud resource scaling. *Proceedings - Symposium on Computer Architecture and High Performance Computing*, **2014**, 310–317. https://doi.org/10.1109/SBAC-PAD.2014.16.