Extreme Scale with Full SQL Language Support in Microsoft SQL Azure

David G. Campbell
Microsoft Corporation, 1 Microsoft Way
Redmond, WA USA
davidc@microsoft.com

Gopal Kakivaya
Microsoft Corporation, 1 Microsoft Way
Redmond, WA USA
gopalk@microsoft.com

Nigel Ellis
Microsoft Corporation, 1 Microsoft Way
Redmond, WA USA
nigele@microsoft.com

ABSTRACT
Cloud SQL Server is an Internet scale relational database service which is currently used by Microsoft delivered services and also offered directly as a fully relational database service known as "SQL Azure". One of the principle design objectives in Cloud SQL Server was to provide true SQL support with full ACID transactions within controlled scale "consistency domains" and provide a relaxed degree of consistency across consistency domains that would be viable to clusters of 1,000's of nodes. In this paper, we describe the implementation of Cloud SQL Server with an emphasis on this core design principle.

Categories & Subject Descriptors
H.2 DATABASE MANAGEMENT, H.2.4 Systems, D.0 GENERAL

General Terms:
Design, Economics, Reliability

Keywords:
Cloud Database, Extreme Scale

1. INTRODUCTION
Cloud SQL Server is a relational database service designed for Cloud computing workloads. It is built upon Microsoft SQL Server technology and uses a partitioned database model over a large scale, shared nothing, infrastructure. Cloud SQL Server's design goal is to provide near 100% compatibility with SQL Server at Internet scale with very low cost of operation. The Cloud SQL Server infrastructure masks or recovers from most hardware and software failures without human intervention. Each database partition is replicated across a set of loosely coupled and dynamically assigned hosts using a custom primary-secondary replication scheme. Cloud SQL Server is currently used to power two services at Microsoft: 1) The SQL Azure service [1] in which a large number of independent databases is served, and 2) Exchange Hosted Archive [2] in which the infrastructure serves a massively scaled, multi-tenant, email archival service.

SIGMOD'10, June 6–10, 2010, Indianapolis, Indiana, USA.

Internet scale services present a complex set of requirements. The workloads can be massive and often require scaling out from thousands to hundreds of thousands of servers. Cost factors require delivering highly reliable services over inexpensive hardware with an operational model that is fully automated. Furthermore, failures of individual components are literally an everyday occurrence at this scale. As such, the system must mask or handle most faults without operator intervention.

Cloud SQL Server's design goals were to achieve the primary cost and operational objectives of today's high scale Internet services while still providing a high degree of compatibility with existing RDBMS facilities. In this paper we discuss the tradeoffs around consistency and scale; describe the implementation approach taken by Cloud SQL Server; and share some of the lessons learned from our work.

2. CONSISTENCY & SCALE
Most commercial RDBMS vendors have approached scaling challenges by scaling up on a single large SMP machine or scaling out through traditional shared-disk or shared-nothing database clusters. One of the primary requirements of the scale-out efforts of the commercial database products has been to provide a "single system image" such that the cluster looked as much as possible as a single system. An implication of this requirement is that the ACID transaction properties must hold across the cluster. Providing ACID level consistency across a cluster of tens of thousands of nodes runs into the challenge of Brewer's CAP conjecture [3].

Lacking a commercial solution for these requirements, many large scale web service providers have developed their own distributed storage systems such as Google's BigTable[4], Yahoo's PNUTS[5], and Amazon's Dynamo[6]. Rather than support full ACID transactions at scale, these systems relaxed consistency to a single storage entity or well defined entity group. Most of these systems have also done away with the logical relational model and SQL compliant query support. Many have come to believe that Internet scale storage solutions and relational model support are mutually exclusive. Cloud SQL Server combines full relational support – within specified "consistency domains" and Internet scale infrastructure to retain the value of the relational model and still achieve the benefits of the Cloud architecture model.

3. IMPLEMENTATION APPROACH
Cloud SQL Server can be broken up into several layers as shown in Figure 1.

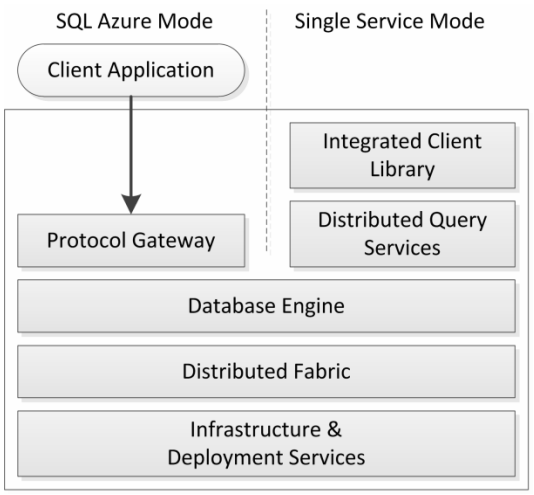

Figure 1: Cloud SQL Server Layers

The "Infrastructure and Deployment Services" (IDS) layer coordinates the activities of the physical machines within a cluster. IDS coordinates and loads the initial software image onto a machine based upon its specific role in the cluster. The roles include: front end machines which handle protocol activity; database nodes which manage the cluster data; and cluster infrastructure roles which perform tasks such as global state management, operational data gathering and processing or other cluster specific tasks. IDS also runs various workflows in response to requests from the distributed fabric such as node restart, shutdown, or software reimage.

The "Distributed Fabric" (DF) layer coordinates many of the low level cluster activities such as managing replica set quorum, global partition management, failure detection, and leadership election for various roles in the cluster. The distributed fabric has been designed as a distinct layer and is currently being used for other services at Microsoft as well.

The "Database Engine" (DE) layer consists of a cooperating set of database servers which are based upon technology from the Microsoft SQL Server 2008 kernel. The architectural approach taken by Cloud SQL Server is to intercept and inject the specific cluster control points with minimal invasion into the core SQL Server code base. This allowed us to retain much of the core relational features of SQL Server. Specific changes made to the SQL Server database engine included:

- Database Virtualization: SQL Server supports a large number of individual databases per server instance. In order to achieve its cost goals, SQL Azure mode further allows multiple logical databases to be hosted in a single physical database. This allows multiple virtual databases to share a common transaction log and to save on memory for the internal database structures in the server. Each logical database is isolated from all other databases using an isolation container that extends the existing SQL schema mechanisms for this purpose.
- Partition and Reconfiguration Management: Cloud SQL Server maintains n-safe durability by replicating each transaction across a set of replicas. At any point in time there is a single master replica and a configurable number of secondary partition replicas. This constraint is actively maintained by the Distributed Fabric's Global Partition Manager (GPM). If a hard failure is detected at any secondary replica, a new secondary node is appointed and a local replica of the partition will be copied to the newly appointed secondary node. If a primary coordinator dies, the GPM will elect one of the remaining secondary nodes to become the primary.
- Deployment: Changes were required to allow the Infrastructure and Deployment Service layer to perform an unattended image based deployment of the engine software without the need to invoke SQL Server's full product setup.

The Distributed Query Services (DQS) layer in Cloud SQL Server today is quite modest and only used in Single Service deployments such as Exchange Hosted Archive. In this mode, partitions across a subset of nodes in a cluster form a single large sharded (partitioned) database. The DQS layer is responsible for routing queries to the appropriate partition for single partition queries and to coordinate queries across partitions for multi-partition queries. As mentioned earlier, Cloud SQL Server currently only supports full ACID properties *within* a single partition so queries coordinated by DQS that span partitions will not have full ACID consistency.

The Protocol Gateway (PG) layer is currently only used in the SQL Azure service mode. SQL Azure supports TDS, the native protocol of SQL Server. The PG is responsible for accepting inbound TDS database connection requests from clients and performing a binding to the node which is currently managing the primary replica. The PG coordinates with the DF layer to locate the primary and will renegotiate in the case of a failure or system initiated reconfiguration which will cause election of a new primary. The PG also masks some failure and reconfiguration events from external clients by renegotiating internal sessions while maintaining the protocol session state between the client and PG.

4. AVAILBILITY & LOAD BALANCING

The Cloud SQL Server cluster system is aware of the layout and topology of all nodes, partitions, racks, and physical network routes. This allows the system to coordinate software upgrades in a way that allows for high data availability during the upgrade. For example, any node can be removed from service by switching primary replicas to other nodes and shutting down secondary replicas. Once cleared, a node may be serviced and brought back into the cluster. Currently, clients may see a transaction failure if a primary election occurs during a transaction but this will only occur for a brief instant while the reconfiguration takes place. Through careful scheduling of activities, Cloud SQL Server maintains a configurable safe number of partition replicas during an upgrade.

Cloud SQL Server also contains support for active load balancing. Some degree of workload balancing can be achieved by reconfiguring (swapping) primary and secondary replicas. A further degree of workload balancing and management can be obtained by actively migrating partition replicas from one node to another.

5. RESTRICTIONS

SQL Azure exposes most of the functionality of SQL Server with some restrictions. The restricted features generally fall into several classes:

1. Features which would be inherently unsafe in a shared cluster environment such as Extended Stored Procedures[1]
2. Features which may lead to excessive resource consumption within a single database that would impact other users on the

[1] Unmanaged extended stored procedures allow user written native code extensions to SQL Server.

same node. SQL Azure contains several resource governors to mitigate this problem but it is an area for further investment.

3. Features which are associated with the server instance rather than a specific database. These items include server level performance and resource monitoring queries as well as access to the server error log. Our approach is to virtualize many of these resources and views to provide database specific versions.

6. LESSONS AND FUTURE WORK

There have been many lessons from building a true relational database service with extreme scale such as:

- The Cloud SQL Server team was formed with staff from the existing SQL Server product, other Microsoft server products, and from other Microsoft service efforts. One key challenge early on was shifting our development methodology to be "service friendly". Many of the SQL Server developers and testers had to unlearn old processes to create new development and test processes that worked at service scale and pace.
- Loose and controlled coupling is essential at extreme scale. Cloud SQL Server faced challenges in that it requires a high level of consistency for coordinating partition management but it also must be decoupled such that no small set of failures can lead to a coupled collapse of the cluster. Ensuring graceful degradation in the face of any failure provided many challenges during development. Ultimately our testing of the DF layer required significant investment in model based testing and resequencing distributed logs to assert global state invariants across the cluster.
- The separation of logical and physical administration also proved challenging in SQL Azure mode. Most RDBMS products use the SQL language for data manipulation, data definition, and for system administration. Profiling Transact-SQL, the dialect of SQL used by SQL Server and SQL Azure, will be an ongoing challenge. One example of this challenge is the "CREATE DATABASE" SQL statement. In its simplest form, CREATE DATABASE, creates a new named database schema and all physical database file naming is handled by the SQL Server or SQL Azure. SQL Server's CREATE DATABASE statement also allows specification of many physical attributes such as the filesystem location of the transaction log file. Obviously, many of these physical attributes are not appropriate for a database service.

7. CONCLUSION

Cloud SQL Server has proven to be a valuable asset to Microsoft and is currently deployed in production with internal Microsoft Services as well as being offered as a relational database service. One of our key design principles was to create an Internet scale data service which retained most of the value of traditional RDBMS systems. By carefully considering the consistency domains in which full ACID properties are provided, Cloud SQL Server strikes an interesting balance of providing full relational capabilities within virtualized databases or specific partitions and providing a relaxed consistency model over a collection of these partitions in large scale service deployments such as that used in Exchange Hosted Archive.Our success in deploying Cloud SQL Server in these two modes has convinced us that we have succeeded in our principle design objective.

8. ACKNOWLEDGMENTS

Our thanks to everyone involved in the design and implementation of Cloud SQL Server.

9. REFERENCES

[1] Microsoft Corp.: Microsoft SQL Azure. http://www.microsoft.com/windowsazure/sqlazure/

[2] Microsoft Corp.: Microsoft Exchange Hosted Archive. http://www.microsoft.com/online/exchange-hosted-services.mspx

[3] E. Brewer. "Towards robust distributed systems", *Principles of Distributed Computing (PODC) Keynote*, July 2000.

[4] Chang, F., J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber, "Bigtable: A Distributed Storage System for Structured Data", *ACM Transactions on Computer Systems*, 26(2): 4:1 - 4:26, 2008.

[5] Cooper, B.F, R. Ramakrishnan, U. Srivastava, A.Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, R. Yerneni, "PNUTS: Yahoo!'s hosted data serving platform," *PVLDB* 1(2): 1277-1288 (2008)

[6] DeCandia, G., D. Hastorun, M. Jampani, et al., "Dynamo: Amazon's Highly Available Key-value Store", *Proc. 21st SOSP*, October 2007, pp. 205-220.