Microsoft Azure SQL Database Telemetry

Frank Bertsch# David J. DeWitt⁺ Willis Lang⁺ Nigel Ellis* ⁺Microsoft Corp. *University of Wisconsin-Madison wilang@microsoft.com, bertsch@cs.wisc.edu, dewitt@microsoft.com, nigel@dooger.org

Abstract

Microsoft operates the Azure SQL Database (ASD) cloud service, one of the dominant relational cloud database services in the market today. To aid the academic community in their research on designing and efficiently operating cloud database services, Microsoft is introducing the release of production-level telemetry traces from the ASD service. This telemetry data set provides, over a wide set of important hardware resources and counters, the consumption level of each customer database replica. The first release will be a multi-month time-series data set that includes the full cluster traces from two different ASD global regions.

Keywords Cloud database, Resources, Telemetry, Data set

Introduction

The rate of cloud growth and cloud service adoption has been breathtaking. In a recent survey, between 2013 and 2014, of over 600 organizations, it was found that the proportion of HR and IT departments that have adopted cloud services increased from slightly under 60% to almost 90% [6]. Therefore, it shouldn't be surprising that major cloud service providers such as Microsoft, Amazon, and Google have all dedicated tremendous time and resources in designing, developing, and delivering cloud services to customers. In particular, the Database-as-a-Service class of cloud services has seen particular focus amongst cloud vendors. This is not only because of the consumer demand for these products, but also because providing these services is particularly challenging due to the desired elasticity of performance and scale and the unpredictability of customers.

A Database-as-a-Service (DBaaS) is a cloud providermanaged system where the goal of the system is to provide

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SoCC '15, August, 2015, Kohala Coast, HI, USA. Copyright © 2015 ACM 978-1-4503-3651-2/15/08...\$15.00. http://dx.doi.org/10.1145/2806777.2806845

both academic and industrial database researchers have been studying the problems that we described above – increasing the efficiency of a DBaaS through increased user density in the servers of a cluster while delivering variable performance objectives (see Section 2). This is generally known as the 'cloud database multi-tenancy problem'. Unfortunately, for academic researchers, generally there is no access to realworld data that represents the actual utilization load generated by DBaaS users and this makes research additionally difficult; typically relying on synthetic workloads and/or other load assumptions. Our goal here is to engage external

flexible and elastic data processing that can expose an array of storage and performance levels to customers at varying price points. Furthermore, these services require no infrastructure commitment, manual upgrades/updates, or back-up routines by the customer. Providing such flexible and elastic data processing systems while operating an efficient service is the main challenge facing DBaaS providers. Unfortunately, these are opposing goals - increasing the operational efficiency of the service generally means increasing the utilization levels of the service by co-locating users together onto fewer physical servers, while high, stable performance is generally achieved by isolating a user onto their own server (potentially very wasteful). The workload of the user, which is essentially a pattern of resource consumption, is the key factor determining whether or not two users may be co-located with each other on the same physical server.

At Microsoft, our DBaaS – Azure SQL Database – serves millions of customer databases worldwide spanning 19 regions around the globe. The large diversity of customers and their workloads can be observed within each region. Each region is backed by multiple clusters of servers. With continuously evolving hardware technology, our servers are configured differently from cluster to cluster, providing different levels of processing power, memory capacity, and I/O latency and bandwidth. Understanding how our current customers behave is crucial not only for the efficiency, availability, and reliability of the day to day operations of the production systems, but also so Microsoft can continue to improve the service moving forward.

With the recent levels of DBaaS adoption and growth, academic researchers by sharing our real-world production system data that reflects actual user behavior so that we may

^{*} Work done while an employee of Microsoft Corp.

all mutually benefit. We hope researchers can use this data to ground the motivations/assumptions, and also improve the results/insights of their research with the hope that Microsoft and other cloud providers may be able to use these insights and approaches to continue improving the quality and efficiency of cloud database services.

While our goal is to provide data that can be used to study Azure SQL Database (ASD) user workloads, our primary responsibility is to serve Microsoft customers, and so we must be sensitive to Microsoft customers' privacy and respect our terms of use obligations. While other types of data such as the actual query logs and workloads may provide great insights in characterizing user behavior, providing these details would be unacceptable for privacy reasons. Instead, the data we will provide is what we deem 'performance telemetry' data, which is essentially the resource utilization traces per database over regular intervals. Monitoring this type of data is the primary means by which ASD makes load balancing and placement decisions. In this paper, we will describe the details of the traces, what types of metrics are captured, and the limitations/nuances of the data (see Section 4.)

The purpose of this paper is to introduce the Microsoft Azure SQL Database telemetry data set that Microsoft is now making freely available to academic researchers. The rest of the paper is as follows: in Section 2, we will discuss some relevant work in cloud database research. We will then briefly describe the architecture of the Azure SQL Database v1 system that the data set is produced from in Section 3. The data set description and parameters will be presented in Section 4 and we will end with our concluding remarks in Section 5.

2. Related Work in Cloud Database Research

As interest in studying the research problems of cloud database systems has risen, we have seen a few common approaches appear repeatedly in the research literature. A common theme across some of these studies is to use wellknown, common database benchmarks as hypothetical cloud workloads, characterize and model the resource consumption of the workloads, and finally, using the models, optimize the cloud system for metrics like cost or performance. Generally, the benchmarks used are transactional (TPC-C) [3, 7, 9, 13], web-transactional (TPC-W) [8], or more cloud-tailored workloads (YCSB) [10, 13]. Sometimes, these benchmarks have even been combined to compose workloads that aim to represent the huge variability in real-world aggregate workloads [5, 11]. In certain ideal cases, usually when there are industrial parties involved in the research, real-world data may be used in the study to model cloud user workloads and resource consumption [12].

These studies generally boil down to traditional computer science problems like optimal workload scheduling (or in the cloud case, 'tenant' placement), or the minimization of data movement during load balancing. Some common metrics for 'cost' are the number of physical servers required to serve a fixed number of cloud users (the 'multi-tenancy problem' of consolidating customer databases) or the minimization of SLA/SLO violations to maximize profit. And, given the benchmarks used, typical performance metrics are transaction throughput, or read/write latency. These optimization goals and evaluation metrics are all of real interest to cloud DBaaS providers and the exciting thing is that these results/insights may find their way into actual service products.

Our aim is to provide academic researchers with the real-world data that is being collected today and being used by DBaaS providers to make real-time decisions, drive the tenant placement, and trigger the load balancing algorithms. We hope that researchers will be able to use this data to augment their studies beyond the synthetic workloads and benchmarks that are typically the only option currently available to them.

3. Azure SQL Database v1 Background

At the time of preparing this paper, the Azure SQL Database service is about to undergo a significant architectural update as it moves to v2 (update 12) [1]. Currently, the overwhelming majority of Azure SQL Database customers are housed within v1 clusters. As a result, the data that is being released is produced from the v1 service clusters. (Since the main purpose of the data release is to provide a view of actual customer behavior and resource usage, providing data from the v1 system as opposed to the v2 system is not a significant pitfall.) In this section we will: (1) describe the key v1 system details to help put the telemetry data in context, a deeper description on Azure SQL Database's architecture can be found in [2]; and (2) provide an overview of the tiered subscription model.

3.1 System Background of Azure SQL Database v1

Cloud database systems can be designed using one of three basic architectural categories: physical databases contained in virtual machines, databases collocated in the same shared database process, and users sharing the same database tables [4]. Azure SQL Database v1's design follows the 'shared process' architecture. Azure SQL Database uses the SQL Server database engine where an instance of the database engine runs as a single process and many customer databases are stored and serve requests from this instance. Depending on the service tier selected by a customer, certain resource isolation mechanisms (e.g., guaranteeing enough CPU or IOPs for a certain transactional throughput) may be in place to provide more stable performance (see Section 3.2.) Each physical server in the cluster runs a single SQL Server instance.

To support high-availability, a customer database is automatically k-way replicated where for the majority of cases, k = 3. For any database, at any point in time, one of the repli-

cas is designated as the primary replica. The primary replica does all of the query processing including updates. Reads are *only* served by the primary replica. Transaction updates are sent to the two secondary replicas and the primary returns the transaction as committed when one of the secondary replicas has sent an ACK of the commit. This means that in certain special cases where k=4, the *quorum commit protocol* requires a primary and two secondary ACKs before a transaction is deemed successful. The data sent to the secondaries are after-images (page images) of each update [2], instead of redo log records. This is designed to reduce the amount of work that needs to be done by the secondary replicas (i.e., reduce the load on the database engine.)

Customer database replicas are placed across a cluster in a specific manner due to the way that servers are organized into groups: upgrade and fault domains. Upgrade domains are used to stage software update rollouts across the cluster, while fault domains define groups of servers that share single points of hardware failures (i.e., networking or power distribution.) These two domains are non-overlapping server groupings and replicas are placed so that no two replicas of the same customer database reside in the same upgrade or fault domain.

Database replicas can be moved from one physical server to another due to cluster management maintenance and capacity management or due to load balancing. The data that we will release includes both the resource consumption of each database replica as well as the replica placement mapping on the cluster over time.

3.2 Tiered Subscription Model

Azure SQL Database currently offers a tiered subscription model that allows customers to choose from databases with different subscription levels, or colloquially, 't-shirt' sizes. T-shirt sizes correspond not only to various levels of performance, but also availability, and reliability objectives. For instance, upper tiers provide *Active* data replication while lower-tiers have *Passive* data replication [1].

The 'original' Azure SQL Database service launched with two t-shirt sizes: Web and Business, which only corresponded to different maximum database sizes. Azure SQL Database's 'current' subscription model consists of three tiers: Basic, Standard, and Premium (Standard and Premium are further subdivided into four and three sub-levels, respectively.) These subscription classes each provide different 'database transaction unit' (DTU) performance objectives (a proprietary transactional benchmark metric in the spirit of TPC-C,) as well as other parameters like the maximum DB size, the number of database worker threads ('Max Workers'), and the number tabular data stream (TDS) sessions ('Max Sessions'). A summary of these details is shown in Table 1 and further details can be found in [1]. In the telemetry data, both the Web and Business tiers have a 'Max Workers' value of 180 and a 'Max Sessions' value of 8000; however, Web and Business tier customers belong to a 'shared re-

Service Tier	DTU	Max Workers	Max Sessions
Web			
Business			
Basic	5	30	300
Standard S0	10	60	600
Standard S1	20	90	900
Standard S2	50	120	1,200
Standard S3	100	200	2,400
Premium P1	100	200	2,400
Premium P2	200	400	4,800
Premium P3	800	1,600	19,200

Table 1. Service Tier Mappings

source pool' where there are no minimum resource settings. The implication is that Web and Business tier customers may potentially suffer from unpredictable performance.

Currently, the Azure SQL Database service has customers that subscribe to the 'original' and 'current' t-shirt tiers and the data that we are releasing will reflect this: researchers will be able to observe the behavior of customers from all of the t-shirt sizes. Customers that are currently subscribed to the 'original' model (Web and Business) will be voluntarily or automatically migrated to the 'current' model (Basic, Standard, and Premium) in the near future, but currently, there are still a vast number of Web and Business customers. As this migration occurs, it is reasonable to assume that the usage pattern of the majority of these 'original' t-shirt model customers remains consistent with our current observations in the data.

4. Performance Telemetry Data

There are many types of data collected by the Azure SQL Database (ASD) monitoring system, including customer database-level, SQL Server instance-level, and operating system-level performance counter data. Our data set contains database-level counters collected at regular intervals. In this section we will describe and provide the schema of this relational data set (Section 4.1), discuss what we think researchers may be able to do with the data (Section 4.2), and describe how Microsoft is providing the data to academic researchers (Section 4.4).

4.1 Data Collection and Description

There are two relational tables being provided: a fact table that describes the machine placement and resource utilization per customer database replica, and a dimension table that lists the machine's upgrade and fault domain (see Section 3.1). Table 2 provides the schema attributes for the fact table. Each row of the table represents a time series data point for a specific customer database replica. A customer database replica is uniquely identified by the composite key of an anonymized unique (1:1) customer database identifier and a half key: (dbuid, machine_id). The type of the replica is provided by the field *replica_type*.

Attribute	Description	
timestamp	The UTC time when this window ends.	
interval_seconds	The length of this time window	
	(when the window starts.)	
dbuid	The anonymized unique customer	
	database id. (This is a 1:1 mapping.)	
replica_type	The type of the replica: primary (1),	
	secondary (2), or transitioning.	
machine_id	A foreign key denoting the machine	
	that this replica resides on during	
	this window.	
max_worker_count	The maximum number of worker	
	threads available to this database.	
cpu_time	The CPU time spent by this database's	
	primary replica in this time	
	window (microseconds.) This should	
	be ignored if replica_type isn't 1.	
cpu_time_sec	The CPU time spent by this database's	
	secondary replica in this time	
	window (microseconds.) This should	
	be ignored if replica_type isn't 2.	
logical_reads_pgs	The number of logical page reads	
	in this time window.	
logical_writes_pgs	The number of logical page writes	
	in this time window.	
phys_reads_issued	The number of physical reads issued	
	in this time window. This should	
	be ignored if replica_type isn't 1.	
phys_read_bytes	The number of physical bytes read in this	
	time window. This should be ignored if	
	replica_type isn't 1.	
phys_writes_issued	The number of physical writes issued	
	in this time window. This should	
	be ignored if replica_type isn't 1.	
phys_write_bytes	The number of physical bytes written in	
	this time window. This should be ignored	
	if replica_type isn't 1.	
log_operations	The number of log operations performed	
	in this time window. I.e., a transaction	
	requires many write operations to the log.	
	This should be ignored if replica_type	
log omanations -	isn't 1.	
log_operations_sec	The number of secondary replica	
	log operations performed in this time window. I.e., a transaction requires many	
	write operations to the log. <i>This should</i> be ignored if replica_type isn't 2.	
	ve ignorea ij replica_type isn t 2.	

Table 2. Fact table schema.

The telemetry collection pipeline emits a row per database replica every 300 seconds (a best-effort target) and each row lists the resource consumption by the replica within this time window (e.g., the total number of database log operations charged to a database replica within every 300 second window.) This means that an ASD server (SQL Server instance) that is housing *R* replicas (primary and secondary replicas) may be producing *R* telemetry records every 300 seconds.

Attribute	Description
machine_id	A key to the fact table.
machine_name	A unique machine name.
upgrade_domain	The upgrade domain for this machine.
fault_domain	The fault domain for this machine.

Table 3. Dimension table schema.

Consequently, each row is tagged with the field *timestamp* denoting the end of the time window (in UTC time) and the field *interval_seconds* which tells us the length of the time window (and provides the start of the window). To reduce the amount of data produced by the telemetry pipeline, a row for a single database replica is *not produced if all of the resource counters report no activity during the time interval*. The implication of this behavior is that from this data set, we cannot tell whether or not a database is idle, or it has been simply dropped by the customer.

The number of maximum worker threads available to this database is provided in <code>max_worker_count</code>. One can identify the service tier of the customer's database from this field, along with the mapping in Table 1. The remaining attribute fields in this schema are the actual resource utilization fields. The resource counters that are provided are for the CPU, I/O (logical and physical), and the database log operations. The details of these counters are all listed in Table 2.

Machine domain information is provided in Table 3. This table includes the upgrade and fault domain values per machine in the cluster. The fact table joins with the dimension table through the *machine_id* column; however, this column is not the unique identifier for a given server. Any given server may have many different *machine_id* values. The *machine_name* field provides a unique value per server.

4.2 Research Questions

As we described in Section 2, there have been many recent studies published that could have taken advantage of our Telemetry data set. We believe that researchers may be aided by our data set to tackle questions like:

- Multi-tenancy Can we improve the customer density levels on database-as-a-service clusters without negatively impacting the user's experience?
- Configuration Can we find patterns of behavior that may give us a taxonomy of customers as to aid us in provisioning hardware resources? Efficient cluster and server configurations?
- Experience Can we provide guidance to customers on what service level objectives (tiers) they actually need?
- Modeling Can we come up with a new benchmark workload that is guided by the behavioral patterns we see in these traces?

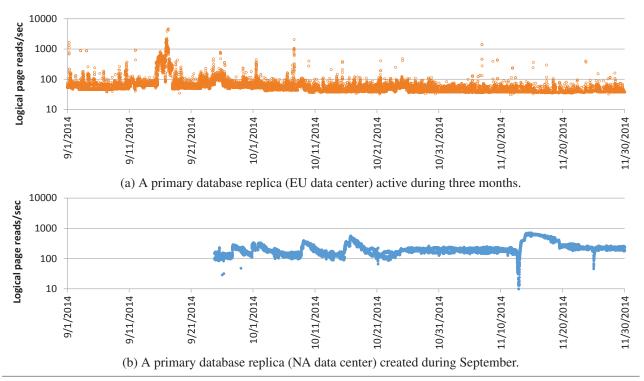


Figure 1. Two example traces of the logical page reads (per second) from two different primary database replicas recorded during a three month period.

4.3 Applications and Limitations

Internally within Microsoft, this data has been used to help identify and characterize certain types of customers through data mining (e.g., heavy-hitters and idle users.) Additionally, in a collaborative effort between the Microsoft Gray Systems Lab and external academic researchers, a study on database placement schemes and cluster provisioning ¹ has been submitted for peer-review publication. Finally, this Telemetry is used in part to inform the load balancing system that action is required on a set of nodes.

Ultimately, the description of the data set here is produced from a single telemetry source; it is by no means the cleanest and most informative data set, nor is it the only type of ASD telemetry Microsoft has collected. The 300 second granularity of the time series is quite coarse and we cannot make very precise statements about a user's workload. Furthermore, the data set does not include the overall ASD system resource consumption necessary to run the service on the cluster. Fortunately, we are continuing to gather other data from other ASD telemetry sources and we hope to expand this data release in response to demand and subject to corporate approval.

4.4 Initial Release and Agreement

Microsoft will release multiple telemetry traces from a couple of clusters housed in a few Microsoft regions around the world. The initial data sets will include data from a North American data center and a European data center. These traces will be long, multi-month traces (some spanning three months or greater). An example of multi-month traces from two customer databases can be found in Figure 1 which shows the *logical_reads_pgs* metric for two distinct customer databases (primary replica) over a period spanning September through November 2014.

The data set will be freely released to academic research teams after we have received a signed Microsoft agreement from *every individual that will have access to the data set*. The release format will essentially be the entire anonymized data set which will include all of the customer database traces across all of the machines in a given cluster. We will also provide other 'processed' versions of the data set that, in our experience, may be more useful under analysis (e.g., time bucket alignment, normalization, etc.) The details of the processing will be described at the URL listed below.

Given the completeness of the release, the agreement will require a Microsoft review (primarily by Microsoft technical staff and in certain cases, by Microsoft legal staff,) of any sort of public disclosure or publication of any data, results, and/or findings. (It is our intention to have a relatively short turn-around time on these reviews.) Additionally, each copy of the data set provided will be watermarked uniquely. Please refer to the agreement document for further details.

Currently, we have already provided this data set to researchers from MIT, Northwestern, and the University of

¹ Project report: http://people.csail.mit.edu/rytaft/masters.pdf

Chicago. They all have the data set we have provided stored in their own institutional servers after signing the release agreement.

To provide some idea of what one should expect in this full data set, we will provide a small sample set of customer database traces for download online. This small data set will contain traces of a set of customer databases over a relatively short period of time and will not require a signed agreement. Details of this sample, such as the number of databases and the length of the traces, will be provided online. It is our hope that after taking a look at the sample data set, researchers will decide to pursue the full data set.

Microsoft reserves the right to change the data set at any time and such changes will be announced online. This may include adding more counters/attributes to the data set or adding new data sets altogether. The ultimate goal is to develop a working relationship with the research community and this may ultimately result in community engagement that helps to determine the parameters of future releases.

The release agreement form, sample data, and further instructions can be found at: http://gsl.azurewebsites.net/ Projects/SQLAzure/AzureTelemetry.aspx

5. Conclusions

In this paper, we introduce the Microsoft Azure SQL Database telemetry data set that will be freely available for academic researchers. The goal of releasing this data set is to help researchers understand real-world user behavior in cloud database environments. This anonymized, time-series trace of the resource consumption of Microsoft Azure SQL Database user workloads is the first data set of its kind publicly released in this manner. We hope that the scale and length of the data set will provide researchers with a valuable resource to tackle the challenges of designing and operating cloud database services.

Acknowledgments

The authors would like to thank T.K. Rengarajan, Ajay Kalhan, Sandu Chirica, Ned Locke, Jignesh Patel, and Jeffrey Naughton for their support and feedback throughout this project and in the preparation of this paper.

References

- [1] Azure SQL Database. Microsoft corporation. http://azure.microsoft.com/en-us/documentation/ services/sql-database/, 2015.
- [2] P. A. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, D. B. Lomet, R. Manne, L. Novik, and T. Talius. Adapting microsoft sql server for cloud computing. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, pages 1255–1263, Los Alamitos, CA, USA, 2011. IEEE Computer Society. ISBN 978-1-4244-8959-6.

- [3] C. Curino, E. P. Jones, S. Madden, and H. Balakrishnan. Workload-aware database monitoring and consolidation. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11, pages 313–324, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0661-4.
- [4] A. J. Elmore, C. Curino, D. Agrawal, and A. El Abbadi. Towards database virtualization for database as a service. *Proc. VLDB Endow.*, 6(11):1194–1195, Aug. 2013. ISSN 2150-8097
- [5] A. J. Elmore, S. Das, A. Pucher, D. Agrawal, A. El Abbadi, and X. Yan. Characterizing tenant behavior for placement and crisis mitigation in multitenant dbmss. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 517–528, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2037-5.
- [6] KPMG International Corp. The cloud takes shape global cloud survey: the implementation challenge. 2013.
- [7] W. Lang, S. Shankar, J. M. Patel, and A. Kalhan. Towards multi-tenant performance slos. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 702–713, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4747-3.
- [8] Z. Liu, H. Hacigümüş, H. J. Moon, Y. Chi, and W.-P. Hsiung. Pmax: Tenant placement in multitenant databases for profit maximization. In *Proceedings of the 16th International Con*ference on Extending Database Technology, EDBT '13, pages 442–453, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1597-5.
- [9] B. Mozafari, C. Curino, A. Jindal, and S. Madden. Performance and resource modeling in highly-concurrent oltp workloads. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 301–312, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2037-5.
- [10] B. Mozafari, C. Curino, and S. Madden. Dbseer: Resource and performance prediction for building a next generation database cloud. In CIDR, 2013.
- [11] J. Schaffner and T. Januschowski. Realistic tenant traces for enterprise dbaas. Technische Berichte des Hasso-Plattner-Instituts für Softwaresystemtechnik an der Universität Potsdam, page 17, 2013.
- [12] J. Schaffner, T. Januschowski, M. Kercher, T. Kraska, H. Plattner, M. J. Franklin, and D. Jacobs. Rtp: Robust tenant placement for elastic in-memory database clusters. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 773–784, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2037-5.
- [13] M. Serafini, E. Mansour, A. Aboulnaga, K. Salem, T. Rafiq, and U. F. Minhas. Accordion: Elastic scalability for database systems supporting distributed transactions. *Proc. VLDB Endow.*, 7(12):1035–1046, Aug. 2014. ISSN 2150-8097.