A Systematic Approach to Adaptive Dimensioning of Data Centers

Wenhong Tian[a]
[a] School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu 611731, China
Email: tian_wenhong@uestc.edu.cn

Abstract— **Cloud data centers (CDCs) provide key infrastructure for Cloud computing. Accurately allocating computing resources is very important for the CDCs to function efficiently. Current allocation of resources in CDCs is mostly dedicated and static. However, workloads for Cloud applications are highly variable which cause poor application performance, poor resource utilization or both. In this paper, considering both blocking and delay probability by applying queuing theories to model different scenarios, systematic adaptive dimensioning methods for CDCs are developed so that right amount of computing resources are allocated for variable workloads to meet quality of service requirements in different situations. Both single CDC and multiple CDCs with and without delay are considered. The proposed methods can be applied to dimension data centers efficiently and dynamically.**

Keywords: Cloud computing; Cloud Data Centers (CDCs); Adaptive Dimensioning Methods.

I. INTRODUCTION

Cloud Computing refers to the applications delivered as services over the Internet, the hardware and systems software in the data centers that provide those services [1] [3]. Cloud data centers (CDCs) consist of both clustering of servers and networking infrastructure. Driven by the economics of large number of low-cost servers with clustering, the reliability and availability of the Cloud system as a whole are improved significantly in distributed computing. Current CDCs, for instance, Google's, Amazon.com's, and Microsoft's, now being built to contain more than 100,000 servers. Google [2] introduces one of CDC architecture: which consists of multiple clusters distributed worldwide. Each cluster has around a few thousand machines, and the geographically distributed setup protects the system against catastrophic data center failures (like those arising from earthquakes and large-scale power failures). The servers on each side of a rack interconnect via a 100-Mbps Ethernet switch that has one or two gigabit uplinks to a core gigabit switch which connects all racks together. In Amazon.com [7], the virtual machine providing the equivalent of a system with a 1.7Ghz x86 processor, 1.75GB of RAM, 160GB of local disk, and 250Mb/s of network bandwidth, is called an computing "instance", other configuration of virtual machines with different computing powers are called different instances. Measuring computing power in a Cloud is introduced in [17] as Cloud equivalence. In this paper, computing resources of the CDCs are considered such as for virtual computing lab [26] and other research applications. Assuming that an application each time requests one or more computing instances such as in VCL [26] [27] and Amazon.com's EC2 [31], this is called full server utility model [19]. Gu et al. [9] present a scheduling strategy on load balancing of VM resources based on genetic algorithm, it claims to achieve load balancing and reduce or avoid dynamic migration according to historical data and current state of the system. Xu et al. [28] develop heterogeneous computing resources tool called HCCloud to help use computing resources in a more efficient, scalable and flexible way. You et al. [29] design an automatic resource allocation strategy based on market Mechanism (ARAS-M), implemented on Xen, and experiment results show that ARAS-M can approximately achieve the equi- librium state between demand and supply. Tian [24] proposes three ways to improve the efficiency of virtual cloud lab by applying queueing model. Tian [25] provides preliminary results of related adaptive dimensioning methods for Cloud data centers; we extend it in this paper. The traffic of the customer requests is highly variable depending on the time of the day. Figure 1 from [19] shows a log-transform results of requests arrival rates from a trace. One example is given in [1] as follows: "When Animoto online made its service available via Facebook, it experienced a demand surge that resulted in growing from 50 servers to 3500 servers in three days. Even if the average utilization of each server was low, no one could have foreseen that resource needs would suddenly double every 12 hours for 3 days. After the peak subsided, traffic fell to a level that was well below the peak. " Overprovisioning and underprovisioning are two coexisting extremes in provisioning data centers. Taking SETI@home project [12] [30] as another example, there are six largest time zones in terms of hosts. Hosts are number of servers or machines for customers applications. These time zones in corresponding to Central Europe (17,000 hosts), Eastern North America (11,003 hosts), Central North America (6,077 hosts)), Western North America (4,900 hosts), Western Europe

Manuscript received Feb. 12, 2013; revised June 14, 2013; accepted July 4, 2013.
This work is supported the National Natural Science Foundation of China (NSFC) Grant 61150110486.

doi:10.4304/jcp.9.2.266-274

(4280 hosts), and Eastern Asia (2396 hosts). New hosts added per day and average new hosts per day are shown in Figure 2 and Figure 3 respectively. It can be seen that customer requests (hosts) are highly variable. We need to develop adaptive dimensioning methods so that right amount of computing resources can be allocated for variable workload to avoid overprovisioning or under-provisioning. An architecture and server migration

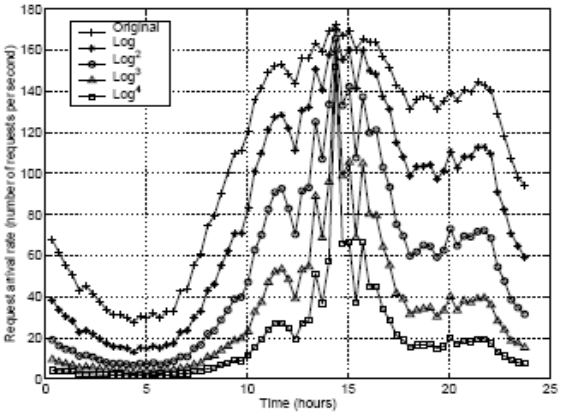

Figure 1. An Example of Requests Arrival Rate from [19]

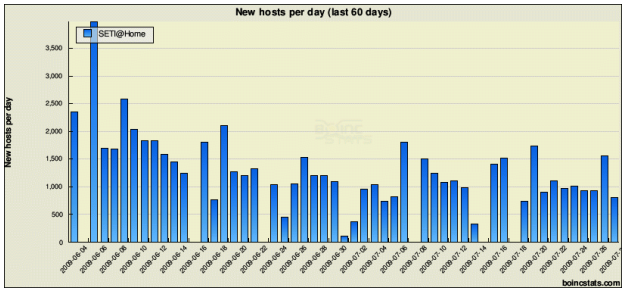

Figure 2. New Hosts For SETI@home [30]

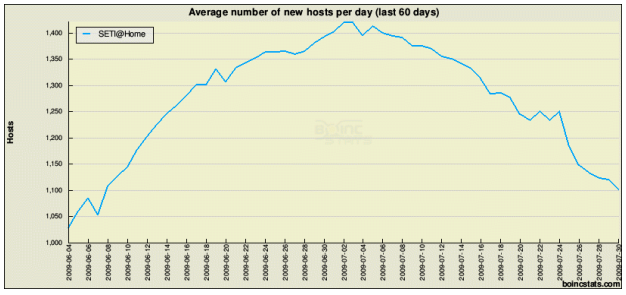

Figure 3. Average New Hosts For SETI@home [30]

approaches have been introduced in [19] where both online and offline server migration strategies are discussed and evaluated. Liu et al. [18] propose an architecture of self-adaptive configuration optimization system which supports dynamic reconfiguration when workloads change using genetic algorithm. An integer linear programming

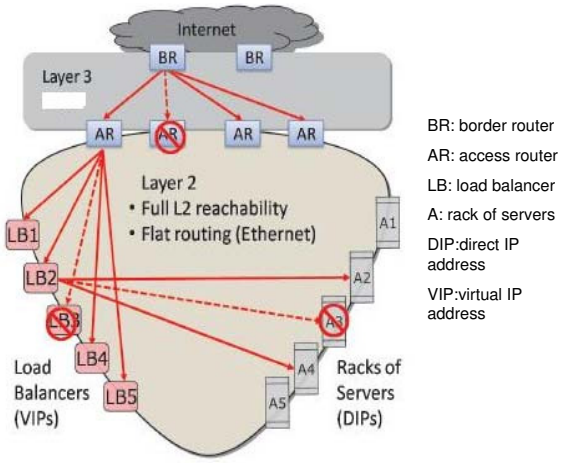

Figure 4. A New Architecture for Data Center [10]

(ILP) approach is provided in [6] to solve the resilient grid/cloud dimensioning problem using failure-dependent backup routes. Next generation architecture for data centers is proposed in [10] and reprinted in Figure 4. In the architecture, layer 2 networks connect all the servers inside a data center and requests are distributed over pools of servers. To increase the total capacity of the CDCs, three layered structure of internetworking is proposed in [10] and reprinted in Figure 5. There are n_1 = 144

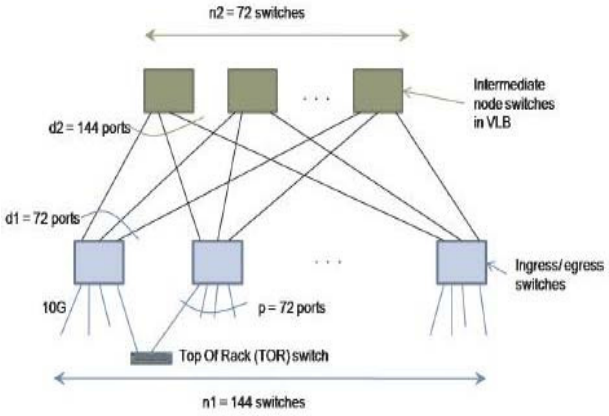

Figure 5. Three layered structure for Data Center [10]

ingress-egress switches, shown in light gray. Each ingress-egress switch connects, through a 10-Gbps port , to every intermediate switch, of which there are n_2 = 72, shown in dark gray. Each 10G bps port can connect 10 servers with 1Gbps bandwidth. It can be computed that the total number of servers 144x72x10=103,680 (or 1,036,800 if each server's bandwidth is 100Mbps). All the servers in the CDCs can be automatically started up or shut down using power concentrators. For this kind of fully connected meshed architecture in a CDC, it is reasonable to model the system as a multi-server queue. Current

CDC architectures are manually configured and cannot automatically adapt to time-varying workloads, they result in poor resource utilization when workload is low (overprovisioning) or significant performance degradation when loads exceed capacity (under-provisioning) as discussed in [19]. To overcome overprovisioning and under-provisioning problems, adaptive dimensioning methods are developed in this paper. The paper is organized as follows. In section II, methods for adaptive dimensioning of single Cloud data center are proposed. Method for multiple federated Cloud data centers is discussed in section III. Finally, the conclusions are given in section IV.

II. Single Cloud Data Center (Cloud)

In this section, models and dimensioning methods for single Cloud data center are discussed on the basis of blocking and non-blocking models.

A. Dimensioning with Blocking Models

Assuming that individual requests constituting the workload are independent to each other; tasks are scheduled in a first-in-first-out manner without preemption; but the incoming requests that find all servers busy are blocked and depart from the system. Erlang loss model can be applied in this case.

1) Erlang Loss Model with Poisson Arrivals for Single Class of Customers (requests): In this case, a CDC with server cluster can be modeled as a M/G/C/C queue, i.e., requests arrivals follow Poisson process and service time distribution can be general and there are total C servers in a CDC. Provisioning optimal total number of servers is one of practical ways to meet the blocking probability and other QoS requirements. In this section, we describe how to calculate the minimum number of servers C of the blocking model so that the maximum blocking probability is less than a pre-specified value ϵ for a given load. Erlang loss formula is given by:

$$B(N,\rho) = \frac{\rho^N/N!}{\sum_{i=0}^{N}\rho^i/i!} \tag{1}$$

where ρ is the offered load to the system, for example measured by the number of requests per seconds. This minimum value of C can be calculated iteratively using equation (1). However, when the required capacity C is very large, this iterative approach becomes CPU intensive since its time complexity is $O(log2(N/\epsilon)N)$. A recursive formula for equation (1) is as follow:

$$B(N,\rho) = \frac{\rho B(N-1,\rho)}{N+1+\rho B(N-1,\rho)}, B(0,\rho) = 1. \tag{2}$$

It is a long-standing conjecture that the optimal number of servers is of the form ρ+$K\sqrt{\rho}$ for Erlang loss model where K is a constant depending on the offered load and blocking probability. This approximation yields very accurate results. Indeed, based on extensive sensitivity tests, the actual optimum and approximate values rarely deviate by more than one server, or by more than one percent, whichever is greater. In this paper, single class traffic is considered for a CDC. Then asymptotic expression for the optimum value of C is obtained as follow:

$$N = \rho + \psi(\epsilon\sqrt{\rho})\sqrt{\rho} \tag{3}$$

where ϵ is the blocking probability requirement, ρ is the offered load (workload) to the system and $\psi(x)$ is the unique solution of the following differential equation

$$\psi'(x) = \frac{-1}{(\psi(x)+x)x)}, \psi(\sqrt{2/\pi}) = 0 \tag{4}$$

In [22], equation (4) is solved to obtain

$$x^{-1}e^{-0.5\psi(x)^2} - \sqrt{2\pi}erf(0.5^{1/2}\psi(x)) - x\sqrt{0.5\pi} = 0 \tag{5}$$

where $erf(.)$ function is defined as follow:

$$erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x exp(-t^2)dt \tag{6}$$

Given x, equation (4) can be easily solved numerically for $\psi(x)$. Applied to equation (3), the requested total number of servers can be obtained.Table I shows the minimum required number of servers for a Erlang loss queue, so that the blocking probability ϵ is less than 0.01. The offered load ρ was varied. For each value of ρ, the minimum required servers is computed using equation (3) (labelled as 'Asm') and also using Erlang loss formula (exact solution labeled as 'Exa') which can be obtained using equation (2) iteratively. Through many numerical examples, that the minimum capacity C obtained using equation (3) is observed to be very closed to the exact solution.

TABLE I.
THE REQUIRED NUMBER OF SERVERS C VS. OFFERED LOAD (ρ)

ρ	Method	C	ρ	Method	C
0.14	Exa	2	100	Exa	118
0.14	Asm	2	100	Asm	118
1	Exa	5	200	Exa	222
1	Asm	5	200	Asm	222
3	Exa	8	500	Exa	527
3	Asm	8	500	Asm	527
10	Exa	18	1000	Exa	1030
10	Asm	18	1000	Asm	1030
40	Exa	53	2000	Exa	2030
40	Asm	53	2000	Asm	2030

2) Erlang Loss Model with Poisson Arrivals for Multiple Classes of Customers: Let us assume that the system has a total C identical servers, and each can provide service to any class of arrivals. Let n=$(n_1, n_2, ..., n_R)$ where n_r is the number of class r customers in the system, and let b=$(b_1, b_2, ..., b_R)$. The total number of busy servers in state C is

$$bn^T = b_1n_1 + b_2n_2 + ... + b_Rn_R. \tag{7}$$

The set of all possible states of the system can be described as

$$S^b = \{n : bn^T \leq C\}. \tag{8}$$

It is well known that the multi-class Erlang loss system has a product-form solution.

$$P(n) = \prod_{i=1}^{R} \frac{\rho_i^{n_i}}{n_i!} G^{-1}(\Omega), \forall n \in \Omega \quad (9)$$

where

$$G(\Omega) = \sum_{n \in \Omega} \prod_{i=1}^{R} \frac{\rho_i^{n_i}}{n_i!} \quad (10)$$

The class-i has offered load $\rho_i = \lambda_i/\mu_i$. The challenge in this model is to obtain the blocking probability for each class. Computing the blocking probabilities by directly enumerating all possible states of the system requires an $O(C^R)$ amount of time. The direct method is computationally cumbersome and grows exponentially fast even for relatively small systems. Several methods have been presented in the literature to avoid the exponential complexity of the computations. One of the most powerful methods for obtaining the blocking probabilities was published independently by Kaufman (1981) [14] and Roberts (1981) [20]. The Kaufman-Roberts method is a recursive algorithm that has a linear complexity, $O(CR)$, and it is considered as a fast method. The recursive formula is as follows:

$$w(K) = \frac{1}{K} \sum_{r=1}^{R} \rho_r b_r w(K - b_r), K = 1, 2, ..., C. \quad (11)$$

where $w(x)$=0 if $x < 0$, $w(0)$=1 and $\rho_r = \lambda_r/\mu_r$. Then, the blocking probability of class r arrivals is given by:

$$BP_r = \frac{\sum_{j=C-(b_r-1)}^{C} w(j)}{\sum_{j=0}^{C} w(j)}, r = 1, 2, ..., R. \quad (12)$$

It is interesting to know that this formula can be applied to the single class model, as a fast way of obtaining the blocking probability. Given the blocking probabilities, the average number of class r customers in the system is

$$E[Q_r] = \rho_r(1 - BP_r), r = 1, 2, ..., R. \quad (13)$$

3) Blocking Model with General Arrival Processes for Single Class of Customers: $M_t/G/N/N$ queue model can be applied for general arrivals which may be time varying. A solution similar to Erlang loss model is provided in [11]:

$$N = b\rho_t + \psi(\frac{\epsilon}{b}\sqrt{b^2\rho_t})\sqrt{b^2\rho_t} \quad (14)$$

where time-varying ρ_t is the offered load to the system and b is the bandwidth (servers) required by each request. A few examples are also introduced in [11]. Let us take one example from [11], arrival rate $\lambda(t)$=40+10sin(2πt/80), requesting b=5 units of bandwidth (similar to servers in CDCs) and desiring no more than 1 percent blocking. Taking every 10 minutes as a measurement and provisioning interval. Figure 6 shows the dimensioning results for eight periods of measurement. Decisions of adaptive provisioning are made at the end of each measurement interval to meet the blocking probability of requirement.

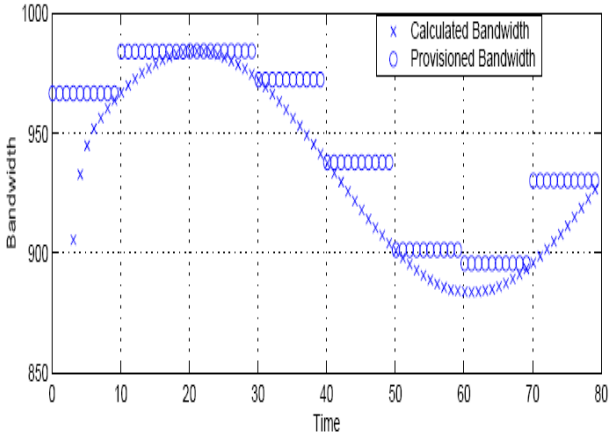

Figure 6. Eight Periods Provisioning Example from [11]

4) Blocking Model with General Arrival Processes for Multiple Classes of Customers: Similar to Poisson arrival processes for multiple classes of customers, the total number of servers in general arrival case can be dimensioned using following equation:

$$C = \sum_{i=1}^{R} b_i\rho_i(t) + \psi(\min_{1 \le i \le R} \frac{\epsilon_i}{b_i} \sqrt{\sum_{i=1}^{R} b_i\rho_i(t)}) \sqrt{\sum_{i=1}^{R} b_i\rho_i(t)} \quad (15)$$

where time-varying $\rho_i(t)$ is the offered load to class-i, ϵ_i and b_i is respectively the blocking probability and the number of servers requests for class-i.

B. Dimensioning with Nonblocking Models

Assuming that individual requests constituting the workload are independent to each other; tasks are scheduled in a first-in-first-out manner without preemption; there are a large number of servers so that requests rarely incur blocking or non-blocking. Then delay models can be applied in this case.

1) Erlang Delay Model: Erlang delay model $M/M/C$ can be applied in this case, where each server represents a server in a CDC, requests who come and find all servers busy will be put into an infinite queue to wait before being served. For example, if the total number of servers in a CDC is 256, then the Erlang delay model will consist of 256 servers. A customer in this queuing system represents an on-demand request for computing resources, those customers who find all servers are busy when coming will wait in a queue as long as necessary for a server to become available. In this model, it is well known that the probability of delay (the fraction of customers who will find all C servers busy and must wait in the queue) is given by the famous Erlang Delay (or Erlang C) formula as shown in equation (16) where C is the number of servers and ρ=λ τ is the total offered load in erlangs,

λ is the arrival rate and τ is the average service time:

$$D(C,A) = \frac{\frac{A^C}{C!(1-A)}}{\sum_{k=0}^{C-1}\frac{A^k}{k!} + \frac{A^C}{C!(1-A)}} \quad (16)$$

where A=ρ/C if $\rho \leq C$; A=1 if ρ=1. And for large C, computing $D(C,A)$ directly is very expensive or can cause computer overflow. Notice that there is a recursive formula for Erlang B and a recursive formula based on Erlang B (equation (2)) can be used for Erlang C as follows:

$$D(C,A) = \frac{B(C,A)}{1 - A(1 - B(C,A))} \quad (17)$$

It is obvious that the larger C is, the smaller delay will be. However, we cannot design the CDC to have as many servers as possible because of finance and space limitation. We can compute a few examples and find that D(0.9, 1)=0.9, D(9, 10)=0.67, D(90, 100)=0.22 and D(900, 1000)=0.0006. In all above cases, the server utilization is 0.9. It shows that large systems are more efficient than small ones. The following equation can be used to predict how long the customers (requests) will have to wait:

$$w = D(C,A)\frac{1}{1-A}\frac{\tau}{C} \quad (18)$$

This holds true even when service is not first come first service (FIFO), for example, LIFO (Last in First out) or service in random order. This is because that interchanging statistically identical customers waiting in the queue does not change the number of customers or the amount of work waiting to be served, see Cooper [5] for more detailed explanations. Tian and Perros [23] introduced one way to dimension the data center with job priorities and QoS constraints using single server queuing model. Based on Erlang delay model itself, it is possible to dimension the system to meet QoS requirements. Jennings et al [13] introduced the following formula for dimensioning Erlang delay model:

$$C = \rho + z_\alpha\sqrt{\rho} \quad (19)$$

where ρ is the mean offered load to the system. After some computation and simplification, z_α satisfies

$$Pr(N(0,1) > z_\alpha) = \alpha = \frac{1}{\sqrt{2\pi}}\int_{z_\alpha}^{\infty} exp(-u^2/2)du \quad (20)$$

where $N(0,1)$ is the standard normal distribution and α is the delay probability of the system. Given α, z_α can be computed explicitly so that the required number of servers C can be found easily from equation (19). It is shown in [13] that using equation (19) for dimensioning of Erlang delay model can meet given delay requirement with appropriate number of servers. Equation (19) and (20) can be easily used to find the appropriate number of servers. In Table II, the probability of delay as a function of the offered load in the Erlang delay queue M/M/C is shown, where C is chosen to satisfy Equation (20) with α=0.005 (z_α=2.576). $C(Asm)$ is obtained using equation (19) while $C(Ite)$ is obtained applying equation (16) iteratively.

TABLE II.
THE REQUIRED NUMBER OF SERVERS C VS. OFFERED LOAD (ρ)

ρ	$\rho + z_\alpha\sqrt{\rho}$	C(Asm)	C(Ite)
1	4.1	5	5
10	18.6	20	21
100	126.3	127	128
200	236.9	237	239
500	558.1	559	560
1000	1082.1	1083	1084
2000	2115.7	2116	2119

2) Nonblocking model with General Arrivals and Services: In this case, requests arrival process and service process can be general.

(1) Considering delay probability. In this case the total number of servers can be found using the following equation:

$$C = \rho_t + z_\alpha\sqrt{\rho_t} \quad (21)$$

where ρ_t is the mean time-varying offered load to the system.

(2) Considering response time. A CDC with server cluster can be modelled as a G/C/C system which has a mean response time R under heavy traffic given by [16]

$$R = \frac{1}{\bar{\mu}} + \frac{\sigma_a^2 + \frac{\sigma_b^2}{C^2}}{2t(1-U)} \quad (22)$$

where σ_a^2 and σ_b^2 are the variance of inter-arrival and service times, respectively, and t is the mean inter-arrival time, $\bar{\mu}$ is the mean service rate and U is the average CPU utilization in the previous measurement interval. U can be obtained by taking average over all servers' CPU utilization. It can be seen that the number of servers C is related to six variables (R,U,$\bar{\mu}$,t,σ_a^2,σ_b^2). Solving the equation for C, we have

$$C = \sqrt{\frac{\sigma_b^2}{2t(1-U)(R - \frac{1}{\bar{\mu}})) - \sigma_a^2}} \quad (23)$$

Given (R, U, $\bar{\mu}$, t, σ_a^2, σ_b^2), the required number of servers for the next interval can be computed using above equation. Normally, the quality of service (QoS) requirements may include U and R. And $\bar{\mu}$,t,σ_a^2, σ_b^2 can be measured. For example, given U=0.5, t=0.2, σ_a^2=0.2 and σ_b^2=100, the total number of required servers is changing with the required response time R, results are shown in Figure 7.

C. Optimized Capacity Provisioning and Bounds

For the given arrival rate and blocking probability requirement of each class, we may use an iterative approach (based on fixed point algorithm) to optimize the total capacity. We are to find the number of total servers required for a CDC to meet the acceptable level of blocking (known as grade-of-service). The optimization problem is to find a minimum total number C for given offered load and acceptable blocking

$$\epsilon : min\{C | max\ blocking\ probability \leq \epsilon\}. \quad (24)$$

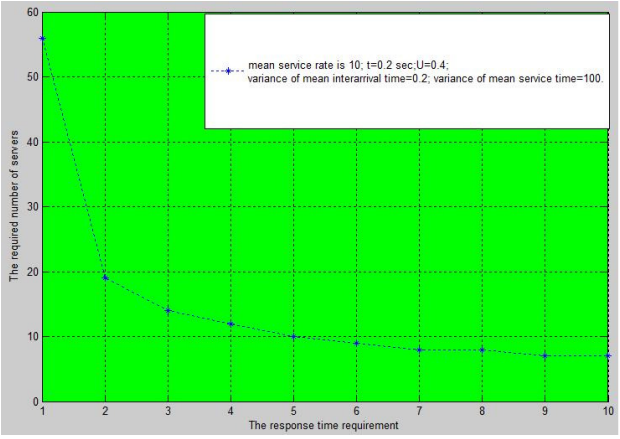

Figure 7. A Dimensioning Example for Non-blocking Model

When needed capacity is very large, iterative approach based on fixed point algorithm will be expensive with time complexity at least $O(CR)$. It is a long-standing conjecture the optimal number of bandwidth (servers) is of form servers=$\rho+K\sqrt{\rho}$ for single class traffic where K is a constant depending on the offered load and blocking probability. In Grassmann [8], he stated that "Extensive sensitivity testing revealed that this approximation yields very accurate results. Indeed, the actual optimum and the approximation rarely deviate by more than one server, or by more than one percent, whichever is greater." For multiclass traffic, similar result was introduced by Hampshire et al. in [10]:

$$C=\sum_{i=1}^{R}b_i\rho_i+\psi(\min_{1\le i\le R}\frac{\epsilon_i}{b_i}\sqrt{\sum_{i=1}^{R}b_i\rho_i})\sqrt{\sum_{i=1}^{R}b_i\rho_i} \quad (25)$$

where ϵ_i is the grade of service (blocking probability) requirement for class-i traffic and $\psi(x)$ is the unique solution to the equation (5). Given x, equation (5) can be solved easily with complexity $O(1)$, i.e., we can get $\psi(x)$ easily given x. Applied to equation (25), we obtain the request total capacity. Because of the asymptotic rule, satisfying the requirements provides more than enough servers for all the other classes. Through many numerical examples, we observed that the total servers obtained using equation (25) is very closed to exact solution. This approach is a very accurate provisioning solution with complexity $O(1)$. Also from Ross [20], we know that the blocking probability of each class is proportional to its servers (bandwidth) requirement when total capacity is very large, i.e.,

$$BP_k\approx b_k\alpha, k=1,2,\ldots,R \quad (26)$$

where α is a constant for given total capacity and offered loads. If we know Qos requirement of the dominant Qos classes, we can know the blocking probabilities of other classes, so we can find total arrival rates for each class easily and then use equation(25) for provisioning. Observing that the asymptotic rule of thumb and the blocking probabilities relationship among different classes, we can find the blocking probability of the dominant Qos classes by inverting the equation (25) as follows:

$$\epsilon_i=\frac{b_i}{\delta\int_0^\infty e^{-0.5t^2+(C-m)t/\delta}dt} \quad (27)$$

where m (called mean of offered load) and δ (called standard deviation of offered load) are defined as:

$$m=\sum_{i=1}^{R}b_i\rho_i;\delta=\sqrt{\sum_{i=1}^{R}b_i^2\rho_i} \quad (28)$$

Equation (27) and (28) can help us find the tight bounds of blocking probabilities given other parameters.

TABLE III.
8 TYPES OF VIRTUAL MACHINES (VMS) IN AMAZON EC2

MEM	CPU (units)	Sto	VM
1.7	1 (1 cores × 1 units)	160	1-1(1)
7.5	4 (2 cores × 2 units)	850	1-2(2)
15.0	8 (4 cores × 2 units)	1690	1-3(3)
17.1	6.5 (2 cores × 3.25 units)	420	2-1(4)
34.2	13 (4 cores × 3.25 units)	850	2-2(5)
68.4	26 (8 cores × 3.25 units)	1690	2-3(6)
1.7	5 (2 cores × 2.5 units)	350	3-1(7)
7.0	20 (8 cores × 2.5 units)	1690	3-2(8)

TABLE IV.
3 TYPES OF PHYSICAL MACHINES (PMS) SUGGESTED

PM	CPU (units)	MEM	Type
1	16 (4 cores × 4 units)	160	1
2	52 (16 cores × 3.25 units)	850	2
3	40 (16 cores × 2.5 units)	1690	3

III. MODELING VIRTUAL MACHINES ALLOCATION

Taking the widely used example of Amazon EC2 [31], we show that a uniform view of different types of virtual machines (VMs) is possible. Table III shows eight types of virtual machines from Amazon EC2 online information. We can therefore form three types of different physical machines based on compute units. In real Cloud data center, for example, a physical machine (PM) with 2 × 68.4GB memory, 16 cores × 3.25 units, 2 × 1690GB storage can be provided. In this or similar way, and uniform view of different types of virtual machines is possibly formed. This kind of classification provides a uniform view of virtualized resources for heterogeneous virtualization platforms e.g., Xen, KVM, VMWare, etc., and brings great benefits for virtual machine management and allocation. Customers only need selecting suitable types of VMs based on their requirements. There are eight types of VMs in EC2 as shown in Table III, where MEM is for memory with unit GB and Sto is for hard disk storage with unit GB. Three types of PMs are considered for heterogeneous case as shown in Table IV. We model the virtual machine allocation on different types of physical machines as a multi-class Erlang loss model (MELM). In the MELM model (As descripted in Section II. A), the capacity of a basic VM request is considered as a server in service side. For example, VM1-1 is considered

TABLE V.
THE BLOCKING PROBABILITY COMPARISON FOR PM TYPE 1

ρ	Ana1	Ana2	Ana3	Sim1	Sim2	Sim3
0.1	0.004	0.005	0.021	0.004	0.005	0.021
0.2	0.013	0.018	0.068	0.013	0.018	0.068
0.3	0.025	0.038	0.128	0.025	0.038	0.128
0.4	0.036	0.062	0.190	0.036	0.062	0.190
0.5	0.046	0.088	0.252	0.046	0.088	0.252
0.6	0.055	0.116	0.309	0.055	0.116	0.309
0.7	0.063	0.143	0.363	0.063	0.143	0.362
0.8	0.069	0.170	0.411	0.069	0.170	0.412
0.9	0.075	0.196	0.455	0.075	0.196	0.455
1.0	0.079	0.222	0.495	0.079	0.221	0.495
2.0	0.107	0.420	0.735	0.107	0.423	0.745
3.0	0.144	0.542	0.838	0.147	0.562	0.852
4.0	0.189	0.624	0.892	0.198	0.612	0.886
5.0	0.229	0.683	0.925	0.235	0.713	0.919

TABLE VI.
THE BLOCKING PROBABILITY COMPARISON FOR PM TYPE 2

ρ	Ana1	Ana2	Ana3	Sim1	Sim2	Sim3
0.1	0.004	0.005	0.021	0.004	0.005	0.021
0.2	0.014	0.018	0.068	0.014	0.018	0.068
0.3	0.026	0.037	0.128	0.026	0.038	0.127
0.4	0.040	0.063	0.191	0.040	0.063	0.191
0.5	0.055	0.091	0.253	0.054	0.090	0.252
0.6	0.069	0.119	0.312	0.069	0.119	0.311
0.7	0.084	0.148	0.366	0.084	0.147	0.367
0.8	0.098	0.176	0.415	0.098	0.175	0.413
0.9	0.112	0.204	0.460	0.112	0.205	0.461
1.0	0.125	0.231	0.501	0.125	0.232	0.500
2.0	0.242	0.440	0.748	0.239	0.444	0.754
3.0	0.328	0.568	0.853	0.328	0.576	0.858
4.0	0.394	0.651	0.906	0.392	0.638	0.882
5.0	0.446	0.710	0.937	0.453	0.697	0.920

as a request of 1 server from PM type 1, VM1-2 is considered as a request of 4 servers from PM type 1 since its requested capacity (CPU, MEM, Sto) is 4 times of VM1-1. Similarly the number of servers (capacity) can be determined for other VM types. Firstly we show the blocking probability tested by both analytical (Ana) and simulation results (Sim) as shown in Table V and Table VI, where $\rho = \lambda/\mu$, is the average total offered load to each class. The total number of PM-1 and PM-2 is 30 respectively in this case. In Table V, Ana1 represents for theoretical results of the blocking probability for VM type 1-1 (1) obtained from equation (12), Sim1 is the simulated results for VM type1-1(1). Similarly Ana2 (Sim2) is for VM type 1-2 (2) and Ana3 (Sim3) is for VM type1-3 (3). In Table VI, Ana1 represents for theoretical results of the blocking probability for VM type 2-1(4) obtained from equation (12), Sim1 is the simulated results for VM type2-1 (4). Similarly Ana2 (Sim2) is for VM type 2-2 (5) and Ana3 (Sim3) is for VM type2-3 (6). Extensive numerical results show that theoretical and simulation results match very well. And for capacity provisioning, we can apply the method introduced in Section II.B. For PM type 3, similar results are obtained as for PM type 1 and 2, so that we omit.

IV. A MODEL FOR MULTIPLE FEDERATED CLOUD DATA CENTERS (CLOUDS)

As introduced in previous sections, each CDC may be modeled by a multi-server queue. There may be many CDCs distributed around world for a company such as Google, Microsoft and SET@HOME. As the example given in SET@HOME, there are six largest centers around the world. When the traffic comes to the data centers, it's distributed based on the load balancer. The probability of portion of traffic is assigned to CDC-i is considered as α_i. In reality, the load balancer distributes the incoming traffic based on the physical configuration of each CDC and dynamic traffic loading in each CDC. A model for multiple federated CDCs is shown in Figure 8 where each CDC is modeled by a multiple server queue as introduced in previous sections. "With/no buffer" refers to two cases with or without buffering. "Federated" refers to all CDCs are connected to a management node by WAN or LAN and incoming workloads are distributed among all CDCs using load balancer. A shared server pool is provided so that more servers can be added to a CDC when workload is higher than pre-specified threshold or some servers can be moved from a CDC to shared server pool when workload is lower than pre-specified threshold. The performance

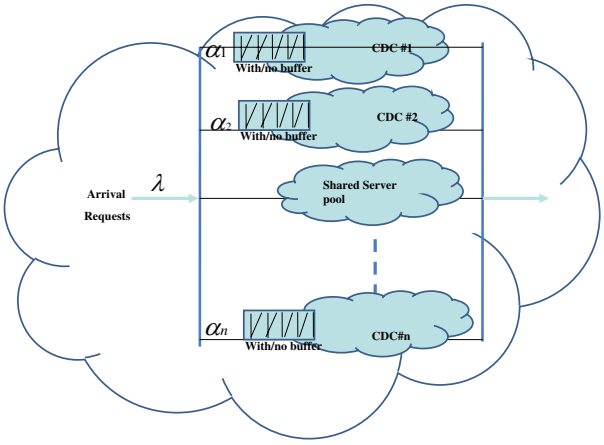

Figure 8. A Model for Multiple Federated Cloud Data Centers

of the multiple CDCs depends on the configuration of the each CDC. Methods introduced in previous section can be applied. For example, let us assume that there are three CDCs and a shared server pool. Using certain time as a measurement interval, for instance, 10 minutes, the expected total time spent in each CDC is about 1 minute. During the 10 minutes interval, if CDC-1 experiences high workload and the expected total time spent in it is more than 1 minute, we can use the methods introduced in section II to find right number of total servers and allocate additional number of servers from shared server pool to CDC-1. Similarly if during another 10 minutes, CDC-2 experiences lower workload than expected and there are many idle servers in it, then some number

of servers can be moved to shared server pool after calculation using methods in previous sections. Adaptive dimensioning process can be summarized as follows:

Summary of Adaptive Dimensioning Process

set ϵ for blocking probability or β for delay probability

step 1: choose the appropriate model from section II

step 2: compute the total number of required servers in a CDC for current interval

step 3: compare to the number of servers currently allocated to a CDC

step 4: make a decision to add more servers to a CDC or move some servers from a CDC to the shared server pool

step 5: update and take action to allocate or de-allocate servers

Another benefit of using federated (compared to isolated) multiple CDCs, is that average total time spent in the system can be reduced. This because that load balancer can distribute workloads based on the current performance of each CDC using measurement in previous intervals and adaptively adjusts the workload distribution among all CDCs. One example is shown in [4], where three federated CDCs are considered. It showed that the average time spent in the system is reduced by more than 50% [4].

V. Conclusion

In this paper, blocking and non-blocking models for Cloud data centers are introduced for dimensioning under highly variable workloads. Adaptive dimensioning methods with numerical examples for each case are provided so that right amount of computing resources is allocated for variable workloads to meet quality of service requirements. In the future, extension to shared server utility model will be considered where all services are run concurrently on all servers in a cluster of a CDC and optimization method is used to determine the fraction of CPU resources allocated to each service on each server. Also obtaining more simulation results on federated CDCs is under study.

Acknowledgment

The authors are grateful to the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

References

[1] M.Armbrust, A.Fox, R. Griffith,A.D. Joseph,R.H. Katz,A.Konwinski,G.Lee,D.A.Patterson, A.Rabkin,I.Stoica, M.Zaharia, Above the Clouds: A Berkeley View of Cloud Computing, Technical Report No. UCB/EECS-2009-28.

[2] L.A. Barroso, J. Dean,U.Holzle, Web Search for a Planet: the Google Cluster Architecture, 2003, IEEE Micro.

[3] G. Boss, et al., Cloud Computing, IBM Corporation white paper, Oct. 2007.

[4] R.Buyya, R.Ranjan and R.N. Calheiros, Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities, in the proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, June 21-24,2009.

[5] R. B. Cooper, Queueing Theory, in the encyclopedia of comuter science, 4th edition, Groves Dictionaries, Inc. 2000, pp.1496-1498.

[6] C. Develder, J. Buysse, M. D. Leenheer, B. Jaumard, B. DhoedtResilient network dimensioning for optical grid/clouds using relocation, in the proceedings of 2012 ICC conference.

[7] S. L. Garfinkel, An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS,Technical Report TR-08-07, 2008, Harvard University.

[8] Grassmann,W. K.: *"Is the Fact That the Emperor Wears No Clothes a Subject Worthy of Publication?"*, Interfaces, 16(1986),pp. 43-46.

[9] J. Gu, J. Hu, T. Zhao, G. Sun, A New Resource Scheduling Strategy Based on Genetic Algorithm in Cloud Computing Environment, JOURNAL OF COMPUTERS, VOL. 7, NO. 1, JANUARY 2012, pp. 42-52.

[10] A.Greenberg, P.Lahiri, D. A. Maltz, P. Patel, S. Sengupta, Towards a Next Generation Data Center Architecture: Scalability and Commoditization, PRESTO'08, August 22, 2008, Seattle, Washington, USA.

[11] R. C.Hampshire, W.A.Massey, D.Mitra and Q. Wang, Provisioning For Bandwidth Sharing and Exchange, Telecommunications Network Design and Management, edited by G. Anandlingam and S. Raghavan, pp. 207-226, 2003.

[12] B.Javadi, D.Kondo, J.-M.Vincent, D. P. Anderson, Mining for Statistical Models of Availability in Large-Scale Distributed Systems: An Empirical Study of SETI@home. 17th Annual Meeting of the IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, Sept 21-23 2009, London.

[13] O.B.Jennings et al., Server Staffing To Meet Time-Varying Demand, Management Science, pp.1383-1394, 1996.

[14] Kaufman, J.: *"Blocking in a Shared Resource Environment"*, IEEE Transactions On Communications, Vol. COM-29, No. 10, October 1981.

[15] F. Kelly, *"Blocking probabilities in large circuit-switched networks"*,Adv. Appl. Prob., vol. 18, pp. 473-505, 1986.

[16] L. Kleinrock, Queueing Systems, Volume II: Computer Applications, John Wiley & Sons, 1976.

[17] D. Kondo, B.Javadi, P. Malecot, F. Cappello, D. P. Anderson , Cost-Benefit Analysis of Cloud Computing versus Desktop Grids, 18th International Heterogeneity in Computing Workshop, May, 2009,Rome.

[18] J. Lu, GQ. Zhang, An Innovative Self-Adaptive Configuration Optimization System in Cloud Computing, in the conference of Dependable, Autonomic and Secure Computing (DASC), 2011, Page(s): 621 - 627.

[19] S.Ranjan, J.Rolia, H.FU and E.Knightly, QoS-Driven Server Migration for Internet Data Centers, In Proceedings of IWQoS 2002.

[20] J.W. Roberts, *"A serverice system with heterogeneous user requirements"*, In Performance of Data Communications Systems and Their Applications, pp. 423-431, 1981.

[21] K.W. Ross, *"Multiservice Loss Models for Broadband Telecommunication Networks"* ,Springer-Verlag London Limited , 1995.

[22] W. Tian and H. G. Perros, Analysis and Provisioning of a Circuit-switched Link with Variable-Demand Customers, In the proceedings of the 20th International Teletraffic Congress, Ottawa, Canada, 17-21 June 2007.

[23] W. Tian, and H. G. Perros, Dimensioning a Virtual Computing Lab with Job Priorities and QoS Constraints, In the proceedings of 2nd International Conference on the Virtual

Computing Initiative , pp.103-110, May 2008, Research Triangle Park, IBM headquarter, NC, USA.
[24] W.Tian, Three Ways to Improve the Efficiency of Virtual/Cloud Computing Lab. In the proceedings of The IEEE International Conference on Apperceiving Computing and Intelligence Analysis 2008 (ICACIA'08), Dec. 2008.
[25] W. Tian, Adaptive Dimensioning of Cloud Data Centers, In the proceedings of DASC'09, the eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009. pp. 5-10.
[26] M. Vouk, et al., "Powered by VCL" - Using Virtual Computing Laboratory (VCL) Technology to Power Cloud Computing, Published in the Prelim. Proceedings of the 2nd International Conference on Virtual Computing Initiative, 15-16 May 2008, RTP, NC, pp. 1-10.
[27] M. A. Vouk, Cloud Computing - Issues, Research and Implementations,ITI08, pp.23-26-31, June, 2008.7.
[28] B. Xu, N. Wang, C. Li, A Cloud Computing Infrastructure on Heterogeneous Computing Resources, JOURNAL OF COMPUTERS, VOL. 6, NO. 8, AUGUST 2011, pp.1789-1796.
[29] X. You, J. Wan, X. Xu, C. Jiang, W. Zhang, J. Zhang, ARAS-M: Automatic Resource Allocation Strategy based on Market Mechanism in Cloud Computing, JOURNAL OF COMPUTERS, VOL. 6, NO. 7, JULY 2011,pp.1287-1296.
[30] http:/boincstats.com/stats/project−graph.php?pr=sah&view=hosts, 2012
[31] Amazon, Amazon Elastic Compute Cloud, http://aws.amazon.com/ec2/, 2013

ShortBio: Dr. Wenhong Tian has a PhD from Computer Science Department of North Carolina State University. His research interests include dynamic resource scheduling algorithms and management in Cloud datacenter, dynamic modeling and performance analysis of communication networks, biocomputing. He published about 30 journal and conference papers in related areas.