A Capacity Planning Framework for Multi-tier Enterprise Services with Real Workloads

Qi Zhang
College of William and Mary
Williamsburg, VA 23187, USA

Ludmila Cherkasova
Hewlett-Packard Labs
Palo Alto, CA 94304, USA

Guy Mathews, Wayne Greene
Hewlett-Packard
Palo Alto, CA 94304, USA

Evgenia Smirni
College of William and Mary
Williamsburg, VA 23187, USA

***Abstract*— With complexity of systems increasing and customer requirements for QoS growing, new methods and modeling techniques that explain large-systems' behavior and help predict their future performance are required to effectively tackle the emerging performance issues. To accurately answer capacity planning questions for an existing production system with a real workload mix, we propose a new capacity planning framework that is based on the following three components: *i)* a *Workload Profiler* that dynamically builds the workload profile; *ii)* a *Regression-based Solver* that is used for deriving the CPU demand of client transactions on a given hardware; and *iii)* an *Analytical model* that is based on a network of queues representing the different tiers. To validate our approach, we conduct a detailed case study using the access logs from two heterogeneous production servers that represent customized client accesses to a popular and actively used HP Open View Service Desk application.**

I. Problem Statement

As IT and application infrastructures become more complex, predicting and controlling the issues surrounding system performance and capacity planning become a difficult and overwhelming task to many organizations. For larger IT projects, it is not uncommon for the cost factors related to performance tuning, performance management, and capacity planning to result in the largest and least controlled expense. Application performance issues have an immediate impact on customer satisfaction. A sudden slowdown of enterprise-wide application can affect a large population of customers, can lead to delayed projects, and ultimately can result in company financial loss. It is not unusual for a piece of new hardware to be added into the infrastructure to alleviate performance issues without fully understanding where the problem really is.

Large-scale enterprise development projects use a collection of mechanisms and interfaces in a dynamic enterprise IT environment to connect different applications where classic, data-processing legacy systems can be integrated with agile web-based front-end applications. Application servers provide a standardized platform for developing and deploying scalable enterprise systems. Application servers are becoming a core component of an enterprise system and an integral part of a new trend towards building service-oriented architectures. Today, the *three-tier architecture* paradigm has become an industry standard for building scalable client-server applications. In multi-tier systems, frequent calls to application servers and databases place a heavy load on resources and may cause throughput bottlenecks and high server-side processing latencies.

With complexity of systems increasing and customer requirements for QoS growing, the research challenge is to design effective system modeling techniques in order not only to explain large-system behavior but also predict and plan their future performance. Capacity planning is a critical area obtaining a lot of attention in IT management aiming at quality of service support and decision making [1]. Typically, preliminary system capacity estimates are done by using synthetic workloads or benchmarks which are created to reflect a "typical application behavior" for "typical client requests". Many of industry benchmarks are build using this principle [9], [10]. Traditional capacity planning methodologies [5], [6] aim to accommodate variations in load under such "typical application behavior", and examine peak loads and system utilization to conclude on the number of clients that can be handled by the system. However, real workloads rarely exhibit this feature. While this performance evaluation approach can be useful at the initial stages of design and development of a future system, it might not be adequate for answering specific questions about the existing production system. Often, a service provider does need to answer the following questions:

- How many additional clients can be supported by the existing system *i)* while still providing the *same* performance guarantees, e.g., response time under 6 sec., and *ii)* assuming that new clients perform activities as already existing clients in the system, i.e., the system processes the *same* type of workload?
- Does the existing system have enough available capacity for processing an additional service for *N* number of clients where the client activities and behaviors are specified as a well-defined subset of the current system activities?
- If the current client population doubles, then what is the expected system response time?

Instead of focusing on loads solely, a robust capacity planning methodology must also consider the changing workload mix since the system capacity directly depends on the types of user performed activities. In this work, we propose a novel capacity planning framework, called *R-Capriccio*, for practical capacity evaluation of existing production systems under real workload. *R-Capriccio* can assist in providing answers for advanced "what-if" scenarios in system capacity analysis where the evaluated system operates under a diverse workload mix. It uses a similar closed multi-tier model as in [7], but in contrast to [7] or any examples in the existing literature of capacity planning, *R-Capriccio* does not use a controlled environment to parameterize the analytic model. Instead of characterizing the overall service time of every server, it uses a statistical regression method to approximate the service cost of individual core transactions. This CPU cost function together with the transaction mix help for approximating system service times that vary with a changing transaction mix.

The use of statistical methods in capacity planning has been proposed in the early 80's [8], [1], but the focus was on a single machine/cluster that is much simpler than current large-scaled multi-tiered systems. Recently statistical methods are getting more attention in computer performance analysis and system performance prediction. In [2] the authors use multiple linear regression techniques for estimating the mean service times of applications in a single-threaded software server. These service times are correlated with the Application Response Measurement package (ARM) data to predict system future performance. In [3],[4] the authors focus on large-scale distributed servers as our work does. They use linear regression for predicting the transaction response time under varying transaction mix. To the best of our knowledge, *R-Capriccio* is a first capacity planning framework which provides a practical, flexible and accurate toolbox for answering capacity planning questions for multi-tier production systems with real workloads. More importantly, it can be used for explaining large-scale system behavior and predicting future system performance.

II. Solution We Are Working on

R-Capriccio is comprised of the following key components:

- *Workload profiler*: The profiler extracts a set of most popular client transactions, called *core* transactions, to characterize the overall site workload and the most popular client sessions at the site.
- *Regression-based solver*: Using statistical regression, the solver approximates the resource cost (CPU demand) of each core transaction on a given hardware. Thus a real workload mix can be directly mapped into the corresponding CPU demand requirements.
- *Analytical model*: For capacity planning of multi-tier applications with session-based workloads, an analytical model based on network of queues is developed, where the queues represent different tiers of the application.

We consider a *web page* accessed by the client and generated by the application as the basic unit of client/server activity in *R-Capriccio* and call it a *transaction*. A web page consists of an HTML file and embedded objects retrieved via the HTTP protocol. The regression technique works well for estimating the CPU cost of transactions that represent a collection of smaller objects, where direct measurements fail. There are no practical means to effectively measure the service times for all these objects, while the accurate CPU consumption estimates are required for capacity planning of the systems operating under real workload mix. Thus, one of the main *problems* is to approximate the *CPU cost* of different client transactions at different tiers, and then use these cost functions for evaluating the system resource requirement under diverse real workloads to accurately size a future system.

A prerequisite for applying our framework is that a service provider collects the following information:

- the application server access log that reflects all processed client requests and client activities at the site;
- CPU utilization at all tiers of the evaluated system.

Workload profiler. The profiler collects a set of workload and system metrics over time: *i)* the average CPU utilization, *ii)* the number of different transactions, *iii)* the number of concurrent sessions, and *iv)* the client think times. Additionally, for each time window (1-hour in our case study), the profiler provides the average CPU utilization as well as the number of transactions N_i for the a set of M most popular client transactions, where $1 \leq i \leq M$. A fragment of the workload profile is shown in Table I.

TABLE I
An example of transaction profile.

Time (hour)	N_1	N_2	N_3	$\cdots$	N_M	U_{CPU} (%)	Think Time (second)
1	21	15	21	$\cdots$	0	13.3201	72.58
2	24	6	8	$\cdots$	0	8.4306	107.06
3	18	2	5	$\cdots$	0	7.4107	160.21
$\cdots$							

Regression-based solver. We use the Non-negative Least Squares Regression to approximate the *CPU cost* C_i of different client transactions (at different tiers) from the following equation:

$$C_0 + \sum_i N_i \cdot C_i = U_{\text{CPU}} \cdot T, \tag{1}$$

where T is the length of the monitoring window; N_i is the number of transactions of the i-th type ($1 \leq i \leq M$); U_{CPU} is the average CPU utilization during this monitoring window; C_i is the average CPU service time of transactions of the i-th type and C_0 is the average CPU overhead related to system activities that "keep the system up".

To solve for C_i, one can choose a regression method from a variety of known methods in the literature. The non-negative least squares regression method used here is to find a set of non-negative values of C_i to minimize the squared error of the measured U_{CPU} and the fitted U'_{CPU} which is computed by applying the solutions of C_i into Eq. (1). Statistical regression works well for estimating the CPU demands of transactions that represent a collection of smaller objects. thus when direct measurement methods are not feasible.

Analytical model: Due to the session-based client behavior, a multi-tier system is usually modeled as a closed system with a network of queues where the queues represent different tiers of the application as shown in Fig. 1. The number of clients in the system is fixed. When a client receives the response from the server, it issues another request after certain think time. This think time is modeled as an infinite server Q_0 in Fig. 1. We use the results of the regression method to derive service time in each queue and parametrize our analytical model of queues. This closed system can be solved efficiently using Mean-Value Analysis (MVA) [1].

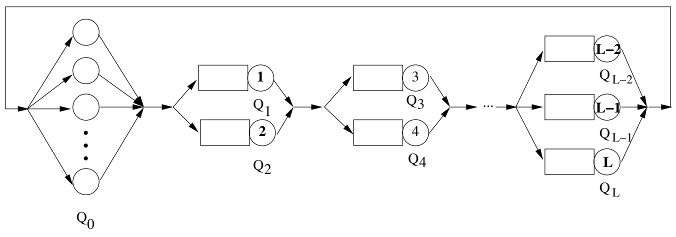

Fig. 1. Queuing network modeling of a multi-tier closed system.

The workload mix of real systems changes over time, hence service times could not be modeled as a fixed distribution for the entire lifetime of the system. Still, one can treat the workload as fixed during shorter time intervals (e.g., 1 hour). *R-Capriccio* executes the capacity planning procedure for each monitoring time window and then combines the results across these time points to get the overall solution.

III. Case Study

To validate *R-Capriccio*, we conducted a detailed case study using the access logs from two heterogeneous production servers that represent customized client accesses to a popular and actively used HP Open View Service Desk application. These traces have a detailed information about each processed request at the OVSD business portal during July 2006, including request arrival and departure time, request URL, and client session ID.

Overall, there are 756 different unique transactions (or transaction types). Consistently with earlier works, the studied workload exhibits a very high degree of reference locality: i.e., a small subset of site transactions is responsible for a very high percentage of client accesses, e.g.,

- the top 10 transaction types accumulate 79.1% of all the client accesses;
- the top 20 transaction types are responsible for 93.6% of the site accesses;
- the top 100 transaction types account for 99.8% of all site accesses.

Fig. 4 shows the percentages of these transaction types in the workload mix over time. Each point in these figures corresponds to the statistics of one hour. The figure shows that the transaction mix is not stable over time. During weekends, 20% of the entire transactions are of the second popular transaction type, while the transactions of the most popular type are only about 5%. During peak times in workdays, 40% of the transactions are of the most popular type, but even during the same workday the transaction mix changes from hour to hour.

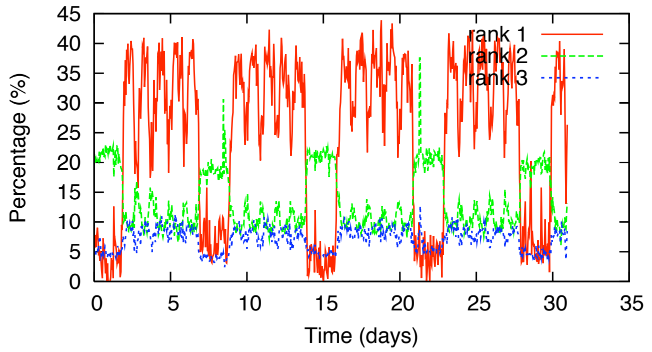

Fig. 2. Portions of the transactions belonging to the second top 3 popular transactions across time.

Solving a large number of equations with 756 variables might result in a long computation time. So, the question is whether accurate performance results can be obtained by approximating the CPU cost of a much smaller set of popular (*core*) transactions. To this end, we divided the OVSD trace into two parts. The first half is used as a training set to solve for the CPU cost C_i using the non-negative LSQ regression method. The second half is treated as a validation set. To capture the changes in user behavior we observe a number of different transactions over fixed length 1-hour time intervals. Fig. 2 (a), (b) show the CDF of the relative errors for training

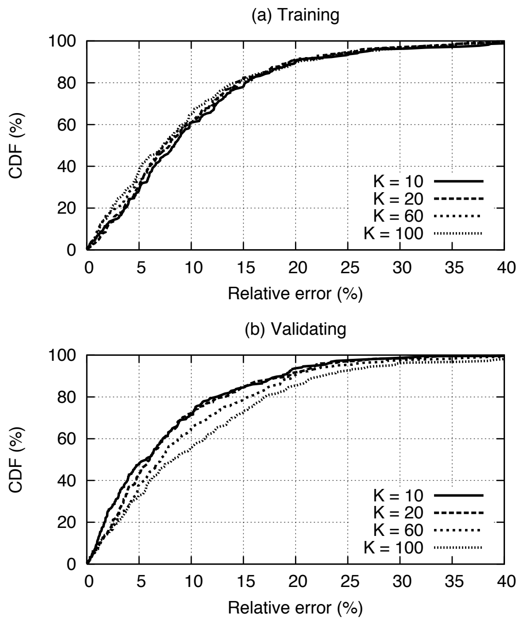

Fig. 3. Server 1. CDF of relative errors under a different number of of core transactions chosen for a regression method: (a) training set, (b) validating set.

and validating sets for server 1 (results for server 2 are similar to those reported in this figure). Overall, the non-negative LSQ regression achieves good results when it is is applied to approximate the CPU cost of the top 10, 20, 60, or 100 most popular transactions. For the training set, at least 70% of the points have relative errors less than 10%. At least 92% of the points have relative errors less than 20%, see Fig. 2 (a). The accuracy of the method for the validating set is only slightly worse, see Fig.2 (b). Prediction accuracy significantly improves when workload locality properties are taken into account, i.e., when we apply regression to the top 20 most popular transactions and omit the rest of rarely accessed transactions. This indicates that the regression method with core transactions has a unique ability to "absorb" some level of *uncertainty* or *noise* always present in real-world data. Combining the knowledge of critical workload features with statistical regression provides an elegant and powerful solution for performance evaluation of complex production systems with real workload.

We use the deduced costs C_i of the 20 most popular transaction types to approximate the average service time for each 1-hour time interval. These service times are the input to the third step of *R-Capriccio* – the analytic model. Fig. 4 shows the validation results by comparing the throughput of the analytic model and the measured transaction throughput of server 1. The analytic model captures the real system behavior well, i.e., 90% of the relative errors are below 18.7%.

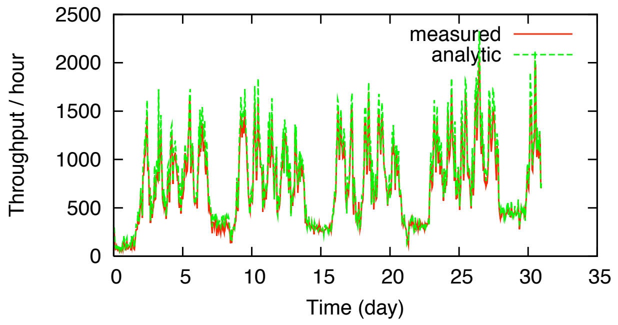

Fig. 4. Server 1. Throughput of transactions: measurements versus analytic model.

Assume that *R-Capriccio* is used to answer the following capacity planning question: How many clients can be supported by the existing system, providing the desirable performance guarantees, e.g., response time, is under Γ_R? The MVA-based analytic model with simple changes can answer this question for each hour. The smallest value among the results of all the hours is the global solution of this system. Table II gives the summary of results in our case study. Because server 2 has a faster CPU, it is expected to have a much higher capacity than server 1. Higher values of threshold allow for a larger number of clients to be supported by the system.

TABLE II

MAXIMUM NUMBER OF CLIENTS UNDER DIFFERENT Γ_R

Γ_R(sec)	Server 1	Server 2	Total
1	472	1349	1821
3	528	1478	2006
6	565	1534	2099
10	608	1580	2188

The capacity of the entire application server composed of these two heterogeneous servers is also determined by the load balancing policy. For example, if the SLA defines that the average transaction response time is not higher than 1 seconds, the studied application server can handle 1821 concurrent clients but only if the load balancer is aware of the heterogeneous capacity of these two servers, it can split the load proportionally to server capacity. If the load balancer partitions transactions equally, capacity reduces to 944, just half of the previous one. Such a big difference indicates the significant impact of a load balancing policy on system capacity as heterogeneous CPU speeds must be taken into account.

IV. CONCLUSION

In this paper, we present *R-Capriccio*, a new capacity planning framework which provides a practical, flexible and accurate toolbox for answering capacity planning and anomaly detection questions for multi-tier production systems with real workloads. More importantly, it can be used for explaining large-scale system behavior and predicting future system performance. We used the access logs from the OVSD application servers to demonstrate and validate the three key components of *R-Capriccio*: the workload profiler, the regression-based solver, and the analytic model. We believe that this framework also provides a natural extension to the existing Open View tool such as Open View Transaction Analyzer (OVTA) and can be easily integrated with it to offer a new attractive service to system administrators and service providers dealing with complex and dynamic multi-tier applications.

REFERENCES

[1] D. Menasce, V. Almeida, L. Dowdy. Capacity Planning and Performance Modeling: from mainframes to client-server systems. Prentice Hall, 1994.

[2] J. Rolia, V. Vetland. Correlating Resource Demand Information with ARM Data for Application Services. In Proc. of the ACM Workshop on Software and Performance, 1998.

[3] T. Kelly. Detecting Performance Anomalies in Global Applications. Second Workshop on Real, Large Distributed Systems (WORLDS'2005), 2005

[4] T. Kelly, A. Zhang. Predicting Performance in Distributed Enterprise Applications. HP Labs Tech Report, HPL-2006-76, May 2006.

[5] Daniel Villela, Prashant Pradhan, Dan Rubenstein. Provisioning Servers in the Application Tier for E-Commerce Systems. In Proc. of IWQoS'04, Montreal, Canada, 2004.

[6] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic Provisioning of Multi-tier Internet Applications. In Proc. of the 2nd IEEE International Conference on Autonomic Computing (ICAC-05), Seattle, June 2005.

[7] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An Analytical Model for Multi-tier Internet Services and its Applications. In Proc. of the ACM SIGMETRICS'2005, Banff, Canada, June 2005.

[8] T. M. Kachigan. A Multi-Dimensional Approach to Capacity Planning. In Proc. of CMG Conference 1980. Boston, MA, 1980.

[9] The Workload for the SPECweb96 Benchmark. URL http://www.specbench.org/osg/web96/workload.html

[10] TPC-W Benchmark. URL http://www.tpc.org