Predicting System Performance for Multi-tenant Database Workloads

Mumtaz Ahmad
University of Waterloo
m4ahmad@uwaterloo.ca

Ivan T. Bowman
Sybase, an SAP Company
ibowman@sybase.com

ABSTRACT

Database consolidation is gaining wide acceptance as a means to reduce the cost and complexity of managing database systems. However, this new trend poses many interesting challenges for understanding and predicting system performance. The consolidated databases in multi-tenant settings share resources and compete with each other for these resources. In this work we present an experimental study to highlight how these interactions can be fairly complex. We argue that individual database staging or workload profiling is not an adequate approach to understanding the performance of the consolidated system. Our initial investigations suggest that machine learning approaches that use monitored data to model the system can work well for important tasks.

Categories and Subject Descriptors

D.4.8 [**Performance**]: Modeling and prediction; H.2.m [**Database Management**]: Miscellaneous

General Terms

Experimentation, Performance

1. INTRODUCTION

Capacity planning is often an important problem for DBMS customers. This problem is becoming increasingly important because database consolidation is gaining wide acceptance as a means to reduce cost and complexity of managing the database systems. The customers run multiple workloads on a single machine to take advantage of improvements in machine capacities, network speed and reliability. The consolidated databases share resources and compete with each other for these resources.

In order to better motivate the problem, we first describe an example. A company "BSalon" provides software, billing, and member services tailored for beauty salons and health spas. The staff of a customer salon or spa use the BSalon software at the point of sale to track inventory and appointments that are used. They also use it for employee time reporting. Salon managers generate reports and forecasts that analyze historical usage. BSalon stores all of the data for an individual customer in a single database file, and hosts all of the data files in their data center. The workload varies between customers of BSalon because of differences in scale (single-salon customers up to national chains of hundreds of salons) and because of customizations of the database schema and operations. The cost of machine resources can be minimized if multiple customer databases are hosted by a single machine. A single machine can host hundreds of databases for smaller customers.

When consolidating workloads in this way, database administrators need answers to questions such as the following:

- Can a new workload be added to an existing machine that is already running a mix of workloads without a resource becoming the bottleneck?
- If a machine is overloaded, which workloads should be moved to other machines in order to restore performance to desired levels?
- Should replicas be created for a database to scale-out the workload for a database? Where should these replicas be placed?

In our experience with SQL Anywhere customers, we have seen that while these questions are quite important to database administrators, they are notoriously hard to answer. The database administrators need significant background knowledge and careful experimentation to come up with best practices [11]. The consolidation of multiple workloads onto a single machine adds an extra dimension of complexity, in particular as the DBA would like to make predictions about configurations that have never been tried before.

While the problems are challenging, DBAs currently do answer these questions using techniques that are generally robust but often inefficient. For example, a DBA may intentionally under-utilize machine capacity in order to be confident that the service agreements will be met.

Problem Definition: There can be many different approaches for consolidating tenant databases onto a single host machine: (a) Each tenant database can run its own database server. This achieves a high level of isolation as each tenant can be managed and maintained individually. (b) Alternatively, the tenant databases can share a database sever and can have shared schema in that server. In this

DBTest '11 June 13, 2011 Athens, Greece

setting, a given table can contain records from multiple tenants and, typically, tenant identifiers are needed to associate records with the tenants. (c) Finally, the tenant databases can share a database server, but each database has its own schema. We use this last approach (c) for our experimental settings.

When multiple tenants are co-located in a database server running on a host machine, the tenants share system, OS and database resources. When workloads are executed concurrently, these workloads interact with each other on shared resources. Concretely speaking, we want to know how resource metrics of the underlying system are impacted by these workload mixes.

In this work, we argue that individual database staging or workload profiling is not an adequate approach to understand the performance of the consolidated system. We select synthetic workloads that are well behaved with regular characteristics and we show that it is difficult to predict the performance of these workloads when they are combined. We describe an investigation into the system performance of mixed workloads. Our initial results suggest that techniques based on machine learning offer some hope.

Outline: This paper is organized as follows. In Section 2 we discuss related work. In Section 3 we present examples from our experimental study that show that when workloads are consolidated, it is difficult to predict system performance. In Section 4 we show the results of applying machine learning techniques to the problem of predicting system resource metrics for given workloads. We present a discussion of the assumptions and avenues for future research in Section 5. Finally we conclude in Section 6.

2. RELATED WORK

Developing multi-tenant solutions for database consolidation is becoming an active area of research [7] [10]. As we discussed in Section 1, there can be many different approaches to databases consolidation. In [8], the authors experimentally compare the approach where the tenants share the database server but have separate schemas to the approach where each tenant is running its own server. They show that the first approach is significantly better. We also use this first setting for our experimental study. In [8] the authors propose the use of staging and analytical models to solve the problem of database placement in this multi-tenant setting. In Section 3 we discuss in detail that this approach has its drawbacks when workloads are contending for resources and interactions are complex.

There are not many works that talk about system performance metrics for database consolidation. Similarly, up to the best of our knowledge there are not works that would discuss interactions among workloads in a general way. The idea of modeling and exploiting interactions among concurrent queries was presented in [1] [2] [3]. And it has been used for scheduling workloads [2] [3] [4], and predicting workload completion times [5] [6]. These works focus on query completion time as a metric of interest and capture interactions at the granularity of concurrent query types. In contrast, this paper focuses on concurrently executing workloads and performance of resource metrics of the host system.

In [9], the authors employ machine learning to predict multiple performance metrics for database queries. They train the models on data consisting of query plan features and performance metrics and then predict performance metrics for given query plan features. Their work focuses on single queries and it is not straight forward to extend these techniques for concurrently executing workloads. The problem of extracting useful plan features for performance prediction is a non-trivial problem. We have discussed it further in Section 3 and Section 5.

In [17], the authors discuss resource provisioning and anomaly detection for multi-tier applications. In particular, they use a linear regression model for estimating CPU utilization demand. This work, however, defines a transaction mix consisting of all the transactions that run during a monitoring window without considering which of these transactions actually ran concurrently and competed for resources. Similarly queuing theoretic models have been proposed for capacity planning in multi-tier applications [14] [15]. However, when workloads are executing in concurrent setting and bottlenecks can shift from one resource to another, it is non-trivial to estimate the parameters for these queueing models.

Another approach to multi-tenant solutions considers mechanisms that minimize the interference between concurrent workloads on the same server. For example, the Lachesis project [13] automatically detects and exploits device characteristics to provide robust performance characteristics in the presence of concurrent query execution. The Argon project [16] uses cache partitioning with disk prefetching, writeback and scheduling to insulate sequential stream efficiency from concurrent disk access. These mechanisms are important tools in implementing database consolidation, but they do not address the ways that workloads may change behaviour when they are run concurrently (for example, due to query plan changes). While complete isolation between tenants prevents this type of interaction, this complete isolation comes at the cost of reduced sharing of resources.

3. RESOURCE CONSUMPTION IN MULTI-TENANCY

In multi-tenant database systems, a database is assigned to a database server where it would be running with other databases hosted on that server. An important consideration is how this assignment of co-located databases is going to impact the resource consumption at the host machine. Unfortunately, there is no systematic way to answer this question.

When databases are co-located, the workloads interact with each other. The interactions can be complex and at many different levels. One commonly used approach is to over-provision the system by computing the individual resource consumption profiles of the workloads. Thus a database and its workload are placed in a staging environment where resource consumption of the workload on a dedicated host is monitored. Once these resource consumption profiles of the workloads are known, an estimate of the capacity of the target system can be made. However, there are some principled flaws in this approach. The interactions among the workloads can happen at many different resources and they can shift from one resource to another. When we profile the workloads, we may see a read-only workload on a database, but when this read-only database is added to a multi-tenant server, it may end up causing a lot of writes in the server because this new database has to share the memory with other workloads. Further, consider the case of

metric	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}
CPU Utilization (%)	28.19	25.38	25.28	25.20	26.10	25.31	50.07	75.08	62.19	58.57	57.86	63.12
Disk milliseconds/Transfer	16.22	6.18	5.92	6.74	14.95	6.37	5.33	6.06	5.93	6.31	6.59	6.86

Table 1: CPU utilization and disk transfer time for different workloads.

	workload mix	CPU Utilization (%)	Disk milliseconds/Transfer
mix 1	(W_2, W_3, W_4)	26.70	7.80
mix 2	(W_{10}, W_{11}, W_{12})	95.76	6.44
mix 3	$(W_1, W_2, \cdots W_{12})$	35.30	53.27
mix 4	$(W_1, W_2, \cdots W_6, W_7, W_8, W_9, W_{11})$	45.86	74.63
mix 5	$(W_1, W_2, \cdots W_6, W_9, W_{10}, W_{11})$	44.43	63.96

Table 2: CPU utilization and disk transfer time for different workloads mixes.

query optimizer that selects the execution plan for a query. The query optimizer uses a cost model to distinguish the candidate plans. All commercial databases aim to put self managing features in their optimizers and cost models. The cost models are augmented by taking into account the current state of the system. This effectively means that plans chosen for queries in a workload can depend upon what other workloads are running on co-located databases. As a matter of fact, within a multi-tenant environment, the plans may differ from one execution to another, because even if we settle on a co-location assignment of databases, the workload mix that is executing at a particular instance can be quite time-dependent. The change in execution plans means that there may be change in resource consumption profiles when a workload runs in different workload mixes.

Another approach can be to build analytical models for different resources and predict resource consumption for a given load. However, this has the limitation that the models can only be hardware-specific. Further, if we are hosting our multi-tenant setup in a data-center and there are virtualization layers underneath, then it may not be even possible to develop such a model. We believe that for many practical scenarios the analytical modeling may be just infeasible.

In this section we present the results of our study that shows that even for the most basic CPU and disk performance metrics it is difficult to predict the workload interactions and come up with useful heuristics. We first detail our experimental set up and assumptions, and then report the results.

In our experiments we use Sybase SQL Anywhere version 12 as our database sever. The server is hosted on a virtual machine running Windows Server 2003 x64 edition. The virtual machine is allocated 4GB physical memory and 4 cores of Intel 2.13 GHz processors

We create 12 different TPC-C and TPC-H databases. There is one-to-one mapping between the workloads and the databases. We explain these databases and accompanying workloads as follows.

We use TPC-H to create 6 different databases. Three databases are of size 1GB and remaining three are that of 500MB. For each database size, we profile the individual TPC-H queries and group them according to their average CPU utilization. We define three classes of queries: light CPU utilization, medium CPU utilization, and all TPC-H queries. We use the QGEN utility of TPC-H to generate multiple instances with varying parameters for queries belonging to each class. Then we create workloads from instances of each query class. Thus a database and its workload can be uniquely identified by <db_size, CPU utilzation>. For example, <500MB, light CPU utilization>, <500MB, all TPC-H queries> and <1GB, all TPC-H queries> are three databases and accompanying workloads

For TPC-C we also create 6 different databases. We use three settings for the number of warehouses: 1, 5 and 10. TPC-C emulates an OLTP application environment and a think time distribution can be used to model user's sessional behavior. For each of the above three cases, we vary number of transactions and think time. The variation in think time changes the time-varying resource consumption profile. Thus we have a database uniquely identified by < num_warehouses, num_transactions, thinktime >.

In order to monitor the system we use operating system's performance monitor (Windows Perfmon). We also use SQL Anywhere's stored procedures to collect server and individual database metrics. We collect statistics for system, server and database performance. Here, we limit the discussion to CPU and disk performance metrics only. In particular we focus on two most important metrics for CPU and disk: average CPU utilization and disk milliseconds/transfer. Average CPU utilization gives us the utilization of the machine for all processors. The metric disk milliseconds/transfer is a direct measure of disk response time. This measurement reflects the complete round-trip time of the request with queuing delays included.

Table 1 shows CPU utilization and disk transfer time for 12 workloads when run in isolation. Table 2 shows these metrics for different workload mixes. There are very interesting observations that can be made from these tables.

Workload mixes *mix* 1 and *mix* 2 show that simple additive models do not work for both metrics. In *mix* 1, the workloads W_2, W_3 and W_4 are running together. In Table 1 we can see that the CPU utilization for all these workloads is around 25% when they are running alone. When these workloads are running together in Table 2, the CPU utilization is still the same. Similarly there is a minuscule increase in disk transfer time, but overall it is very clear to see that when these workloads are running together in a mix, there is no major impact on the disk and CPU performance for the metrics that we are monitoring. Now contrast this observation with *mix* 2 where W_{10}, W_{11} and W_{12} are running together. Once again, we can see that the metrics are not simply additive. However notice the significant increase in CPU utilization while the disk transfer time is not impacted at all. From these examples, we can see that if we are just given the individual workload profiles, it can be very difficult to come up with an estimation of the resource utilization in workload mixes.

The workload *mix* 3 presents the case when all 12 work-

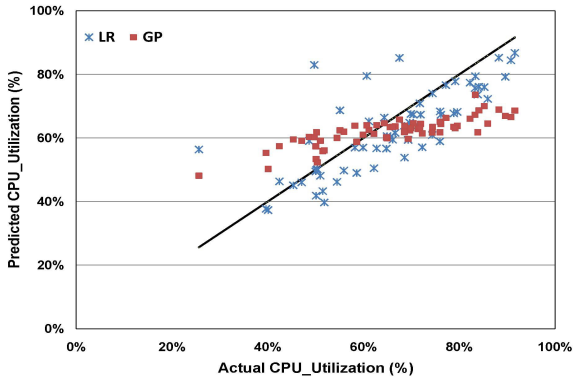

Figure 1: Scatter plot for actual and predicted CPU utilization

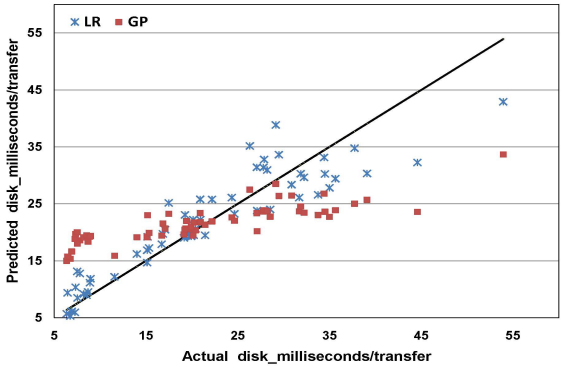

Figure 2: Scatter plot for actual and predicted disk transfer rate

loads $W_1, W_2, \cdots, W_{12}$ are running in a mix. The values for the metrics are not surprising. When all the workloads are running together, there is maximum contention for the resources, and a resource either becomes bottleneck or it is under-utilized because of another resource becoming the bottleneck. Thus we see that there is significant increase in disk transfer time and CPU seems to be under-utilized. However, this is not the worst case of disk performance that we have observed. The workload *mix* 4 presents the worst case of the disk performance that we observed in our experimental study. In this mix, ten different workloads, $W_1, W_2, \cdots W_6, W_7, W_8, W_9$ and W_{11} are running. There is a very interesting observation here. In *mix* 4, we have a subset of total workloads (10 out of 12) running together and CPU utilization is around 45.86% while disk transfer time is 74.63 milliseconds per transfer. Thus one would think that since disk has already become a severe bottleneck, running another workload in addition to these 10 workloads will further degrade its performance. However, as we have seen in *mix* 3 where all 12 workloads are running, this is not the case. In *mix* 3, the disk is still a bottleneck but it is performing slightly better as compared to *mix* 4: the disk transfer time in *mix* 3 is 53.27 milliseconds per transfer as compared to 74.63 milliseconds per transfer in *mix* 4. On the other hand CPU in *mix* 3 is more under-utilized as compared to *mix* 4. Similar observations can be made for *mix* 5

Based on the examples that we have presented we can draw the following conclusions:

- When workloads are running concurrently, the resource interactions can be fairly complex.
- Simple additive models do not work even for the most basic and important resource metrics.
- The bottlenecks and resource utilization can keep on shifting from one mix to another.

Next we present the initial results of our efforts to predict the resource metrics for concurrently running workloads.

4. PERFORMANCE MODELING

Our objective is to predict the resource metrics of the system when databases are consolidated in multi-tenant settings. In our case, we have 12 workloads. It means there are $2^{12} = 4096$ workload combinations. So if we want to absolutely answer the question about utilization of a resource, then we would have to force the data center to try all these different combinations of co-location assignment for databases. This is an infeasible task with ever increasing number of databases. We show here that in practice we do not need an exhaustive set of these combinations

Fundamentally, given a mix of workloads we want to predict what a particular resource metric is going to be. This can be thought of a *what-if* question of the form

$$\hat{y} = f(W_1, W_2, \ldots, W_n)$$

where $\hat{y}$ is the predicted value of the performance metric for which we want to answer the *what-if* question. For every performance metric of interest, we collect samples of the type $(y, W_1, W_2, \ldots, W_n)$, where y is the actual value of the metric. Once we have collected these samples, the *what-if* question becomes a regression problem, where our goal is to train a suitable model $f(.)$ on these samples. There are many candidate model approaches from the machine learning literature and accuracy can vary widely among these model choices. In our investigations we have tested linear regression (LR) and Gaussian processes (GP). As we discuss later, these models can give reasonable accuracy for the metrics that we are monitoring.

We collect 200 workload sample mixes and use 135 for training and remaining 65 for testing. The figures 1 and 2 show the scatter plot for actual and predicted values for our two resource metrics.

We measure the predictive accuracy of the models by mean relative error *(MRE)*. The mean relative error is given by $\frac{1}{S}\sum_{i=1}^{S}\frac{|\hat{y}_i - y_i|}{y_i}$. Table 3 shows *MRE* for CPU utilization and disk milliseconds/transfer. An error of less than 20% is highly desirable and difficult to achieve for resource consumption metrics [9]. Both modeling approaches are very effective in predicting the CPU utilization of the system. For disk transfer rate, we find that linear regression is reasonably accurate in predicting the disk milliseconds/transfer. The error is relatively higher for Gaussian processes for disk milliseconds/transfer.

The metric MRE gives us an idea of the average error, but an interesting question is how rigidly one should interpret this number in practice? The reason is that, even though it is difficult to understand and predict resource interactions, the

metric	LR	GP
CPU Utilization (%)	12.83	15.44
Disk milliseconds/Transfer	17.41	48.03

Table 3: MRE for modeling approaches.

metric	LR	GP
CPU Utilization (%)	11.10	14.10
Disk milliseconds/Transfer	8.42	11.42

Table 4: MRE for modeling approaches, after applying heuristics.

problem that a DBA wants to solve is quite flexible in many practical scenarios. Consider the case of CPU utilization. Frequently a DBA is not interested in absolute error of the prediction. As we motivated in the beginning of this paper, he or she may want to know, when a group of databases are co-tenants in a server, is CPU going to be a bottleneck? Is CPU highly, moderately or lightly utilized? For instance, if the actual CPU utilization of a database consolidation onto a machine is 5% and our model predicts it to be 10%, there is an error of 100% in the prediction. However from the perspective of a DBA this may be the "perfect" prediction. It tells the DBA conclusively that the system is going to be lightly loaded and it is good enough for his task at hand.

The case of disk transfer time is even more interesting. We consulted user manuals and online technical support blogs to have a better understanding of this metric. It emerges from the best practices, that a disk transfer rate of less than 15 ms/transfer is considered "ideal" and it implies excellent health of the disk system. Similarly, a disk transfer rate of 20 ms/transfer is typically considered acceptable. This effectively means that if actual disk transfer time is 5 ms/transfer and our prediction is 7.5 ms/transfer, it may be a case of misplaced focus, if we try hard to bring down this error of 50% within 10%-20% error range.

In a nutshell, the values for resource metrics can only be interpreted in the context of the system settings, and in most practical cases simple heuristics can help a lot in a better interpretation of the modeling results. In our experimental study we are already getting reasonable accuracy for both metrics by using linear regression. The value of *MRE* for Gaussian process for disk transfer time is 48.03%. We add the two following heuristics to better interpret the predicted data of our CPU and disk metrics: (a) for CPU, we ignore the error if the predicted is within 5% of the actual, (b) for disk, if both actual and predicted transfer times are under 20ms, we ignore the error. When we process the data the results are shown in Table 4. Notice that the error for for Gaussian process, that was of more than 48% is now less than 12%. Basically this tells us that, for a given assignment of co-located databases, even by using Gaussian process, we can predict with high degree of accuracy whether disk is becoming bottleneck or not. In many practical scenarios, this may be all that is needed.

5. DISCUSSION

As discussed above the fundamental problem is that we want to understand the system performance for database assignment in multi-tenant environments. In this section we discuss our assumptions and avenues for further research.

Collecting training data: In Section 4, we assume that we have training samples and we can learn a model from these training samples. An important question is how and when these samples of workload mixes are collected? We believe that there can be two approaches for this: *active sampling* and *passive sampling*. In passive sampling, the training data is collected by just passively monitoring the production systems in data center. In this setting it can be assumed that the DBA will be assigning databases to different hosts based on heuristics and best practices. In case of active sampling, in addition to using the best practices for database assignment, a DBA can actively try to cover space of database combinations. For instance, in our study we had 12 databases, and $2^{12} = 4096$ combinations. A DBA can use approaches from the design of experiments to cover this space in an efficient way. There are many techniques in the design of experiments that try to cover the space with respect to certain criteria, e.g., Latin Hypercube design, maximin distance design and minimal distance design, stratified random sampling etc. [12]. In our study we use simple random sampling to pick a set of 200 mixes from 4096 mixes. This effectively simulates the behavior when a DBA is randomly assigning the databases to multi-tenant hosts. Further note that there is a non-stationary nature of concurrent workload mixes for a given assignment. When we assign W_1, W_2, W_3 and W_4 to a multi-tenant host, in theory, we can collect monitoring data for all $2^4 = 16$ mixes from this assignment.

Defining a workload: In this paper we have assumed a one-to-one mapping between workloads and databases. We have used standard TPC benchmarks which give us this flexibility that a start and end time of the workload can be defined clearly and steady behavior is assumed in between. In real world scenarios, one may not have this level of control and flexibility, and the workloads can have time-dependent nature and activity cycles. For a given database, the day time activity can be quite different from evening and weekend activity. Similarly the activity during certain vacation periods may be entirely different from typical activity. In some cases these boundaries may not be even clear and properly defining a workload becomes a non-trivial problem. In our study we work with clearly defined workloads by using standard benchmarks.

Workload features for prediction: In Section 4 the input to model $f(.)$ is the workload vector $(W_1, W_2, \ldots, W_n)$. It is essentially a bit vector where a value of 1 for a workload indicates that the workload is present in the mix. A very interesting question is what other features of a workload or database we can use for predicting the system performance? For instance, in [9], the authors use query plan features for predicting multiple performance metrics for database queries. However, the work in [9] does not take into account the concurrent execution of workloads and queries. The fact that plans may change in concurrent environment and the time-dependent nature of workloads means that the problem of extracting useful plan features from workloads for performance prediction is a non-trivial problem. We believe that it is an important avenue for further research.

Choice of metric: It is very important to choose the proper metric for task at hand. For instance, in order to understand disk performance we monitor disk milliseconds/transfer. It is a direct metric that gives us first hand information about disk response time. Our experimental results show that our modeling techniques give us reasonable accuracy while pre-

dicting this metric. Further, an interpretation of this metric is very intuitive and helps us understand the performance of our system. Now consider another disk related metric: average disk queue length. This is a secondary metric that Windows perfmon gives us after applying Little's Law on other metrics and in certain scenarios it may have no meaning at all because the assumptions of underlying formula are invalid. Thus the proper choice of metric can be very important for a better understanding of system performance

6. CONCLUSIONS

In this paper, we show that machine learning techniques can be useful in predicting the performance of the underlying system when workloads are running on co-located databases in multi-tenant settings. We show that in this multi-tenant setting the workloads contend for shared resources and the nature of these interactions can not be predicted by developing individual workload profiles. One needs to take into account the co-located databases and concurrent mix of workloads. We present the results that show that machine learning approaches can be used to predict the system performance with a reasonable degree of accuracy. We outline research directions for a better understanding of the problem of database assignment in multi-tenant environments.

7. REFERENCES

[1] M. Ahmad, A. Aboulnaga, and S. Babu. Query interactions in database workloads. In *DBTest*, 2009.

[2] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala. Modeling and exploiting query interactions in database systems. In *CIKM*, 2008.

[3] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala. Qshuffler: Getting the query mix right. In *ICDE*, 2008.

[4] M. Ahmad, A. Aboulnaga, S. Babu, and K. Munagala. Interaction-aware scheduling of report-generation workloads. *The VLDB Journal*, 2011.

[5] M. Ahmad, S. Duan, A. Aboulnaga, and S. Babu. Interaction-aware prediction of business intelligence workload completion times. In *ICDE*, 2010.

[6] M. Ahmad, S. Duan, A. Aboulnaga, and S. Babu. Predicting completion times of batch query workloads using interaction-aware models and simulation. In *EDBT*, 2011.

[7] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques. In *SIGMOD*, 2008.

[8] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich. Relationalcloud: a database service for the cloud. In *CIDR*, 2011.

[9] A. Ganapathi, H. A. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. I. Jordan, and D. A. Patterson. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *ICDE*, 2009.

[10] M. Hui, D. Jiang, G. Li, and Y. Zhou. Supporting database applications as a service. In *ICDE*, 2009.

[11] G. Paulley and I. T. Bowman. Capacity planning with SQL Anywhere. Technical Report 1056535, Sybase, an SAP Company, March 2008.

[12] T. J. Santner, W. B., and N. W. *The Design and Analysis of Computer Experiments*. Springer-Verlag, 2003.

[13] J. Schindler, A. Ailamaki, and G. R. Ganger. Lachesis: robust database storage management based on device-specific performance characteristics. In *VLDB*, 2003.

[14] B. Urgaonkar, G. Pacifici, P. J. Shenoy, M. Spreitzer, and A. N. Tantawi. An analytical model for multi-tier internet services and its applications. In *SIGMETRICS*, 2005.

[15] B. Urgaonkar, P. J. Shenoy, A. Chandra, and P. Goyal. Dynamic provisioning of multi-tier internet applications. In *ICAC*, 2005.

[16] M. Wachs, M. Abd-El-Malek, E. Thereska, and G. R. Ganger. Argon: performance insulation for shared storage servers. In *FAST*, 2007.

[17] Q. Zhang, L. Cherkasova, G. Mathews, W. Greene, and E. Smirni. R-Capriccio: a capacity planning and anomaly detection tool for enterprise services with live workloads. In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, 2007.