Storing and Processing Sensor Networks Data in Public Clouds

Aysun Simitci
# Table of Contents

- Introduction .................................................................................................................. 2
- Cloud Databases ........................................................................................................... 2
- Advantages and Disadvantages of Cloud Databases .................................................... 3
- Amazon Relational Database Service ........................................................................... 3
- Microsoft SQL Azure .................................................................................................... 4
- Comparison of Cloud Databases .................................................................................. 4
  - Cloud Blob Storage ..................................................................................................... 5
- Windows Azure Blob Storage ........................................................................................ 5
  - About Page Blobs ......................................................................................................... 5
  - HTTP Operations on Blob Service Resources ............................................................ 5
  - Storage Pricing .............................................................................................................. 6
- Amazon Simple Storage Service (Amazon S3) .............................................................. 6
  - Amazon S3 Functionality ............................................................................................. 6
  - Storage Pricing ............................................................................................................. 7
  - Request Pricing ............................................................................................................ 7
  - Transferring Large Amounts of Data .......................................................................... 7
  - Cloud Table Storage .................................................................................................... 7
- Windows Azure Table Storage ...................................................................................... 7
  - PartitionKeys and RowKeys Drive Performance and Scalability .................................. 8
- Amazon SimpleDB ......................................................................................................... 8
  - Amazon SimpleDB Functionality .............................................................................. 8
  - Pricing .......................................................................................................................... 9
  - Amazon SimpleDB Pricing .......................................................................................... 9
  - Windows Azure Table Storage Pricing ......................................................................... 9
  - Summary of Cloud Storage Alternative ....................................................................... 10
  - Sensor Cloud Gateway ............................................................................................... 11
- Design and Architecture ............................................................................................... 11
- Database Design ........................................................................................................... 11
- Code Samples ............................................................................................................... 13
- Output Samples ............................................................................................................. 14
**Introduction**

In this research, alternatives for storing sensor data in the public cloud are surveyed.

Storing data in the public cloud requires unique considerations compared to private grid networks. First, public clouds have higher network latencies and require optimizing the data transmission. For example, data can be buffered in gateway servers and forwarded to the cloud in an optimized way. Secondly, public clouds require special consideration for the security of the transmission. Third, to provide reliability and availability, the data will need to be replicated between multiple cloud systems.

In summary, the proposed system will store sensor data using optimized gateways to multiple cloud systems using secure protocols.

Cloud computing is the next stage in evolution of the Internet. The cloud in cloud computing provides the means through which everything — from computing power to computing infrastructure, applications, business processes to personal collaboration — can be delivered to you as a service wherever and whenever you need.

In general the cloud is fluid and can easily expand and contract. This elasticity means that users can request additional resources on demand and just as easily release those resources when they’re no longer needed. This elasticity is one of the main reasons individual, business, and IT users are moving to the cloud.

There are three types of cloud storage services common among many cloud vendors:

- Binary Large Object (BLOB) Service, the simplest way to store text or binary data (Windows Azure Blob Storage, Amazon S3)
- Table Service is better for large amounts of data that need additional structure, which works exceptionally well with applications that need to work with data in a very detailed manner via queries (Windows Azure Table Storage, Amazon SimpleDB).
- Queue Service for reliable, persistent messaging between Web and Worker role instances (Windows Azure Queue Service, Amazon Queues).

**Cloud Databases**

Cloud databases are web-based services, designed for running queries on structured data stored on cloud data services. Most of the time, these services work in conjunction with cloud compute resources to provide users the capability to store, process, and query data sets within the cloud environment.
These services are designed to make web-scale computing easier and more cost-effective for developers. Traditionally, this type of functionality was provided using a clustered relational database that requires a sizable investment. Implementations of this nature brought on more complexity and often required the services of a database administrator to maintain it.

By comparison to traditional approaches, cloud databases are easy to use and provide the core functionality of a database (e.g., real-time lookup and simple querying of structured data) without inheriting the operational complexity involved in traditional implementations.

**Advantages and Disadvantages of Cloud Databases**
The benefits of using a cloud-based relational solution are the same as the benefits offered by the rest of the cloud-based platform:

- No hardware and physical installation is required.
- Patches and updates are applied automatically.
- High availability and fault tolerance are built in.
- Provisioning is simple and you can deploy multiple databases.
- Databases can be scaled up or down based on business needs.
- The infrastructure supports multitenancy (multiple users).
- There is integration with existing database tools and technologies.
- In most services, you have the option for the pay-as-you-go pricing.

Databases are repositories for information with links within the information that help make the data searchable. Distributed databases, like Amazon RDS, spread information among physically dispersed hardware. But to the client, the information seems to be located in one place.

There are currently no standards to convert a centralized database into a cloud solution. Each service provides its own conversion tool.

**Amazon Relational Database Service**
Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. This service provides elastic capacities and at the same time handles time-consuming database administration tasks. Because of automated database backups and snapshots, the Amazon RDS is highly reliable: a database instance can be recovered for any point in time or recovery point that lies within the agreed retention period.

Amazon CloudWatch allows users to monitor the utilization of the computing and storage capacities of their database instances and to scale the available resources vertically using a simple API call as needed. In connection with highly demanding applications involving many read operations, it is possible to scale out by launching so-called Read Replica instances. A corresponding high-availability offering allows the provisioning of synchronously replicated database instances without additional costs in multiple availability regions as a safeguard against failures at a single location. This way, it is possible to mask maintenance windows as RDS switches the database services transparently between the locations.
Amazon RDS enables access to all MySQL database functions so it is easy to migrate existing applications while maintaining the preferred database tools and programming languages. If an existing application already uses a MySQL database, the data can be exported with mysqldump and then be piped directly into Amazon RDS.

Amazon RDS selects the optimum configuration parameters for database instances, taking the relevant computing resources and storage capacity requirements into account. However, it is also possible to change the default setting through configuration management APIs. It is not possible to set the database parameters by directly accessing the servers through the SSH.

For the management of its database services, Amazon not only offers command line tools and libraries for various programming languages, but also a convenient web-based management console.

**Microsoft SQL Azure**

Microsoft SQL Azure is a cloud-based relational database service built on SQL Server technologies. It is a highly available, scalable, multi-tenant database service hosted by Microsoft in the cloud. SQL Azure helps to ease provisioning and deployment of multiple databases. Developers do not have to install, setup, and patch or manage any software. High availability and fault tolerance is built-in and no physical administration is required.

Customers can use existing knowledge in T-SQL development and a familiar relational data model for symmetry with existing on-premises databases. Additionally, customers can get productive on SQL Azure quickly by using the same development and management tools they use for on-premises databases.

Microsoft is using a modified version of the SQL Server engine to provide a highly available and scalable database service on the Windows Azure Platform. SQL Azure supports T-SQL, so you can use almost any technology that produces queries.

Windows Azure tables are optimized for large-scale, cheap data access. However, Windows Azure tables do not support standard relational database management system (RDBMS) features such as referential integrity, the SQL language, or the ecosystem of tools around SQL Server. SQL Azure, on the other hand, is meant to extend your existing investments in SQL into the cloud.

### Comparison of Cloud Databases

<table>
<thead>
<tr>
<th></th>
<th>Microsoft SQL Azure</th>
<th>Amazon Relational Database Service</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Security</strong></td>
<td>Firewall, Active directory, Windows Live ID, Facebook</td>
<td>Firewall, Amazon Account</td>
</tr>
<tr>
<td><strong>Language Support</strong></td>
<td>ADO.NET, ASP.NET, C#, PHP, JDBC, Java</td>
<td>Java, PHP, Python, Ruby, .NET</td>
</tr>
<tr>
<td><strong>Price</strong></td>
<td>$10 – $1000 per month (1GB – 100GB)</td>
<td>$.11 – $2.60 per hour + $0.10 per GB-month</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>No geo-replication</td>
<td>Can get another datacenter for double the price</td>
</tr>
</tbody>
</table>
As the above table shows, both SQL Azure and Amazon RDS have mostly similar features. However, SQL Azure is oriented towards Microsoft technologies while Amazon RDS works best with open source tools.

Both cloud databases provide the benefits of cloud services such as usage based pricing, scalability, ease of use, and automated administration. When compared to other cloud storage technologies like table and blob storage, cloud databases provide easier migration from existing databases and provide complex query capabilities.

**Cloud Blob Storage**

A blob (Binary Large Object) is a collection of binary data stored as a single entity in a database management system. Blobs are typically images, audio or other multimedia objects, though sometimes binary executable code is stored as a blob. Database support for blobs is not universal.

**Windows Azure Blob Storage**

The storage service offers two types of blobs, block blobs and page blobs. You specify the blob type when you create the blob. Once the blob has been created, its type cannot be changed, and it can be updated only by using operations appropriate for that blob type, i.e., writing a block or list of blocks to a block blob, and writing pages to a page blob.

Block blobs let you upload large blobs efficiently. Block blobs are comprised of blocks, each of which is identified by a block ID. You create or modify a block blob by writing a set of blocks and committing them by their block IDs. Each block can be a different size, up to a maximum of 4 MB. The maximum size for a block blob is 200 GB, and a block blob can include no more than 50,000 blocks. If you are writing a block blob that is no more than 64 MB in size, you can upload it in its entirety with a single write operation.

**About Page Blobs**

Page blobs are a collection of 512-byte pages optimized for random read and write operations. To create a page blob, you initialize the page blob and specify the maximum size the page blob will grow. To add or update the contents of a page blob, you write a page or pages by specifying an offset and a range that align to 512-byte page boundaries. A write to a page blob can overwrite just one page, some pages, or up to 4 MB of the page blob. Writes to page blobs happens in-place and are immediately committed to the blob. The maximum size for a page blob is 1 TB.

**HTTP Operations on Blob Service Resources**

This topic lists HTTP operations supported by each resource type. The Blob service exposes the following resource types via the REST API:
Account: A storage account is a globally uniquely identified entity within the storage system. The account is the parent namespace for the Blob service. All containers are associated with an account.

Containers: A container is a user-defined set of blobs within an account. A container resource has no associated content, only properties and metadata.

Blobs: A blob is an entity representing a set of content. A blob resource includes content, properties, and metadata.

Storage Pricing

<table>
<thead>
<tr>
<th>Standard pay-as-you-go pricing for storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.15 per GB stored per month based on the daily average</td>
</tr>
<tr>
<td>$0.01 per 10,000 storage transactions</td>
</tr>
<tr>
<td>North America and Europe regions: $0.15 per GB out</td>
</tr>
</tbody>
</table>

Amazon Simple Storage Service (Amazon S3)

Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites.

Amazon S3 Functionality

Amazon S3 is intentionally built with a minimal feature set.

- Write, read, and delete objects containing from 1 byte to 5 terabytes of data each. The number of objects you can store is unlimited.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
- A bucket can be stored in one of several Regions. Amazon S3 is currently available in the US Standard, EU (Ireland), US West (Northern California), Asia Pacific (Singapore), Asia Pacific (Tokyo) and GovCloud (US) Regions.
- Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.
- Uses standards-based REST and SOAP interfaces designed to work with any Internet-development toolkit.
- Built to be flexible so that protocol or functional layers can easily be added. The default download protocol is HTTP
- Reliability backed with the Amazon S3 Service Level Agreement.

**Storage Pricing**

<table>
<thead>
<tr>
<th>Storage Pricing</th>
<th>Standard Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>First 1 TB / month</td>
<td>$0.140 per GB</td>
</tr>
<tr>
<td>49 TB / month</td>
<td>$0.125 per GB</td>
</tr>
<tr>
<td>500 TB / month</td>
<td>$0.095 per GB</td>
</tr>
<tr>
<td>Over 5000 TB / month</td>
<td>$0.055 per GB</td>
</tr>
</tbody>
</table>

**Request Pricing**

<table>
<thead>
<tr>
<th>Request Pricing</th>
<th>Pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT, COPY, POST, or LIST Requests</td>
<td>$0.01 per 1,000 requests</td>
</tr>
<tr>
<td>GET and all other Requests †</td>
<td>$0.01 per 1,000 requests</td>
</tr>
</tbody>
</table>

**Transferring Large Amounts Of Data**
AWS Import/Export accelerates moving large amounts of data into and out of AWS using portable storage devices for transport. AWS transfers your data directly onto and off of storage devices using Amazon’s high-speed internal network and bypassing the Internet. For significant data sets, AWS Import/Export is often faster than Internet transfer and more cost effective than upgrading your connectivity.

**Cloud Table Storage**
Cloud Tables store data as collections of entities. Entities are similar to rows. An entity has a primary key and a set of properties. A property is a name/value pair, similar to a column.

**Windows Azure Table Storage**
Azure stores information a few ways, but the two that focus on persisting structured data are SQL Azure and Windows Azure Table storage. The first is a relational database and aligns fairly closely with SQL
Server. It has tables with defined schema, keys, relationships and other constraints, and you connect to it using a connection string just as you do with SQL Server and other databases.

Windows Azure Table services provide the potential to store enormous amounts of data, while enabling efficient access and persistence. The services simplify storage, saving you from jumping through all the hoops required to work with a relational database—constraints, views, indices, relationships and stored procedures. You just deal with data. Windows Azure Tables use keys that enable efficient querying, and you can employ one—the PartitionKey—for load balancing when the table service decides it’s time to spread your table over multiple servers. A table doesn’t have a specified schema. It’s simply a structured container of rows (or entities) that doesn’t care what a row looks like. You can have a table that stores one particular type, but you can also store rows with varying structures in a single table.

**PartitionKeys and RowKeys Drive Performance and Scalability**

Many developers are used to a system of primary keys, foreign keys and constraints between the two. With Windows Azure Table storage, you have to let go of these concepts or you’ll have difficulty grasping its system of keys.

In Windows Azure Tables, the string PartitionKey and RowKey properties work together as an index for your table, so when defining them, you must consider how your data is queried. Together, the properties also provide for uniqueness, acting as a primary key for the row. Each entity in a table must have a unique PartitionKey/RowKey combination.

But you need to consider more than querying when defining a PartitionKey, because it’s also used for physically partitioning the tables, which provides for load balancing and scalability.

**Amazon SimpleDB**

Amazon SimpleDB web service is for indexing and querying data. It’s used with two other Amazon products to store, process, and query data sets in the cloud. Amazon likens the database to a spreadsheet in that it has columns and rows with attributes and items stored in each. Unlike a spreadsheet, however, each cell can have multiple values and each item can have its own set of associated attributes. Amazon then automatically indexes the data.

This service works in close conjunction with Amazon Simple Storage Service (AmazonS3) and Amazon Elastic Compute Cloud (Amazon EC2), collectively providing the ability to store, process, and query data sets in the cloud.

Amazon offers the feature because traditional relational databases require a sizable upfront expense. They are also complex to design and often require the employment of a database administrator. Amazon SimpleDB is—as the name says—simpler. It requires no schema, automatically indexes data, and provides a simple API for storage and access.

**Amazon SimpleDB Functionality**

Amazon SimpleDB provides a simple web services interface to create and store multiple data sets, query your data easily, and return the results. Your data is automatically indexed, making it easy to quickly find
the information that you need. There is no need to pre-define a schema or change a schema if new data is added later. And scale-out is as simple as creating new domains, rather than building out new servers.

**Highly available** – Amazon SimpleDB automatically creates multiple geographically distributed copies of each data item you store. This provides high availability and durability – in the unlikely event that one replica fails, Amazon SimpleDB can failover to another replica in the system.

**Secure** – Amazon SimpleDB provides an https end point to ensure secure, encrypted communication between your application or client and your domain. In addition, through integration with AWS Identity and Access Management, you can establish user or group-level control over access to specific SimpleDB domains and operations.

### Pricing

**Amazon SimpleDB Pricing**

**Machine Utilization**
- First 25 Amazon SimpleDB Machine Hours consumed per month are free
- $0.140 per Amazon SimpleDB Machine Hour consumed thereafter

**Data Transfer**

<table>
<thead>
<tr>
<th>Data Transfer IN</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>All data transfer in</td>
<td>$0.000 per GB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Transfer OUT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1GB / month</td>
<td>$0.000 per GB</td>
</tr>
<tr>
<td>Up to 10 TB / month</td>
<td>$0.120 per GB</td>
</tr>
<tr>
<td>100 TB / month</td>
<td>$0.070 per GB</td>
</tr>
<tr>
<td>350 TB / month</td>
<td>$0.050 per GB</td>
</tr>
</tbody>
</table>

**Structured Data Storage**
- First 1 GB stored per month is free*
- $0.250 per GB-month thereafter

**Windows Azure Table Storage Pricing**

**Storage**
- $0.15 per GB stored per month
- $0.01 per 10,000 storage transactions

**Data Transfers**
- North America and Europe regions $0.15 per GB out
- Asia Pacific region $0.20 per GB out
All inbound data transfers are at no charge.

**Summary of Cloud Storage Alternative**
Among three cloud storage services (Blobs, Tables, Databases) studied in this project, tables and databases are more appropriate to store sensor data. Blob storage does not provide structured storage and query capabilities required for working with sensor data.

Cloud Databases like SQL Azure and Amazon RDS provide data-processing capabilities through queries, transactions and stored procedures that are executed on the server side, and only the results are returned to the app. If you have an application that requires data processing over large data sets, then a cloud database is a good choice. If you have an app that stores and retrieves (scans/filters) large datasets but does not require data processing, then Table Storage is a better choice.

Cloud table storage services like Windows Azure Table Storage and Amazon SimpleDB make sense if you don’t need a relational store or access is limited to a single table at a time and doesn’t require joins. In this case, your data sets would be small and joins could be handled client-side.

If you have more data than the maximum amount supported by SQL Azure (which is currently 50GB for a single instance), a better option is Windows Azure Table Storage. Note that size limitation can be overcome with some data partitioning, but that could drive up the SQL Azure costs. The same space in Windows Azure Table Storage would probably be less expensive and has partitioning built-in by a declared partition key.

Storing and processing sensor data will require complex queries and relationships. In addition, the amount of storage space required for sensor data should be within the limits of cloud databases. For these reasons, a cloud database service would be the best choice for sensor data.
Sensor Cloud Gateway

Sensor Cloud Gateway is a tool that gathers real time sensor data and forwards it to cloud databases. In the following sections the design and implementation details will be discussed.

Design and Architecture

Sensor Cloud Gateway utilizes the Sensor Server and Connector sensor frameworks. Figure 1 shows the general design of Sensor Cloud Gateway architecture.

Figure 1 Sensor Cloud Gateway Architecture

As Figure 1 shows, real time sensor data is first collected by the sensor server. Sensor Cloud Gateway uses the Connector library and APIs to access data from the Sensor Server. It also uses JDBC SQL driver to connect to the SQL Azure database. Later, the sensor data in the database will be used to analyze and visualize historical and real-time data.

Database Design

The sensor database is implemented in Microsoft SQL Azure. Database structures are configured using the Azure portal web page.

Figure 2 shows the two database tables used for sensor data.
Figure 2: Sensor Database Tables

Figure 3 shows data fields of the Sensor Data Table. These are the record fields generated by the sensors and collected by the Sensor Server. These fields are representative for a temperature sensor. However, the field types can be easily modified to store all kinds of sensor data.

Figure 3: Sensor Data Table

Figure 4 shows the Sensor Information Table which can be used to store metadata about the sensor devices themselves, such as location information.
Figure 4: Sensor Information Table

**Code Samples**

In this section, highlights from Sensor Cloud Gateway code will be presented.

The following code excerpt shows how a connection to the SQL Azure can be made in Java using the JDBC SQL Driver. The connection string contains the database name, username, and password.

```java
// Load the SQLServerDriver class
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

// build the connection string
String connectionUrl = "jdbc:sqlserver://databasename.database.windows.net:" +
                      "database-SensorDatabase;" +
                      "user=username@databasename;" +
                      "password=databasepassword";

// get a connection to the database
con = DriverManager.getConnection(connectionUrl);
System.out.println("Connected.");
```

The next code sample shows how a connection to the Sensor Server can be made using the Connector class.

```java
// Connect to the sensor server using the connections in file.txt
Connector system = new Connector("files.txt", true, 0);

// Open data channel from "master"
FileInputStream file = new FileInputStream("master");
String temp = null;
String[] data = null;

// Read each line from this file
Scanner input = new Scanner(file);
```
The Connector class gets the address of the Sensor Server from the Files.txt file, and connects to the “master” sensor channel.

Files.txt:

master ftp://username:password@hercules.uwb.edu:55555

The next code excerpt shows how SQL Prepared Statement class can be used to construct a SQL INSERT query statement to insert a sensor record to the SQL Azure database using the connection obtained above.

```java
String SQL = "INSERT INTO SensorDataTable (MacAddress, TempF," +
             "RSSI, Battery, Period, Time) VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement pstmt = con.prepareStatement(SQL);

// set the record values
pstmt.setString(1, data[2]); // MacAddress
pstmt.setFloat(2, Float.parseFloat( data[3])); // TempF
pstmt.setFloat(3, Float.parseFloat( data[4])); // RSSI
pstmt.setInt(4, Integer.parseInt(data[5])); // Battery
pstmt.setInt(5, Integer.parseInt(data[1])); // Period
pstmt.setTimestamp(6, Timestamp.valueOf(data[0])); // Time

// execute the insert query
int count = pstmt.executeUpdate();
System.out.println("ROWS INSERTED: " + count);
pstmt.close();
```

The sensor record fields are set one by one and then the query statement is executed and the number of rows inserted into the database is printed.

**Output Samples**

Before executing the Sensor Cloud Gateway, first you need to start the SensorServer using the following Java commands.

```
java -cp Connector.jar:. SensorServer
```

Then, you can start the SensorGateway with the following command.

```
java -cp Connector.jar:sqljdbc4.jar:. SensorGateway
```

While both SensorServer and SensorGateway are executing, sensor data will be stored in SQL Azure tables.
Figure 5: Sample Data from the SensorDatabase

Figure 5 shows a sample snapshot of the sensor data in the SQL Azure database.

**Future Work**

In the next step of this project, the software implementation will be expanded to include sensor data processing and visualization running as cloud services. These services will be based on the sensor data layout developed in this research. It will have visualization capabilities to analyze the data at multiple resolutions and time frames.

Figure 6 shows an example of data visualization over the Internet. The graph shows the comparison of two market stocks. This is an inspiration for visualizing sensor data through a web-service. Similar to this example, sensor data visualization will have compare option for comparing months, years, etc., and multiple sensors.

The visualization tool will have options and settings for several data mining techniques. These will include statistical measurements and trend analysis.

The web-service will be implemented as an Azure web role and it will be running in the Azure Cloud Service. Since it is a cloud service, it will be accessible everywhere.
Figure 6: Graphing service example

http://investing.money.msn.com/investments/charts?symbol

Conclusions

Storing and processing sensor data requires complex queries and relationships. In addition, the amount of storage space required for sensor data should be within the limits of cloud databases. For these reasons, a cloud database service is the best choice for sensor data.

Sensor database implemented in SQL Azure was simple to develop and execute. The data is easily accessible through JDBC drivers. This sensor data framework will be the basis of the future sensor data visualization cloud service.

References


Brunetti, Roberto (2011) WindowsAzureStepByStep - Chapter 9


http://www.microsoft.com/windowsazure/features/storage/

http://www.microsoft.com/windowsazure/pricing/

http://aws.amazon.com/s3/

http://aws.amazon.com/simpledb/pricing/