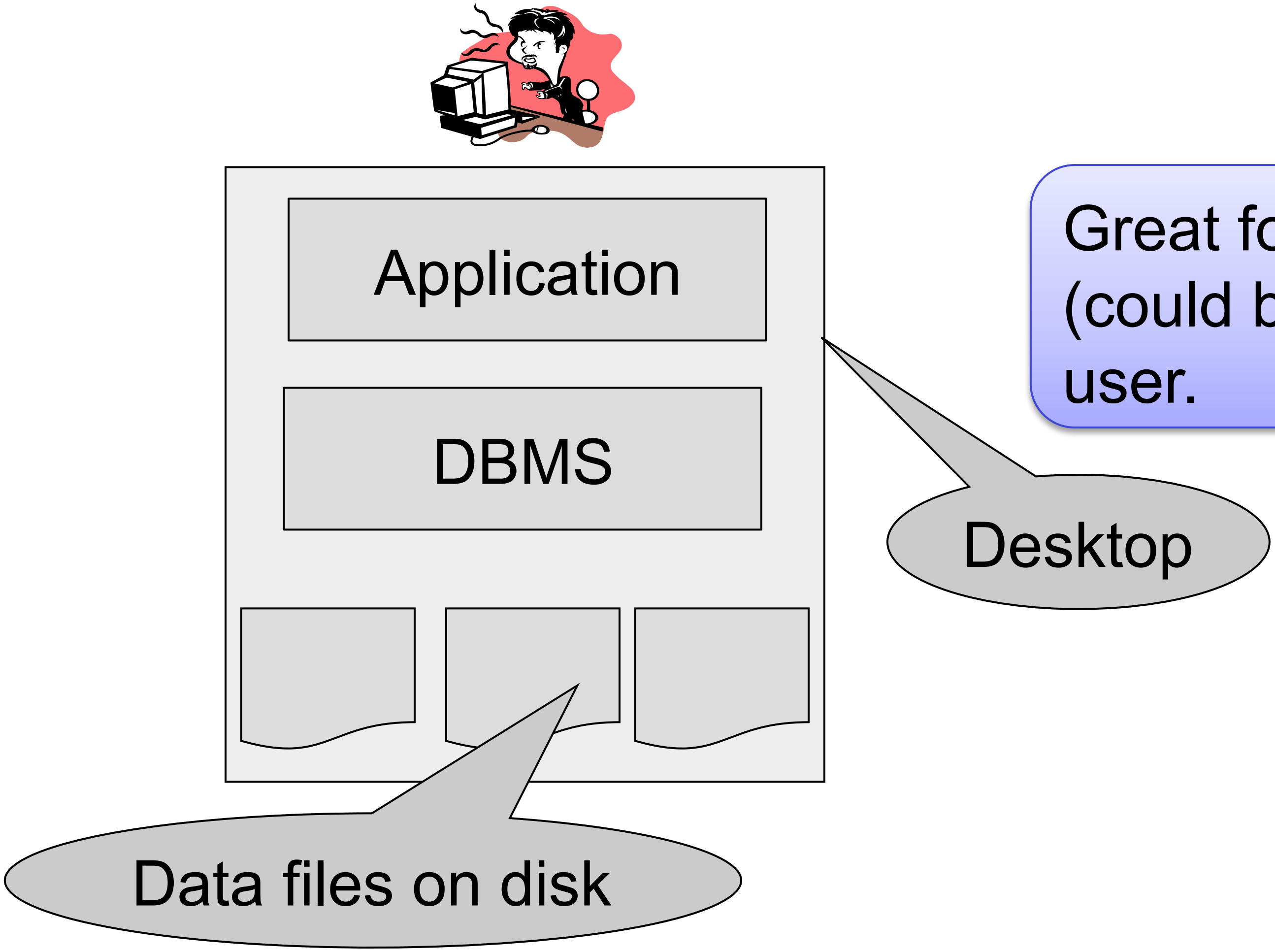
Introduction to Data Management CSE 344

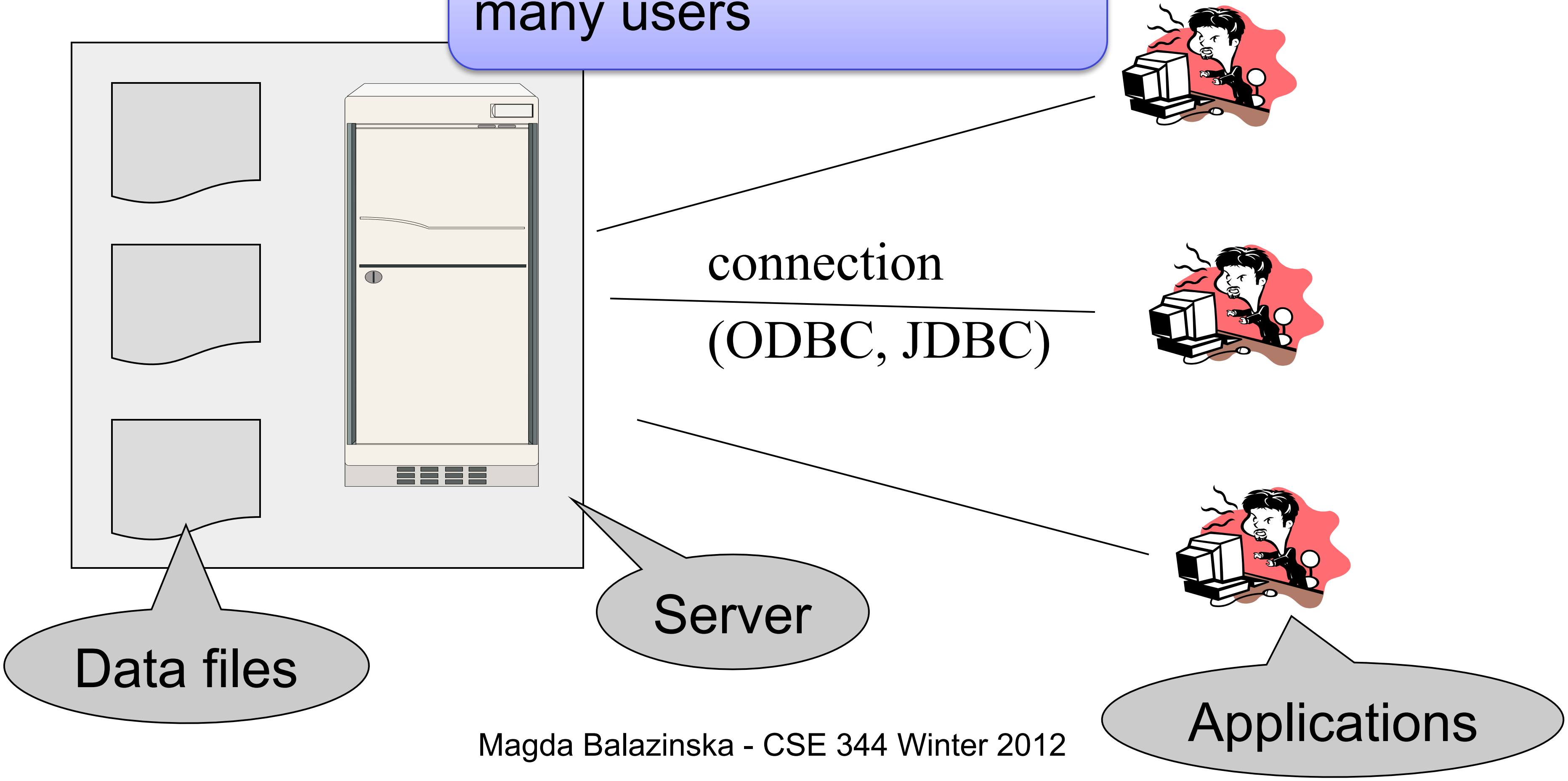
Lecture 25: DBMS-as-a-service and NoSQL

DBMS Deployment: Local



Great for one application (could be more) and one user.

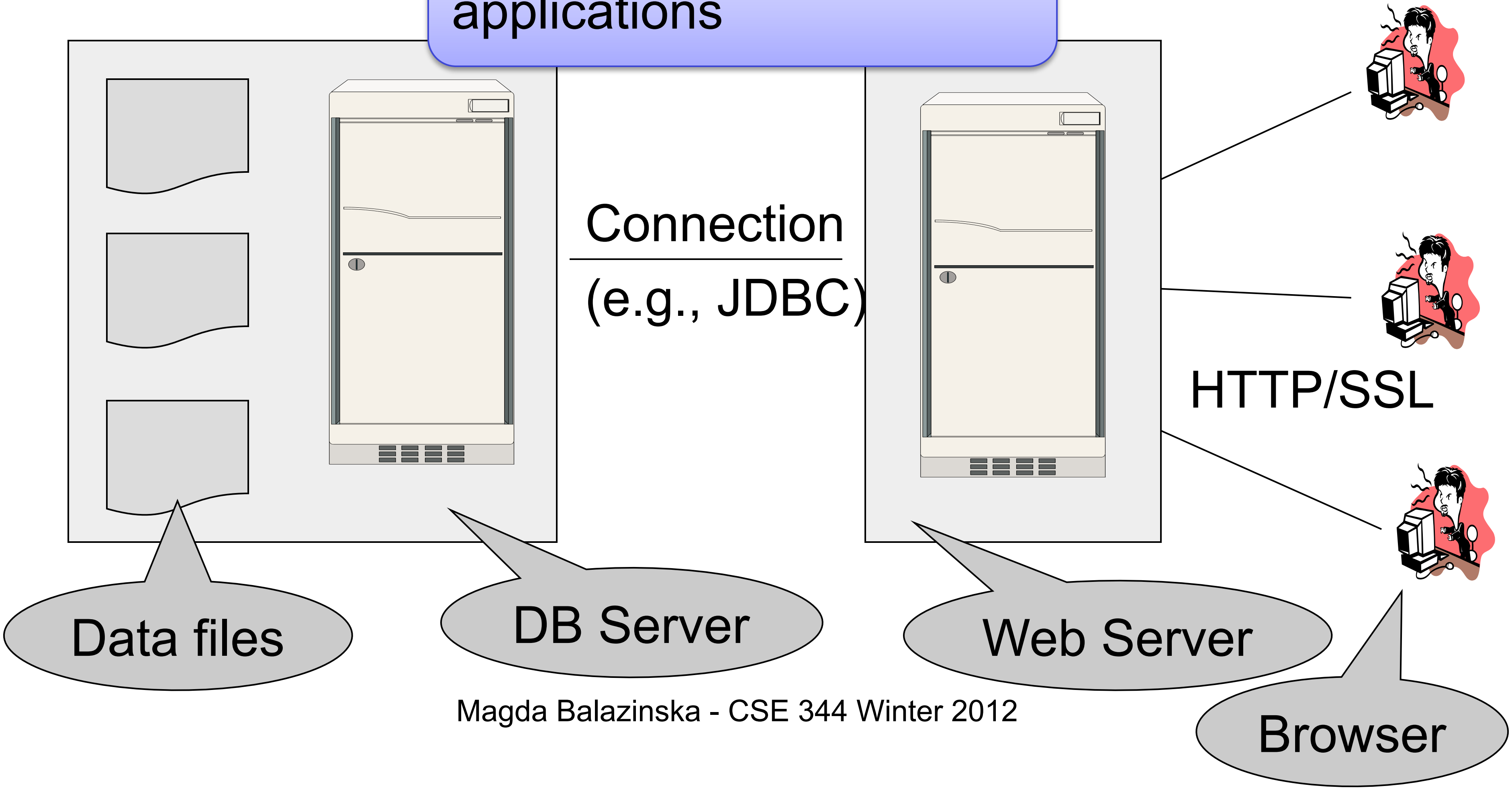
DBMS Deployment: Client/Server



Great for many apps and many users

DBMS Deployment: 3 Tiers

Great for web-based applications



Challenges

Deploying and operating a DBMS is hard!

- Acquire and setup hardware
- Install Web server and DBMS server
- Configure and tune them
- Configure for failures and variable load
  - Need to manage failures and load variations!
  - Difficult to react to changing workloads!
    - Acquire/configure new machines

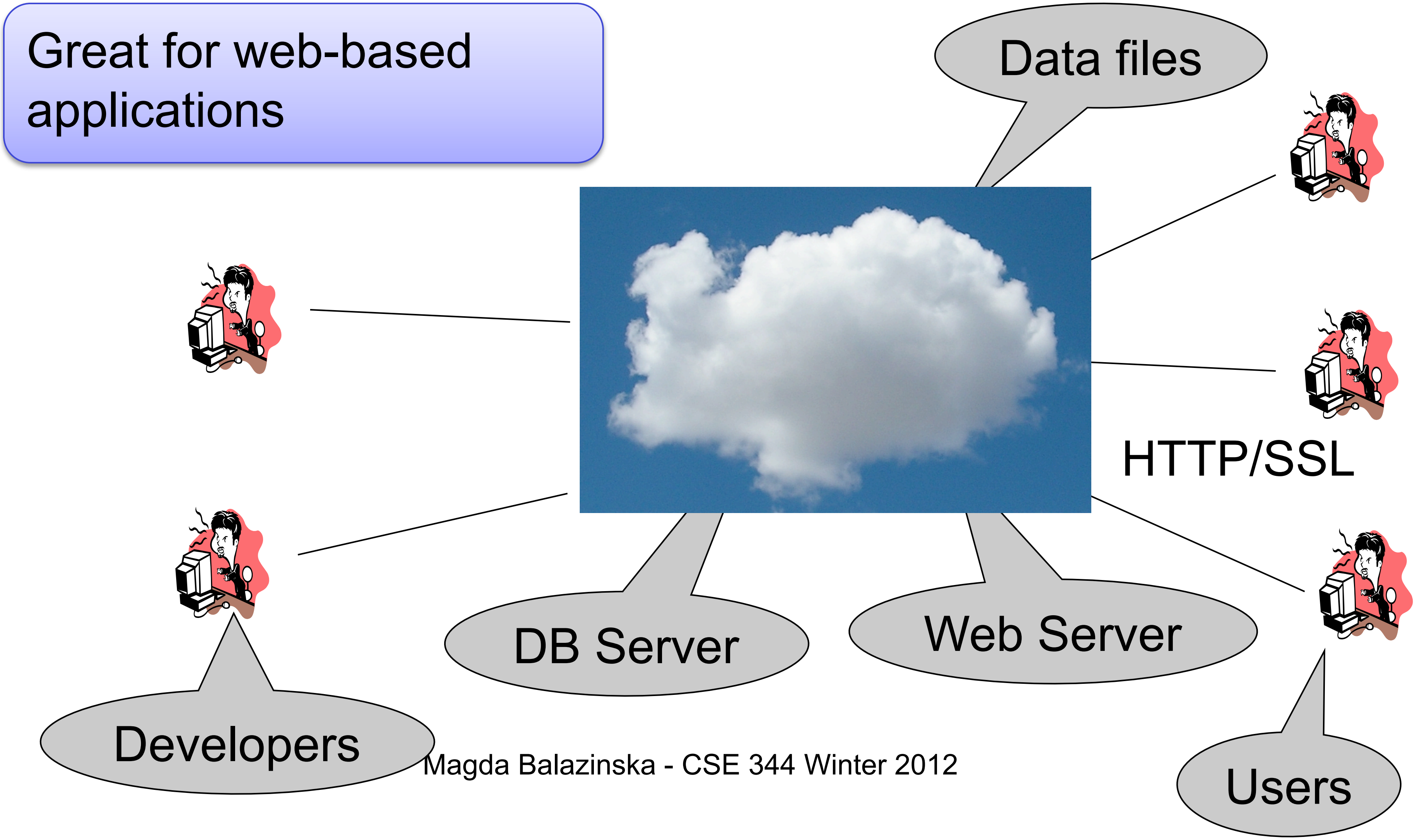
Cloud Computing

- A definition
  - "Style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet"

- Basic idea
  - Developer focuses on application logic
  - Infrastructure, software, and data hosted by someone else in their "cloud"
  - Hence all operations tasks handled by cloud service provider

DBMS Deployment: Cloud

Great for web-based applications



Cloud Computing History

- “Computation may someday be organized as a public utility” (John McCarthy – 1960)
- Late 1990’s: Infrastructure as a Service (i.e., rent machines)
- Late 1990s’: Software as a service (e.g., Hotmail, Salesforce)
- Early 2000s: Web services
- 2006: Amazon Web Services
- And now it’s a craze!

Levels of Service

- Infrastructure as a Service (IaaS)
  - Example Amazon EC2
- Platform as a Service (PaaS)
  - Example Microsoft Azure, Google App Engine
- Software as a Service (SaaS)
  - Example Google Docs

Basic Features for Data Management as a Service

- Some sort of data storage and query capabilities
- Operations and admin handled by cloud provider
  - Includes high availability, upgrades, etc.
- **Elastic scalability**
  - Can grow and shrink resources dynamically
  - No capital expenditures and fast provisioning
- **Pay-per-use**
  - Clients pay exactly for the resources they consume

Types of Data Management as a Service

Three different types exist at the moment

- Relational data management systems (e.g., SQL Azure)
- Simplified data mgmt systems (e.g., Amazon SimpleDB)
  - Also called “NoSQL” systems. We will see why in a few slides
  - Highly scalable
- Analysis services such as Amazon Elastic MapReduce

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 Relational Database Service (Amazon RDS)
- Amazon SimpleDB™
- Amazon Elastic MapReduce™
- And more…
- And growing…

Amazon EC2

- Amazon Elastic Compute Cloud (Amazon EC2™)

- Rent compute power on demand (“server instances”)
  - Select required capacity: small, large, or extra large instance
  - Share resources with other users (multitenant): **Virtual machines**
  - 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 TB of data each
  - Objects are stored in buckets. User chooses geographic area
  - A bucket can be accessed from anywhere
  - Authentication
  - Reliability

Amazon RDS

- Amazon Relational DB Service (Amazon RDS^{TM})
  - Web service that facilitates set up, operations, and scaling of a relational database in the cloud
  - Full capabilities of a familiar MySQL or Oracle DBMS
- Some key features
  - Automated patches of DBMS
  - Automated backups for user-defined retention period
  - Elastic scalability but can only **scale-up**
    - Make your instance more powerful (CPU and memory)
    - Attach more storage to your instance
  - Can scale-out only by adding ***read* replicas**

NoSQL Motivation

- Scaling a relational DBMS is hard
- We saw how to scale queries with parallel DBMSs
- Much more difficult to scale ***transactions***
  - Need to partition the database across multiple machines
  - If a transaction touches one machine, life is good
  - If a transaction touches multiple machines, ACID becomes extremely expensive! Need what is called two-phase commit
- Replication
  - Replication can also help to increase throughput
  - Create multiple copies of each database partition
  - Spread queries across these replicas
  - Easy for reads but writes, once again, become expensive!

NoSQL Systems

- Goal: elastic and highly scalable data management
  - Basic data storage, basic querying, and atomic updates
  - More flexible than a relational DBMS: no fixed schema!
  - Highly scalable!
  - But to scale-out, give up on complex queries
    - No joins (or limited joins)
  - Gives up on ACID: instead eventually consistent
  - No transactions! Or limited transactions
- Caveat: Hard to build apps without ACID guarantees
- Today: Many NoSQL systems provide choice between strong consistency and eventual consistency

Amazon SimpleDB

- An example of a NoSQL data management system

- **Partitioning**
  - Data partitioned into domains: queries run within domain
  - Domains seem to be unit of replication. Limit 10GB
  - Can use domains to manually create parallelism

- **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 is limited to 5s and 1MB output (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”
  - Today, can choose between consistent or eventually consistent read
- **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.”

Amazon Elastic MapReduce

- "Web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data"
- Hosted Hadoop framework on top of EC2 and S3
- Support for Hive and Pig
- User specifies
  - Data location in S3
  - Query
  - Number of machines
- System sets-up the cluster, runs query, and shuts down

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 of Clouds

- 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