DBSeer: Resource and Performance Prediction for Building a Next Generation Database Cloud*

Barzan Mozafari
MIT - CSAIL
barzan@csail.mit.edu

Carlo Curino
Microsoft - CISL
ccurino@microsoft.com

Samuel Madden
MIT - CSAIL
madden@csail.mit.edu

ABSTRACT

Cloud computing is characterized by shared infrastructure and a decoupling between its operators and tenants. These two characteristics impose new challenges to databases applications hosted in the cloud, namely: (i) how to price database services, (ii) how to isolate database tenants, and (iii) how to optimize database performance on this shared infrastructure. We argue that today's solutions, based on virtual-machines, do not properly address these challenges. We hint at new research directions to tackle these problems and argue that these three challenges share a common need for accurate predictive models of *performance and resource utilization*. We present our approach, called *DBSeer*, with our initial results on predictive models for the important class of OLTP/Web workloads and show how they can be used to address these challenges.

1. INTRODUCTION

The defining attributes of cloud computing are a shared infrastructure, and a layer of indirection that decouples the responsibility of running the infrastructure from the applications that use it. This decoupling simplifies some aspects of application deployment, as the cloud provider manages reliability, security, and mapping of software processes onto physical machines. Additionally, sharing infrastructure and administration has clear economic benefits. In most cloud services, these properties are delivered via virtualization, where a fraction of the CPU, disk, and memory resources on a physical machine are dedicated to each application running on it. In these cases, the cloud provider manages the placement of the virtual machines (VMs) and the operation of the physical infrastructure under them, while users manage the operating systems and applications running in the VMs. VMs provide "hard isolation" as there is little or no sharing of resources or interaction between applications hosted on the same physical machine. Unfortunately, this way of providing cloud services is non-ideal for performance-sensitive applications like databases for three key reasons:

1. For ease of accounting, cloud vendors require customers to do capacity planning in terms of virtualized hardware resources instead of measurable performance metrics of a database.

*This research is supported by a grant from Quanta Computer, Inc. The authors are also grateful to participants of Dagstuhl seminar on Database Workload Management.

6th Biennial Conference on Innovative Data Systems Research (CIDR Õ13) January 6-9, 2013, Asilomar, California, USA.

2. Hard isolation prevents multiplexing and sharing of resources when multiple databases are co-located on a machine, causing considerable over-provisioning and commensurate loss of performance.

3. Higher degree of multi-tenancy and the decoupling of cloud service providers from app-developers make tuning/provisioning of the DBMS in the cloud more challenging than in conventional database deployments.

These three limitations are primary reasons why many businesses are reluctant to use the VM-based database-as-a-service (DBaaS) [10]. Our belief is that, to fully realize the potential of cloud computing in the database world, an alternative approach is needed. Part of the solution is to avoid VMs, which introduce a performance bottleneck. By using the same DBMS to host multiple databases owned by different users of the cloud, it is possible to avoid some of the shortcomings of a VM-based approach. Initial results have shown that this alternative can have substantial benefits [5, 6].

Simply discarding VMs and running multiple databases in the same DBMS instance, however, does not address the three limitations discussed above. In this paper, we argue that what is needed is innovation in several key technical areas:

Resource Attribution and Application-level SLAs. New research is needed to provide mechanisms for mapping high-level metrics such as transaction throughput and latency into underlying resources, e.g., CPU cores, RAM capacity, disk I/O, network bandwidth. This will enable providers to expose billing schemes and capacity planning tools that are more understandable to users.

Soft Isolation. In order to avoid the performance drawbacks of VM-based hard isolation, we need to provide alternative mechanisms that achieve soft-isolation, i.e., allow for resource sharing when the system is underutilized, but enforce quotas when necessary. We argue this could be done as a closed-loop admission-control in the DBMS, which in turn requires a deep understanding of the performance-resource tradeoff of users' workloads.

Multi-tenant, Multi-machine DBMS Tuning. Workload-specific DBMS tuning is crucial for high performance. In a decoupled, multi-tenant setting such as a DBaaS, neither providers nor users have the right combination of visibility/access/expertise to perform proper tuning and provisioning. In current DBaaS offerings, users are allowed to select one of a small number of fixed configurations. We show that this is inefficient, and argue for new, automated solutions that achieve workload-specific DBMS tuning in a multi-tenant fashion, where we observe multiple workloads running on a pool of machines, extract performance/resource characteristics of each workload (without ever observing it in isolation), and solve workload-to-DBMS placement and DBMS tuning problems (i.e., selecting groups of workloads and assigning them to DBMS configurations in an optimal fashion). We call this *multi-tenant, multi-machine DBMS tuning*.

Each of these three areas requires substantial innovation. However, there is a key enabling technology that underlies these three

problems: **Resource and Performance Prediction.** Specifically, all the above problems require predictive models that, for a given set of workloads running on a machine, can predict the resource utilization (e.g., CPU usage, disk IOs/sec) required for a certain level of performance (e.g., transactional latency and throughput), and can estimate the achievable performance with a given set of resources. These predictive models will help the three aforementioned problems as follows: (i) for the attribution/SLA problem, they will allow cloud providers to determine how much to charge per each type of transaction run by a user; (ii) for soft isolation, they will allow the database service to determine what resources are needed to meet certain SLAs, to co-locate tenants appropriately, and to perform admission control; (iii) for tuning, accurate estimation of the resource requirements of a given workload, will allow for setting database parameters appropriately.

Building such models is tricky, as the performance of a given workload or transaction varies depending on other transactions running in the system. For example, a given transaction may do many I/Os when running alone, but fetch few pages in the presence of another transaction that reads the same data. Alternatively, a given transaction may do more I/Os in the presence of other transactions that access many pages and put pressure on the buffer pool.

We have developed a tool for resource and performance prediction, called DBSeer. In this paper, we describe encouraging preliminary results, showing that DBSeer can solve the performance modeling problem for transactional workloads in MySQL (Section 2). We then describe how such models might be used to tackle the three technical challenges described above, in Sections 3–5.

2. RESOURCE PREDICTION

To fully realize the next generation of database clouds, we need to be able to accurately predict resource consumption and performance of a given workload. Specifically, we focus on answering questions like: "How many disk IOs/sec are needed to run TPC-C at 2000 transactions per second (TPS) on this DBMS configuration?" These questions are of central importance in all three of the challenges described above (see Sections 3–5).

In our preliminary efforts, in DBSeer, we have focused on the hard-class of highly concurrent transactional (a.k.a. OLTP) workloads. With the exception of [1] where simple regression techniques have been examined for OLTP settings, prior work on performance prediction has only focused on OLAP databases [2, 7]. What makes OLTP uniquely challenging is the presence of a large number of highly concurrent, short-lived, and lock-prone transactions that do small, random reads and writes. Due to high degree of concurrency and competition for non-linear resources (e.g., locks, cache, I/O), even small changes in load can lead to a large change in the performance of transactions.

In DBSeer, we have developed prediction models for MySQL that we empirically show to be highly accurate. Although more research is needed to generalize these models to other DBMSs, our initial results are highly promising as even these MySQL-specific results can benefit millions of MySQL users. In the remainder of this section, we briefly summarize our approach in DBSeer, which is based on a combination of unsupervised machine learning, regression analysis and resource-specific models to predict the resource utilization and performance metrics of transactions.

2.1 Clustering Transactions

Transactional workloads are mostly composed of many instances of a few transaction templates, where each template involves the same or slightly different operations, but with the different constants in each instance. Based on this observation, DBSeer first clusters transactions (from a query log) into classes that execute the same or similar templates, and characterizes a workload by the rate and proportion of transactions that it runs from each template. For example, in TPC-C there are 5 classes (corresponding to the five transaction types in the benchmark). With this automatic clustering, DBSeer can classify and reason about incoming transactions. Moreover, once the user gives a name to each extracted class (say, by looking at a few instances of each), pricing schemes can be made more intuitive to users, e.g. by assigning a cost to each transaction based on its human-recognizable name.

2.2 Modeling Resources

In DBSeer, we have developed models for predicting CPU, RAM, network, disk I/O, and number of acquired locks per each table. These profiles are built by only observing query logs and OS-level statistics of a system *in situ*, i.e., as it is currently deployed, without controlling the system's execution. DBSeer passively observes the natural fluctuations in the incoming transactions and the corresponding change in the usage of different resources, and uses this information to estimate parameters of its resource models. DBSeer can then use these models to make predictions about resource usages for a certain mix of transactions at a certain throughput.

DBSeer uses simple linear models for CPU, network usage and sequential I/O and they are fairly linear for typical OLTP workloads. Other models, especially for predicting RAM, random I/O, and degree of lock contention are considerably more sophisticated. Due to space constraints, here we only focus on our disk-write model.

Estimating Disk Writes. Disk writes in transactional databases consist of log writes and data writes. Log writes are mostly sequential, and linearly proportional to the number and type of transactions executed (since each transaction writes the same number of log records regardless of other concurrent transactions). As a result, log writes are largely independent of configuration parameters (with the exception of group-commit features). We thus model them using linear regression. Data writes, on the contrary, are non-linear, tend to be random, and are dependent on a multitude of configuration parameters (e.g., buffer-pool size, log-size, flushing algorithms).

DBSeer models data writes as follows. First, to predict the rate at which pages are being dirtied, DBSeer uses a cache model (e.g., [11]). Then, to model data writes, DBSeer relies on a notion of *conservation of flow*, which says that, in the steady state, the rate at which pages are dirtied must match the rate at which dirty pages are flushed. The challenge is in predicting exactly when these flushes will happen, given an input transaction mix and rate. This question can be answered using a monte-carlo simulation, but this is typically very expensive. Instead, we have developed a simple iterative algorithm that converges faster, provides very accurate predictions, and also offers insight into how different transactions contribute to the overall I/O seen by the system.

The details of the algorithm are beyond the scope of this vision paper—the interested reader can find more details in [9]. Instead, we present some evidence that this approach provides accurate estimates on two benchmarks: TPC-C and Wikipedia.

We evaluate DBSeer's model by training on logs of a given workload (TPC-C and Wikipedia) running at a certain transaction rate and mixture, and testing the accuracy at predicting the same workload when running at a different rate and with a different transaction mixture. Note that both transaction rate and mixture have been changed: this makes for a challenging scenario.

Fig 1 shows the accuracy of DBSeer's model for predicting the data flush rate. The notation "wiki 100 - 900" means we ran wikipedia at 100tps and tested at 900tps (with a different transaction

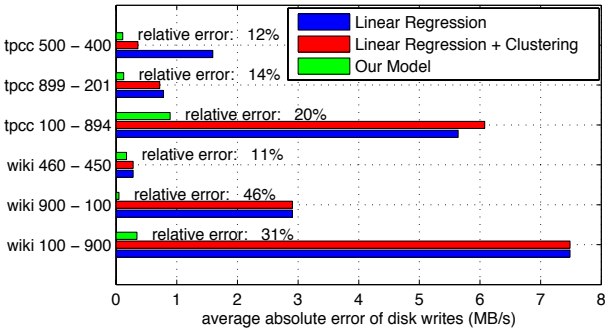

Figure 1: Disk write flush rate prediction for Wikipedia and TPC-C (Our model's percentage error is annotated on the plot).

mixture as well). The set of experiments we present shows both small and large differences between training and test configurations.

In summary, using DBSeer's model, we are able to accurately predict the volume of disk writes for arbitrary mixtures and transaction rates (always below 1MB/sec error), with a dramatic improvement of the accuracy compared to the previous work [1] based on linear regression, i.e. $10.8\times$ on average and up to $71\times$ more accurate. The most impressive improvements are observed for the more realistic workload (Wikipedia)—where data sizes and access skews stress the non-linear characteristics of the system.

2.3 Modeling Performance

In this section, we discuss how DBSeer uses resource models for predicting maximum transaction throughput. We have also validated DBSeer's models on predicting latency, and found similar results, but omit the details due to space constraints. First, using the individual models for different resources, DBSeer estimates the resource utilization at a given TPS rate. Then, to determine the maximum throughput of the system, DBSeer identifies the TPS at which each model predicts the resource will be saturated (e.g., the point where our I/O model predicts the disk will be saturated) and then report the minimum TPS (for the bottleneck resource).

To evaluate this algorithm, we randomly generated 20 mixtures of TPC-C with different ratios of transaction types. We grouped these 20 mixtures into three subsets: I/O bound mixtures, lock-bound mixtures, and CPU-bound mixtures (only a few of our mixtures were I/O bound, and approximately equal numbers were CPU and lock bound). In Fig 2, we show the average relative error of estimating the maximum throughput on different subsets of the mixtures. Here, we compare our models to a few baselines[1]: a simple linear regression on the CPU vs TPS ("LR for CPU"), and a simple linear regression on the number of page flushes vs TPS ("LR for PageFlush"). We also tried enhanced models where we performed a separate regression for each transaction type ("LR + Clust for CPU").

Our model's average error ranges between 0-25%, with its worst error on lock-bound mixtures. Our I/O model produces error that is less than 1% on average, thanks to our accurate models for disk writes. Note that the regression-based CPU models perform much better on CPU bound workloads, and that the regression-based page flush model does better on the I/O bound workload, but in all cases our model does better.

Although we omitted the details of our models, these results show that it is possible to build models that accurately predict resource utilization across a range of transaction mixes and rates. The next three sections are dedicated to describing how such models can be leveraged to provide: (i) better attribution/pricing, (ii) soft-isolation, and (iii) cloud DBMS tuning.

[1]We chose linear regression as a baseline since it has been proposed by previous work work [1] as superior for predicting disk I/O compared to other types of regression such as Gaussian processes.

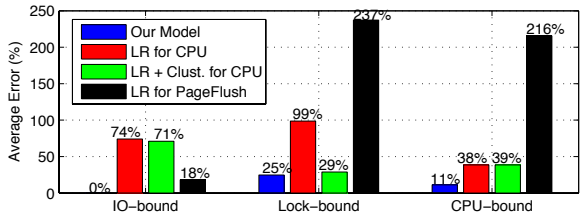

Figure 2: Max throughput prediction

3. RESOURCE ATTRIBUTION

As noted in the introduction, current cloud services charge customers by the number and sizes of VM instances. From the perspective of database users, it would be greatly preferable to pay for an application-level SLA – such as a guaranteed latency per transaction or minimum transaction throughput. Additionally, billing would be better done in terms of application-based resource usage, such as a fixed cost per transaction. While such SLAs/billing schemes have been the subject of research proposals (e.g., see [4] for piecewise linear SLAs), supporting them in real commercial settings not only requires efficient performance isolation mechanisms (see Section 4), but also requires models like those described in Section 2 that can correlate a workload's performance to its resource utilization, in order to determine the price to charge for a given level of service.

A challenge here is that resources in a shared database system will be used concurrently by multiple different transactions, and the utilization of some resources *do not scale linearly* with the number of concurrent tenants or transactions. For instance, suppose workloads A and B generate 100 and 40 physical IOs/sec, respectively, when run in isolation. When run concurrently (on the same DBMS), the overall number of IOs/sec will likely not be 140; it may be less (e.g., if workloads share data) or more (e.g., if workloads cause each other's pages to be evicted from buffer pool). Therefore, being able to *attribute* non-linear resources such as disk, memory, locks (and sometimes CPU), is a challenging problem both from a research perspective as well as from an economic one.

Additionally, users only want to pay for what they are *actually* using, regardless of what other tenants are doing in the system. This also plays a role in the notion of "stability" of the billing, which should be independent of the efficiency/optimization choices made by the provider. An alternative solution is to construct a pricing scheme that aligns provider's and users' interests, so that the users are assured that any optimization performed by the provider will be beneficial to both parties. Thus, the research problem here is to find a billing scheme that is (i) expressed in terms of high-level SLAs and performance goals that are directly relatable to users' business objectives, (ii) proportional to the actual resources used to deliver those SLAs, and (iii) stable to provider optimizations, and (iv) intuitive as users prefer simpler pricing schemes.

4. SOFT ISOLATION

VMs are relatively good at providing *hard performance isolation* between different tenants. However, as mentioned in the introduction, hard isolation leads to underutilization of resources and inefficiency. What is needed is a way to provide tenants with dedicated resources when they need them, but allow tenants to use unused resources that were reserved for other tenants, when they are underutilized. Moreover, VM's success at hard isolation does *not* always extend to virtualizing I/Os, e.g. VMs sharing the same physical hard disk interfere with each other. Setting hard quotas per tenant (say, in terms of IOs/sec) is not effective either, as hard disks' bandwidth (bytes transferred/sec) varies depending on the randomness/sequentiality of the I/O requests. The alternative is *soft performance isolation*. Soft isolation can be achieved via *admission*

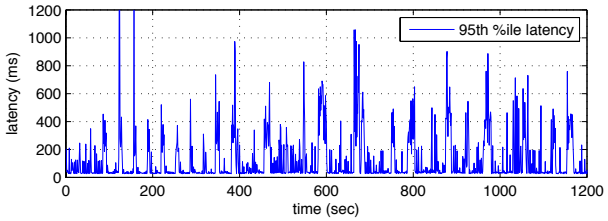

Figure 3: Erratic performance of YCSB when run on a MySQL-based DBaaS offering.

control mechanisms (i.e., throttling of certain transactions or tenants). Such mechanisms already exist in some database systems such as Teradata[2], but these throttles must be manually configured, i.e. users need to define rules for dealing with every possible situation.

Our goal is to replace this tedious, manual process of adjusting numerous throttling rules with a closed-loop control framework that adjusts the admission rates of different tenants and transaction classes in real-time. Application of feedback control to data-intensive services has been limited to data movement and replication mechanisms (e.g., see [12, 8]) to cope with load changes or SLA violations. However, designing a control framework for soft performance isolation in a DBaaS is an orthogonal but challenging task. Migrating tenants might be reasonable when overload persists, but delivering consistent performance under stringent SLAs is a real-time task which requires automatic throttling of different requests. Such a framework has to minimize the overall cost of SLA violations, which is where predictive models of resource and performance (e.g., those built in DBSeer) need to be incorporated into the control mechanism. Using such models, the controller can map the load (i.e. admission rates) to their resource consumption and in turn, to their performance impact on the SLAs. This approach will also eliminate the problem of I/O interference between different tenants, as the system will be able to estimate the effect of admitting each transaction on the total number of (sequential and random) physical I/Os. We believe such an approach can guarantee that tenants receive reserved resources while providing substantial cost savings due to resource sharing and soft isolation.

5. CLOUD DBMS TUNING

It is well-known that DBMSs require careful, workload-specific tuning to achieve good performance. In the cloud, multi-tenancy and the decoupling of the service provider from the application developer make this problem much harder by: (i) reducing the DBA's visibility into the workloads, e.g. they observe collective resource consumption rather than individual consumption of each workload, and (ii) preventing much of the coordination between administrators and application developers, which is often instrumental in achieving good performance. The current (insufficient) solution is to offer one or a small number of fixed configurations to the user to choose from. Even the most advanced auto-tuning features of academic proposals [3] and commercial DBMSs do not address the problem of re-assigning workloads to machines in a way that would allow for more specialized tuning. As an example, in Fig 3, we report the results of running a YCSB benchmark on one of the major commercial DBaaS offerings (based on MySQL). It is easy to notice the erratic performance behavior of latency. We speculate that this is due to a mismatch between the write pressure of our workload and the log-recycling policy in the MySQL configuration used by the DBaaS provider. Proper tuning of MySQL on equivalent hardware resources leads to a much more stable performance.

The performance/resource models, such as those presented in Section 2, can be used to estimate the performance impact of different tuning parameters, and the effect of various changes in resource allocation amongst different databases running on a shared DBMS. Additionally, workload models and our transaction clustering approach allow us to extract performance/resource characteristics of a single workload, even when it is only observed on a shared DBMS. Using such knowledge, it should be possible to explore the optimization space by leveraging a combination of workload-placement [5] and auto-tuning [3]. Clearly, much research is needed in this space, and the models we present are only the first step in this direction. A more immediate application of our models consists in automatically assigning workloads to a set of differently tuned DBMSs (e.g., with different buffer pool sizes, write-back policies, log sizes, with group-commit or not) which we also plan to explore.

[2] `http://developer.teradata.com/blog/carrie/2009/08/complex-throttle-rules-can-tie-you-up-in-knots`

6. CONCLUSIONS

In this paper, we argued that current approaches to cloud computing suffer from their own advantages: infrastructure sharing and the decoupling of applications from providers of the service are why clouds are popular, but these features also cripple performance-sensitive services such as databases. We argued for three key needs before clouds are appropriate for database services: (i) pricing schemes that reflect their operational costs but are also simple and intuitive to users, (ii) performance efficient mechanisms to isolate the performance of tenants from each other, while allowing soft-sharing of resources, and (iii) workload-specific tuning for each tenant. We illustrated that all these three problems enjoy a common denominator which is developing models and tools for predicting resource utilization and performance. We introduced a new tool, called DBSeer, that provides such models for OLTP databases, and presented some initial results that are quite encouraging. Finally, we discussed a number of other issues that need to be addressed to fully tackle each of these three challenges.

7. REFERENCES

[1] M. Ahmad and I. T. Bowman. Predicting system performance for multi-tenant database workloads. In *DBTest*, 2011.

[2] 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.

[3] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. In *ICDE'07*, pages 826–835, 2007.

[4] Y. Chi, H. J. Moon, and H. Hacigümüş. icbs: incremental cost-based scheduling under piecewise linear slas. *PVLDB*, 4(9), 2011.

[5] C. Curino, E. P. C. Jones, S. Madden, and H. Balakrishnan. Workload-aware database monitoring and consolidation. In *SIGMOD Conference*, pages 313–324, 2011.

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

[7] J. Duggan, U. Cetintemel, O. Papaemmanouil, and E. Upfal. Performance prediction for concurrent database workloads. In *SIGMOD*, 2011.

[8] H. C. Lim, S. Babu, and J. S. Chase. Automated control for elastic storage. In *ICAC*, 2010.

[9] B. Mozafari, C. Curino, and S. Madden. Performance and resource modeling in highly-concurrent oltp workloads. Technical report, MIT, February 2012.

[10] C. Talbot. Virtualization, cloud having little impact on databases. Web Page, Jan. 2012. `http://tinyurl.com/VM-talbot`.

[11] D. N. Tran, P. C. Huynh, Y. C. Tay, and A. K. H. Tung. A new approach to dynamic self-tuning of database buffers. *Trans. Storage*, 4(1), 2008.

[12] B. Trushkowsky, P. Bodík, A. Fox, M. J. Franklin, M. I. Jordan, and D. A. Patterson. The scads director: scaling a distributed storage system under stringent performance requirements. In *FAST*, 2011.