Systems Engineering at MITRE
CLOUD COMPUTING SERIES

# Database as a Service: A Marketplace Assessment

*Lawrence Pizette*
*Toby Cabot*

MITRE

*For their collaboration and help with this paper, we would like to thank Sri Vasireddy and the team from Amazon, Scott Frohman, Shannon Sullivan, Jim Young and the team from Google, and Marc Langlois and the team from Microsoft.*



January 2012

## Table of Contents

**THE BIG PICTURE:** Public DBaaS offerings may provide a ripe opportunity for reducing costs, but there are many considerations for Government IT decision makers.

# Database as a Service: A Marketplace Assessment

*Lawrence Pizette*
*Toby Cabot*

## 1.0 Introduction

*Database as a Service* (DBaaS), a form of Platform as a Service (PaaS), is currently found in the public marketplace in three broad capabilities—online general relational databases, non-relational databases, and the ability to operate virtual machine images loaded with common open source databases such as MySQL or similar commercial databases. These three approaches provide Government IT leadership with a wide range of capabilities and potential complexities.

The analysis is intended for the chief information officer (CIO) and project-level decision makers in Government that are considering employing DBaaS products, but would like greater visibility into product benefits, risks, appropriate usage, and trade-offs. In this paper we evaluate four public DBaaS offerings, contrasting their features and capabilities. Two of the services, Amazon Relational Database Service (RDS) and Microsoft SQL Azure, offer structured query language (SQL)-compliant database products. The remaining two services, Google Datastore and Amazon SimpleDB, provide NoSQL interfaces, and offer proprietary interfaces for storing data in less complex structures. The products we compared are presented in Figure 1-1, and are summarized as follows:

***Amazon RDS***—Amazon's MySQL RDS offering provides an implementation of MySQL on a virtual operating system.

***Microsoft SQL Azure***—Microsoft SQL Azure is a relational database management system (RDBMS) product offering a SQL Server-like experience in a cloud. Microsoft controls many of the database configuration details, allowing the user to focus on the schema, data, and application layer.

***Google AppEngine Datastore***—Google's NoSQL Datastore is integrated with their App Engine PaaS offering. Google states that Datastore is intended to provide robust, scalable storage for App Engine Web applications rather than a general purpose database service.[1]

***Amazon SimpleDB***—Amazon's SimpleDB is a NoSQL database offering that provides users with an application programming interface (API) for writing and reading data. SimpleDB is automatically configured in their base service offering to copy data across Amazon Web Service's (AWS's) availability zones for redundancy.

| | Amazon RDS (MySQL) | Microsoft SQL Azure | Google Datastore | Amazon SimpleDB |
|---|---|---|---|---|
| **Type** | **RDBMS** | **RDBMS** | **NoSQL** | **NoSQL** |
| **Maximum amount of data that can be stored** | 1 terabyte per database[2] | 50 gigabytes per database[3] | Not published for entire database, but 1 MB limit on a subset of data (called an entity). Limit to the number of indexes. | 10 gigabytes per database domain (roughly equivalent to an RDBMS table)[4] |
| **Ease of software portability with similar, locally hosted capability** | High. MySQL instantiation in cloud is very similar to the local instantiated version. | High. Most SQL Server features are available in SQL Azure. | Medium/Low. Requires Java Data Objects or Datastore-specific interface and use of App Engine. | Medium. Requires SimpleDB-specific interface. |
| **Transaction capabilities** | Yes | Yes | Yes | Yes |
| **Configurability and ability to tune database** | High. MySQL instantiation in cloud. | Medium. Can create indexes and stored procedures, but no control over memory allocation or similar resources. | Low | Low |
| **Database accessible as "stand-alone" offering** | Yes | Yes | No. Requires Google App Engine application layer. | Yes |
| **FISMA Certified** | No | No | No | No |
| **Can designate where data is stored (e.g., region or data center)** | Yes | Yes | No | Yes |
| **Replication** | Yes | Yes | Yes | Yes |

Figure 1-1. Common Consideration

## 1.1 Common Considerations for Comparing DBaaS Offerings

While DBaaS provides a ripe opportunity for reducing costs and achieving the Federal CIO's vision, there are many considerations for Government IT decision makers in placing data into a cloud-based environment.

***Data Sizing***—Many DBaaS offerings have limits on the size of the data set that can be stored on their systems. For example, SQL Azure allows up to 50 gigabytes (GB) per database instance while Amazon RDS allows up to 1 terabyte (TB).

***Portability***—Portability and adherence to standards is a critical issue for ensuring Continuity of Operations (COOP) and to mitigate business risk (e.g., a provider going out of business or raising rates). The ability to instantiate a replicated version of the data "off-cloud" or in another cloud offering can provide Federal IT leadership with an extra level of assurance that they will not suffer a loss of data. This can be facilitated by standards, such as the use of a standard database query language (e.g., SQL).

***Transaction Capabilities***—Transaction capabilities are an essential feature for databases that need to provide guaranteed reads and writes. For example, financial systems that move money need to provide their users with an absolute certainty that the entire transaction either succeeded or failed. This level of guaranteed transaction is frequently referred to as an "ACID"[5] transaction. Because ACID transactions require processing and storage to ensure that all the data is either written or deleted as a unit, there is an intrinsic overhead. If this level of guarantee is not needed, there could be an opportunity for lower cost, better scalability, or faster performance through non-ACID transactions.

***Configurability***—DBaaS offerings may provide capabilities that reduce the amount of configuration options available to database administrators. For some applications, if more configurability options are managed by the platform owner rather than the

customer's database administrator, it can reduce the amount of effort expended to maintain the database. For others, the inability to tune and control all aspects of the database, such as memory management, can be a limiting constraint in obtaining performance.

***Database Accessibility***—Most DBaaSs offer a predefined set of connectivity mechanisms that will directly impact adoption and use. There are three general approaches. First, RDBMS offerings are typically accessible through industry standard database drivers such as Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC). These drivers allow for applications external to the service to access the database through a standard connection, facilitating interoperability. Second, NoSQL services typically provide interfaces that use standards-based, Service-Oriented Architecture (SOA) protocols, such as SOAP or REST, with Hypertext Transfer Protocol (HTTP) and a vendor-specific API definition. These services may provide software development kits in common source-code languages to facilitate the adoption. Third, some NoSQL databases may be restricted to accessing data through software running in the vendor's ecosystem. This approach may increase security, but it also significantly limits portability and interoperability.

***Certification and Accreditation (e.g., FISMA)***—Prior certification and accreditation can facilitate the adoption of a cloud platform. In order to mitigate the potential expense and risk of performing a new C&A evaluation on a DBaaS offering, Federal leaders can consider acquiring an Infrastructure as a Service (IaaS) offering through the General Services Administration's (GSA) apps.gov blanket purchase order hosting their own database software. This option would require the consuming organization to acquire their own database licenses, implement redundancy features such as replication, and patch their software as necessary.

As of May 2011, we know of no DBaaS offering that has gone through a government C&A evaluation. Nevertheless, given the government's drive to employ clouds, we believe it is only a matter of time before a Federal Information Security Management Act (FISMA) certified DBaaS offering becomes available.

***Data Integrity, Security, and Storage Location***—Ensuring appropriate security and data integrity controls are in place and codified in contractual terms is essential to ensure that data will be handled appropriately. As part of these efforts, a project or agency-specific cloud service can be acquired with terms and conditions covering its usage (e.g., background checks on personnel, data kept in the continental United States, vendor reporting, audit records), and appropriate certification and accreditation (C&A) activities (e.g., FISMA Moderate certification). For example, the recent GSA IaaS blanket purchase order award specified requirements for FISMA Moderate. For additional security, federal IT leaders can employ encryption of sensitive information while it is in transit and stored in the cloud. This extra protection of sensitive information derived through encryption can extend the level of control that Government leaders have over their data stored outside their premises. However, there are trade-offs. For example, encrypted information cannot be directly used within database queries and will need to be retrieved for unencryption and processing.

***Availability and Replication***—The ability to ensure that data is available and not lost will be a key consideration. Ensuring access to data can come through enforcement of service-level agreements (SLA) metrics such as up time, replication across a cloud provider's regions, and replication or movement of the data across cloud providers or to the consuming organization's data center.

- Replication across a cloud provider's hardware within a region may ameliorate the effects of a localized hardware or software failure.
- Replication across a cloud provider's geographic regions may ameliorate the effects of a network outage, natural disaster, or other regional event.
- Replication across multiple cloud providers or back to the consuming organization's Federal IT infrastructure may provide the most COOP benefit through full geographic and IT stack independence.

Many providers such as Microsoft and Amazon offer replication of the data across hardware within a specific region as part of a packaged service. Within a given vendor, replication across geographies is usually more expensive and may result in significant data transfer fees.

***Identity and Access Management***—Enterprise scale key management and access controls are essential for Government organizations considering

adopting a cloud platform. Controls for database offerings can include key pairs for networked access to the cloud, user names and passwords for access to management portals and billing information, and user names and passwords for access to databases. Government IT leaders will need to ensure that these controls are fine-grained so that individual users can only have access to the level of control and data needed for their specific role. For example, a database administrator (DBA) may need full control over several database instances and a user may only need access to one database at a user level without the ability to grant privileges or change tuning parameters.

***Cost***—Federal leadership will want to compare the costs of cloud-based approaches from different vendors and locally hosted options. Cloud-based vendors typically charge for the amount of data stored and the volume of data moved in and out of the database or cloud platform. Therefore, remotely hosted input/output (I/O) intensive applications can drive significant costs. As shown in Figure 1-2, there is not a consistent pricing model across vendors. Therefore, a financial analysis with a set of usage assumptions should be conducted in order to understand the cost trade-offs. The financial analysis should include additional costs that may arise from acquiring a cloud-based service, including porting applications, migrating data, and performing requisite C&A activities. Ongoing operations costs will include monitoring the vendor's performance.

| | Amazon RDS (MySQL) | Microsoft SQL Azure | Google Datastore | Amazon SimpleDB |
|---|---|---|---|---|
| Example Pricing for Processing (Refer to Sections 2-5 for details)* | Ranges from $0.11 per RDS hour for smallest instance to $2.60 per hour for largest instance | Ranges from $9.99 per database with up to 1 GB of storage to $499.95 per database with up to 50 GB of storage per month | $0.10 per App Engine CPU hour (required for accessing Datastore) | $0.14 per SimpleDB unit hour |
| Example "On-demand" Pricing for Data Transfers (Refer to Sections 2-5 for details)* | Inbound $0.10 per GB and outbound ranges from $0.15 per GB to $0.08 per GB, depending on volume | Inbound $0.10 per GB<br>Outbound $0.15 per GB | Inbound $0.10 per GB<br>Outbound $0.12 per GB | Inbound $0.10 per GB and outbound from $0.15 to $0.08 per GB, depending on volume. |
| Example Monthly Pricing for data storage (Refer to Sections 2-5 for details)* | $0.10 per GB plus $0.10 per 1 million I/O requests | Included in processing pricing | $0.15 per GB | $0.25 per GB |

Figure 1-2. Pricing

* Pricing can be a function of specific customer offerings, introductory specials, low volume free tiers, and level of availability/data replication. In addition, it can vary for different regions and data centers.

***Management Portal***—The ability to manage a cloud-based offering over a WAN through a browser interface can simplify use and control for Government operations organizations. Additional APIs may be provided for enhanced capabilities and customization that many Government organizations require. The combined portal and API options will likely be a benefit for Federal organizations looking to adopt cloud services; nevertheless, security architects will need to be aware of all potential attack surfaces to ensure that they are controlled appropriately.

***Performance***—Ensuring that performance meets the user needs is essential for providing cloud-based

| | Average writes per second | Average reads per second |
|---|---|---|
| Amazon MySQL RDS* | 2,567 | 2,551 |
| Microsoft SQL Azure | 406 | 410 |
| Google Datastore** | 288 | 200 |
| Amazon SimpleDB | 208 | 63 |

Figure 1-3. Single Test Client Write/Read Performance

* Large database, no replication (default)
** Master/Slave configuration (default)

database services. As shown in Figure 1-3, we assessed four DBaaS offerings with a single test client simulating the load from a single user or application running on the provider's network. We wrote and read records of approximately 100 characters to and from each database. For perspective, 100 characters is approximately equal to the amount of data in a name and address record, a simple ordering transaction, entry in an accounting system, or a positional record that includes latitude, longitude, altitude and some additional descriptive information. Also we simulated concurrent access to a database from five multiple applications or users. As shown in Figure 1-4, additional throughput can be achieved with concurrency. For the data shown in Figures 1-3 and 1-4, we benchmarked each DBaaS using the default configuration rather than exhaustively "tuning" each system. We felt this was the best approach to providing meaningful data for Government IT leaders.

While production system results would likely be different for Government organizations due to factors such as WAN latency and the complexity of the data, the test results demonstrate that different database services can have dramatically different performance characteristics. Before embarking on a full scale migration to a database service, Federal IT leaders should consider piloting services to understand the performance results that they will achieve using anticipated loads and production configurations.

***RDBMS vs. NoSQL***—The relational database model has been the foundation of databases for the past several decades. RDBMSs, offered by companies such as Oracle, IBM, and Sybase, and open source software (OSS) products such as MySQL and PostgreSQL provide a structured, relationship-based format for storage and an industry standard language (i.e., SQL) for queries. They offer many advanced features such as transactions and logging to ensure data integrity. These features have become indispensable for many systems across the Government, including financial transactions, ordering, and defense capabilities. RDBMSs manage the majority of data in Government systems. In return for these features, the RDBMS implementations have become expensive both in terms of the licensing costs and the capacity of the hardware needed to implement their features.

| | Average writes per second | Average reads per second |
|---|---|---|
| Amazon MySQL RDS* | 7576 | 7905 |
| Microsoft SQL Azure | 1737 | 1893 |
| Google Datastore** | N/A | N/A |
| Amazon SimpleDB | 689 | 281 |

Figure 1-4. Five Test Client Concurrent Write/Read Performance

* Large database, no replication (default)
** Master/Slave configuration (default)

While RDBMS databases are widely deployed and successful, they have shortcomings for some applications that have been filled by the growing use of "NoSQL" databases. Rather than conforming to SQL standards and providing relational data modeling, NoSQL databases typically offer fewer transactional guarantees than RDBMSs in exchange for greater flexibility and scalability. NoSQL databases tend to be less complex than mature RDBMSs and scale horizontally across lower-cost hardware. Noel Yuhanna of Forrester Research writes, "NoSQL is also a movement, a community of developers driving innovation in these new technologies to support dynamic, flexible schemas, storage optimized for Web scale, and easy access to unstructured and semistructured data."[6] Unlike RDBMSs, which share a common relational data model, several different types of databases, such as column-oriented, key-value, and document-oriented, are considered "NoSQL" databases.

NoSQL databases tend to be used in applications that do not require the same level of data consistency guarantees that RDBMS systems provide but that require throughput levels that would be prohibitively expensive for RDBMSs to support. ShutterFly, a Web-based photo site, uses MongoDB, a document-oriented NoSQL database, to store 6 billion images with up to 10,000 transactions per second.[7] Yuhanna adds, "Key-value store databases have been around since databases began, but today's key-value stores handle Web scale—thousands of servers and millions of users—with extremely fast, optimized storage and retrieval. Key-value stores accomplish this by leaving out many features of relational databases, implementing only features that extreme Web apps need."[8]

***Use of White Paper***—The information in this paper can be used by Federal leadership to understand the options and considerations for migrating a database to a community or public cloud environment. There are many considerations, such as performance, cost, COOP, and security, and providers are addressing these issues in different ways. Sections 2 through 5 provide the reader with insight into these factors by describing four offerings by three major cloud providers. Lastly, cloud-based products are being launched frequently in today's marketplace, and we anticipate that this white paper can be used as a framework for evaluating future offerings.

## 2.0 Amazon SimpleDB™ [9]

AWS's SimpleDB is a cloud-based NoSQL offering that stores data in "domains" with attribute-value pairs. SimpleDB has minimal configuration options, which simplifies the usage of SimpleDB, but also limits the levers that users have for tuning the database for their needs. For example, there are no configurations for indexing or memory cache size. As SimpleDB does not provide many tuning options, Government organizations using SimpleDB will likely implement a data access layer for applications to provide transaction support and improve performance and scalability.

***Cost***—AWS measures processor utilization by SimpleDB units defined to be approximately the same as a "circa 2007 1.7 GHz Xeon processor" hour. The first 25 hours per month are free, and subsequent hours are $0.14 per unit hour.[10] Data transfers inbound cost $0.10 per GB, and data transfers outbound range in cost from $0.15 per GB to $0.08 per GB, depending on usage; the first 1 GB per month is free.[11] Monthly data storage costs are $0.25 per GB.

***Maximum Database Size***—Federal organizations that can readily segment their data across many SimpleDB domains will be more likely to use SimpleDB.[12] Individual SimpleDB data domains cannot grow beyond 10 GB. However, organizations can create up to 250 domains with the default limit. More domains can be created with AWS' authorization.[13] In addition, while other databases handle multiple types of data, SimpleDB provides storage for UTF-8 standard data only.[14]

***Security Considerations***—AWS Identity and Access Management (IAM) provides a Web service that can be used to establish customized policies and access control for groups and individuals across a variety of AWS products, including SimpleDB.[15] For enterprise-scale Government organizations, fine-grained access credentials, such as keys, user names, and passwords, are important for maintaining an

### *CASE STUDY FROM INDUSTRY*

Alexa Web Search crawled the Internet every night and generated a Web-scale datastore with terabytes of data. They wanted to allow users to run custom queries against this data and generate up to 10 million results.[16] To provide this service, Alexa's architecture team leveraged a combination of AWS services that included EC2, S3, SQS, and SimpleDB. SimpleDB was used for status information because it was "schema-less." AWS' Jinesh Varia wrote, "There is no need to provide the structure of the record beforehand. Every controller can define its own structure and append data to a 'job' item."[17] SimpleDB allowed components of the architecture to independently and asynchronously read and write state information (e.g., status of jobs in-process). While a good fit for state information, SimpleDB, which had a 10 GB limit per domain, was not used for the nightly multi-terabyte Internet crawl.

In the Federal domain, the performance needs of this type of system would be similar to control information for large scale processing, such as payroll and inventory. Given that this information is used for state, reliability is essential for recovering from outages in the rest of the system. While this state information needs to be highly reliable, the size of the control data is significantly less than the production database.

appropriate security posture and helping with gathering audit information and facilitating employee role changes and turnover.

***Performance***—Using Amazon's Elastic Computing Cloud (EC2) to host a client application, we assessed the write and read performance capability of SimpleDB. Robust performance capabilities for database writes and reads are essential to the scalability and adequacy of many classes of government systems that use batches of data or that render data to users via a Web-based, thin client.

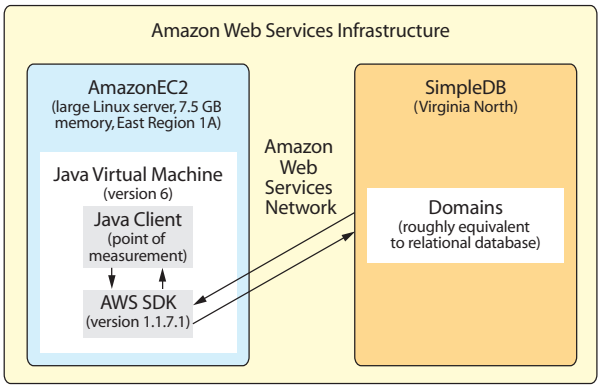


Figure 2-1. Evaluation of SimpleDB

As shown in Figure 2-1, the test client was hosted in a large EC2 instance running a Java Virtual Machine (JVM). Within the JVM, the test client software leveraged the API from the AWS SimpleDB software development kit (SDK). The SDK sent and received SOA, RESTful style messages to and from SimpleDB. The test measured the amount of time it took for the messages to be sent on the AWS network, for SimpleDB to do its work, and for a message to be returned on the AWS network.

We simulated the effect of multiple users requesting services from the database concurrently by employing multiple test clients. Handling multiple concurrent users is important for scaling databases to support enterprise class systems. Web-based systems, financial systems, and inventory systems are all examples that could anticipate this type of usage.

With the multiple test clients, we executed five request pipelines to the database. As shown in Figure 2-2, multiple test applications performed better than single tests with writes being faster than reads.[18]

| | Average writes per second | Average reads per second |
|---|---|---|
| 1 client JVM | 208 | 63 |
| 5 client JVMs | 689 | 281 |

Figure 2-2. AWS SimpleDB Performance

On several occasions, the test clients received messages from SimpleDB, indicating that SimpleDB could not handle more requests and to try again. This "throttling" error message was returned to the requesting application. SimpleDB does not automatically retry the action; retries need to be requested by the application layer.

## 3.0 Amazon MySQL Relational Database Service™

Amazon RDS is a cloud-based RDBMS offered by AWS. At the time of the evaluation, RDS was based solely on MySQL. Since that time, AWS has introduced an Oracle-based RDS service.

RDS MySQL is based upon the ubiquitous open source MySQL database software. As a result, it is fully compliant with SQL standards and can host many capabilities developed for the locally instantiated version. AWS states, "Amazon RDS gives you access to the full capabilities of a familiar MySQL database. This means the code, applications, and tools you already use today with your existing MySQL databases work seamlessly with Amazon RDS."[19] In addition, Amazon RDS allows the consuming organization to select the database server capabilities that they need, including the memory size of the server, number of virtual CPU cores, and the I/O performance. This can be helpful for IT leadership looking to scale and ensure they meet their customer needs. It is also a differentiating factor from other DBaaS offerings that provide a uniform platform with limited performance options.

As with all multi-tenant, cloud DBaaS offerings, the decision process for adopting a cloud-based offering needs to balance the potential cost savings, scalability, performance, and other factors against the risks. For organizations looking to migrate their locally instantiated MySQL database to the cloud, RDS provides a service with the same underlying software product

### *CASE STUDY FROM INDUSTRY*

Airbnb, a vacation rental firm, kept its main database in Amazon RDS. The consistency between locally hosted MySQL and Amazon RDS MySQL facilitated the migration to AWS.[20] A significant architecture consideration for Airbnb was that Amazon provided the underlying replication infrastructure. "Amazon RDS supports asynchronous master-slave replication," wrote Tobi Knaup.[21] Knaup added that the hot standby, which ran in a different AWS Availability Zone, was updated synchronously with no replication lag. Therefore, if the master database failed, the standby was promoted to the new master with no loss of data.[22]

In the Federal domain, this level of integrity would be an important consideration for a system processing large volumes of data that cannot experience loss of information. The performance needs of this type of system would be similar to on-line inventory or logistics systems.

instantiated in the cloud. Therefore, database schema and associated database code (e.g., stored procedures) can be easily ported from the locally instantiated version to the cloud-based version.

RDS easily integrates with DBA tools and common MySQL development tools such as the MySQL Development WorkBench. The ability for operations and development staff to use familiar tools and have a high level of control over the database software can be an important factor in selecting a cloud-based DBaaS offering. As RDS is an instantiation of MySQL in the cloud, RDS affords the technical staff the levers for tuning the database, monitoring and managing their environment (e.g., memory), and tuning performance. For organizations that would like to maintain a high level of administration and tuning control in the cloud, this capability is an important feature.

***Cost***—Amazon lists the following on-demand instance per hour prices for a single availability zone:[23]

- Standard-Memory DB Instance Class
  - Small DB Instance: $0.11
  - Large DB Instance: $0.44
  - Extra Large DB Instance: $0.88
- High-Memory DB Instance Class
  - Extra Large DB Instance $0.65
  - Double Extra Large DB Instance $1.30
  - Quadruple Extra Large DB Instance $2.60

Amazon also charges for data transfers. All inbound data transfers are billed at $0.10 per GB. Outbound data transfers are volume based as follows:

- First 1 GB/month free
- Up to 10 TB/month $0.15 per GB
- Next 40 TB/month $0.11 per GB
- Next 100 TB/month $0.09 per GB
- Greater than 150 TB/month $0.08 per GB

Provisioned database storage is billed at $0.10 per GB. In addition, there is a charge of $0.10 per 1 million I/O requests.

As AWS has multiple offerings and updates prices regularly, these costs should be used as an example.

***Maximum Database Size***—As of May 2011, Amazon RDS provides the ability for each database instance to be configured with a minimum of 5 GB to a maximum of 1 TB of associated storage capacity. This amount of storage is sufficient for many database needs, but could be quickly oversubscribed for large volume transaction systems, sensor data, or binary images (such as photos or scanned documents).

***Security Considerations***—As with SimpleDB, AWS IAM provides a Web service that can be used with RDS to establish customized policies and access control for groups and individuals. For enterprise-scale Government organizations, fine-grained access credentials, such as keys, user names, and passwords, are important for maintaining an appropriate security posture and helping with gathering audit information and facilitating employee role changes and turnover.

***Performance***—Using Amazon's EC2 to host a client application, we assessed the write and read performance capability of RDS. Robust performance capabilities for database writes and reads are essential to the scalability and adequacy of many classes of government systems that use batches of data or that render data to users via a Web-based, thin client.

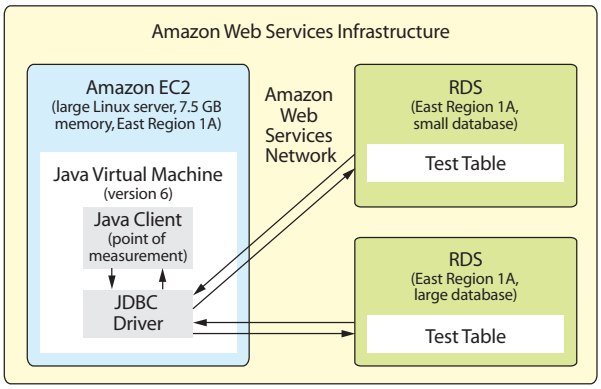


Figure 3-1. RDS Test Environment

As shown in Figure 3-1, the test client was hosted in a large EC2 instance running a JVM. Within the JVM, the test client software leveraged an industry standard JDBC driver to connect with the database. The test measured the amount of time it took for the messages to be sent on the AWS network, for RDS to do its work, and for a message to be returned on the AWS network.

We simulated the effect of multiple users requesting services from the database concurrently by employing a "multi-threaded" test client. Handling multiple concurrent users is important for scaling databases to support enterprise class systems. Web-based systems, financial systems, and inventory systems are all examples that could anticipate this type of usage.

With the multiple threads, we executed five separate pipelines of requests to the database. As shown in Figure 3-2, multi-threaded applications performed better than single threaded tests, but the effect was more pronounced with the large database than the small database. This result is as anticipated. As the small database reaches the maximum of its capabilities, the benefits of concurrent access decrease. However, the large database instance, which has extra memory and superior I/O, was able to process the additional concurrent transactions more efficiently.

| | Average writes per second | Average reads per second |
|---|---|---|
| Single threaded JVM/small database | 2,042 | 1,983 |
| Multi-threaded JVM/ small database | 4,782 | 4,173 |
| Single threaded JVM/large database | 2,567 | 2,551 |
| Multi-threaded JVM/ large database | 7,576 | 7,905 |

Figure 3-2. RDS Performance Results

## 4.0 Google Apps Datastore™ [24]

Google's App Engine is a PaaS offering that offers an integrated application and database environment. It supports applications written in Python, the Go open source project, and Java software languages. For each of these languages, App Engine provides a virtual application server environment that supports webpage generation and background processing.

App Engine implements a key-value Datastore for applications running in the App Engine environment. Datastore is not intended to be used for storage that is independent of the Google application layer as it cannot be accessed from outside of App Engine. Because the App Engine Datastore is so closely tied to the rest of the App Engine environment (and vice versa), the decision to use the Datastore is not an independent decision—it is part of the decision of whether to write an application for App Engine or use a different PaaS engine or an IaaS provider.

Google provides three App Engine software development kits: one for each programming environment that App Engine supports. Each kit includes an App Engine specific API and a run-time environment that developers can use to run their applications locally for development and local testing. Java Data Objects (JDOs) can also be used for industry standard access. As the App Engine specific APIs may limit future portability options, architects and developers should consider using the JDO access for future portability.

There are two additional categories of factors that Federal IT leadership should consider when using Google's Datastore. The first category is driven by Google's PaaS architecture, and the second category

### CASE STUDY FROM INDUSTRY

Giftag, a Web application provided by Best Buy, provided a gift registry capability for Internet Explorer and Firefox. Giftag enabled users to add items to wish lists and share the wish lists via the Internet or Facebook.[25] Prior to moving to the cloud, the Giftag developers had technical skills and a positive experience with Django, a Python Web framework. Given that Google App Engine offered a Python-based environment, the technology match was a consideration in their choice of cloud platforms.[26] Additionally, they viewed scalability, cost, and ease of deploying the application to an operational environment as an important factor.[27]

In the Federal domain, the performance needs of this type of system would be similar to existing systems for conveying information to the public or for registration (e.g., "cash for clunkers rebates"). Availability and reliability are highly desirable attributes for citizen satisfaction, but not at the level of national security systems.

is inherent to Datastore's NoSQL architecture. Due to Google's PaaS architecture, IT leadership must trust Google to provide the entire infrastructure not just the database. They need to cede control of capabilities such as where the code operates and where the data is stored and backed up. Additionally, interoperability beyond simple Web-based applications is challenging. Access to an application's Datastore, for example, is not possible except from the application. For example, developers cannot assume that they can plug a third-party report-generation tool into their applications.

***Cost***—App Engine applications may store up to 1 GB at no cost. Additional storage rates are:

- Master/Slave $0.15 per GB per month
- High Replication $0.45 per GB per month.

Google charges $0.10 per GB for incoming data transfers and $0.12 per GB for outbound data transfers.[28] App Engine CPU usage (for writing and reading Datastore information) is billed at $0.10 per CPU hour.

The App Engine Datastore supports two modes of operation: Master/Slave and High Replication. Master/Slave is faster and less resource-intensive (therefore lower cost), but can occasionally suffer planned downtime.[29] High Replication is more expensive but offers stronger availability guarantees. High Replication is also rate-limited to one write per second per resource group. Therefore, it should be used only where the developer is certain that the application write rates will not exceed that limit.[30] Because of this constraint, we did not test the performance of High Replication.

***Maximum Database Size***—Maximum database sizes are not published for "billing enabled" applications.[31] Nevertheless, there are published limits that can indirectly determine size. For example, each entity (roughly equivalent to a row in an RDBMS database) is limited to 1 MB.[32] The maximum number of values for an entity can be 5,000, and the number of indexes is restricted to 200.[33]

***Security Considerations***—Datastore databases are managed through a Web-based administration portal, which uses Google Accounts as an authentication mechanism. This approach provides coarse-grained access that may not be sufficient for Government organizations that need controls for specific users.

As a dedicated back-end storage mechanism for App Engine applications, Datastore does not provide direct access for development tools or applications owned by different accounts. From a functionality perspective, this may be considered limiting; however, from a security perspective, it minimizes the avenues into a Datastore database.

***Performance***—We used the Google App Engine PaaS environment to assess read and write performance capabilities of the App Engine Datastore. Due to resource quota limitations, we were restricted to running smaller-scale tests than those conducted for other database products in this report.[34] We wrote and read 100,000 records rather than the 1 million records in the other tests. While the duration was shorter, we were still able to obtain a number for writes and reads per second.

Google's App Engine is a PaaS-style capability that provides very abstract runtime support. Therefore, the geographic location of the data center and the precise servers that hosted our application and database service was not known to us.

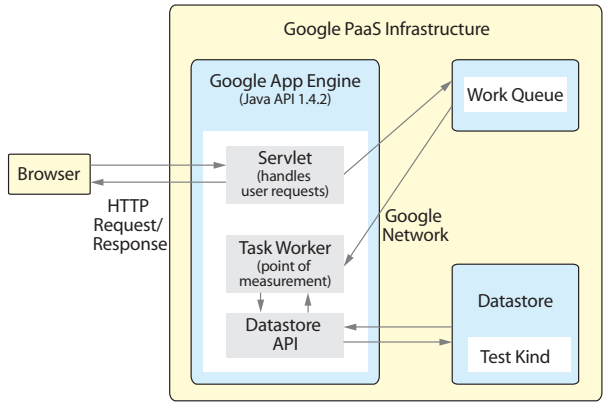


Figure 4-1. Google PaaS App Engine and Datastore

As shown in Figure 4-1, the test client was hosted in the Google App Engine cloud. The test measured the amount of time it took for the messages to be sent on the Google network, for Datastore to do its work, and for the messages to be returned on the Google network.

Over time, App Engine enforces resource constraints on applications that may be too restrictive for Government IT leadership. Individual application requests will be killed if they take longer than App Engine considers acceptable. In addition, applications are constrained in their aggregate resource consumption over longer periods.[35] Examples of the first type of constraint include time limits on how long an individual Web request can take, how long a background process can take, and how long an HTTP client-side request from App Engine to the outside world can take. Examples of the second type of constraint include per-minute and per-day resource quotas for CPU time, database storage, and API calls. Quota restrictions and timeouts are documented and available to developers and platform administrators when acquiring the service, but not necessarily end-user consumers of the service. Quotas and restrictions are, in general, less restrictive for paid apps and more restrictive for no-cost apps; IT leadership may request that Google lift some restrictions, but they should not assume that Google will do so.

| | Average writes per second | Average reads per second |
|---|---|---|
| Master Slave Test | 288 | 200 |

Figure 4-2. Google Datastore Performance

App Engine applications are constrained in their ability to open connections to resources outside the App Engine environment. This would make using a database management system other than the App Engine Datastore difficult and inefficient. It is important that Federal IT leadership considering implementing an App Engine application understand the Datastore's data model thoroughly.

As shown in Figure 4-2, the average write performance of a single App Engine task writing 100,000 entities was 288 entities per second. (For the other databases in this document, we wrote 1 million database records.)

The average read performance of a single task reading 50,000 entities of a single entity kind was 200 entities per second.

## 5.0 Microsoft SQL Azure™ [36]

Microsoft offers an SQL-based RDBMS in the cloud with its SQL Azure Database. The similarity of cloud-based RDBMS offerings, such as SQL Azure and Amazon's RDS, with existing RDBMS products benefits a development community that relies largely on traditional relational models of data sets and hopes to model normalized entity relations in its data stores. The Microsoft cloud database model should look familiar to Microsoft SQL Server users; a high level of compatibility is suggested between the products. For example, familiar concepts such as tables, schemas, indexes, views, stored procedures, and triggers can be found in the online service.

The use of the "remote service in a cloud" paradigm removes some workload from MS SQL Server administrators. For example, Microsoft writes, "Unlike administration for an on-premise instance of SQL Server, SQL Azure abstracts the logical administration from the physical administration; you continue to administer databases, logins, users, and roles, but Microsoft administers the physical hardware such as hard drives, servers, and storage. … SQL Azure automatically replicates all data to

### *CASE STUDY FROM INDUSTRY*

Xerox Corporation ported an on-premise enterprise print capability to a public cloud environment. This capability allowed mobile users to find printers with their smartphones and route printouts. As the on-premise version leveraged Microsoft SQL Server for the database component, Xerox selected Microsoft SQL Azure for cloud storage.[37] This approach allowed them to reuse their prior investments in SQL Server-based technology and .NET, and minimize the technical challenges of porting to a cloud based environment.[38] They were also able to minimize their skills-based challenges because the development team was trained on Microsoft products.

Xerox used SQL Azure for "user account information, job information, device information, print job metadata, and other such data," but the actual print files were stored in Azure Blob Storage, not SQL Azure.[39] Azure Blob Storage had different pricing and characteristics than SQL Azure. For example, unlike SQL Azure, Blob Storage was not limited to 10 GB (Web edition) or 50 GB (Business edition).[40]

In the Federal domain, the performance needs of this type of system would be similar to existing print capabilities or on-line public information systems. While availability and reliability would be desired, they are not mission-critical attributes (beyond the potential loss of fees for usage paid by citizens to the government).

provide high availably. SQL Azure also manages load balancing and, in case of a server failure, transparent fail-over. To provide this level of physical administration, you cannot control the physical resources of SQL Azure."[41] By giving up control of physical administration details, the consumer relies on Microsoft to control the high availability architecture.

As with all multi-tenant, cloud DBaaS offerings, the decision process for usage needs to balance the potential cost savings, scalability, performance, and other factors against the risks. As SQL Azure leverages many Microsoft-specific concepts and capabilities, organizations that are already using Microsoft development and operational environments will have an easier time transitioning to SQL Azure. The database schema and database code (e.g., stored procedures) can be easily ported from SQL Server, but database administration tools and processes may not be useable. For example, Microsoft does not give consumers the ability to tune usage of memory and other underlying configuration options.

***Cost***—Example SQL Azure monthly prices are listed below.[42] Specific costs can be affected by vendor offerings, such as introductory specials, and regional differences in pricing.

- Web Edition
  - $9.99 per database up to 1 GB per month
  - $49.95 per database up to 5 GB per month
- Business Edition
  - $99.99 per database up to 10 GB per month
  - $199.98 per database up to 20 GB per month
  - $299.97 per database up to 30 GB per month
  - $399.96 per database up to 40 GB per month
  - $499.95 per database up to 50 GB per month
- Data transfer costs:
  - Inbound $0.10 per GB
  - Outbound $0.15 per GB.

Other than size and price, the features are the same for the Web and Business versions; however, in later offerings, capabilities for the Business Edition may go beyond the Web version.[43] Example technologies that may be offered in the Business Edition are for optimization and application access capabilities that are currently available in SQL Server but not SQL Azure.

***Maximum Database Size***—As of April 2011, Microsoft requires databases to be under 10 GB per instance for Web databases and 50 GB per instance for Business databases. The 10 GB or 50 GB offerings may be sufficient storage for a small system or single purpose database, but it is significantly smaller than the storage needs of many enterprise databases.

***Security Considerations***—SQL Azure allows an administrator user name and password for each database.[44] Once these credentials have been created, an administrator can establish users with access that is more restricted. In addition to user name and password authentication, an administrator can require requestors be from a specified Internet Protocol (IP) address, range of IP addresses, or within the Microsoft Azure ecosystem (e.g., Windows Azure).

***Performance***—Using Microsoft Azure to host a client application, we assessed the write and read performance capability of SQL Azure. Robust performance capabilities for database writes and reads are essential to the scalability and adequacy of many classes of government systems that use batches of data or that render data to users via a Web-based, thin client.

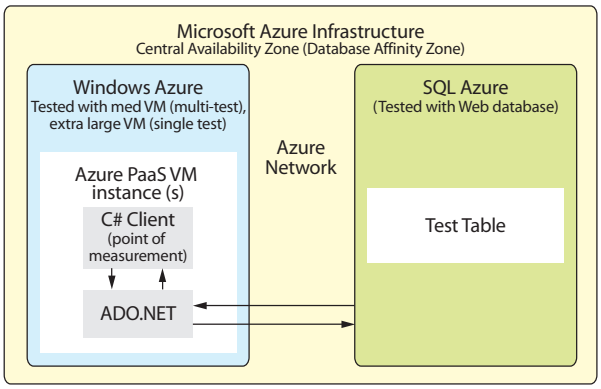


Figure 5-1. SQL Azure Test Environment

As shown in Figure 5-1, the C# test client was hosted in a Windows Azure instance running within the same "database affinity zone" as SQL Azure. The test measured the amount of time it took the messages to be sent on the Microsoft Azure network, for SQL Azure to do its work, and for the messages to be returned on the Microsoft Azure network.

| | Average writes per second | Average reads per second |
|---|---|---|
| Single client instance | 406 | 410 |
| Multi-client instance | 1,737 | 1,893 |

Figure 5-2. SQL Azure Benchmark Results

The test was to write and read 1 million records with approximately 100 characters of data per record. The objective was to test under specific circumstances, rather than tune the test for the fastest possible execution. The default configuration of all environments was utilized for the test.

The test results showed that writes and reads were executed at approximately the same rate (see Figure 5-2). As anticipated, the multi-virtual machine (VM) test performed better than single VM tests. The test with five VMs showed that the database could handle the extra concurrent load with a near linear improvement in performance.

## Acronyms

| Acronym | Definition |
|---|---|
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| C&A | Certification and Accreditation |
| CIO | Chief Information Officer |
| COOP | Continuity of Operations |
| CPU | Central Processing Unit |
| DBaaS | Database as a Service |
| DBA | Database Administrator |
| EC2 | Elastic Cloud Computing |
| FFRDC | Federally Funded Research and Development Centers |
| FISMA | Federal Information Security Management Act |
| GB | Gigabyte |
| GSA | General Services Administration |
| http | Hypertext Transfer Protocol |
| I/O | Input/Output |
| IaaS | Infrastructure as a Service |
| IAM | Identity and Access Management |
| IP | Internet Protocol |
| IT | Information Technology |
| JDBC | Java Database Connection |
| JDO | Java Data Objects |
| JVM | Java Virtual Machine |
| NoSQL | No Structure Query Language |
| PaaS | Platform as a Service |
| RDBMS | Relational Database Management System |
| RDS | Relational Database Service |
| SDK | Software Development Kit |
| SLA | Service-Level Agreement |
| SOA | Service-Oriented Architecture |
| SQL | Structured Query Language |
| TB | Terabyte |
| VM | Virtual Machine |
| WAN | Wide Area Network |

## References


[1] "Datastore Overview," *Google App Engine*
http://code.google.com/appengine/docs/python/datastore/overview.html.

[2] "Amazon Relational Database Service Pricing," *Amazon Web Services,*
http://aws.amazon.com/rds/pricing/.

[3] "Database Count and Size Limits," *MSDN Library,*
http://msdn.microsoft.com/en-us/library/ee336245.aspx#dcasl.

[4] "How much data can I store?,"
http://aws.amazon.com/simpledb/faqs/#How_much_data_can_I_store.

[5] ACID stands for atomicity, consistency, isolation, and durability. For more information on ACID transactions, refer to
http://msdn.microsoft.com/en-us/library/aa719484(v=vs.71).aspx

[6] Yuhanna, N., M. Gilpin, and A. Knoll, November 19, 2010, "Stay Alert To Database Technology Innovation,"
http://www.forrester.com/rb/Research/stay_alert_to_database_technology_innovation/q/id/57947/t/2.

[7] Harrison, G., January 28, 2011, "Real World NoSQL: MongoDB at Shutterfly," *The New York Times,* accessed November 14, 2011
http://www.nytimes.com/external/gigaom/2011/01/28/28gigaom-real-world-nosql-mongodb-at-shutterfly-165.html?partner=rss&emc=rss.

[8] Yuhanna, N., M. Gilpin, and A. Knoll, November 19, 2010, "Stay Alert To Database Technology Innovation," accessed November 14, 2011
http://www.forrester.com/rb/Research/stay_alert_to_database_technology_innovation/q/id/57947/t/2.

[9] Information in this subsection is drawn in part from the online manuals provided by Amazon.com.
http://aws.amazon.com/documentation/simpledb/.

[10] "Amazon SimpleDB Pricing," *Amazon Web Services,*
http://aws.amazon.com/simpledb/pricing/.

[11] ibid.

[12] Conceptually SimpleDB domains are similar to an unrelated RDBMS database table.

[13] "Amazon SimpleDB FAQs: How much data can I store?," *Amazon Web Services,*
http://aws.amazon.com/simpledb/faqs/#How_much_data_can_I_store.

[14] "Amazon SimpleDB FAQs: What kind of data can I store?," *Amazon Web Services,*
http://aws.amazon.com/simpledb/faqs/#What_kind_of_data_can_I_store.

[15] 2011, "AWS Identity and Access Management Using IAM API Version 2010-05-08," *Amazon Web Services,*
http://awsdocs.s3.amazonaws.com/IAM/latest/iam-ug.pdf.

[16] Varia, J., June 2008, Cloud Architectures, *Amazon Web Services.*

[17] ibid.

[18] Typically writes would not be faster than reads, but in this case it is reasonable. The write test required 25 times fewer round trips between EC2 and SimpleDB, due to the batching capability provided by the SDK.

[19] "Amazon Relational Database Service (Amazon RDS)," *Amazon Web Services,* accessed November 14, 2011
http://aws.amazon.com/rds/.

[20] http://aws.amazon.com/solutions/case-studies/airbnb/

[21] http://nerds.airbnb.com/mysql-in-the-cloud-at-airbnb

[22] ibid.

[23] "Amazon Relational Database Service Pricing," *Amazon Web Services,* accessed November 14, 2011
http://aws.amazon.com/rds/pricing/.

[24] Information in this subsection is drawn in part from the online manuals provided by Google at http://code.google.com/appengine/.

[25] http://code.google.com/appengine/casestudies.html#giftag

[26] http://www.youtube.com/watch?v=uwFvCz4pkMQ

[27] ibid.

[28] "Billing and Budgeting Resources," *Google App Engine,* http://code.google.com/appengine/docs/billing.html#Billable_Quota_Unit_Cost.

[29] "Choosing a Datastore (Java)," *Google App Engine,* http://code.google.com/appengine/docs/java/datastore/hr/

[30] "Using the High Replication Datastore," *Google App Engine,* http://code.google.com/appengine/docs/java/datastore/hr/overview.html.

[31] "Quotas," *Google App Engine,* http://code.google.com/appengine/docs/quotas.html.

[32] "Datastore Overview," *Google App Engine,* http://code.google.com/appengine/docs/python/datastore/overview.html#Quotas_and_Limits.

[33] "Quotas," *Google App Engine,* http://code.google.com/appengine/docs/quotas.html.

[34] After we ran our tests, Google introduced a new feature "Backends" which allows requests to run indefinitely. Refer to http://code.google.com/appengine/docs/python/backends/overview.html#Properties_of_Backends

[35] "Quotas," Google App Engine http://code.google.com/appengine/docs/quotas.html.

[36] Lee, J., G. Malcolm, and A. Matthews, September 2009, http://go.microsoft.com/?linkid=9686976. Microsoft.com.

[37] http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?CaseStudyID=4000008986

[38] http://www.youtube.com/watch?v=zHr8gwwJyzg

[39] http://www.microsoft.com/windowsazure/pricing/

[40] ibid.

[41] "SQL Azure Overview," *MSDN Library,* http://msdn.microsoft.com/en-us/library/ee336241.aspx.

[42] "Pricing Overview," *Microsoft Windows Azure,* http://www.microsoft.com/windowsazure/pricing/#sql.

[43] ibid.

[44] "Exercise 1: Preparing Your SQL Azure Account," *MSDN Library,* http://msdn.microsoft.com/en-us/gg282144.

MITRE

www.mitre.org


Approved for Public Release
Distribution Unlimited
Case Number: 11-4727
Document Number: MTR110536

MITRE

www.mitre.org