CSE 544
Principles of Database Management Systems

Magdalena Balazinska

Winter 2009

Lecture 13 – Databases as a Service

Final Project Grading

- Final project report (worth 25%)
  - Due Friday, March 20th at 5pm (you still have 4 weeks!)
  - Graded in two parts: project content and report content
  - **Part 1: Project (15%)**
    - Are the project accomplishments reasonable given the timeframe?
    - Has the problem been deeply studied?
    - Is the approach thought through? Does it leverage prior work?
    - Is the approach implemented?
    - Is the evaluation systematic, thorough? Are results well analyzed?
  - **Part 2: Report: see website for required report content (10%)**
    - Is the problem well motivated? Is the problem statement clear?
    - Is the approach clear? Examples and figures!
    - Is the evaluation reasonably thorough? Is the analysis clear?

Final Project Grading (continued)

- Final project presentations (10%)
  - 8 minutes / team + 2 minutes for questions
  - All team members must talk
  - Grading guidelines are posted on project website
  - We will only grade the project presentation not the project content

- Two days for talks: Monday and Thursday of finals week
  - You can pick whichever day you prefer
  - We will split talks into two or more sessions
  - You must attend ALL the talks in your session
  - You don't need to come to the other sessions

Announcement

- Next lecture: guest speaker, Dr. David Lomet from MSR
- No reading and no paper review
- Content not on the final
- Lecture in CSE 403

References

- Amazon SimpleDB Website
  - Part of the Amazon Web services
- Google App Engine Datastore Website
  - Part of the Google App Engine
- Microsoft SQL Data Services Website
  - Part of the Azure platform

Motivation

- Running a DBMS is challenging
  - Need to hire a skilled database administrator (DBA)
  - Need to provision machines (hardware, software, configuration)
  - Problems:
    - If business picks up, may need to scale quickly
    - Workload varies over time
- Solution: Use a DBMS service
  - All machines are hosted in service provider's data centers
  - Data resides in those data centers
  - Pay-per-use policy
  - Elastic scalability
  - Zero administration

Basic Features

- Data storage and query capabilities
- High availability guarantees
- Operations and administration tasks handled by provider
- **Elastic scalability**: Clients pay exactly for the resources they consume; consumption can grow/shrink dynamically
  - No capital expenditures
  - Fast provisioning

Outline

- Overview of all three systems
  - Amazon Web Services and SimpleDB
  - Google App Engine and Google App Engine Datastore
  - Microsoft Azure platform and SQL Data Services
- Common technical features and differences
- Discussion
  - Technical challenges behind databases as a service
  - Broader impacts of databases as a service

Amazon Web Services

- Since 2006
- “Infrastructure web services platform in the cloud”

- Amazon Elastic Compute Cloud (Amazon EC2™)
- Amazon Simple Storage Service (Amazon S3™)
- Amazon SimpleDB™
- Amazon CloudFront™
- Amazon Simple Queue Service (Amazon SQS™)

Amazon EC2

- Amazon Elastic Compute Cloud (Amazon EC2™)

- Rent compute power on demand (“server instances”)
  - Select required power: small, large, or extra large instance
  - Share resources with other users
  - Variety of operating systems

- Includes: Amazon Elastic Block Store
  - Off-instance storage that persists independent from life of instance
  - Highly available and highly reliable

Amazon S3

- Amazon Simple Storage Service (Amazon S3™)
  - “Storage for the Internet”
  - “Web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.”

- Some key features
  - Write, read, and delete uniquely identified objects containing from 1 byte to 5 gigabytes of data each
  - Objects are stored in buckets, located in US or Europe
  - A bucket can be accessed from anywhere
  - Authentication
  - Reliability

Amazon SimpleDB

- "Web service providing the core database functions of data indexing and querying"

- **Partitioning**
  - Data partitioned into domains: queries run within domain

- **Schema**
  - No fixed schema
  - Objects are defined with attribute-value pairs

Amazon SimpleDB (2/3)

- **Indexing**
  - Automatically indexes all attributes
- **Support for writing**
  - PUT and DELETE items in a domain
- **Support for querying**
  - GET by key
  - Selection + sort
  - A simple form of aggregation: count
  - Query execution time is limited to 5 second (but can continue)

```
select output_list
from domain_name
[where expression]
[sort_instructions]
[limit limit]
```

Amazon SimpleDB (3/3)

- **Availability and consistency**
  - “Fully indexed data is stored redundantly across multiple servers and data centers”
  - “Takes time for the update to propagate to all storage locations. The data will eventually be consistent, but an immediate read might not show the change”

- **Integration with other services**
  - “Developers can run their applications in Amazon EC2 and store their data objects in Amazon S3.”
  - “Amazon SimpleDB can then be used to query the object metadata from within the application in Amazon EC2 and return pointers to the objects stored in Amazon S3.”

Google App Engine

- “Run your web applications on Google's infrastructure”

- Key features
  - Dynamic web serving, with full support for common web technologies: apps serve web requests
  - Persistent storage with queries, sorting and transactions
  - Automatic scaling and load balancing
  - APIs for authenticating users and sending email
  - A fully featured local development environment that simulates Google App Engine on your computer
- Limitation: applications must be written in Python

Google App Engine Datastore (1/3)

- "Distributed data storage service that features a query engine and transactions"

- **Partitioning**
  - Data partitioned into "entity groups"
  - Entities of the same group are stored together for efficient execution of transactions
- **Schema**
  - Each entity has a key and properties that can be either
    - Named values of one of several supported data types (includes list)
    - References to other entities
  - Flexible schema: different entities can have different properties

Google App Engine Datastore (2/3)

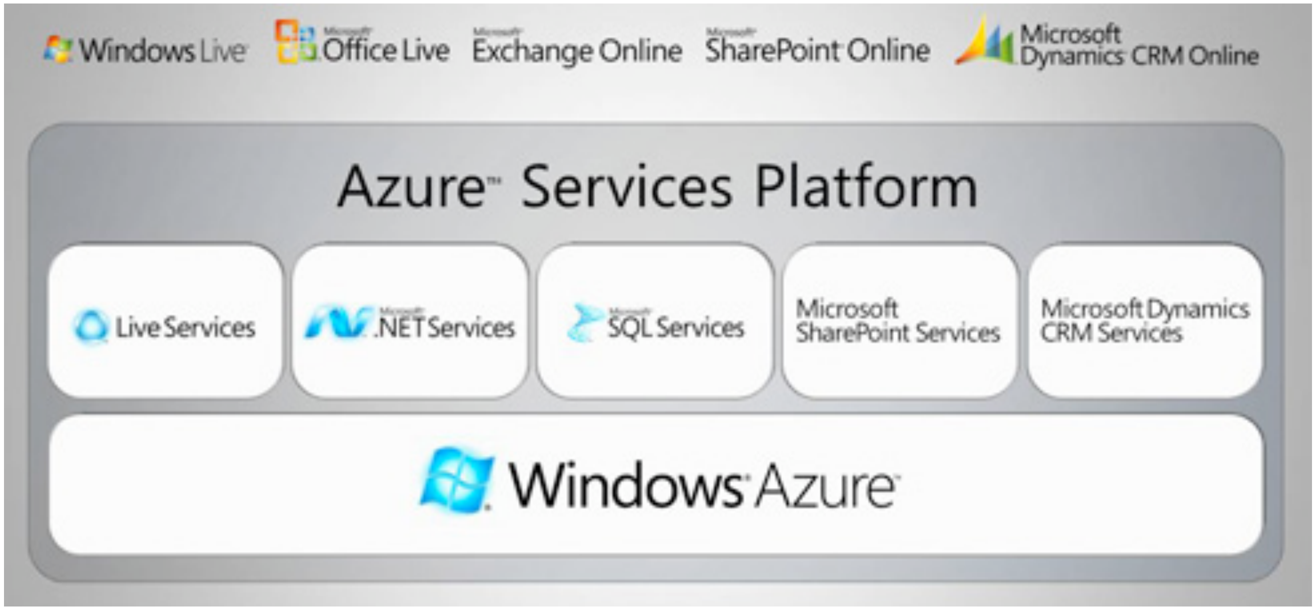
- **Indexing**
  - Applications define indexes: must have one index per query type
- **Support for writing**
  - PUT and DELETE entities
- **Support for querying**
  - Fetch an entity using its key
  - Execute a query: selection + sort
  - Language bindings: either invoke methods or write GQL
  - Lazy query evaluation: query executes when user accesses results

Google App Engine Datastore (3/3)

- **Availability and consistency**
  - Every datastore write operation (put/delete) is atomic
  - Support transactions
    - All operations must operate on entities in the same entity group
    - Cannot perform queries; can only get entities by their keys
  - Optimistic concurrency control

Microsoft Azure Platform

- “Internet-scale cloud computing and services platform”
- “Provides an operating system and a set of developer services that can be used individually or together”


Windows Live
Microsoft Office Live
Microsoft Exchange Online
Microsoft SharePoint Online
Microsoft Dynamics CRM Online
Azure™ Services Platform
Live Services
Microsoft .NET Services
Microsoft SQL Services
Microsoft SharePoint Services
Microsoft Dynamics CRM Services
Windows Azure™

SQL Data Services (1/3)

- "Highly scalable, on-demand data storage and query processing utility services"

- **Partitioning**
  - Three-Level Containment Model (the "ACE" Concept)
  - ACE: Authorities, Containers and Entities
    - Container is largest domain for search and update.
- **Schema**
  - Flexible schema again
  - Entities have keys
  - Each entity inside a container can store any number of user-defined properties and corresponding values

SQL Data Services (2/3)

- **Indexing**
  - Automatically indexes the data
- **Support for writing**
  - Entity = smallest object that can be updated
  - Can retrieve an entire entity; add, update, delete properties; and then replace the original entity with the updated one.
- **Support for querying**
  - Selection with sort
  - Support limited form of join
  - Cannot return more than 500 entities at a time

```
from e in entities
[where condition]
[orderby property 1 [,property 2, …]]
select e
```

SQL Data Services (3/3)

- **Availability and consistency**
  - Availability: Multiple geo-replicated copies of the data
  - Consistency: Transactional consistency across copies

Summary of all three Systems

- **Partitioning**: in all systems data is partitioned
- **Schema**: flexible schema
  - Different entities can have different attributes
- **Indexing**: all systems answer queries using indexes
- **Write** operations: put and delete
  - Some systems support transactions on objects within a group
- **Query** interface: primarily selection + sort
- **Availability and consistency**
  - All systems strive to achieve high availability
  - Some systems have strong consistency others weak

Outline

- Overview of all three systems
  - Amazon Web Services and SimpleDB
  - Google App Engine and Google App Engine Datastore
  - Microsoft Azure platform and SQL Data Services
- Common technical features and differences
- Discussion
  - Technical challenges behind databases as a service
  - Broader impacts of databases as a service

Challenges of DBMS as a Service

- Scalability requirements
  - Large data volumes and large numbers of clients
  - Variable and heavy workloads
- High performance requirements: interactive web services
- Consistency and high availability guarantees
- Service Level Agreements
- Security

Broader Impacts

- Cost-effective solution for building web services

- Content providers focus only on their application logic
  - Service providers take care of administration
  - Service providers take care of operations

- Security/privacy concerns: all data stored in data centers