Scheduling Cloud Capacity for Time- Varying Customer Demand

Brian Bouterse
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
bmbouter@gmail.com

Harry Perros
Department of Computer Science
North Carolina State University
Raleigh, NC, USA
hp@ncsu.edu

***Abstract*—As utility computing resources become more ubiquitous, service providers increasingly look to the cloud for an in-full or in-part infrastructure to serve utility computing customers on demand. Given the costs associated with cloud infrastructure, dynamic scheduling of cloud resources can significantly lower costs while providing an acceptable service level. We investigated several methods for predicting the required cloud capacity in the presence of time-varying customer demand of application environments. We evaluated and compared their performance, using historical data of the Virtual Computing Laboratory (VCL) at North Carolina State University. We show that a simple heuristic, whereby we continuously maintain a fixed reserve capacity, performs better than the other methods.**

***Index Terms*— capacity planning, auto scaling, application delivery, VCL, virtualization, non-stationary traffic, non-homogeneous traffic, traffic characterization, traffic prediction**

I. Introduction

Service providers have the challenging problem of satisfying customer demand while minimizing their expenses on infrastructure. Historically, a service provider owns the entire infrastructure, which is typically dimensioned for peak demand. Cloud computing provides an alternative way to use an infrastructure without owning it, which is to rent computing resources by the hour. The cloud infrastructure is provided by the cloud provider. This frees a service provider from the financial burden of owning equipment, but requires scheduling of resources in order to satisfy dynamically varying customer demand. Users rely on the service provider to get access to applications hosted in cloud environments. Service providers try to minimize the cost of virtual machines. In this paper, we consider the problem of capacity planning of cloud resources for a service provider delivering compute-intensive desktop applications based on customer demand that varies as a function of time. We define capacity as the number of application users a cloud is capable of servicing concurrently. We assume that the service provider has unlimited access to virtual machines in a cloud environment. We propose and compare several models for managing the required number of virtual machines, so that the demand is satisfied within a given blocking probability while the total number of unused resources is kept as low as possible. For this, we use historical data from the Virtual Computing Laboratory (VCL) at North Carolina State University [9]. The VCL offers computing services to 42,000 students and faculty who use it to run applications for teaching and research. The VCL as a service provider contains the right challenges to practically benefit from a schedule of bringing online and offline cloud computing resources.

Existing cloud computing provisioning models explore the capacity-planning problem from the service provider perspective [4], [5], [14]. The focus of this paper is client-orchestrated provisioning of cloud computing infrastructure, which is a separate concern from the cloud optimization by the service provider [6]. In Roy et al. [8] the authors explore client-orchestrated cloud provisioning models that rely on predictive techniques using a single statistical model to predict near-term workload demand [13]. Urgaonkar [13] uses a hybrid model of predictive and reactive decision-making, and use a workload predictor proposed by Rolia et al. [7]. Our research compares several statistical workload prediction techniques, one of which is the model proposed by Roy et al. [8]. Specifically, we analyze and compare the provisioning effectiveness of several statistical traffic prediction techniques, and a heuristic-based technique. We explore both reactive and predictive provisioning techniques. Client-orchestrated cloud provisioning policies with real-time demand needs are also explored [2] and [11], assuming advanced resource reservation by users. This paper assumes that application reservations are not known in advance. Cloud infrastructure boot times make this problem non-trivial.

This paper is organized as follows. Section 2 presents the system under study. In section 3, we assume that we have a very reliable forecast of the number of requests per five-minute intervals for a given period of time. Using this forecast, we determine the necessary capacity for all five-minute interval for the same period of time using the Erlang loss model so that the blocking probability is less or equal to 0.01. In section 4, we present four forecasting models for predicting the demand for every five-minute interval. Based, on the predicted demand for the next five-minute interval, we determine the necessary capacity using the Erlang loss model so that the blocking probability is less or equal to 0.01. In section 5, we present a simple heuristic model where a fixed reserve capacity is maintained. The performance of each model, expressed in terms of the blocking probability and the total unused seat capacity, is evaluated for different traffic conditions. In addition, in section 6, we compare these five models using a composite metric that combines both the blocking probability and the total unused capacity. We show that the reserve capacity model performs the best among the models studied. Finally, the conclusions are given in section 7.

II. The System Under Study

We assume that a service provider rents virtual machines from a cloud dynamically in order to match the anticipated customer demand, which varies over time. Each virtual machine is a part of the cluster, and we assume an unlimited supply of virtual machines. Each virtual machine has *N application seats* for concurrent independent users to use. In practice, VCL uses virtual machines that have two user application seats, but experimenting with a range of virtual machine user seat capacities per virtual machine only marginally changed blocking probability. In this paper, we assume that *N*=2, unless otherwise noted. Servers are requested dynamically as customer demand increases, and they are also released dynamically as demand decreases. The increases and decreases are an integer number of servers. We assume an infinite population of users. If a user arrives at a time when there are no seats available, the user is blocked and leaves the system without returning.

The proposed models are tested using historical data from the VCL collected for two years. Year 1 began on July 1 2008, and year 2 on July 1 2009. Users that took longer than 8 hours were removed from the data set. This is because the maximum on-demand reservation in VCL is 8 hours. Longer reservations are also accepted but they are provisioned in a dedicated way, which is not within the scope of this study. The resultant data set consists of a total of 175,554 requests in the first year and 232,626 in the second year. Each request is associated with the time at which it was issued and the total service time used, i.e., the total time the user holds a seat. The number of requests computed over 5 minute intervals for year 1 is shown in figure 1, and the histogram of the service times for the same year is shown in figure 2.

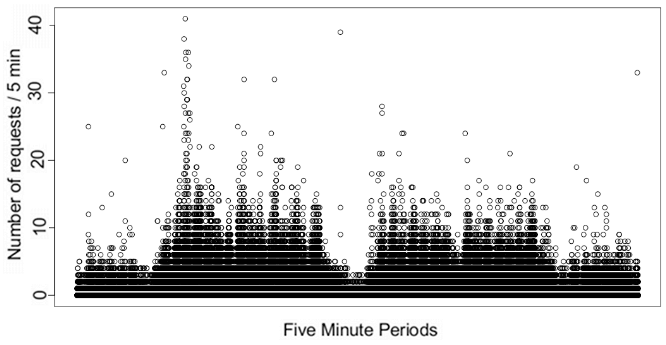

Fig. 1. Year 1 number of requests per 5 minute interval

As can be seen in figure 1, the arrival rate varies in time, daily and weekly. The arrival traffic also varies seasonally with traffic characteristics as the school calendar transitions through the fall and spring semesters, summer sessions, and dead periods. For instance, the midpoint in figure 1 has very low arrival traffic, which corresponds with the academic December holiday.

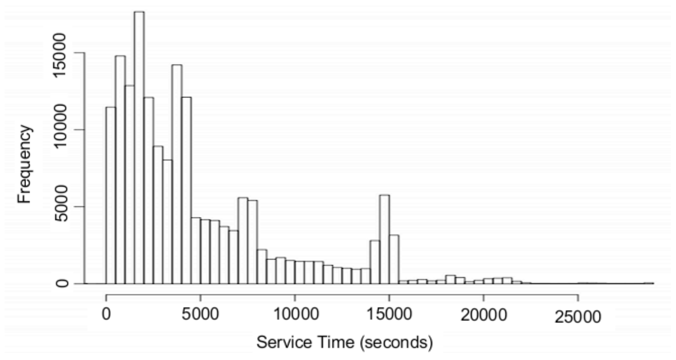

Fig. 2. The service time distribution of year 1 data

We assume a delay between the time that a number of virtual machines are requested to the time they become available. This reflects the boot time to bring up a new virtual machine and we assume that it is independent of the requested number of virtual machines. Based on empirical observation of current cloud infrastructures by the authors, the boot time is set to five minutes. A shut down delay is also introduced in order to avoid “thrashing”, see Groskinsky [3], whereby a virtual machine is released and immediately after a new virtual machine is requested. Such an oscillation can cause an unnecessary boot time. The shut down time is set to five minutes. After five minutes, the released virtual machines are returned to the service provider’s pool of free virtual machines in the cloud infrastructure. When a new virtual machine is requested, we first examine whether there are any released virtual machines that have not been returned to the service provider’s pool yet. If there is one, then it is put into service without incurring a boot time. In addition, a released virtual machine is removed from active service only if all seats of the virtual machine are empty. Finally, we note that virtual machines are assigned priorities, and a user is always assigned to the highest priority available virtual machine. The user assignment policy groups users densly onto VMs in the cluster.

We now proceed to describe the five models for predicting the capacity of the system under study. The performance of each model is expressed in terms of blocking probability and total unused seat-hours. The blocking probability is defined as the probability that a request will be denied because there are no seats available. To calculate the blocking probability and the total number of unused seat-hours of each model, we developed a simulation, implemented in SimPy [10], which models the system described above, with the provisioning of the capacity obtained by the model under test.

III. Poisson Capacity Planning Using A Forecasting Period with pre-known demand

In this section, we assume that we have a very reliable forecast of the number of requests per five-minute interval for an entire period of time. For each five-minute interval, we calculate the required capacity so that the blocking probability is 1%. We do this using the Erlang-loss queueing model. That is, the system under study is modeled as an Erlang-loss queueing model where each server in the queueing model represents an application seat. (A server in the queueing model should not be confused with a virtual machine in the physical system under study.) The arrival process is assumed to be Poisson distributed with a time-dependent arrival rate, equal to the number of requests forecasted to occur during each five-minute interval over the entire period of the forecast. The mean service time in the Erlang-loss is equal to 1.377 hours, which is calculated from the year 1 data of the VCL. For a given capacity, arrival and service rates, the blocking probability can be obtained using the well-known Erlang B loss equation, which is not dependent on the distribution of the service time. This expression is obtained under the assumption that the arrival rate is stationary. However, in this case the arrival rate is non-stationary since it varies for each five-minute interval. In Alnowibet and Perros [1], it was observed that the time-dependent blocking probability BP(t) can be obtained approximately using the Erlang B formula. That is,

$$BP(t) = \frac{\rho(t)^s / s!}{\sum_{i=0}^{s} \rho(t)^i / i!}. \tag{1}$$

where $\rho(t)=\lambda(t)/\mu$ is the traffic intensity, with μ equal to the service rate, $\lambda(t)$ the time-dependent arrival rate, and s be the number of application seats. The authors Alnowibet and Perros observe that the approximation has an extremely low relative error. Consequently, for each five-minute interval we calculate the blocking probability using the forecasted number of requests for the same interval.

Using a simple search algorithm we can find the value of s, the total number of seats required so that the blocking probability is less than or equal to 1%. Repeating this for all successive five-minute intervals, yields a curve of the required seat capacity so that the blocking probability of 0.01 is satisfied. In figure 3, we show this curve for year 1, consisting of 105,120 five-minute observations. The number of virtual machines is obtained by dividing the seat capacity by the number of application seats per virtual machine and rounding up. Given that we know the required virtual machine application seat capacity, we can construct an optimum schedule to provision and de-provision virtual machines, taking into account the boot time and shut down times.

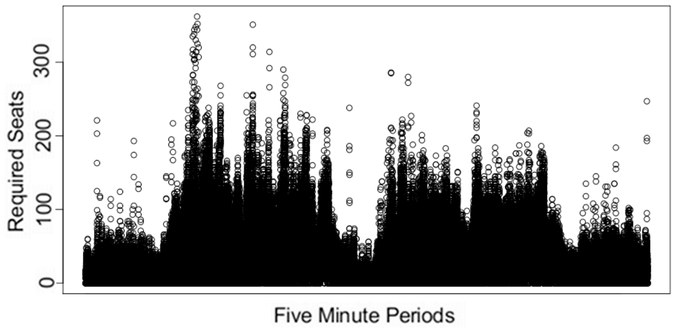

Fig. 3. The required number of seats for each five-minute interval for year 1 such that the blocking probability is less or equal to 0.01.

Running the simulation on the year 1 VCL data, we obtain a blocking probability of 0.00899 and total unused seat-hours of 167,476. The forecast used as input to this scheme is the actual historical data, and consequently there are no forecasting errors. The unused seat-hours are due to the calculation of the capacity at each five-minute interval using the Erlang B formula which assumes that the arrival process is Poisson distributed, and also due to over-dimensioning that arises from the fact that there are *N* seats per virtual machine. The error of using the Erlang B formula for approximating non-stationary arrivals is negligible as reported by Alnowibet and Perros [1]. When tested with different *N* values of seats per virtual machine, the error only changed marginally. In view of this, the main source of error in these results is due to the Poisson assumption. However, in practice, this source of error is less than the error due to forecasting demand.

IV. MODELS FOR FORECASTING CUSTOMER DEMAND

In this section, we present four different forecasting models that are used to predict the number of requests in the next five-minute interval. Based on the predicted customer demand, the Erlang B formula is used as in section III to determine the capacity of the system so that the blocking probability is less or equal to 0.1. Accordingly, a provisioning or a deprovisioning action may be initiated.

A. Moving Average Model

The moving average model with parameter *k* determines the customer demand for the next five-minute interval s_{t+1} based on the previous *k* five-minute interval actual observations x_t, x_{t-1}, ... , x_{t-k+1}, as follows:

$$s(k) = \frac{x_t + x_{t-1} + \ldots + x_{t-k+1}}{k} \tag{2}$$

Using this model in our simulation we obtained the blocking probability, seat utilization, and total unused seat-hours for year 1, for various values of *k*. The results are summarized in table 1. As before, we assumed *N*=2 application seats per virtual machine, and boot time and shut down times equal to five minutes. We observe that the moving average model meets the requirement of the blocking probability and with *k*=30 it achieves the unused seat hours of 265,110.

TABLE 1. THE MOVING AVERAGE MODEL FOR DIFFERENT VALUES OF *K* FOR YEAR 1

k	Blocking Probability	Seat Utilization	Unused Seat-Hours
1	0.0002	0.36485	420758.3
3	0.00088	0.43478	313958.4
10	0.00031	0.46837	274287.7
30	0.00085	0.47673	265110.1
100	0.0305	0.46494	269908.8

Once optimized on the VCL year 1 data, we want to evaluate how well the optimized model generalized to traffic environments with different arrival and service time behaviors. Using *k*=30, we run the simulation using the moving average model against artificial traffic scenarios obtained by scaling the number of requests per five-minute intervals and the service times from year 1 by an offered traffic scaling factor and by a service time scaling factor respectively. Both scaling factors were varied from 0.5 to 1.5. The case where they are both equal to 1 corresponds to the year 1 data. We note that the scaling causes peaks and valleys in the offered traffic to become steeper. The results are presented in figure 4.

Figure 4 shows that this model successfully manages blocking probability to below 0.01 for increases of up to 20% in the service time, or increases in the number of requests per five-minute interval of up to 50%. We note that year 2 represents an increase in traffic volume of 32%, and an increase in service time of 5%. Using the actual year 2 data, we obtained a blocking probability of 0.00412 and total unused seat-hours of 257,177 hours. These year 2 results are similar to the equivalent artificially scaled year 1 traffic scenario.

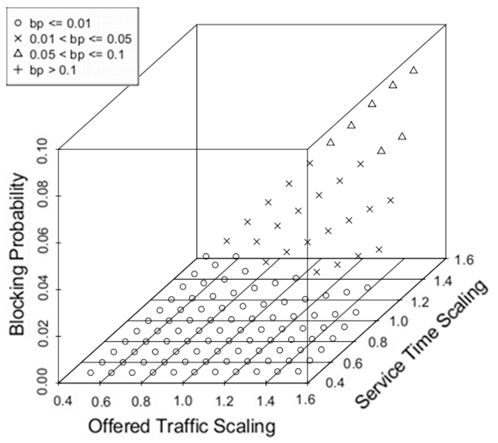

Fig. 4. The blocking probability of the moving average model with $k = 30$ versus the scaling factor for the number of requests and the service time.

With 265,110 and 257,177 unused seat-hours in year 1 and year 2 respectively, the model is over-allocating capacity, which explains why the blocking probability is so low, and the unused hours high. The previous model described in section III, gave rise to 167,476 unused seat-hours for year 1 where we assumed perfect knowledge of demand. The moving average model performs approximately 100,000 hours worse, but it does not require perfect knowledge of the demand over a period of time.

B. Exponential Moving Average Model

An exponential moving average model is also used in the simulation to determine its feasibility. The model is as follows:

$$s_1 = x_0$$
$$s_t = \alpha x_{t-1} + (1-\alpha)s_{t-1} = s_{t-1} + \alpha(x_{t-1} - s_{t-1}), \; t > 1 \qquad (3)$$

where s_t and x_t are the estimated and observed values for the t^{th} five-minute interval, and α, $0 \le \alpha \le 1$. This model was used in the simulation for various values of α for year 1 and the results are shown in table 2.

Almost all values of α satisfy the blocking probability requirement and for α=0.05 we obtain a blocking probability of 0.00099 and 265,484 unused seat-hours. Using α=0.05, we run the exponential moving average model against the same scaled artificial traffic scenarios as in section 4a. The results are presented in figure 5.

TABLE 2. EXPONENTIAL MOVING AVERAGE MODEL FOR VARIOUS VALUES OF A FOR YEAR 1.

α	Blocking Probability	Seat Utilization	Unused Seat-Hours
0.01	0.02898	0.46442	270415.2
0.05	0.00099	0.47633	265483.9
0.1	0.00021	0.47215	270191.1
0.3	0.00042	0.4495	295902.1
0.7	0.00034	0.40191	359612.8
0.9	0.00025	0.3771	399222.0
1.0	0.0002	0.36485	420758.3

Figure 5 shows that this model successfully manages blocking probability to below 0.01 for increases of up to 10% in the service time, or increases in the number of requests per five-minute interval of up to 40%. For year 2 data, we obtained a blocking probability of 0.00473 and total unused seat-hours of 257,697 hours, which are similar to the equivalent artificially scaled year 1 traffic scenario.

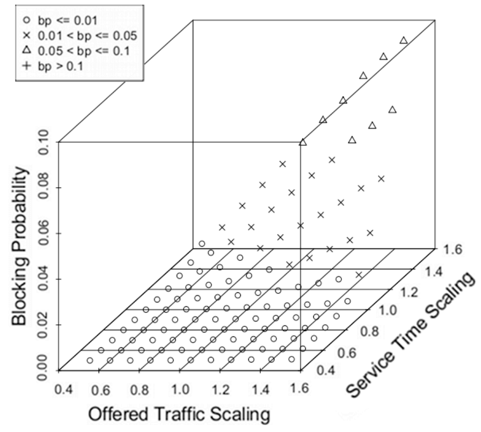

Fig. 5. The exponential moving average blocking probability model with α=0.05 versus the scaling factor for the traffic intensity and service time.

C. An Autoregressive Model

A second order autoregressive model with parameters c, $\{\varphi_1,\varphi_2,\ldots,\varphi_n\}$, and ε_t is defined as follows:

$$s_t = c + \sum_{i=1}^{n} \phi_i s_{t-i} + \varepsilon_t \qquad (4)$$

where s_t is the estimated value for the t^{th} five-minute interval, and ε_t is normally distributed with mean 0 and variance σ^2. This model was also proposed by Schaffer [9] to predict virtual machine workload. Using the year 1 data set, the parameters of this model are optimized assuming an order of two and c=0. Using the Yule-Walker method [12], we obtain φ_1=0.4108, φ_2=0.3368, and σ^2=2.98. Based on experimentation, increasing the modeling order beyond 2 yields only marginal improvements, and using c values other than 0 introduce long-term bias in prediction error. We use this model in our simulation for year 1, and we obtained a blocking probability of 0.00475 and unused seat-hours of 497,610. We run this autoregressive model against the same artificial traffic scenarios as in section 4a and the results are presented in figure

6. We note that this model successfully manages blocking probability to below 0.01 for increases of up to 10% in the service time, or increases in the number of requests per five-minute interval of up to 40%. Using the actual year 2 data in the simulation, we obtained a blocking probability of 0.01031 and 471,683 unused seat-hours, which are similar to the results of the equivalent artificially scaled year 1 traffic scenario.

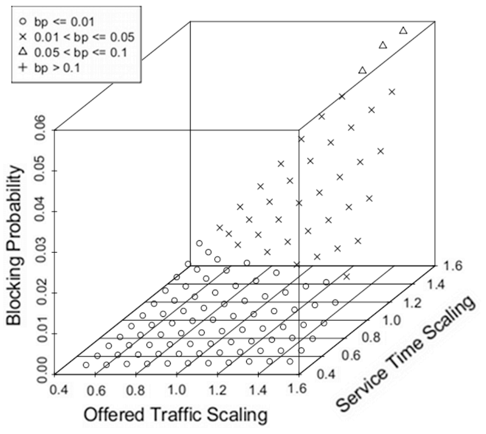

Fig. 6. The blocking probability of the autoregressive model with φ_1=0.4108, φ_2=0.3368, and σ^2 of 2.98 versus the scaling factor for the number of requests and the service time.

D. A Mixed Autoregressive Model

The calendar year in an academic environment can be broken down, from the point of view of customer demand for VCL, to the following three periods: a) Fall and Spring semesters, b) Summer sessions 1 and 2, and c) exam periods.

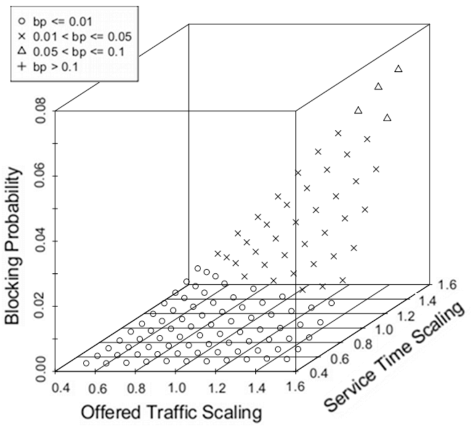

Fig. 7. The blocking probability of the mixed autoregressive model with parameters from table 3 versus the scaling factor for the number of requests and the service time.

An autoregressive model was obtained using the Yule-Walker method [12] for each of these three periods using the year 1 data set. The model parameters are given in table 3.

TABLE 3. C, Φ_1, Φ_2, AND Σ^2 FOR THE EACH TIME PERIOD FOR YEAR 1.

Time Period	c	φ_1	φ_2	σ^2
Fall/Spring Classes	0	0.3993	0.3226	4.197
Summer Sessions	0	0.2604	0.1966	0.9281
Exams	0	0.2530	0.2044	1.638

We refer to these three models as the *mixed autoregressive* model. Using the mixed autoregressive model in the simulation with the demand from year 1, we obtained a blocking probability of 0.00636 and 443,440 unused seat-hours. As in the previous models, the mixed autoregressive model was run against the artificial traffic scenarios from section 4a and the results are presented in figure 7.

Figure 7 shows that this model successfully manages blocking probability to below 0.01 for increases of up to 10% in the service time, or increases in the number of requests per five-minute interval of up to 20%. For year 2, we obtain a blocking probability of 0.01494 (which does not meet the required level of 0.01) and total unused seat-hours of 415,460.

V. The Reserve Capacity Model

In this section we present a simple heuristic for determining the capacity of the VCL, whereby we continuously maintain R available seats. Specifically, every 300 seconds, the current level of usage is observed and according to the number of available seats it maybe adjusted up or down so that the number of available seats is at least R. We refer to this model as the *reserve capacity* model. Using the year 1 data, we experiment with different values of R. The results are summarized in table 4. We observe that for R=12, the blocking probability is 0.00895 and the total unused seat-hours is 118,028. Using R=12, we run the reserve capacity model against the artificial traffic from section 4a, and the results are shown in figure 8.

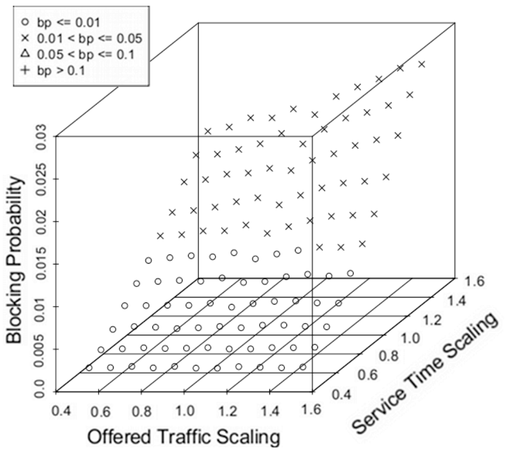

Fig. 8. The blocking probability of the reserve capacity model with R=12 versus the scaling factor for the number of requests and the service time.

TABLE 4. RESULTS FOR DIFFERENT R VALUES FOR YEAR 1

Reserved Capacity (R)	Blocking Probability	Seat Utilization	Unused Seat-Hours
5	0.05519	0.79352	59645.7
10	0.01336	0.70308	100932.1
12	0.00895	0.6703	118027.7
15	0.00558	0.62563	144049.9
20	0.00275	0.56286	187384.7

Figure 8 shows that this model yields a blocking probability to below 0.01 for increases of up to 20% in the service time, but it cannot tolerate any increase in the offered traffic load. Simulation from actual year 2 data yields a blocking probability of 0.01036 and unused seat-hours of 120,922, similar to the equivalent artificially scaled year 1 traffic scenario.

VI. Composite Metric

In this section, we compare the performance of the four forecasting models presented in section IV and the reserve capacity model presented above. We did not include the first model presented in section III, since it is deemed unrealistic. For this comparison, we combined the two performance metrics, i.e., the blocking probability and the unused hours, into a single composite metric, following the method proposed by Yiltas and Perros [15]. This composite metric is calculated for all the methods in order to compare their performance. For each metric, all the observed values for all the models and for all the traffic scenarios investigated (a total of 121) are sorted in an ascending order, and assigned an integer equal to their position in the list. Metric values that tie are assigned the same position integer. The integer values are then normalized by the sum *(U(U – U)/2)*, where *U* is the total number of observations, that is, *U*=5x121=605. The composite metric of a model is the sum of the two normalized integers, one for the blocking probability and one for the total unused hours. In figure 9, we give a plot of the composite metric for each model and for each traffic case.

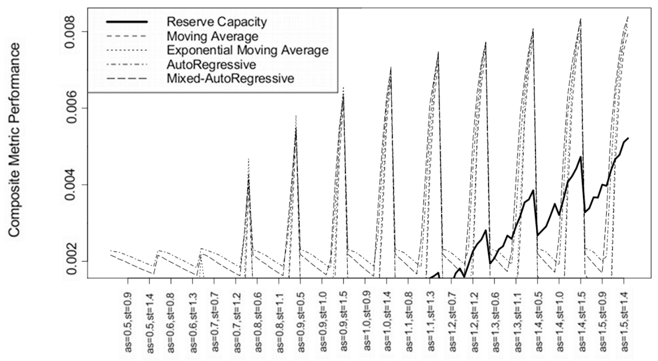

Fig. 9. Composite metrics of the five policy models versus traffic scenarios.

The x-axis lists the traffic cases as follows. The scaling factor for the arrival rate increasing from 0.5 to 1.5 from left to right, and for each arrival rate value the service time scaling increases from 0.5 to 1.5 for left to right. All the composite metrics show a periodic trend as the service time scaling is increased when the arrival scaling is already above 1.0. The smaller the composite metric, the better the performance of the model. We note that the reserve capacity model (thick line) outperforms all the other models.

VII. Conclusions

In this paper, we examine several different models for dimensioning the capacity of cloud-based system in the presence of time-varying customer demand. It appears that the reserve capacity model significantly outperforms all models as is shown clearly by the composite metric graph of Figure 9. For better performance year-after-year, *R* should be recalibrated every year.

The reserve capacity model outperforms all Poisson-based traffic capacity planning models because the total error of an optimized reserve capacity strategy has 118,028 unused seat-hours while a Poisson approximated provisioning schedule that assumes pre-known demand has 167,476. This may suggest that the Poisson treatment of the VCL arrival process is not a good assumption. However, this needs to be further investigated.

Among the traffic prediction methods that use the Erlang loss model to calculate the required capacity, it seems that the mixed autoregressive model outperforms the autoregressive model by an average of 11.4%.

References

[1] Alnowibet, K.A. and Perros, H., “Nonstationary analysis of the loss queue and of queueing networks of loss queue,” European J. of Operational Research, vol. 196, pp. 1015-1030, 2009.

[2] Beckman, P. et al., “SPRUCE: A system for supporting urgent high-performance computing,” Grid-Based Problem Solving Environments, pp. 295-311, 2007.

[3] Groskinsky, B. et al., “An investigation of adaptive capacity control schemes in a dynamic traffic environment,” IEICE Trans. on Commun. E Series B, vol. 84, pp. 263-274, 2001.

[4] Jiang, J et al., “An Innovative Self-Adaptive Configuration Optimization System in Cloud Computing,” IEEE Ninth Int. Conf. on Dependable, Autonomic and Secure Computing, pp. 621-627, 2011.

[5] Li, B. et al., “Enacloud: An energy-saving application live placement approach for cloud computing environments,” IEEE Int. Conf. on Cloud Computing, pp. 17-24, 2009.

[6] Lim, H.C., et al. “Automated control in cloud computing: challenges and opportunities,” Proc. of the 1st workshop on Automated control for datacenters and clouds, pp. 13-18, 2009.

[7] Rolia, J. et al., “Statistical service assurances for applications in utility grid environments,” J. of Perf. Evaluation, vol. 58, pp. 319-339, 2004.

[8] Roy, N. et al., “Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting,” Proc. of the IEEE 4th Int. Conf. on Cloud Comp., pp. 500-507, 2011.

[9] Schaffer, H.E., “NCSU's virtual computing lab: a cloud computing solution,” J. or Comp., vol. 42, pp. 94-97, 2009.

[10] SimPy Developer Team (2012, 6, 13). *SimPy Simulation Package* [Online]. Available http://simpy.sourceforge.net/

[11] Sotomayor, B. et al., “Capacity leasing in cloud systems using the opennebula engine,” Workshop on Cloud Comp. and its Applicat., vol 3, 2008.

[12] Stoica, P. and Moses, R.L., “AR Signals” in *Introduction to spectral analysis*, Prentice Hall Upper Saddle River, NJ, 1997, ch. 3 sec. 3.4.1, pp. 89.

[13] Urgaonkar, B., “Agile dynamic provisioning of multi-tier internet applications,” ACM Trans. on Autonomous and Adaptive Syst., vol. 3, 2008.

[14] Van, H.N. et al., “SLA-aware virtual resource management for cloud infrastructures,” Ninth IEEE Int. Conf. on Comp. and Inform. Technology, pp. 357-362, 2009.

[15] Yiltas, D. and Perros, H. “A composite QoS metric for multi-attribute QoS-based multi-domain routing,” IET Comm., vol. 5, pp. 327-336, 2011.