# Introduction

INF 551

Wensheng Wu

# Logistics

- Instructor email: wenshenw@usc.edu
- Class meeting times:
  - 3:30-6:20pm, Tue, SOS B44
- Office hours:
  - 4-5pm, MW, GER 204

# Logistics

- TAs
  - Yucheng Guo <yuchengg@usc.edu>
- Office hours
  - Thursday 4-5pm
  - SAL computing lab

# Blackboard

- Discussion forums
  - Posting general and homework questions
  - Do not post solutions
  - Please actively participate in helping others!
  - Do not abuse forum (an academic misconduct!)

- Check frequently for updates

# Prerequisites

- Programming skills:
  - Python (e.g., for Spark), Java (e.g., for Hadoop)
- Unix-like environment
  - E.g., Ubuntu, Virtual machine, Amazon EC2 (we will use this)
- Basic knowledge of algorithms and data structures
  - Sorting, hashing, etc. (CS 570)
- Basic probability and statistics

# Textbooks

- Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*, 2015 (selected chapters only). Available free at: http://pages.cs.wisc.edu/~remzi/OSTEP/

- Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. Database Systems: The Complete Book (Second Edition), Prentice Hall, 2009. (selected chapters only)
  - http://infolab.stanford.edu/~ullman/dscb.html

- Jiawei Han, Micheline Kamber, and Jian Pei. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2011, 3rd Edition (selected chapters only).

# Grading structure

- Homework assignments: 30%
  - About 5, may involve programming
- Weekly quizzes:  30%
  - Typically, based on last week's materials
- Final exam: 30%
  - Closed-book and notes
- Lab session (and possible student presentation): 10%

# Lab sessions

- Flipped:
  - Task and details posted before class
  - Bring questions to class

- Typically utilize last 15-30 mins of class

# Grading scale

- 94 – 100 = A  74 - 76 = C
- 90 – 93 = A-  70 - 73 = C-
- 87 – 89 = B+  67 - 69 = D+
- 84 – 86 = B  64 - 66 = D
- 80 – 83 = B-  60 - 63 = D-
- 77 – 79 = C+  Below 60 is an F

# Borderline

- Active participation in in-class discussion & forum helps
  - Sign-up sheets may be handed out in class

- Grades are not negotiable!

# Late Policy

- Homework will be submitted to Blackboard
  - 10% for every 24 hours late
  - No credit after 3 days

- Make up for quizzes are permitted only when
  - You have a medical/family emergency
  - Let me know in advance
  - Proof (e.g., medical note) is required

# Late Policy

- Quiz will be given in the beginning of class
  - You are responsible for missing quiz due to tardiness
  - No make up will be given for tardiness!
- You are responsible for scheduling conflicts
  - With job interviews, job fairs, etc.

# Grading Corrections

- All homework & quiz grades are final one week after grades are posted

- Wasting instructor and staff times on non-legitimate regrading requests may result in losing more points

# Academic Integrity

- **Cheating will not be tolerated**

- **All parties involved will receive a grade of F for the course and be reported to SJACS** WITHOUT EXCEPTION
  - USC Student Judicial Affairs and Community Standards

# Now, movie time ☺

- Explain big data:
  - https://www.youtube.com/watch?v=7D1CQ_LOizA

- Questions:
  - Where does big data come from?
  - What characteristics doe it have?
  - What big data technologies were mentioned?

# Internet Traffic in 2012

- 4.8 zettabyte = 4.8 billion terabytes

- Zettabyte (1000 exabytes)
- Exabyte
- Petabyte
- Terabyte
- Gigabyte
- Megabyte
- Kilobyte

# Major topics

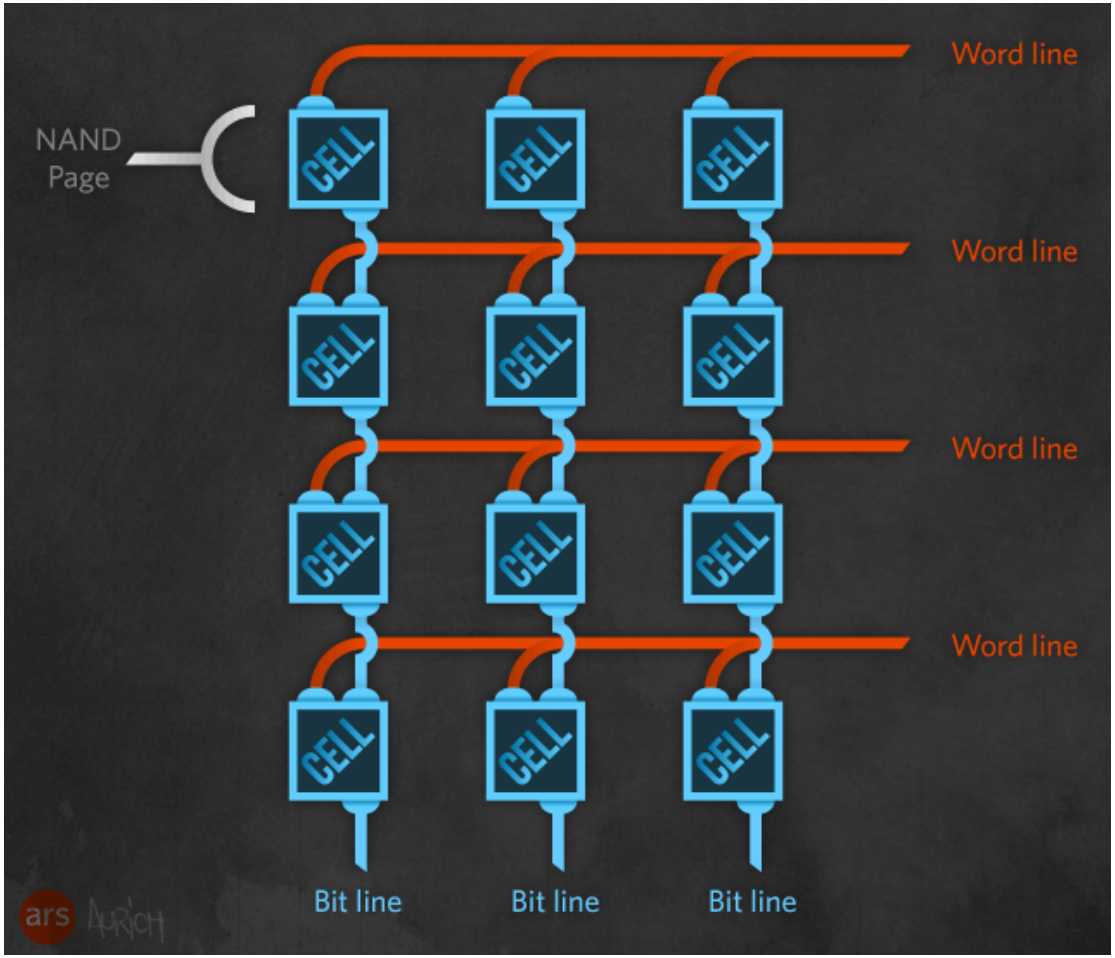
- Storage systems 
- File systems & file formats
- Database management systems
- Big data solution stack
- Data warehousing

# Storage Systems

- Hard disk
- SSD (Solid state drive)


# Internal of hard disk

# NAND flash


Word line
NAND
Page
CELL
Word line
Word line
Word line
Bit line
Bit line
Bit line
ars

# Major topics

- Storage systems
- File systems & file formats 
- Database management systems
- Big data solution stack
- Data warehousing

# File Systems

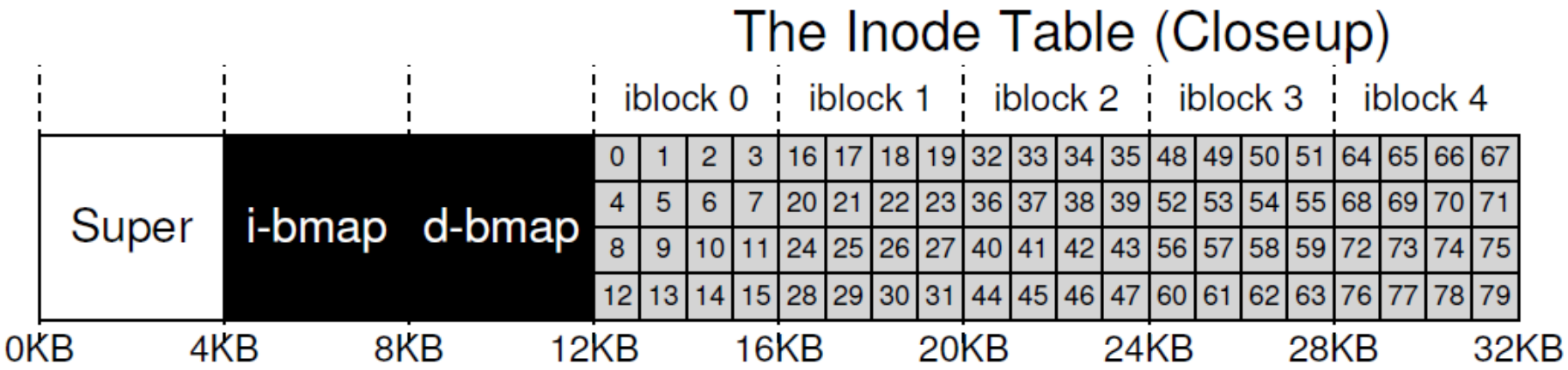
- Standalone
  - Single machine
- Network
  - Client-server
- Distributed (e.g., Hadoop)
  - A number of data servers

# Standalone file systems

- Data structures
  - Data blocks
  - Metadata blocks (Inodes)
  - Bitmap blocks (for space allocation)
- Access paths
  - Read
  - write

# Inode (index node)

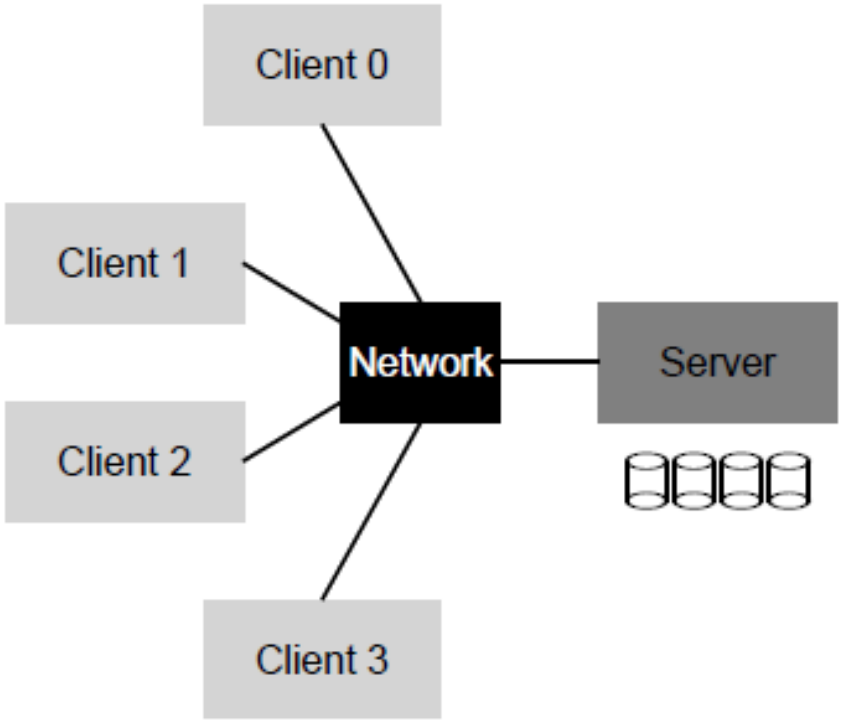
- Each is identified by a number
  - Low-level number of file name: inumber
- Can figure out location of inode from inumber



# Network file system

- Client-server architecture
  - Sun network file system

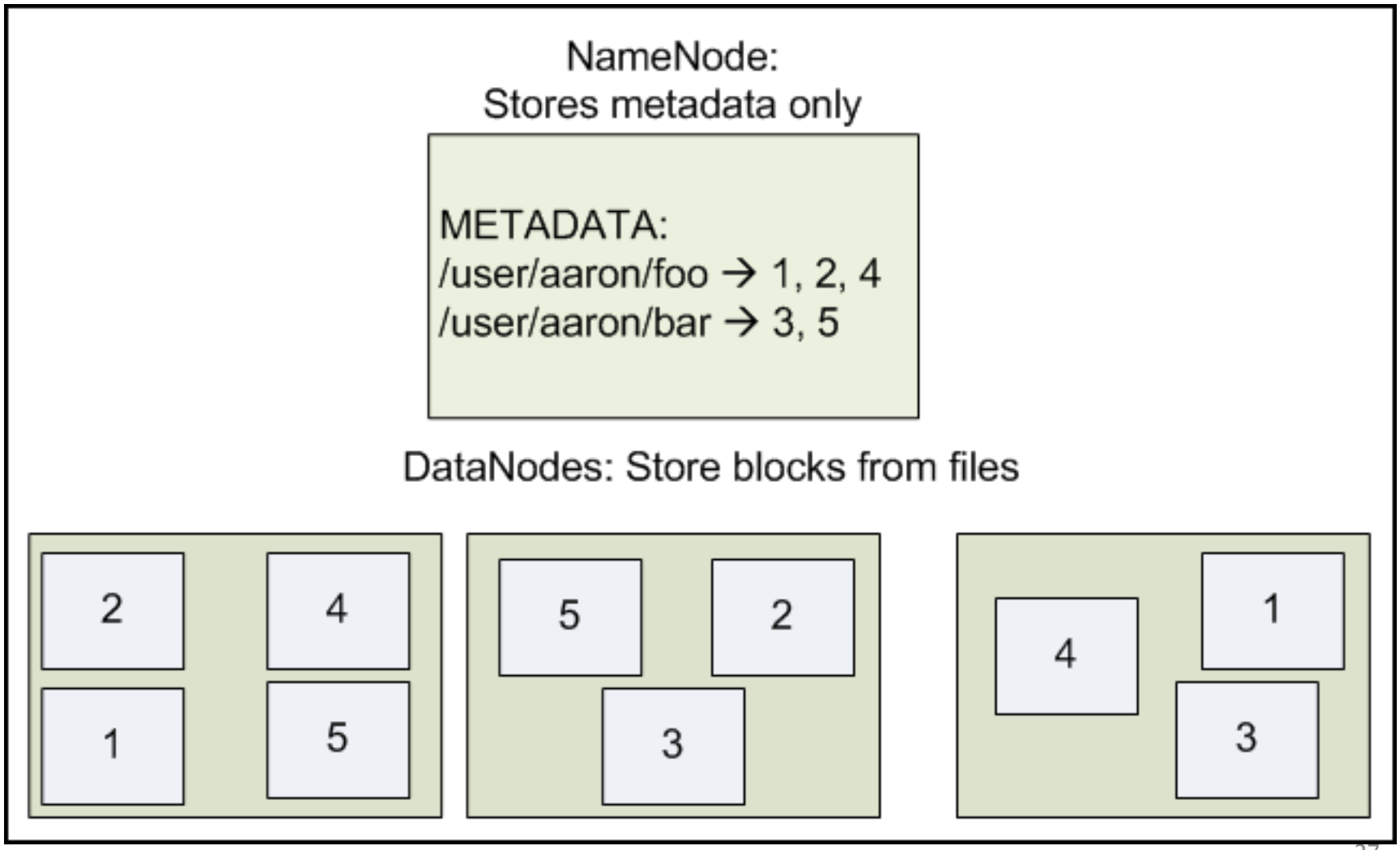
- Key concept:
  - stateless file handle



# Distributed file systems

- Hadoop HDFS (GFS)
  - Data are distributed among data nodes
- Replication
  - Automatic creation of replica (typically 3 copies/data)
- Fault-tolerant
  - Automatic recovery from node failure

# HDFS architecture



# Major topics

- Storage systems
- File systems & file formats 
- Database management systems
- Data warehousing
- Big data solution stack

# File Formats

- JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

# HTML

```
<h1> Bibliography </h1>
<p> <i> Foundations of Databases </i>
      Abiteboul, Hull, Vianu
      <br> Addison Wesley, 1995
<p> <i> Data on the Web </i>
      Abiteoul, Buneman, Suciu
      <br> Morgan Kaufmann, 1999
```

# XML

```
<bibliography>
      <book>   <title> Foundations… </title>
                  <author> Abiteboul </author>
                  <author> Hull </author>
                  <author> Vianu </author>
                  <publisher> Addison Wesley </publisher>
                  <year> 1995 </year>
      </book>
      …
</bibliography>
```

XML describes the content

# Major topics

- Storage systems
- File systems & file formats
- Database management systems 
- Data warehousing
- Big data solution stack

# Relational DBMS

- Data models
  - ER
  - Relational
- Schema
  - Normal forms: BCNF

# RDBMS

- Query languages
  - Relational algebra
  - SQL, constraints, views
- Data organization
  - Records and blocks
  - Index structure: B+-tree

# RDBMS

- Query execution algorithms
  - External sorting
  - One-pass
  - Nested-loop
  - Multiple-pass algorithms

# RDBMS

- Rigid schema
- Strong consistency is the key design goal
  - Never read old data
  - Suitable for mission-critical applications, e.g., banking
- But may suffer from low availability

# RDBMS

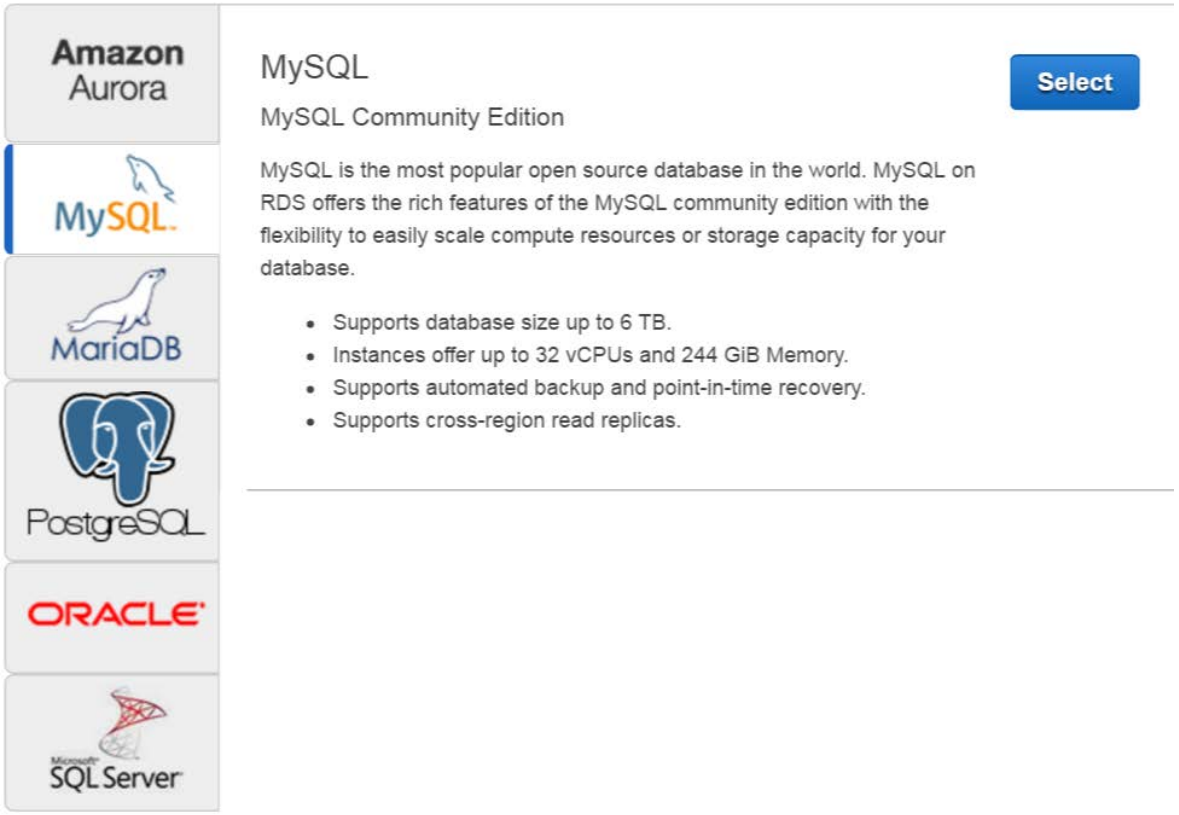
- Hard to scale out
  - Horizontal partitioning/sharding possible
  - But would need distributed storage & computing support like Hadoop & MapReduce

# RDBMS Examples

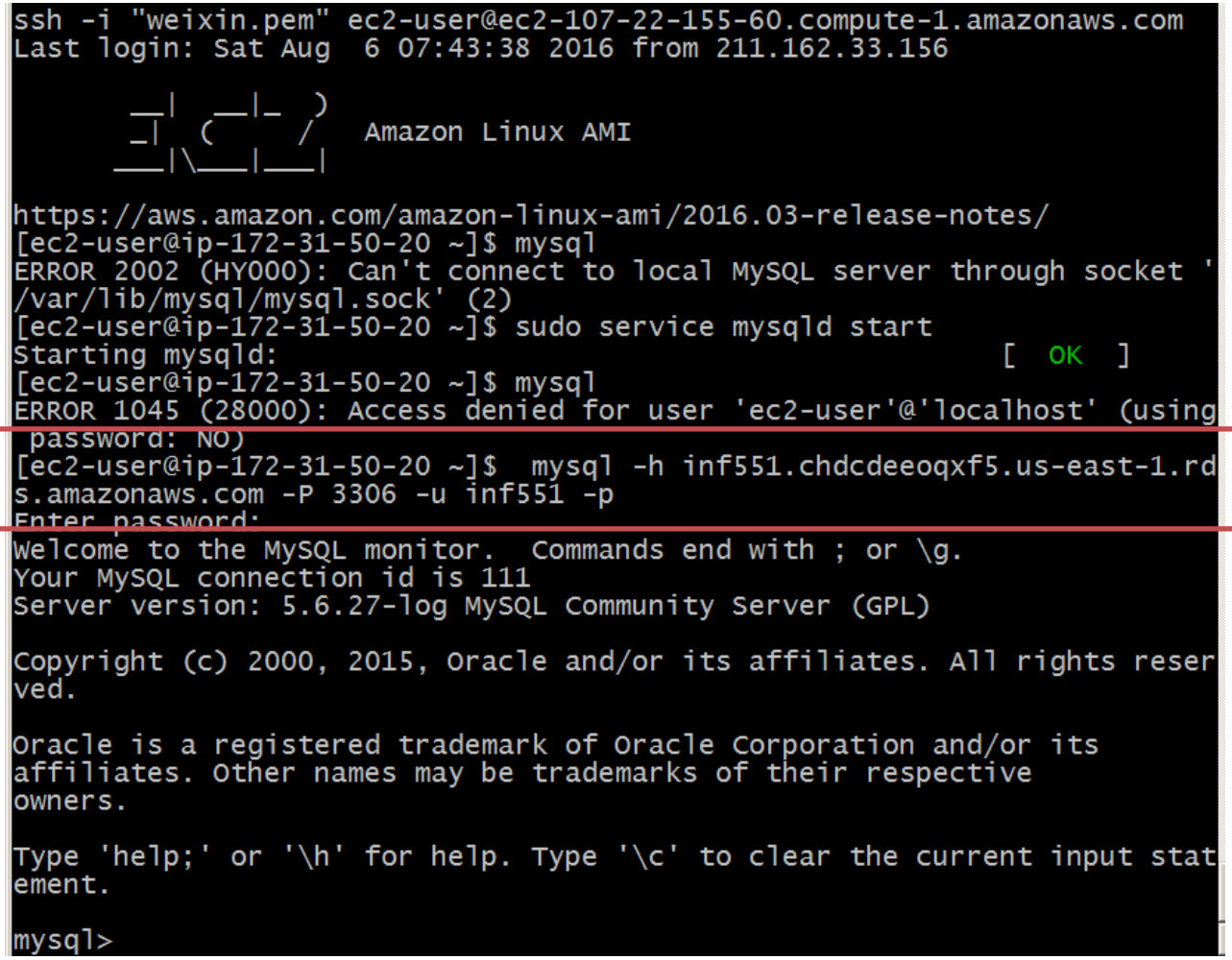
- MySQL (can be installed in Amazon AWS EC2)
- Amazon RDS (Relational database service)
  - DBMS in the cloud
  - Database as a service
- Data warehouse on RDBMS
  - OLAP

# Amazon RDS: Database-as-a-service

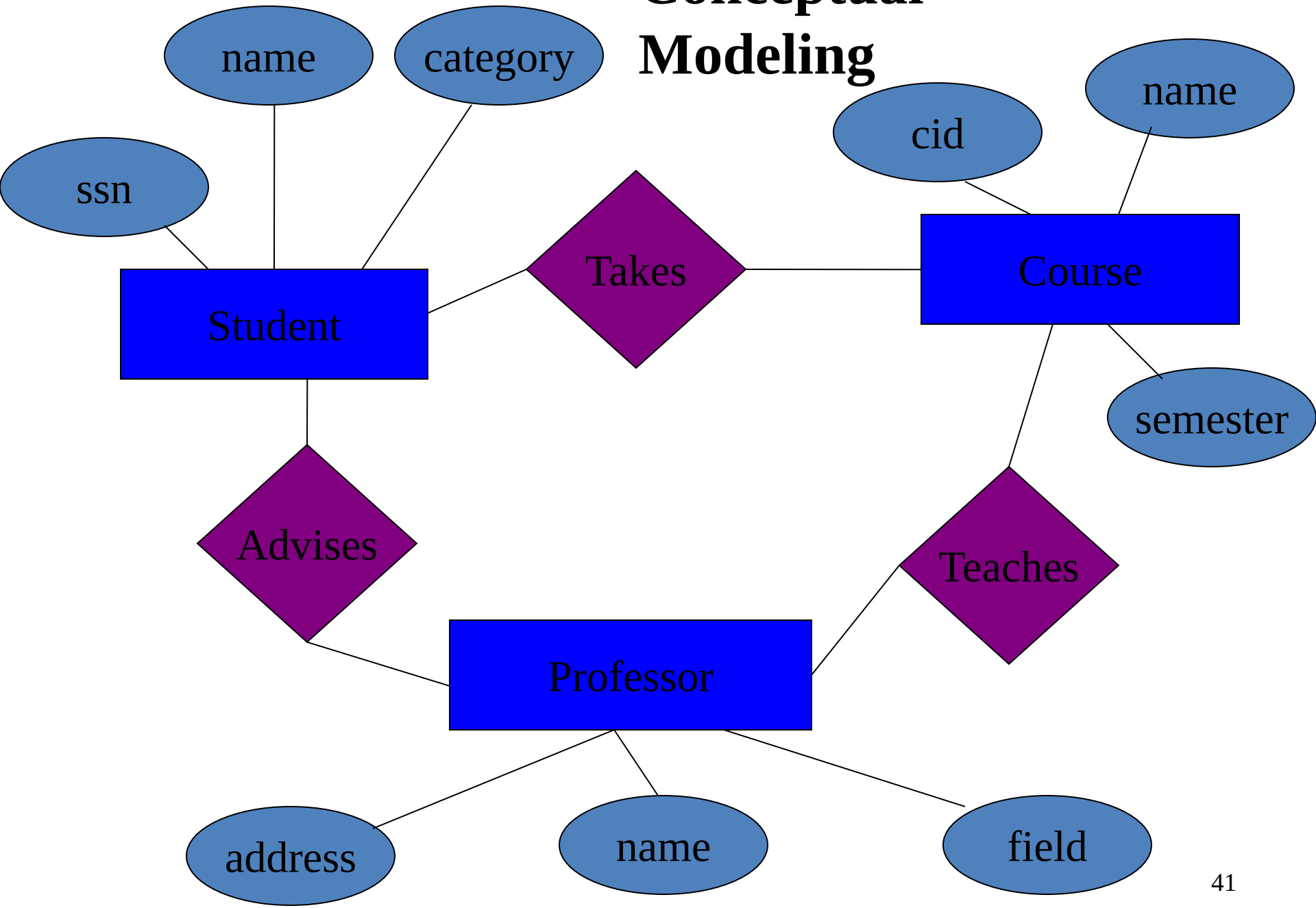
- MySQL, PostgreSQL, Oracle, SQL Server, etc.



# Access MySQL from EC2



# Conceptual Modeling



# Schema Design and Implementation

- Tables:

Students:

| SSN | Name | Category |
|---|---|---|
| 123-45-6789 | Charles | undergrad |
| 234-56-7890 | Dan | grad |
| | ... | ... |

Takes:

| SSN | CID |
|---|---|
| 123-45-6789 | CSE444 |
| 123-45-6789 | CSE444 |
| 234-56-7890 | CSE142 |
| | ... |

Courses:

| CID | Name | Semster |
|---|---|---|
| CSE444 | Databases | fall |
| CSE541 | Operating systems | spring |

- Separates the logical view from the physical view of the data.

# Querying a Database

- Find all courses that “Mary” takes
- S(tructured) Q(uery) L(anguage)

```
select  C.name
from    Students S, Takes T, Courses C
where   S.name = “Mary” and
        S.ssn = T.ssn and T.cid = C.cid
```

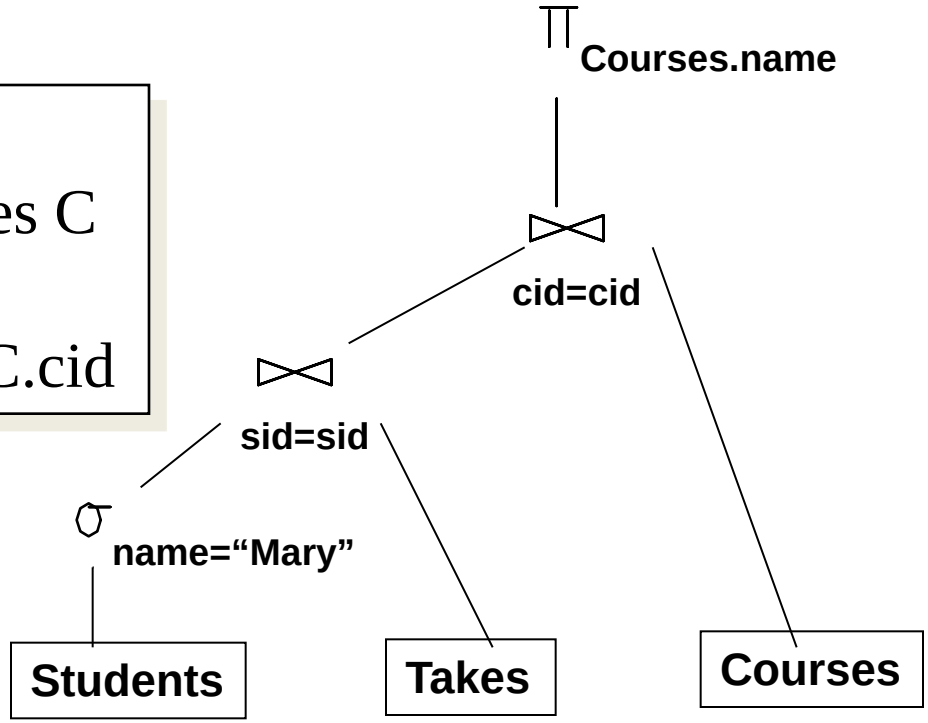
- Query processor figures out how to answer the query efficiently.

# Query Optimization

**Goal:**

*Declarative SQL query* → *Imperative query execution plan:*

```
select  C.name
from Students S, Takes T, Courses C
where S.name="Mary" and
        S.ssn = T.ssn and T.cid = C.cid
```




*Plan:* tree of Relational Algebra operators,
choice of algorithms at each operator

# Major topics

- Storage systems
- File systems & file formats
- Database management systems
- Big data solution stack 
- Data warehousing

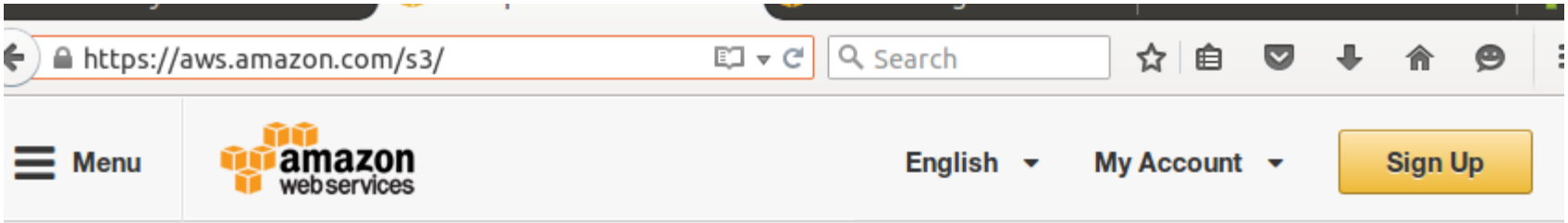
# Topics

- Big data management & analytics
  - Cloud data storage (Amazon S3)
  - NoSQL (Amazon DynamoDB, Cassandra, MongoDB)
  - MapReduce
  - Apache Hadoop
  - Apache Spark
  - Apache Hive

# Cloud data storage

- Amazon S3 (simple storage service)
  - Ideal for storing large binary files
  - E.g., audio, video, image
  - Simple RESTful web service

- Eventual consistency for high availability




PRODUCTS & SERVICES

- Amazon S3
- Product Details
- Storage Classes
- Pricing
- Getting Started
- FAQs
- Resources
- Amazon S3 SLA

RELATED LINKS

- AWS Management Console
- Documentation
- Release Notes

# Amazon S3

Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable object storage. Amazon S3 is easy to use, with a simple web service interface to store and retrieve any amount of data from anywhere on the web. With Amazon S3, you pay only for the storage you actually use. There is no minimum fee and no setup cost.

Amazon S3 offers a range of storage classes designed for different use cases including Amazon S3 Standard for general-purpose storage of frequently accessed data, Amazon S3 Standard - Infrequent

**Get Started with AWS Today**

Try Amazon S3 for Free

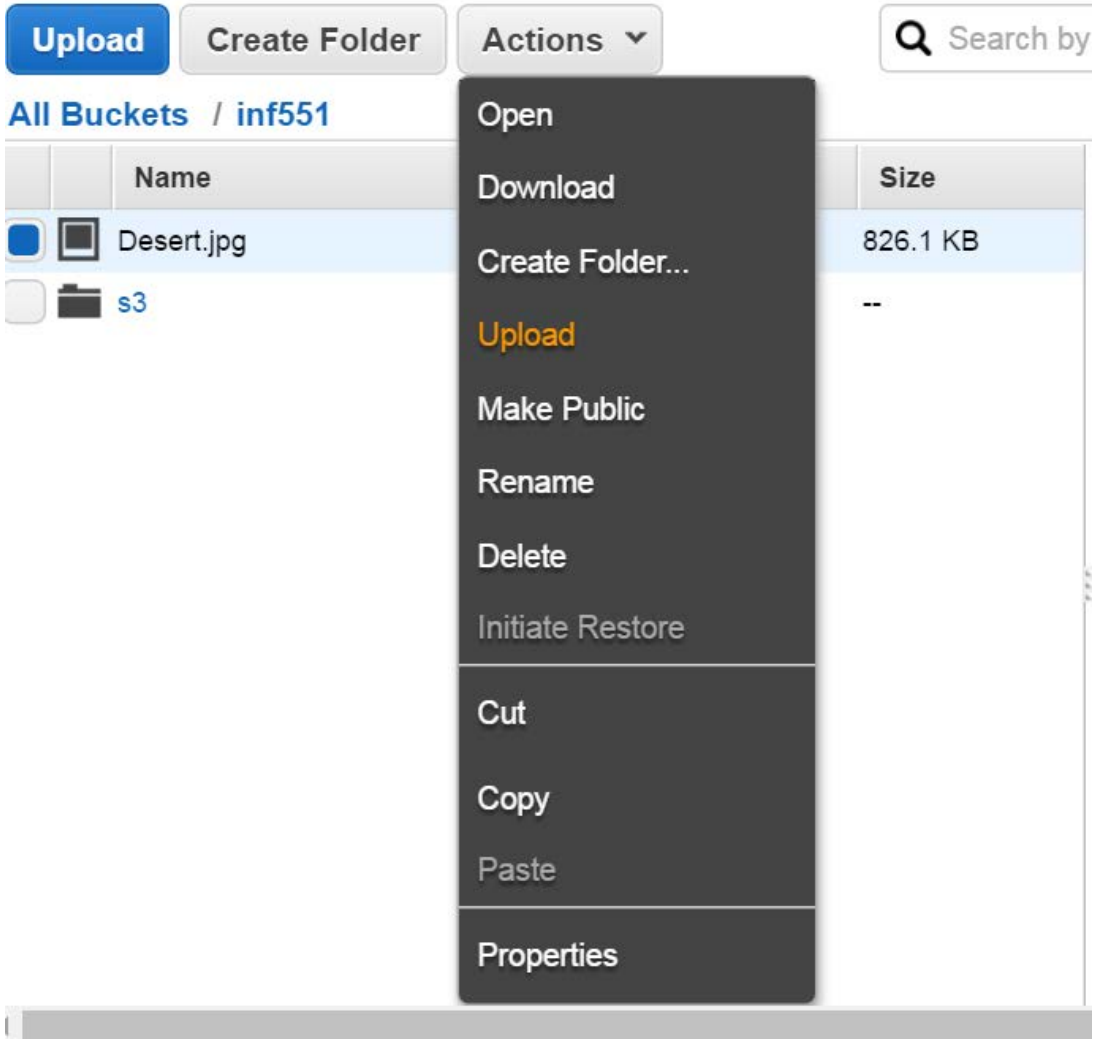
AWS Free Tier includes 5GB storage, 20,000 Get Requests, and 2,000 Put Requests with Amazon S3.

View AWS Free Tier Details »

## In Recent News

**New: Amazon VPC**

# Upload a file



https://s3.amazonaws.com/inf551/Desert.jpg

# NoSQL

- Not only SQL
- Flexible schemas
  - e.g., JSON documents or key-value pairs
  - Ideal for managing a mix of structured, semi-structured, and unstructured data
- High availability
- Weaker (e.g., eventual) consistency model

# Example NoSQL databases

- Amazon DynamoDB
  - Row store
  - row = item = a collection of key-value pairs
- Apache Cassandra
  - Wide column store
  - Google's Bigtable clone
- MongoDB
  - Manage JSON documents

# Key techniques

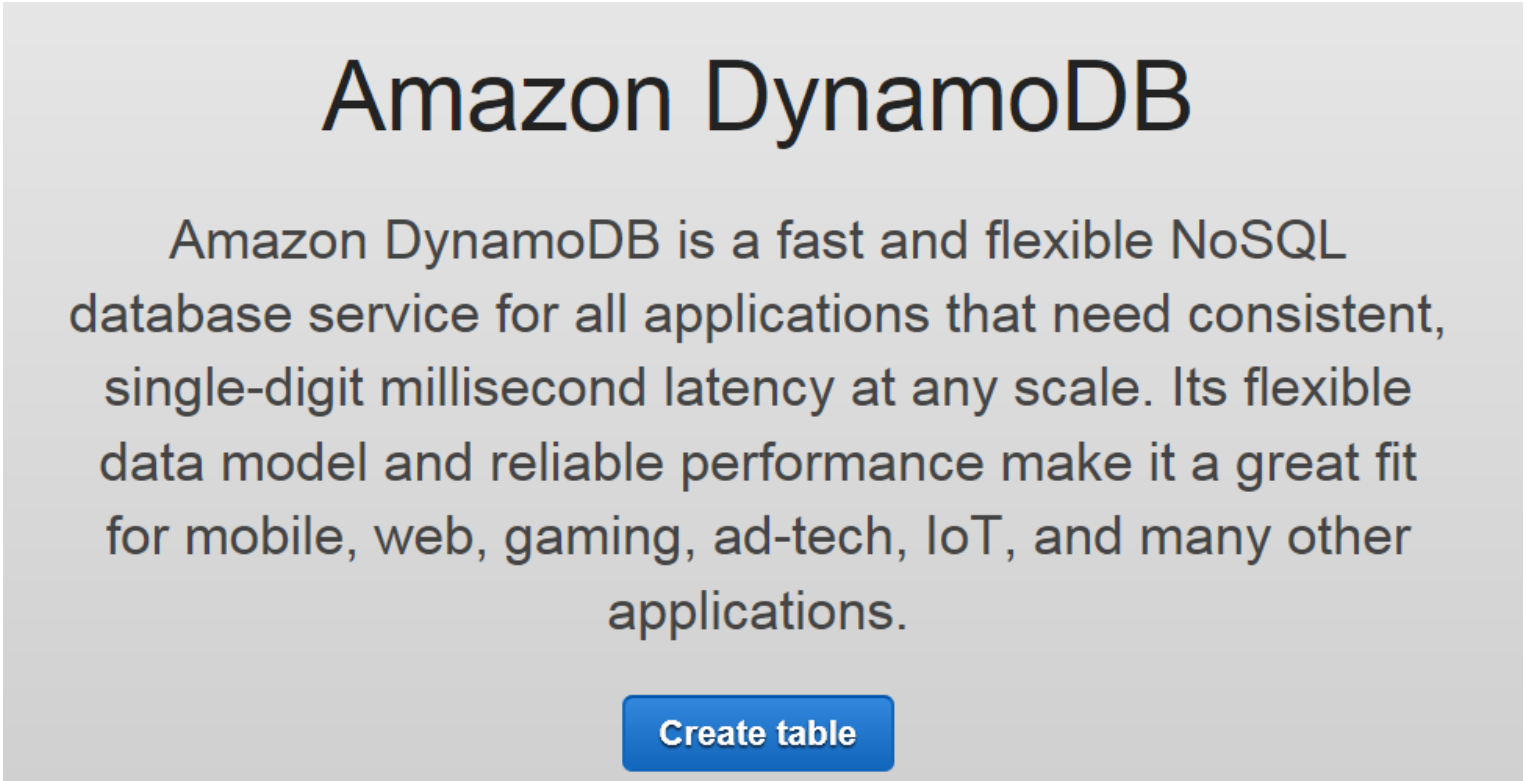
- Consistent hashing (Cassandra, Dynamo)
  - Avoid moving too much data when adding new machines
- Efficient writes
  - Append-only
  - No overwrites
  - Avoid random seek
  - But compaction needed later

# Key techniques

- Compaction
  - Introduced in Google "Bigtable" paper
  - Merge multiple versions of data
  - Remove expired or deleted data

# DynamoDB

- https://console.aws.amazon.com/dynamodb/home?region=us-east-1#gettingStarted:



# Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name*** Books

**Primary key*** Partition key

Author String

Add sort key

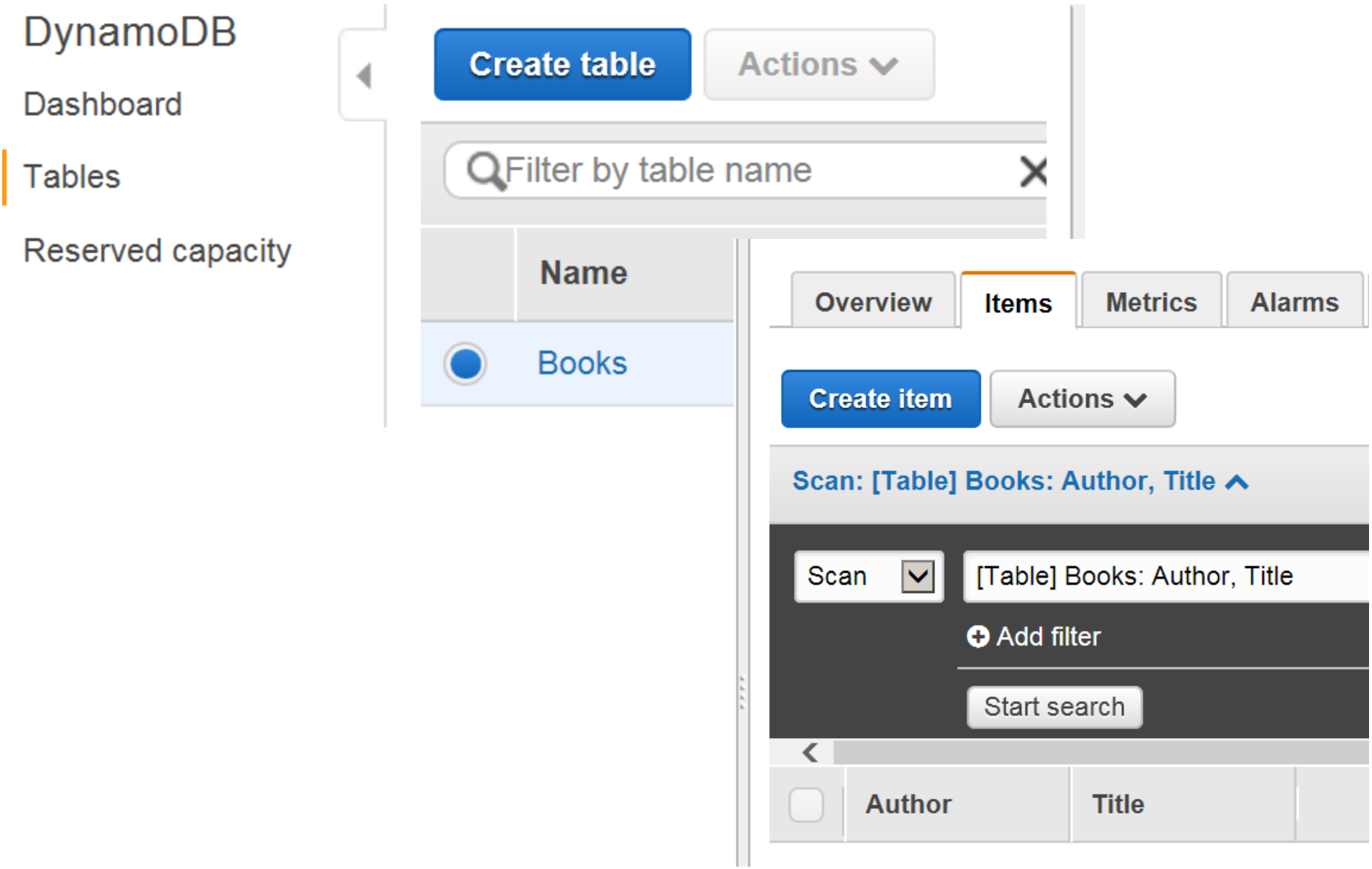
Title String

## Table settings

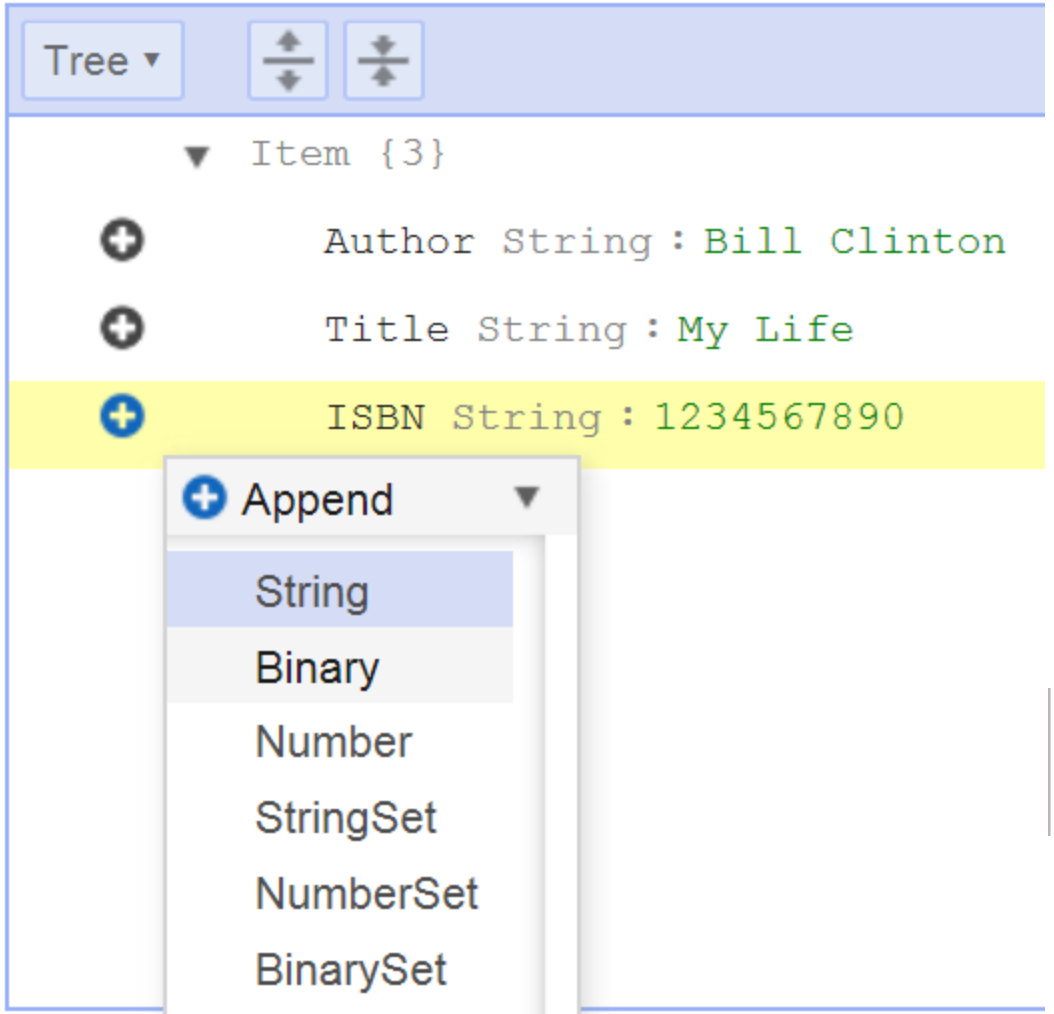
Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

# Insert items



# May add new attributes




| Author | Title | ISBN |
|---|---|---|
| Bill Clinton | My Life | 1234567890 |

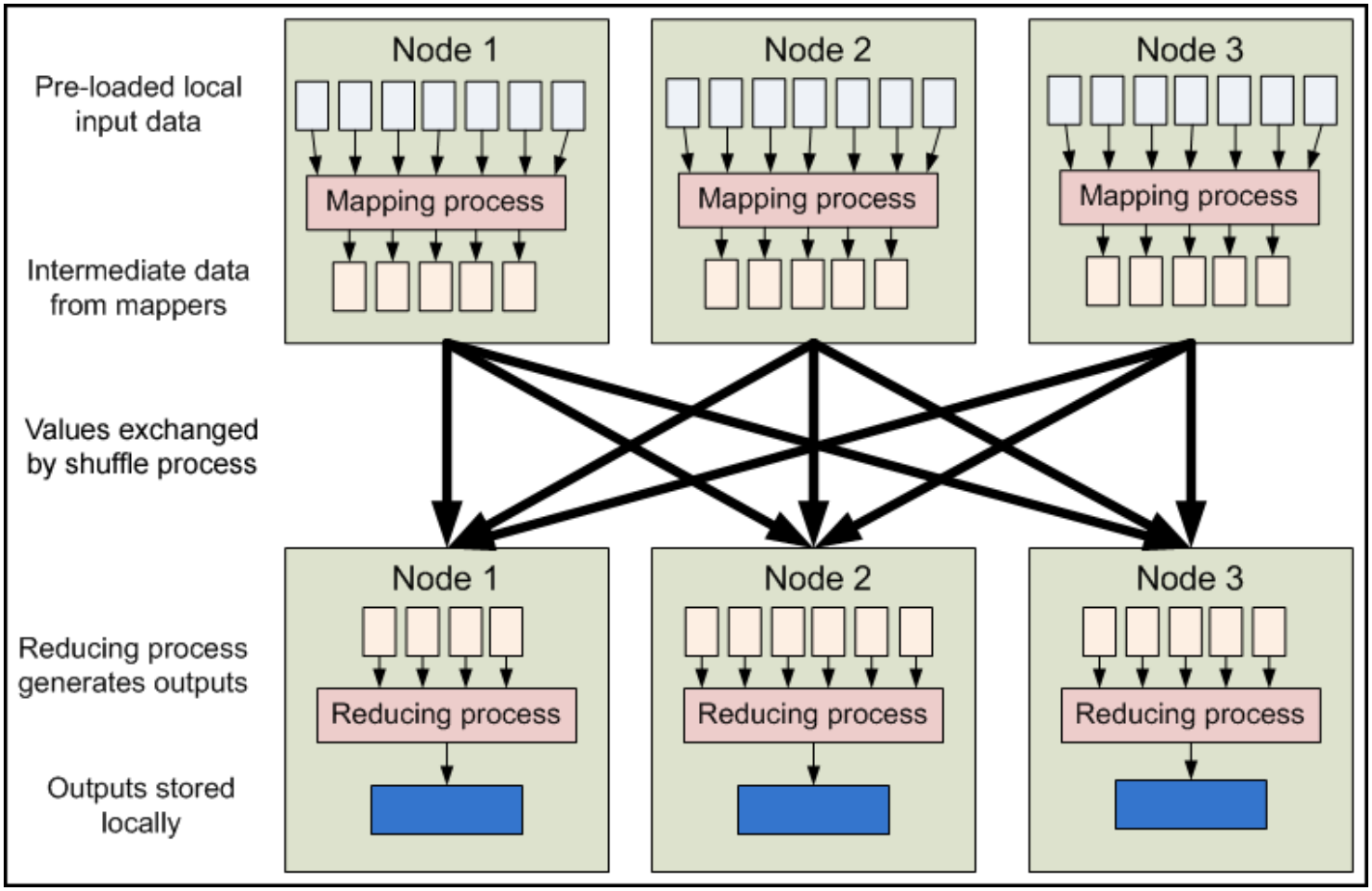
# Topics

- Big data management & analytics
  - Cloud data storage (Amazon S3)
  - NoSQL (Amazon DynamoDB, Cassandra, MongoDB)
  - MapReduce 
  - Apache Hadoop
  - Apache Spark
  - Apache Hive

# MapReduce

- Map
  - <k, v> => list of <k', v'>
- Reduce:
  - <k', list of v'> => list of <k'', v''>
- Write MapReduce programs on Hadoop
  - Using Java

# MapReduce


Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Node 2
Node 3
Mapping process
Mapping process
Mapping process
Node 1
Node 2
Node 3
Reducing process
Reducing process
Reducing process

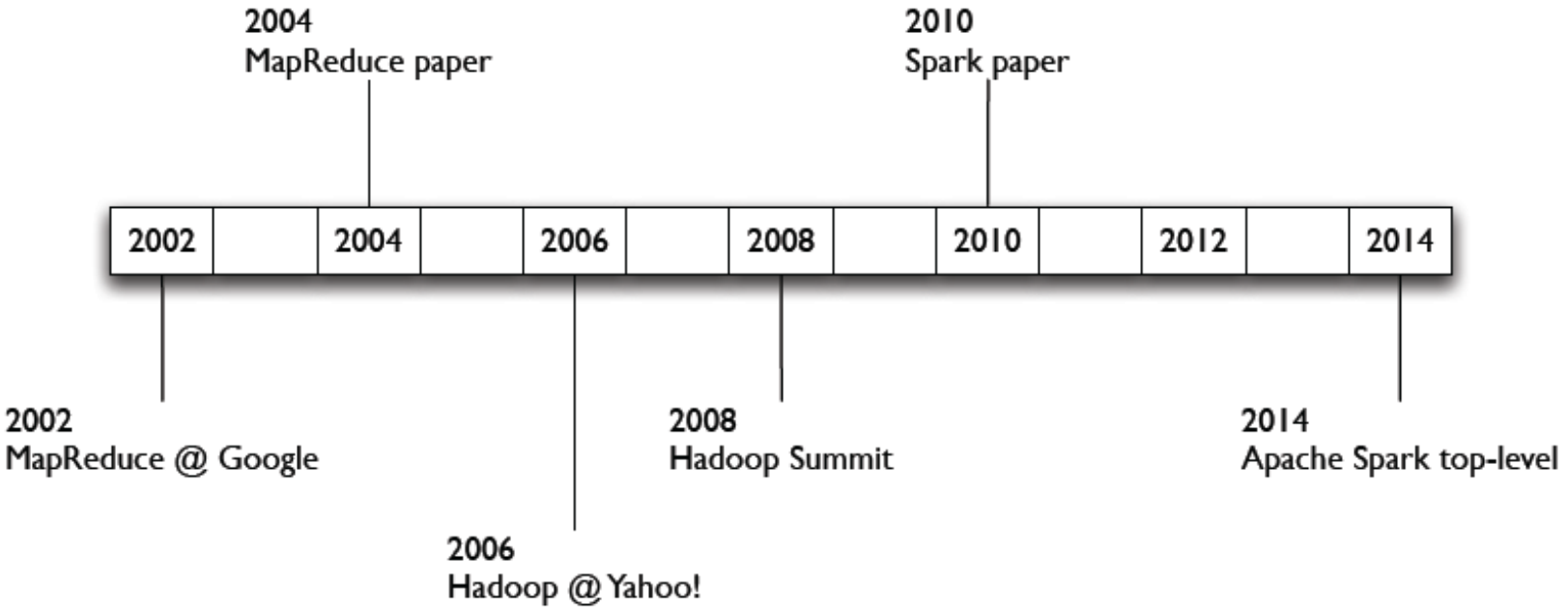
# Limitations of MapReduce

- Acyclic dataflow
  - difficult to reuse computed data
  - ill-suited for iterative algorithms
  - e.g., k-means, PageRank, logistic regression
- Batch processing
  - Not suitable for interactive data analysis

# In-memory MapReduce (Spark)

- Key concepts
  - RDD (resilient distributed dataset)
  - Transformations
  - Actions

# Apache Spark: history

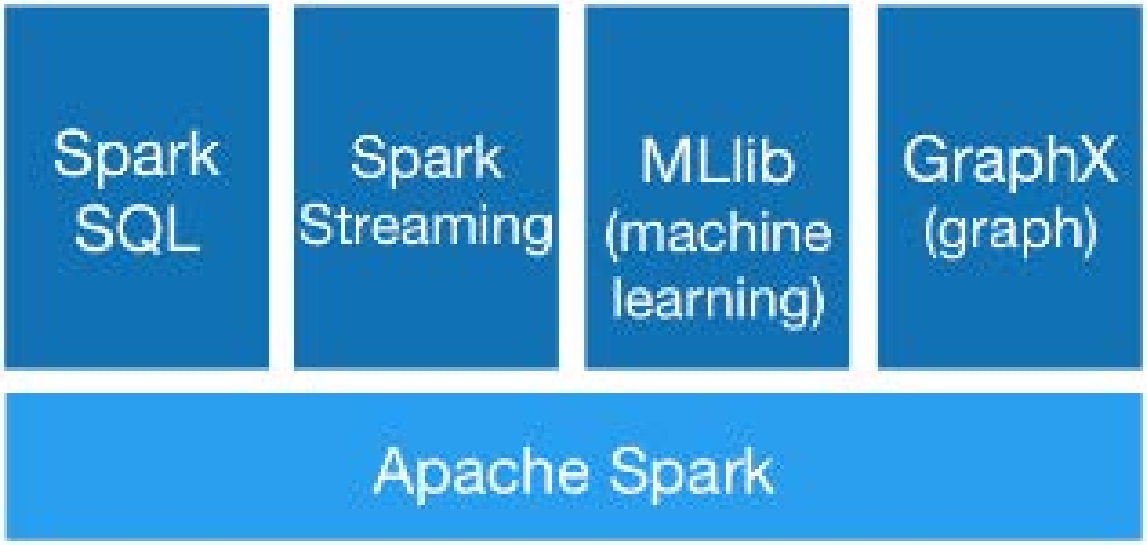


# Spark

- Support working sets through RDD
  - reuse & fault-tolerance
- 10x faster than Hadoop in iterative jobs
- Interactively explore 39GB with sub-second response time

# Spark

- Combine SQL, streaming, and complex analytics



# Spark

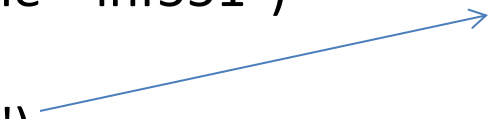
- Run on Hadoop, Cassandra, HBase, etc.





# wc.py

```
from pyspark import SparkContext
from operator import add

sc = SparkContext(appName="inf551")

lines = sc.textFile('hello.txt')

counts = lines.flatMap(lambda x: x.split(' ')) \
          .map(lambda x: (x, 1)) \
          .reduceByKey(add)

output = counts.collect()

for v in output:
   print '%s, %s' % (v[0], v[1])
```



# Major topics

- Storage systems
- File systems & file formats
- Database management systems
- Big data solution stack
- Data warehousing 

# Data warehousing

- Multidimensional data model
  - Star vs snowflake schema
- OLAP operations: rollup, drill-down, etc.
- Materialized views
- Index
  - Bitmap
  - Run-length encoding
  - Join index

# What is a Warehouse?

- Collection of diverse data
  - subject oriented, e.g., sales
  - aimed at executive, decision maker
  - often a copy of operational data
  - with value-added data (e.g., summaries, history)
  - integrated
  - time-varying: historical data, discovering trend
  - non-volatile: once in warehouse, data do not change

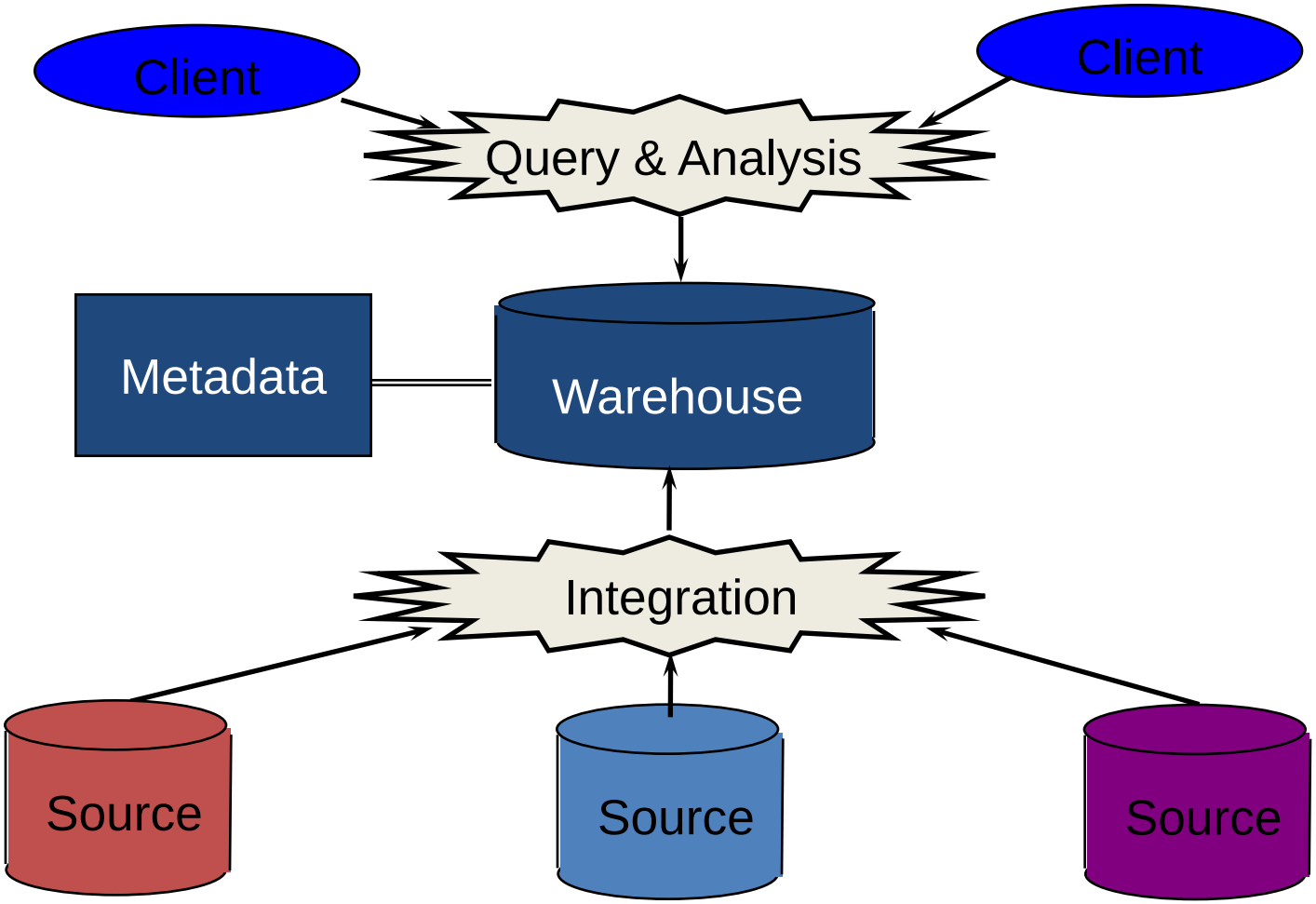


more

# What is a Warehouse?

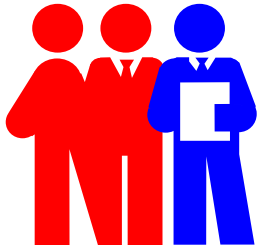
- Collection of tools
  - gathering data
  - cleansing, integrating, ...
  - querying, reporting, analysis
  - data mining
  - monitoring, administering warehouse

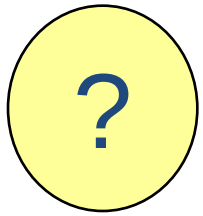
# Warehouse Architecture

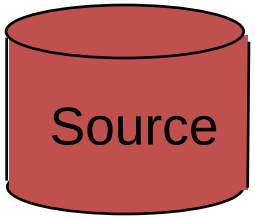


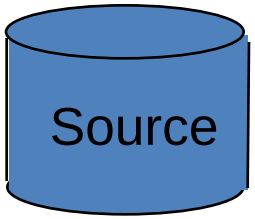
# Why a Warehouse?

- Two Approaches to Integration:
  - Warehouse (Eager)
  - Query-Driven (Lazy)

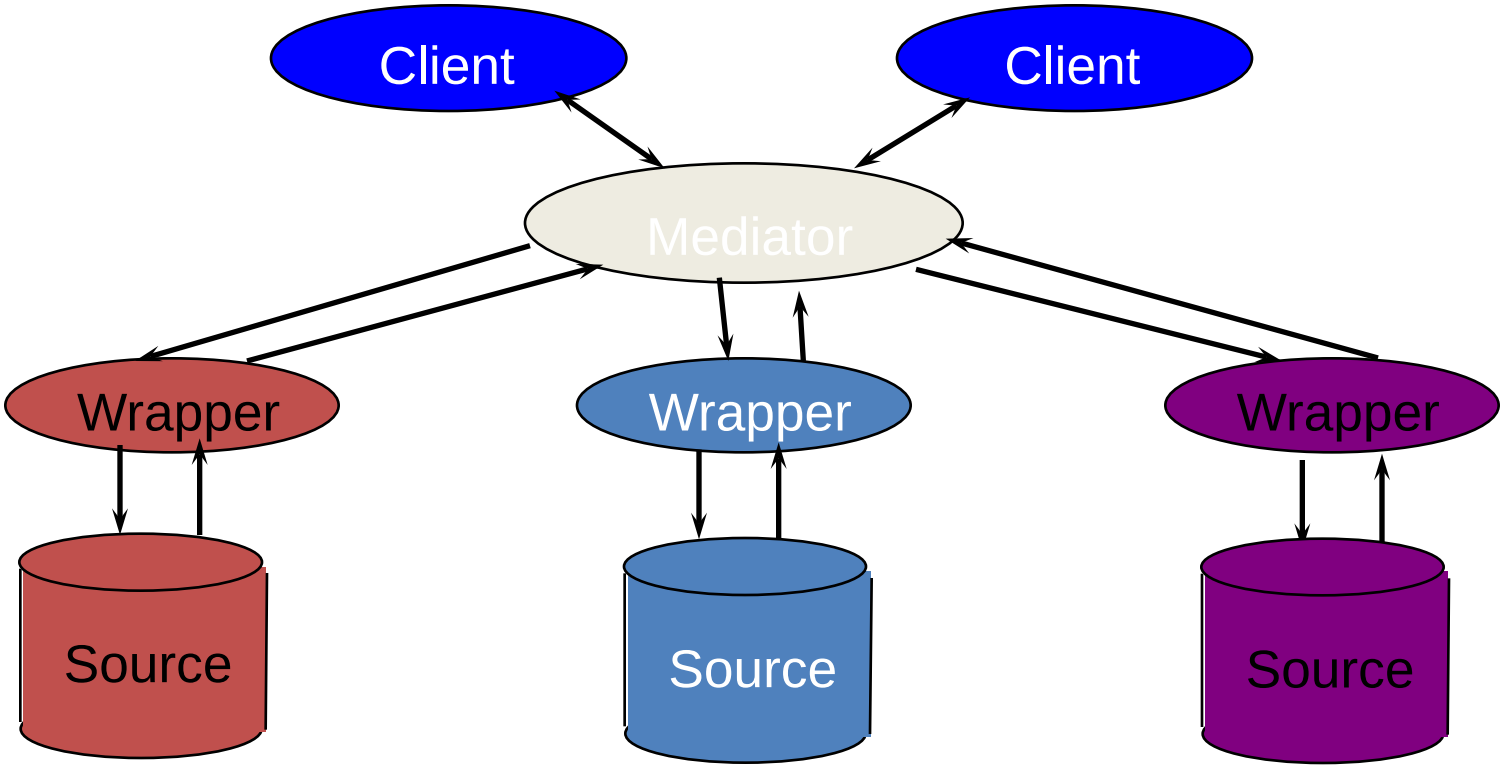




# Query-Driven Approach



# Advantages of Warehousing

- High query performance
- Local processing at sources unaffected
- Can operate when sources unavailable
- Extra information at warehouse
  - Modify, summarize (store aggregates)
  - Add historical information

# Disadvantages of Warehousing

- Decide what to store in advance
- Can only query data stored in warehouse
  - Data get stale
- Must detect source changes & update warehouse

# Advantages of Query-Driven

- No need to copy data, less storage
- No need to purchase data
- More up-to-date data
- Query needs can be unknown

# Disadvantages of Query-Driven

- Inefficient/delay in query processing
  - source unreliable
  - slow network
  - expensive translation, filtering, merging
- Sources might not permit ad-hoc queries
  - Examples?

# OLTP vs. OLAP

- OLTP: On Line Transaction Processing
  - Describes processing at operational sites (order entry in POS/online, banking transactions, etc.)
- OLAP: On Line Analytical Processing
  - Describes processing (answering analytical queries: aggregation, rollup/drilldown, slice/dice, etc.) at warehouse

# OLTP vs. OLAP

## OLTP

- Mostly updates
- Many small transactions
- Mb-Tb of data
- Raw data
- Clerical users
- Up-to-date data
- Consistency, recoverability critical

## OLAP

- Mostly reads
- Queries long, complex
- Gb-Tb of data
- Summarized, consolidated data
- Decision-makers, analysts as users
- Historical data
- Query performance critical

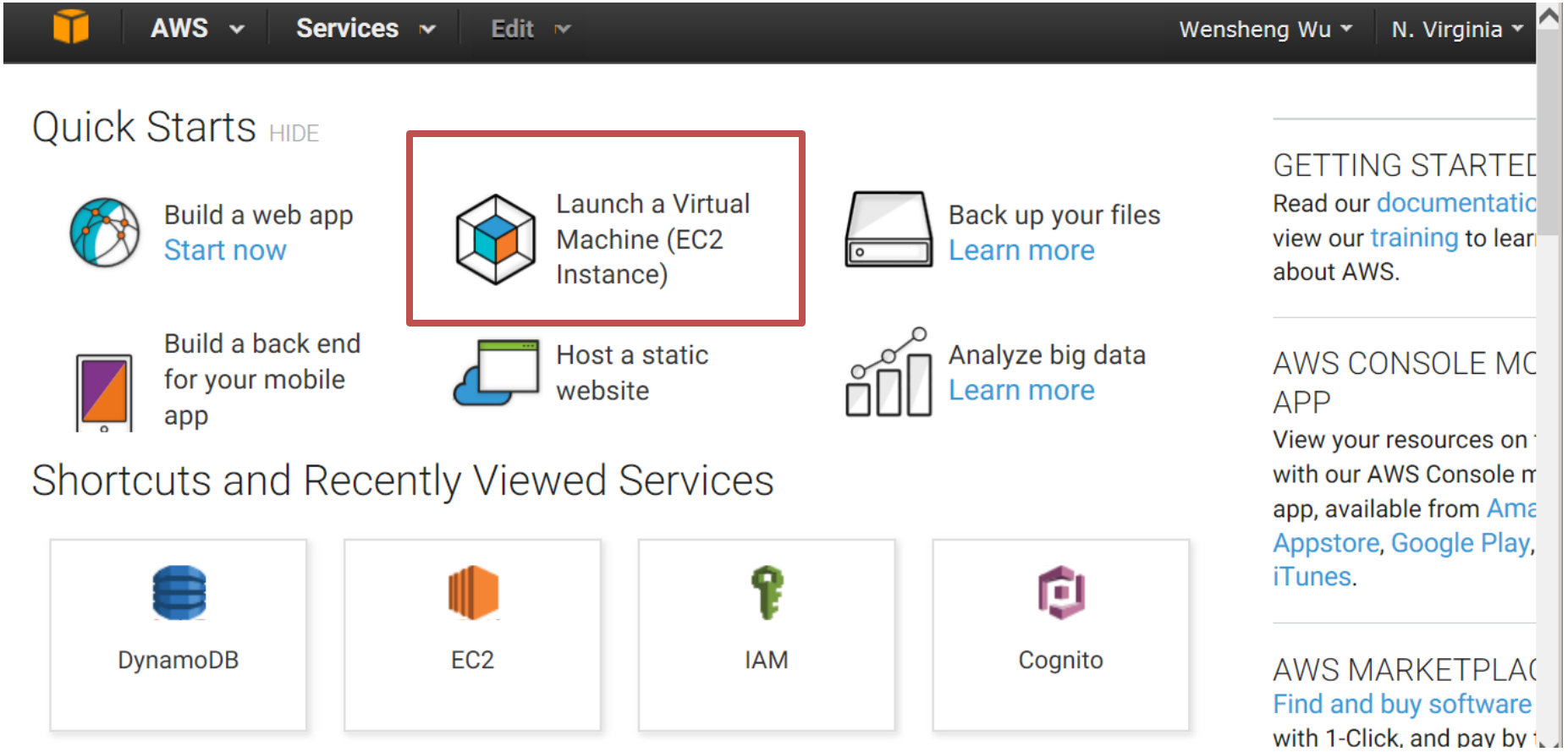
# Big data ETL & Warehousing

- Apache pig
  - Focus on ETL & data transformations
  - Compile transformations into MapReduce jobs
  - Pig latin script is procedural (step-by-step)

- Apache Hive
  - Declarative HiveQL (SQL-like)
  - Queries are turned into MapReduce jobs

# After class: sign up for AWS

# Dashboard after logging on



# Lab session

- Task: Setting up an EC2 instance

- Details: see lab session slides to be posted…