Introduction

INF 551

Wensheng Wu

Logistics

- Instructor email: wenshenw@usc.edu
- Class meeting times:
  - 10-11:50am, MW, THH 118
  - 4-5:50pm, MW, KAP 144
- Office hours:
  - 9-9:45am, MW, GER 204 (please email me first)

Logistics

- TAs
  - Morning section: Karan Maheshwari, kdmahesh@usc.edu
  - Afternoon section: Hsin-Yu Chang, hsinyuch@usc.edu

- Office hours
  - TBD
  - SAL computing lab (lobby)

Blackboard

- Discussion forums
  - You may post 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 & shell commands
  – 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).

Additional readings

- Links can be found in Syllabus (schedule)

- Base your reviews on the lectures
  - I will not ask questions on the reading materials that are not covered in the lectures

Grading structure

• Homework	20%
• Weekly quizzes	25%
• Midterm	15%
• Final	25%
• Lab sessions	5%
• Group project	10%

Grading scale

- [93, 100] = A
- [90, 93) = A-
- [87, 90) = B+
- [83, 87) = B
- [80, 83) = B-
- [77, 80) = C+
- [73, 77) = C
- … (see Syllabus for complete breakdown)

Lab sessions

- Flipped:
  - Task and details posted before class
  - Bring questions to class
- Typically utilize last 15-30 mins of class

Quizzes

- First quiz on 3rd week
- Last quiz on last week
- Based on previous week's materials

Exams

- Closed-notes & book
- Midterm: 10/9, Wednesday, in-class
- Final:
  – Morning section: 12/16, Monday, 8-10am
  – Afternoon section: 12/11, Wednesday, 4:30-6:30pm
  – same classroom as class meeting

Calculator

- Bring one to the tests

- If calculator is needed, we will either announce or state it on the tests

- Otherwise, no electronic devices are allowed

Group project

- Form a group of no more than 2 people

- Done in phases
  - Proposal
  - Midterm report
  - Final report

- I will talk about project ideas

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 Monday's classes
  - 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
- Final exam grades (& all grades) are final after final exam grading review time (to be announced right before/after final)
- Please submit reasonable regrading requests
  - Irrational requests (e.g., simply asking for more points or special treatments) may result in reduction of your grades

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

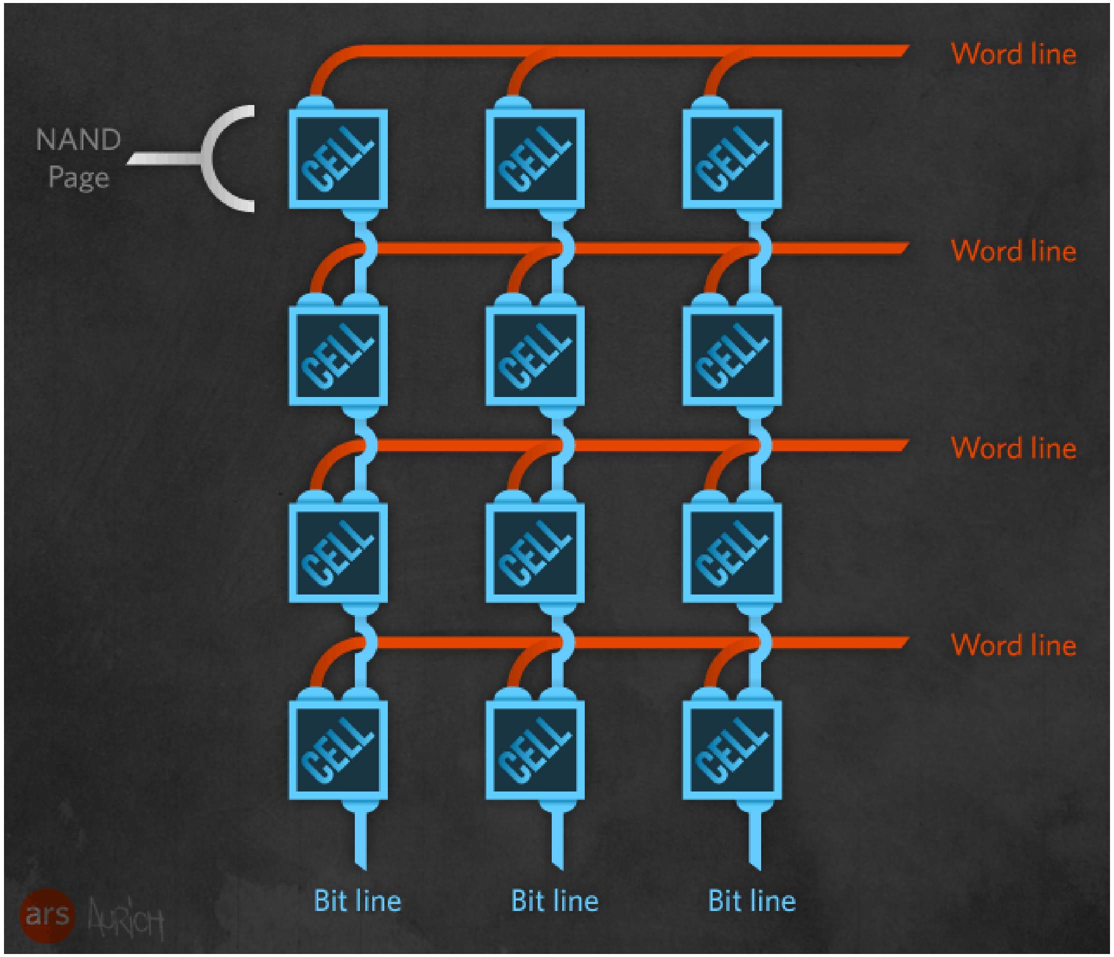
- Storage systems
- File systems & file formats
- Database management systems
- Big data solution stack
- Data warehousing (if time permits)

Storage Systems

- Hard disk
- SSD (Solid state drive)

Internal of hard disk

NAND flash



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

Latencies: read, write, and erase

	SLC	MLC	TLC	HDD	RAM
P/E cycles	100k	10k	5k	*	*
Bits per cell	1	2	3	*	*
Seek latency (μs)	*	*	*	9000	*
Read latency (μs)	25	50	100	2000-7000	0.04-0.1
Write latency (μs)	250	900	1500	2000-7000	0.04-0.1
Erase latency (μs)	1500	3000	5000	*	*
Notes	* metric is not applicable for that type of memory				
Sources	P/E cycles [20] SLC/MLC latencies [1] TLC latencies [23] Hard disk drive latencies [18, 19, 25] RAM latencies [30, 52] L1 and L2 cache latencies [52]				

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

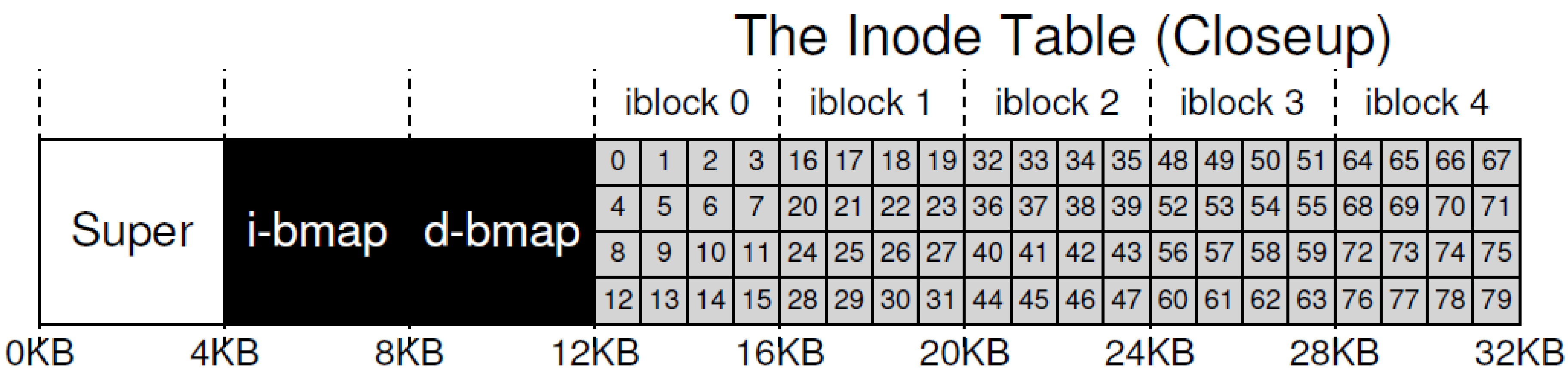
Standalone file systems

- Data structures
  - Data blocks
  - Metadata blocks (Inodes)
  - Bitmap blocks (for space allocation)

- Access paths
  - Read
  - write

Inode (index node)

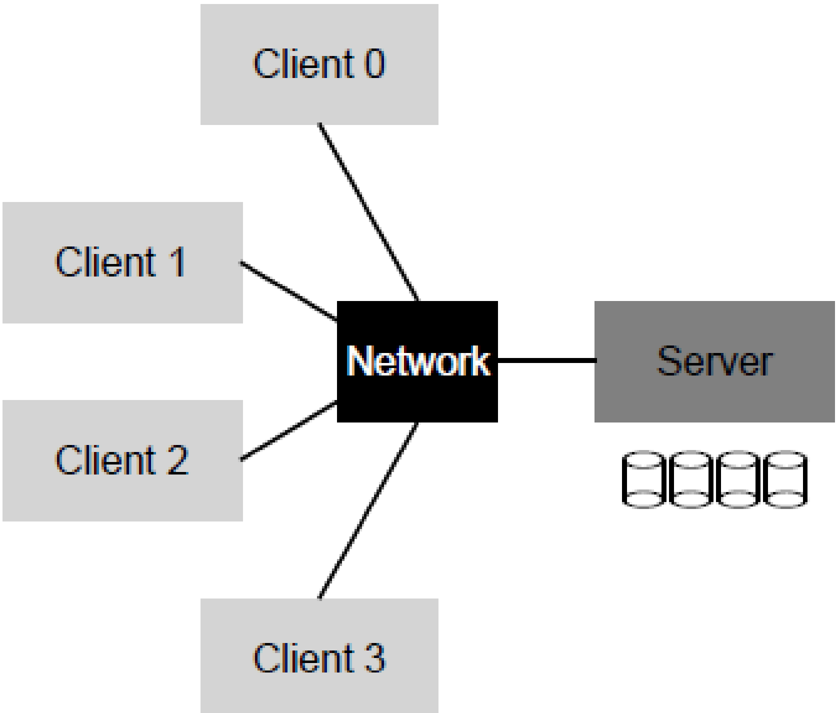
- Each is identified by a number
  - Low-level number of file name: inumber
- Can figure out location of inode from inumber



Network file system

- Client-server architecture
  - Sun's network file system

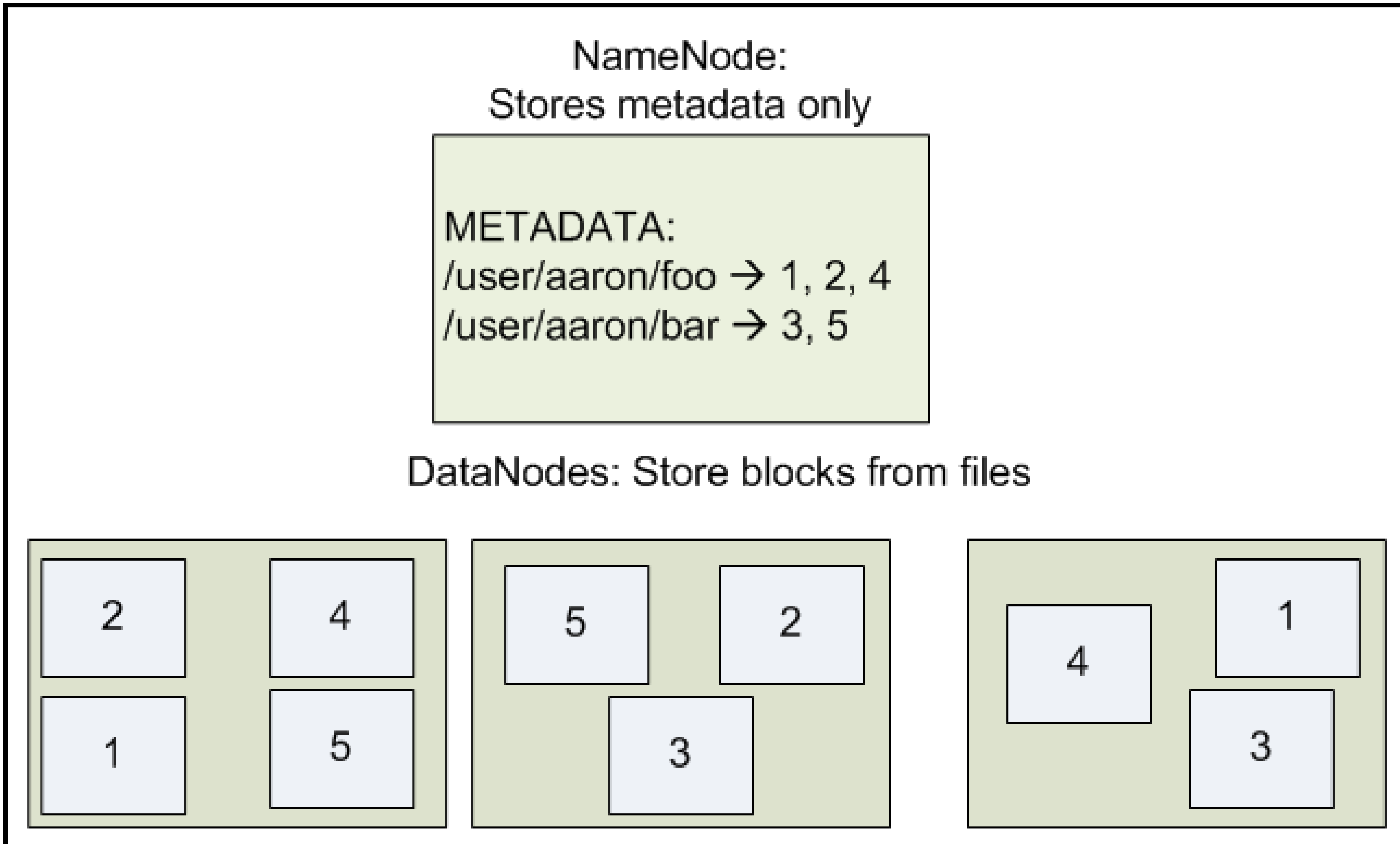
- Key concept:
  - stateless file handle



Distributed file systems

- Hadoop HDFS (after GFS)
  – Data are distributed among data nodes
- Replication
  – Automatic creation of replica (typically 2 or 3 copies/replica of 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

XML usages

- Software configurations files
  - E.g., HDFS
- Android app development
  - Layout resource files, e.g., activity_main.xml
- Java archive (.jar file)
  - Manifest.xml

Android app resource file

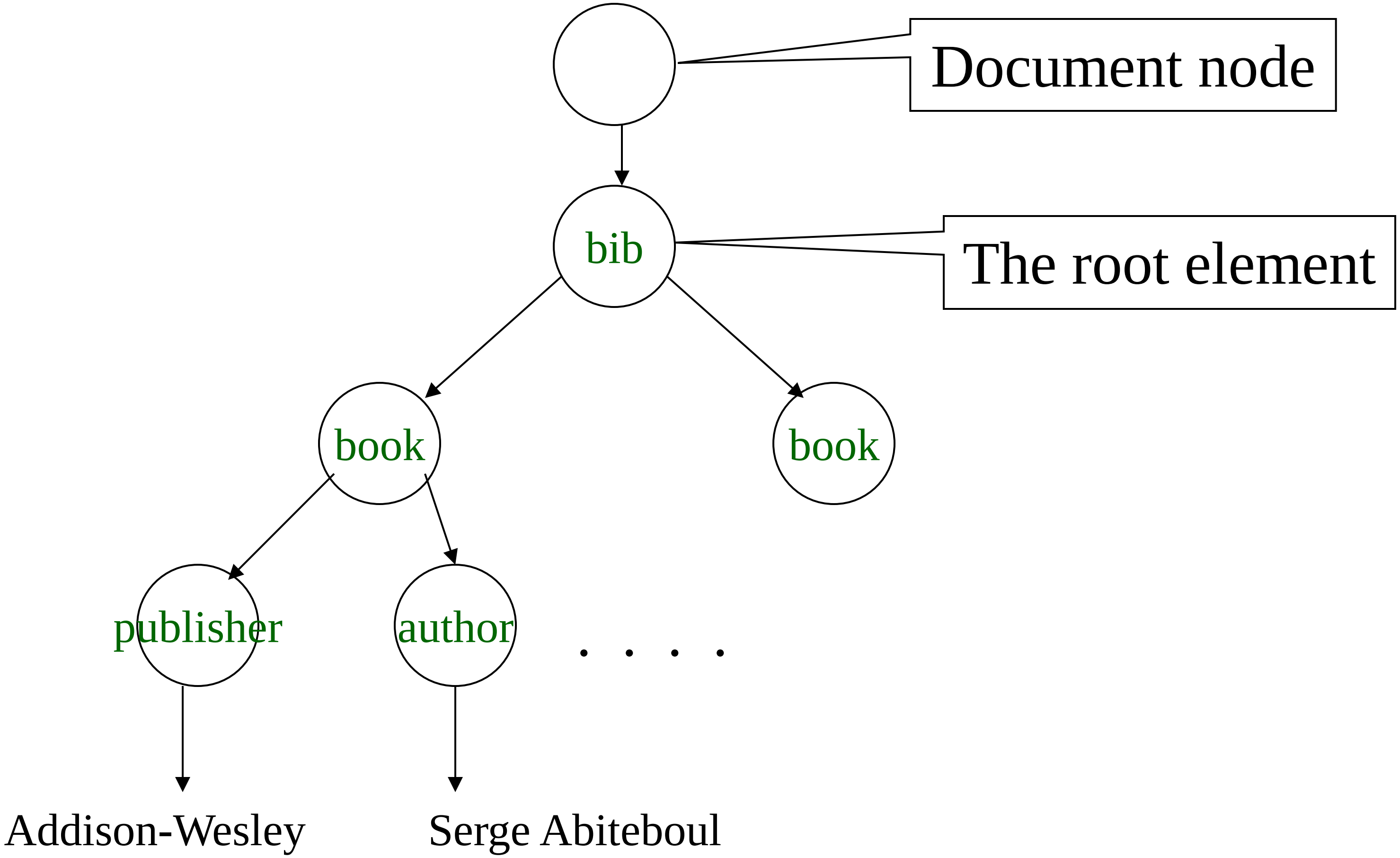
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <android.support.v4.view.ViewPager
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/tabs" />
```

Manifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.firebase.quickstart.database">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Firebase Database"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="Firebase Database"
            android:theme="@style/AppTheme" />
        <activity android:name=".NewPostActivity" />
        <activity android:name=".SignInActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```xml
<bib>
...
<book price="35">
     <publisher>Addison-Wesley</publisher>
      <author>Serge Abiteboul</author>
      <author><first-name>Rick</first-name><last-name>Hull</last-name></author>
      <author age="20">Victor Vianu</author>
      <title>Foundations of Databases</title>
      <year>1995</year>
     <price>38.8</price>
</book>
<book price="55">
      <publisher>Freeman</publisher>
      <author>Jeffrey D. Ullman</author>
      <title>Principles of Database and Knowledge Base Systems</title>
      <year>1998</year>
</book>
...
</bib>
```

Data Model for XPath



XPath: Simple Expressions

/bib/book/year

Result: <year> 1995 </year>

<year> 1998 </year>

/bib/paper/year

Result: empty (there were no papers)

Major topics

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

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 algorithms
  - Nested-loop join
  - 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

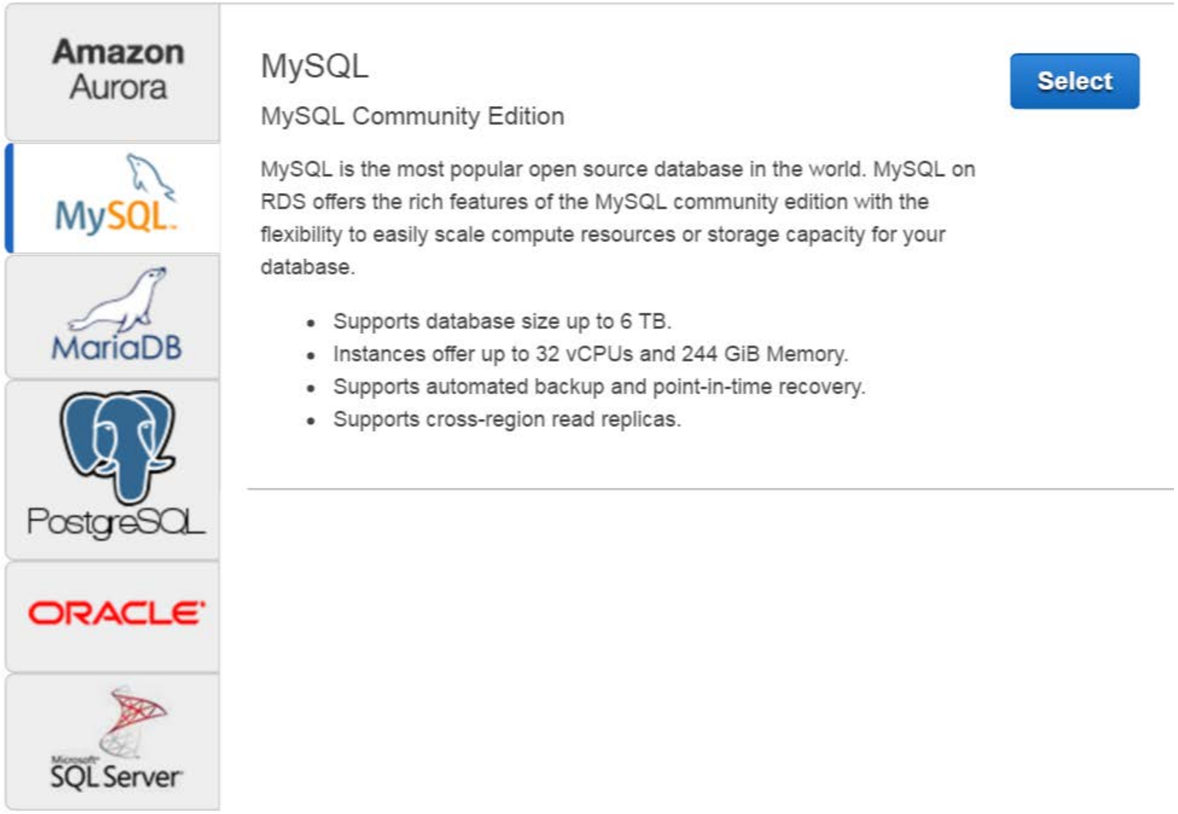
- MySQL (can be installed in Amazon AWS EC2)

- Amazon RDS (Relational database service)
  - DBMS in the cloud
  - Database as a service

- Data warehouse on RDBMS
  - OLAP

Amazon RDS: Database-as-a-service

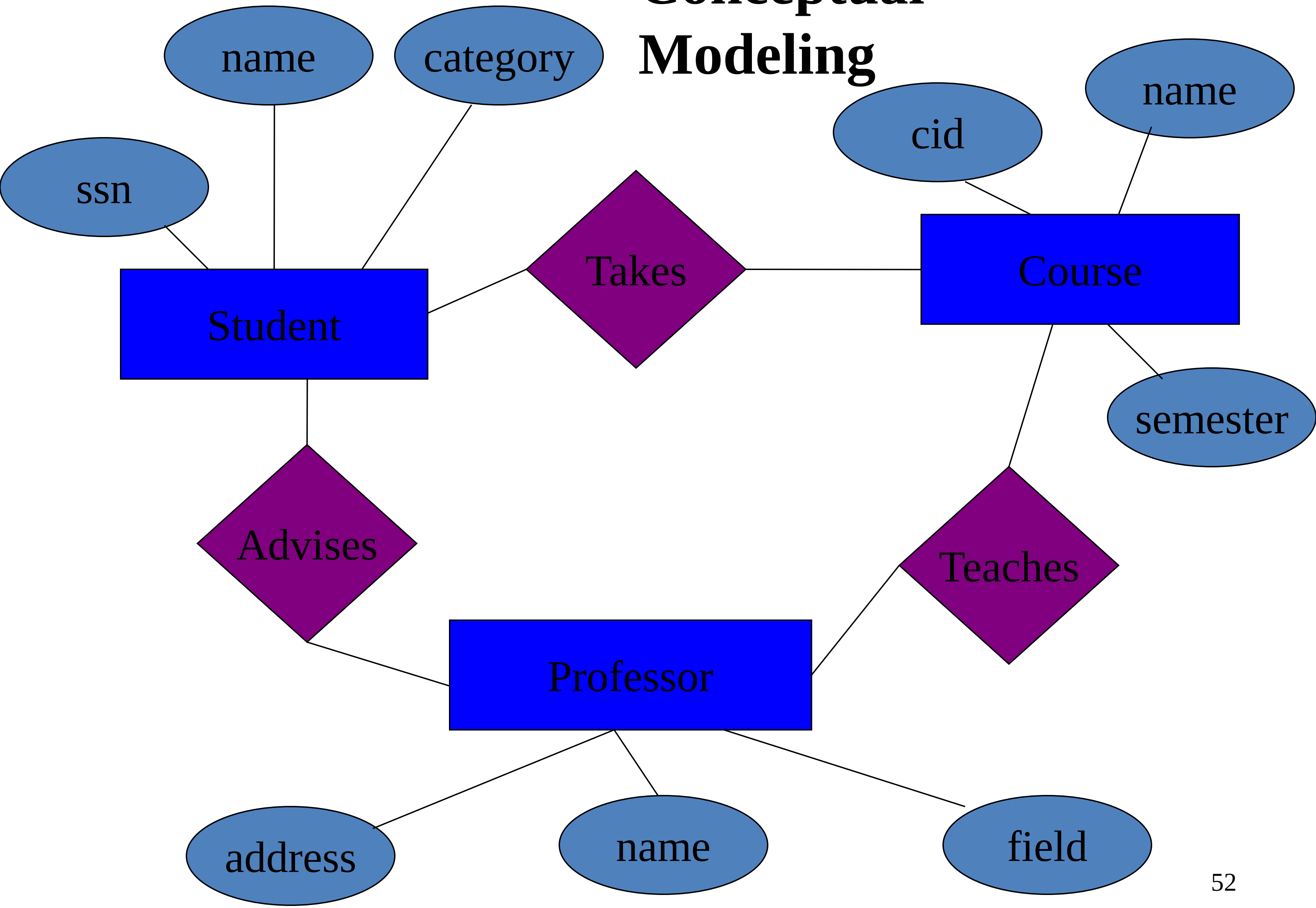
- MySQL, PostgreSQL, Oracle, SQL Server, etc.



Access MySQL from EC2

```
ssh -i "weixin.pem" ec2-user@ec2-107-22-155-60.compute-1.amazonaws.com
Last login: Sat Aug  6 07:43:38 2016 from 211.162.33.156

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2016.03-release-notes/
[ec2-user@ip-172-31-50-20 ~]$ mysql
ERROR 2002 (HY000): Can't connect to local MySQL server through socket '
/var/lib/mysql/mysql.sock' (2)
[ec2-user@ip-172-31-50-20 ~]$ sudo service mysqld start
Starting mysqld:                                           [  OK  ]
[ec2-user@ip-172-31-50-20 ~]$ mysql
ERROR 1045 (28000): Access denied for user 'ec2-user'@'localhost' (using
 password: NO)
[ec2-user@ip-172-31-50-20 ~]$  mysql -h inf551.chdcdeeoqxf5.us-east-1.rd
s.amazonaws.com -P 3306 -u inf551 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 111
Server version: 5.6.27-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reser
ved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input stat
ement.

mysql>
```

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)

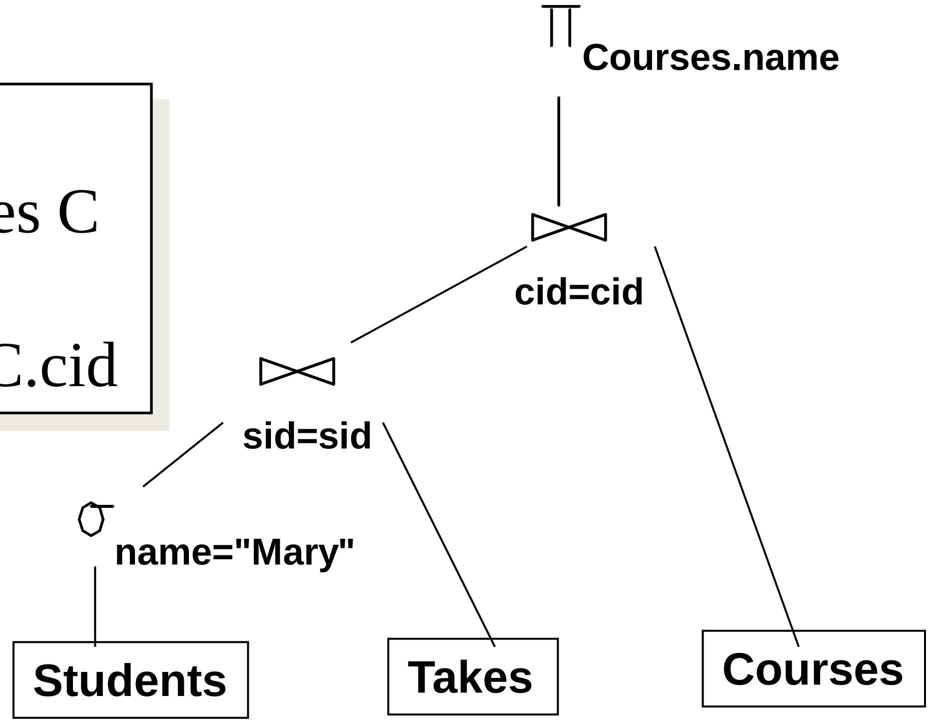
```
select  C.name
from    Students S, Takes T, Courses C
where   S.name = "Mary" and
        S.ssn = T.ssn and T.cid = C.cid
```

- 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

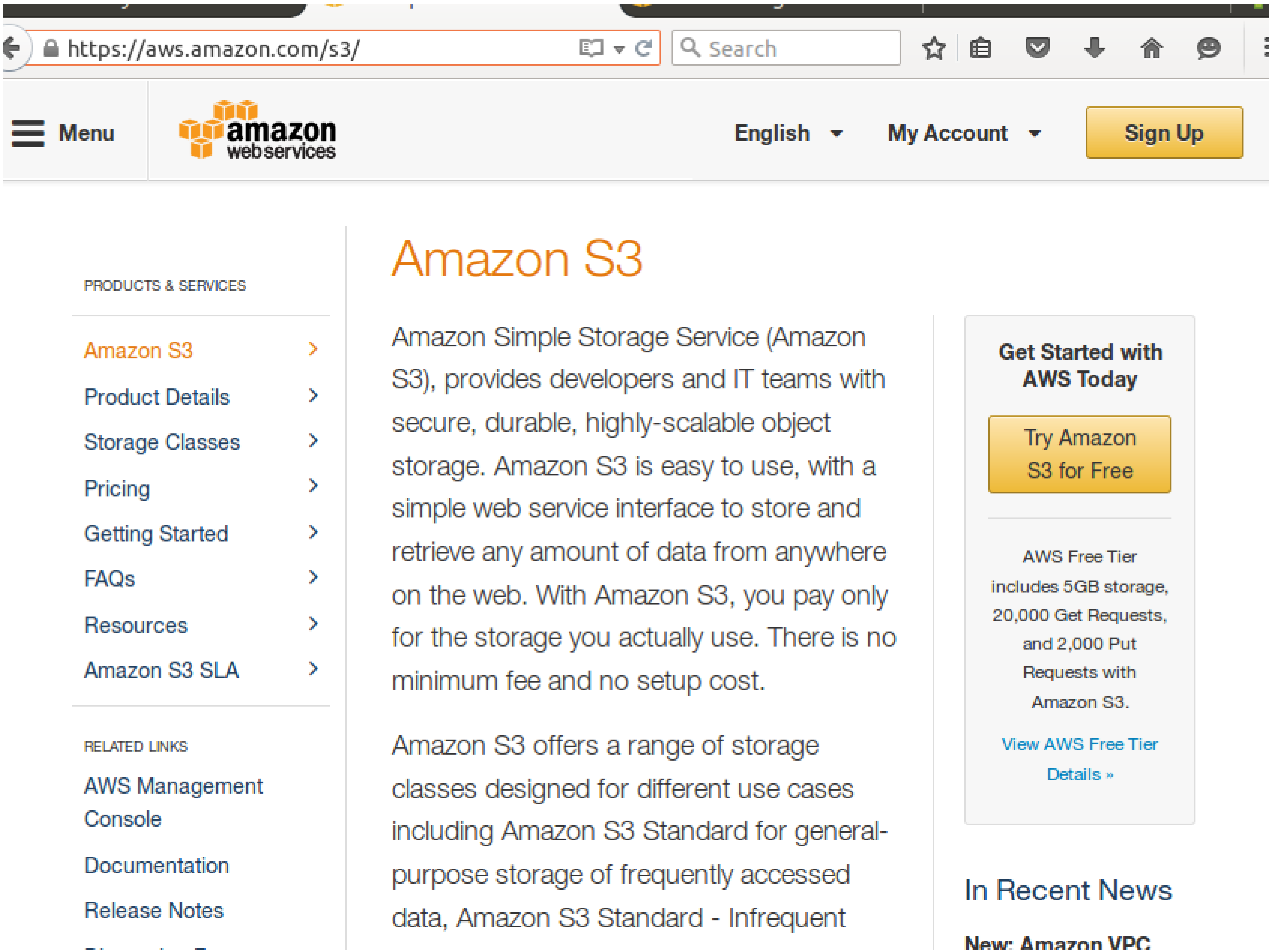
- Storage systems
- File systems & file formats
- Database management systems
- Big data solution stack
- Data warehousing

Topics

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

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



https://aws.amazon.com/s3/

Search

Menu

amazon webservices

English

My Account

Sign Up

Amazon S3

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 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

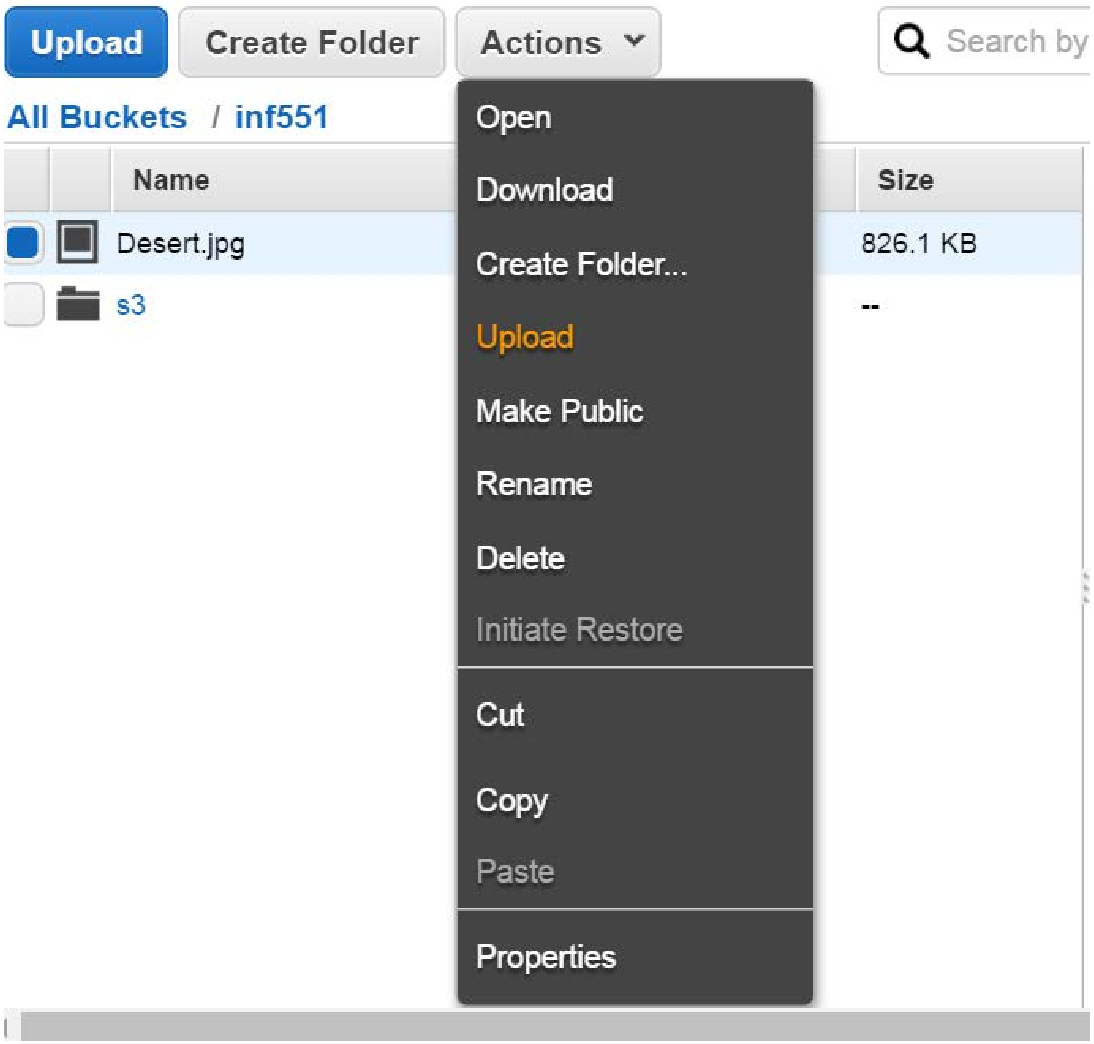
AWS Free Tier includes 5GB storage, 20,000 Get Requests, and 2,000 Put Requests with Amazon S3.

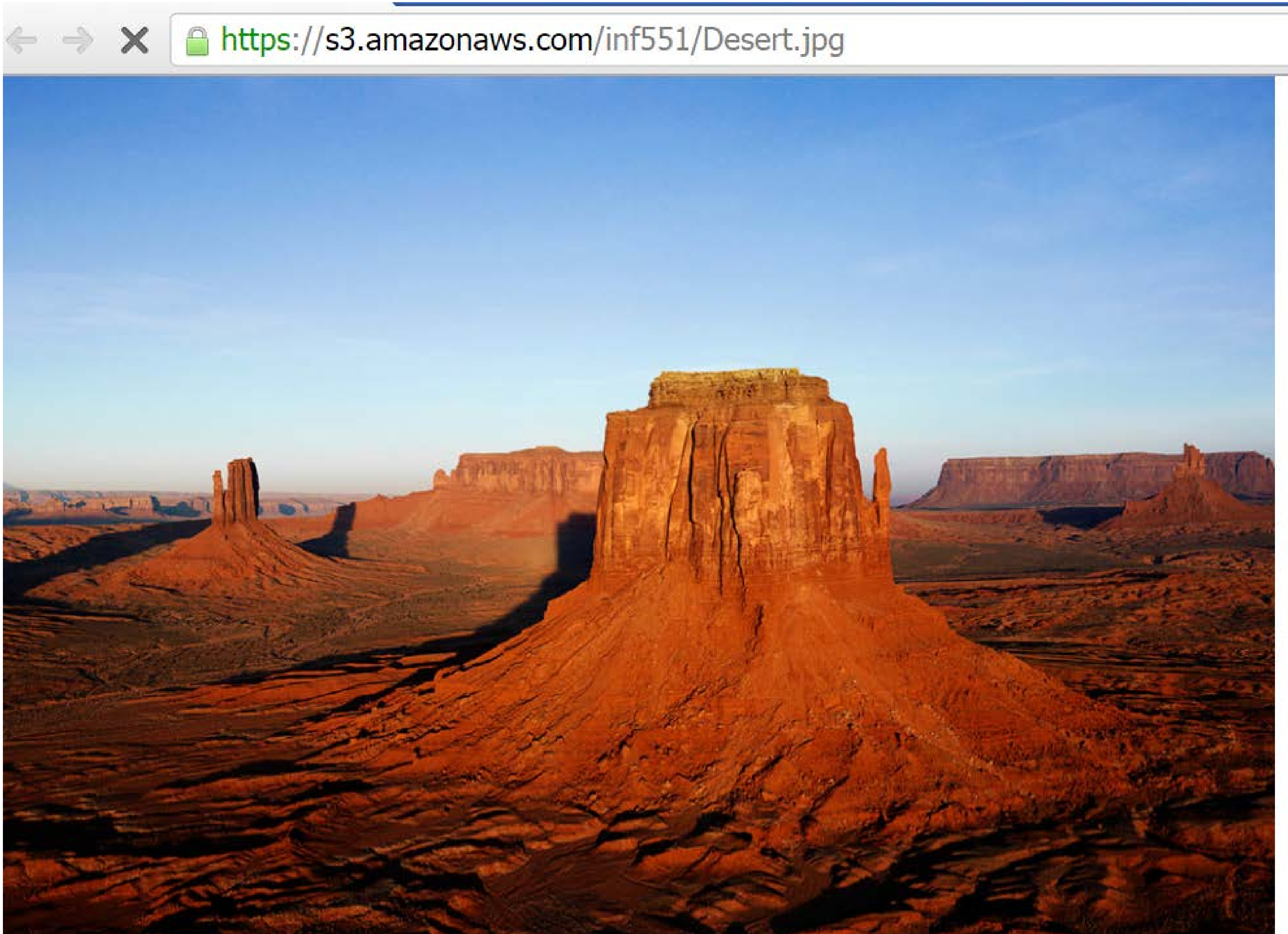
View AWS Free Tier Details »

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

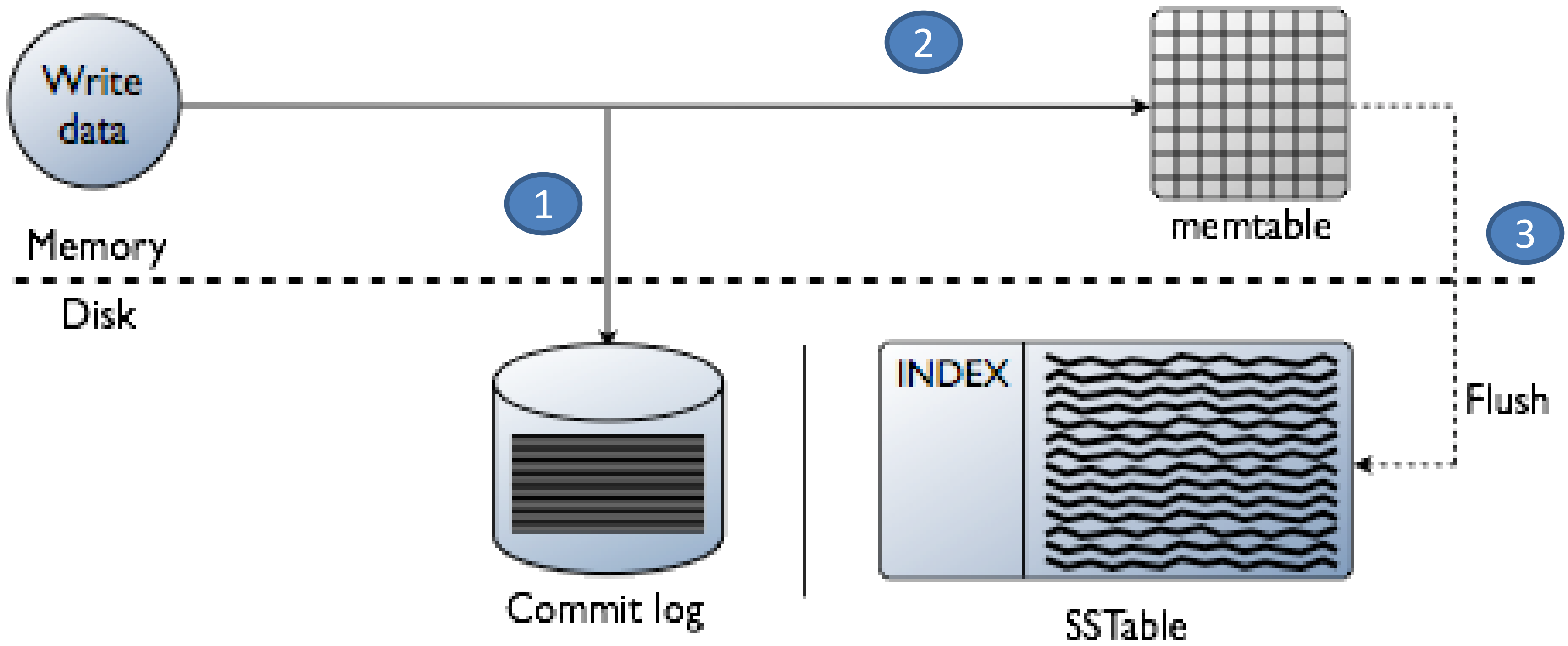
Example NoSQL databases

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

Key techniques

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

Write path in Cassandra

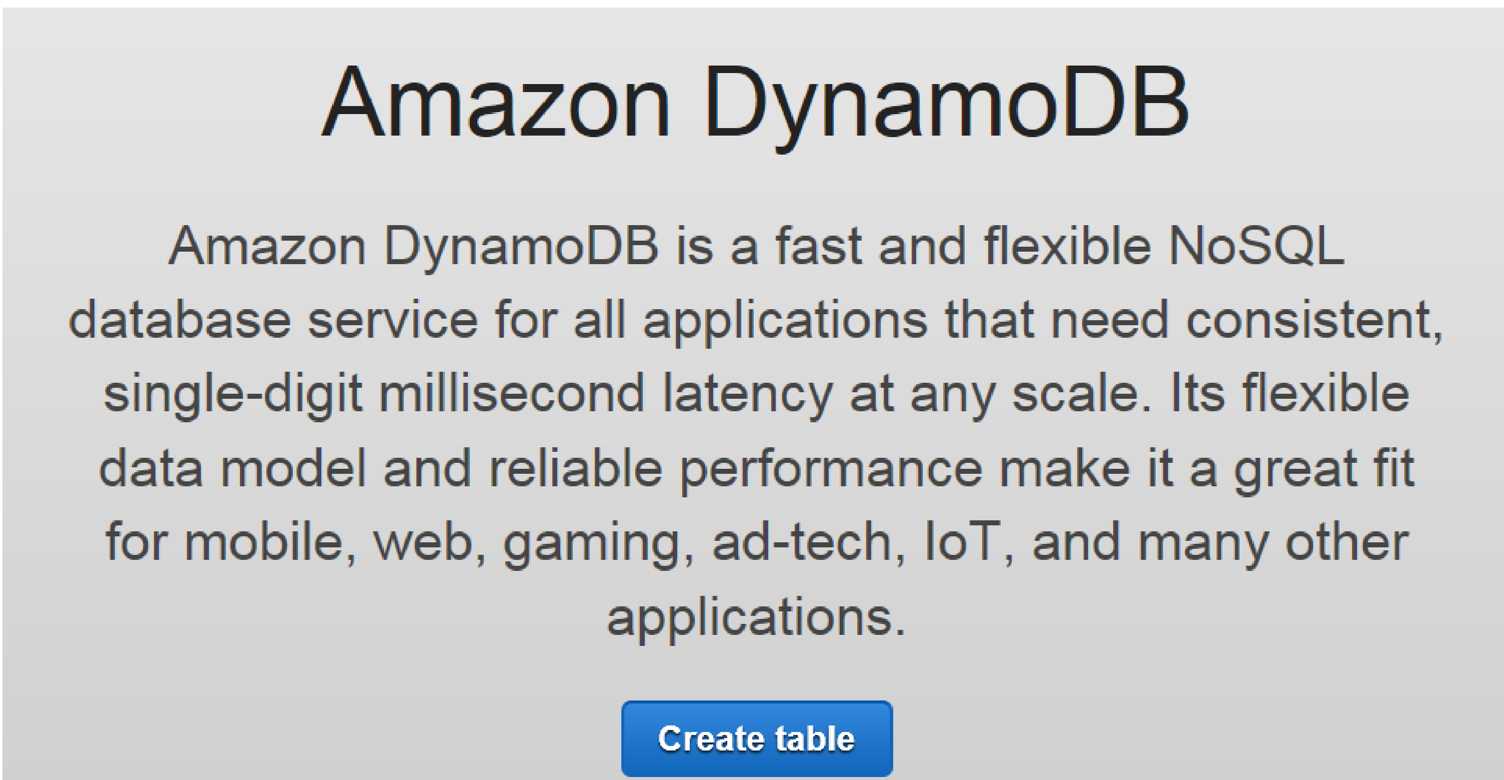


Key techniques

- 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

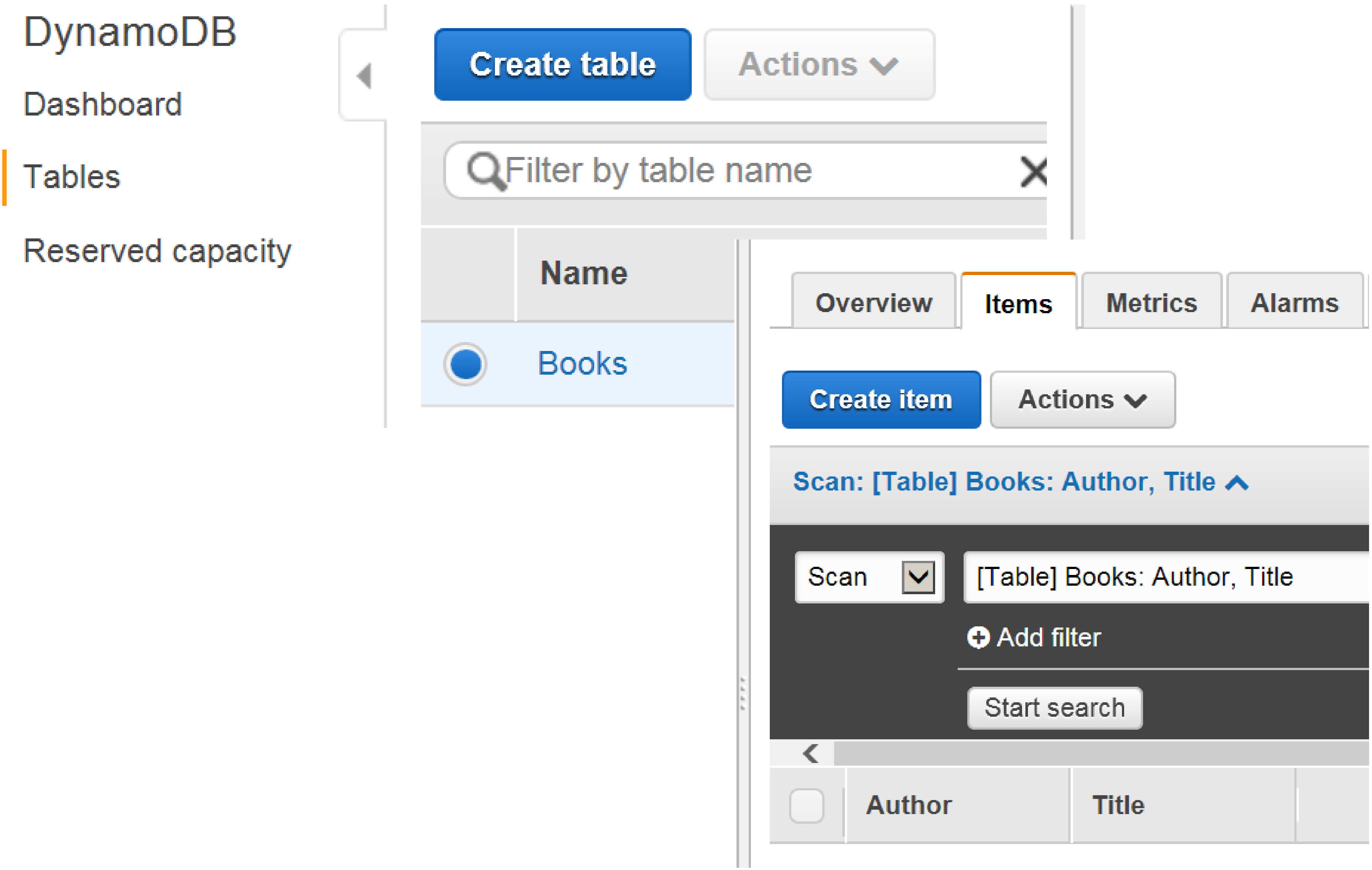
Title — String

Table settings

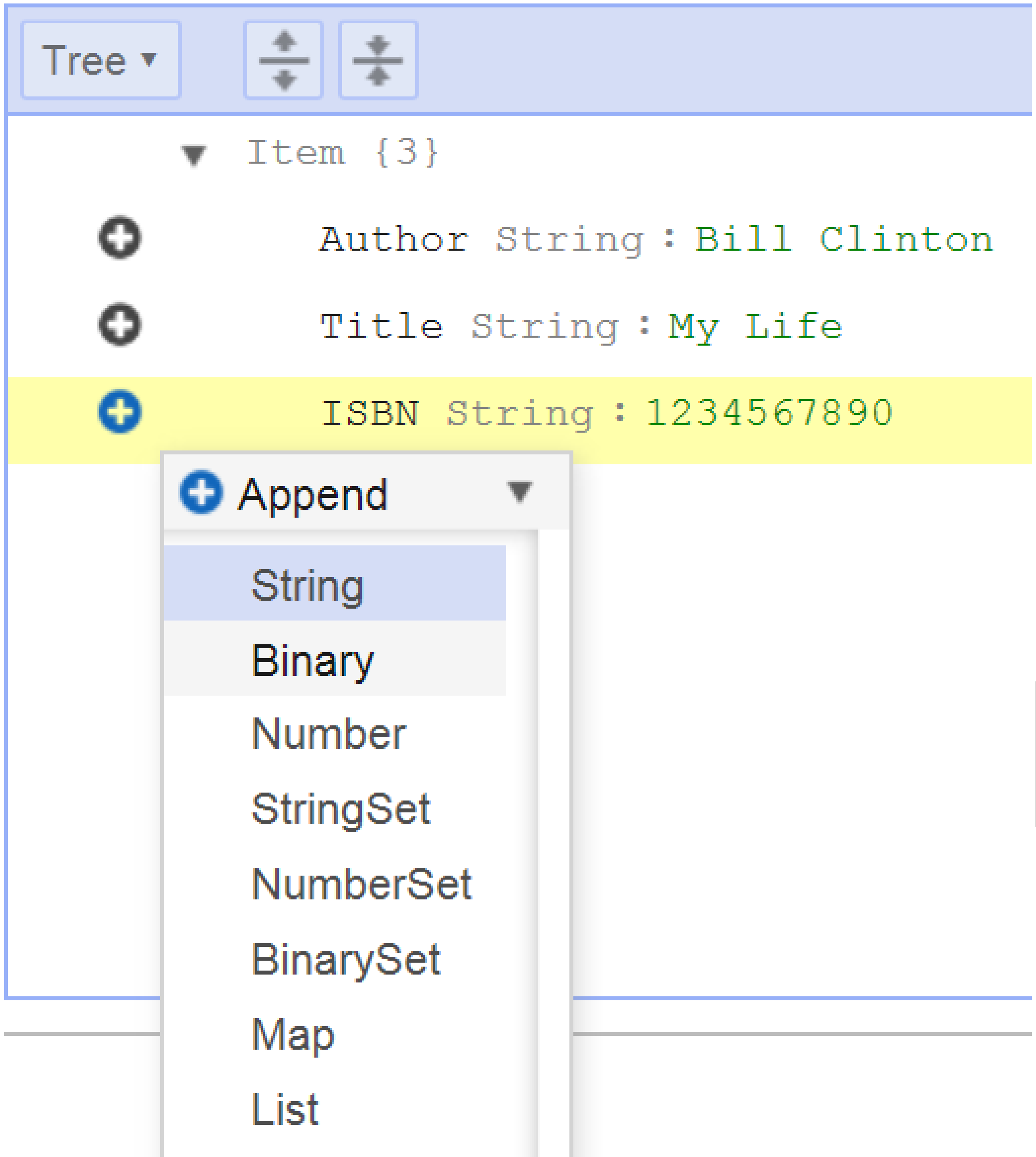
Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

☑ Use default settings

Insert items

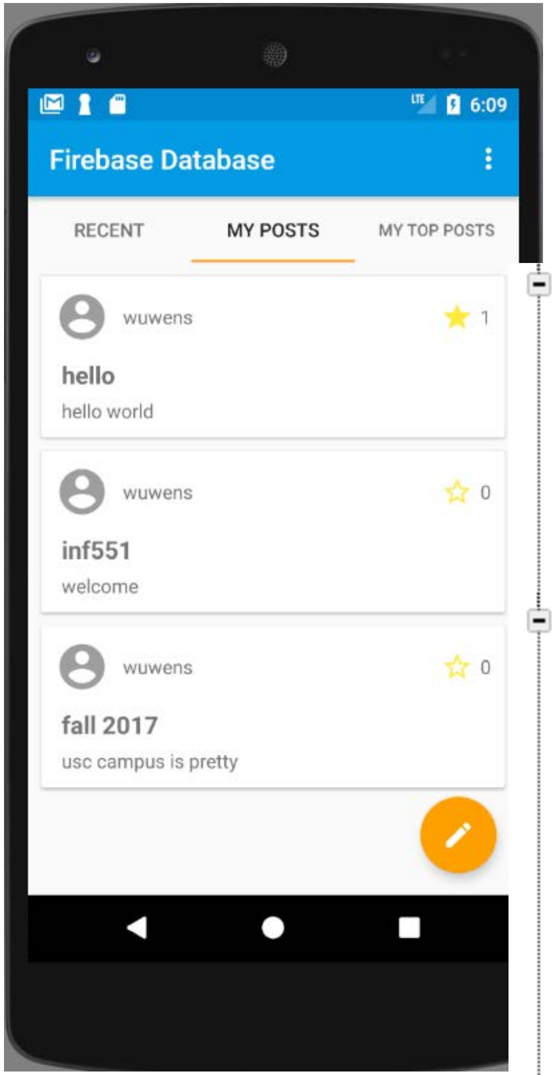


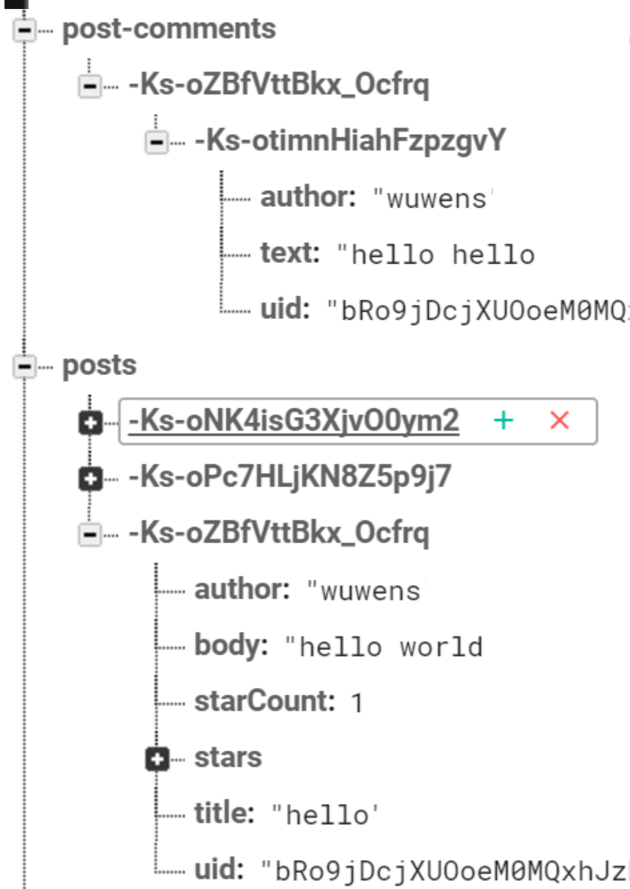
May add new attributes



	Author	Title	ISBN
	Bill Clinton	My Life	1234567890

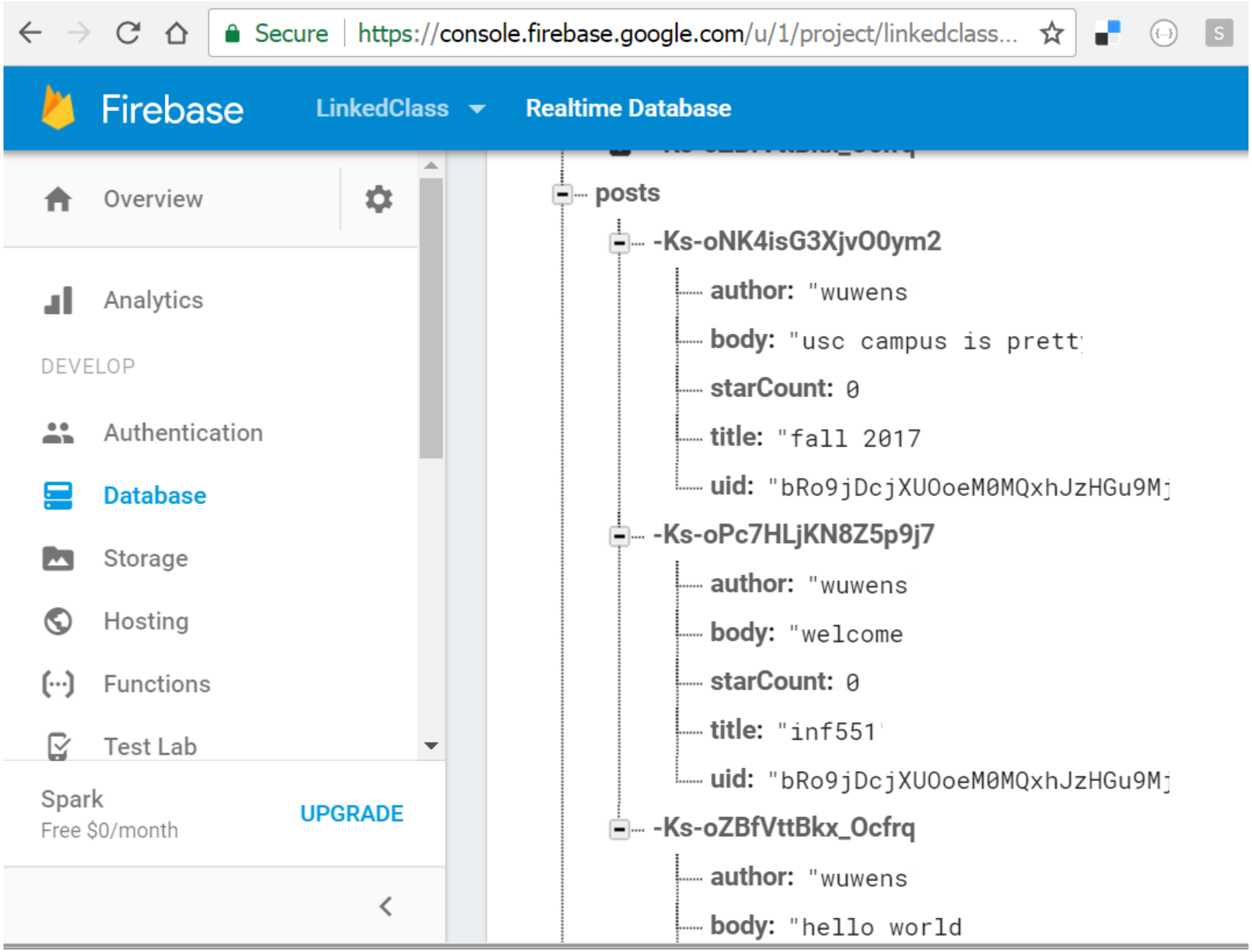
Firebase: a cloud database






```
"post-comments" : {
  "-Ks-oZBfVttBkx_Ocfrq" : {
    "-Ks-otimnHiahFzpzgvY" : {
      "author" : "wuwens",
      "text" : "hello hello",
      "uid" : "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2"
    }
  }
},
"posts" : {
  "-Ks-oNK4isG3XjvO0ym2" : {
    "author" : "wuwens",
    "body" : "usc campus is pretty",
    "starCount" : 0,
    "title" : "fall 2017",
    "uid" : "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2"
  },
  "-Ks-oPc7HLjKN8Z5p9j7" : {
    "author" : "wuwens",
    "body" : "welcome",
    "starCount" : 0,
    "title" : "inf551",
    "uid" : "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2"
  },
  "-Ks-oZBfVttBkx_Ocfrq" : {
    "author" : "wuwens",
    "body" : "hello world",
    "starCount" : 1,
    "stars" : {
      "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2" : true
    },
    "title" : "hello",
    "uid" : "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2"
  }
},
```

Firebase


Secure
https://console.firebase.google.com/u/1/project/linkedclass...
Firebase
LinkedClass
Realtime Database
Overview
Analytics
DEVELOP
Authentication
Database
Storage
Hosting
Functions
Test Lab
Spark
Free $0/month
UPGRADE
posts
-Ks-oNK4isG3XjvO0ym2
author: "wuwens
body: "usc campus is prett
starCount: 0
title: "fall 2017
uid: "bRo9jDcjXUOoeM0MQxhJzHGu9Mj
-Ks-oPc7HLjKN8Z5p9j7
author: "wuwens
body: "welcome
starCount: 0
title: "inf551
uid: "bRo9jDcjXUOoeM0MQxhJzHGu9Mj
-Ks-oZBfVttBkx_Ocfrq
author: "wuwens
body: "hello world

Topics

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

Roots in functional programming

- Functional programming languages:
  - Python, Lisp (list processor), Scheme, Erlang, Haskell
- Two functions:
  - Map: mapping a list => list
  - Reduce: reducing a list => value
- map() and reduce() in Python
  - https://docs.python.org/2/library/functions.html#map

map() and reduce() in Python

- list = [1, 2, 3]
- def sqr(x): return x ** 2
- list1 = map(sqr, list)

What are the value of list1 and z?

- def add(x, y): return x + y
- z = reduce(add, list)

Lambda function

- Anonymous function (not bound to a name)
- list = [1, 2, 3]
- list1 = map(lambda x: x ** 2, list)
- z = reduce(lambda x, y: x + y, list)

How is reduce() in Python evaluated?

- z = reduce(f, list) where f is add function

- Initially, z (an accumulator) is set to list[0]
- Next, repeat z = add(z, list[i]) for each i > 0
- Return final z

- Example: z = reduce(add, [1, 2, 3])
  - i = 0, z = 1; i = 1, z = 3; i = 2, z = 6

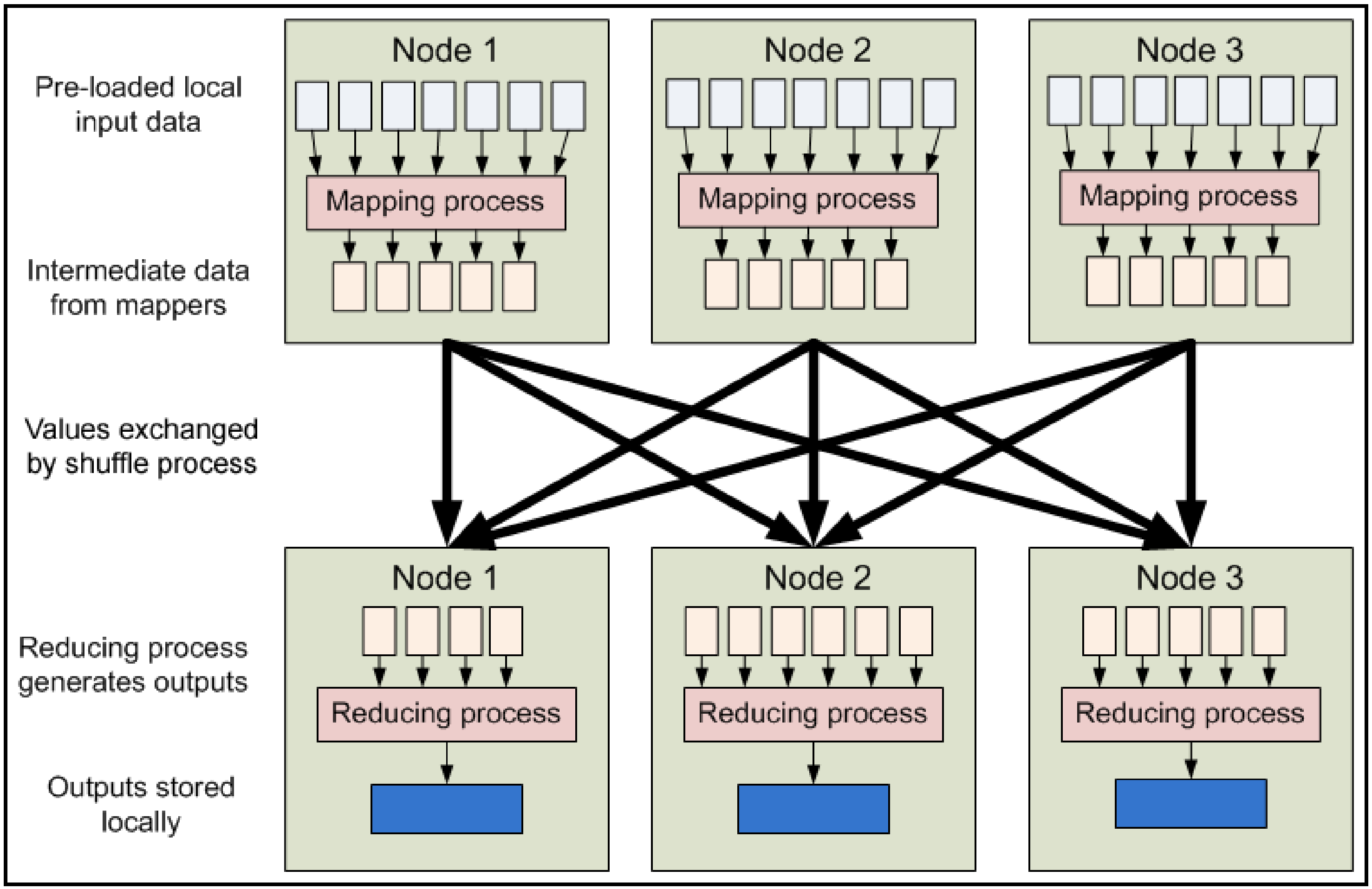
Hadoop MapReduce

- 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

WordCount: mapper

Object can be replaced with LongWritable

Data types of input key-value

Data types of output key-value

```
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
}
```

Key-value pairs with specified data types

WordCount: reducer

Data types of input key-value

Data types of output key-value

```
public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
  }
}
```

A list of values

Characteristics of Hadoop

- Acyclic data flow model
  - Data loaded from stable storage (e.g., HDFS)
  - Processed through a sequence of steps
  - Results written to disk

- Batch processing
  - No interactions permitted during processing

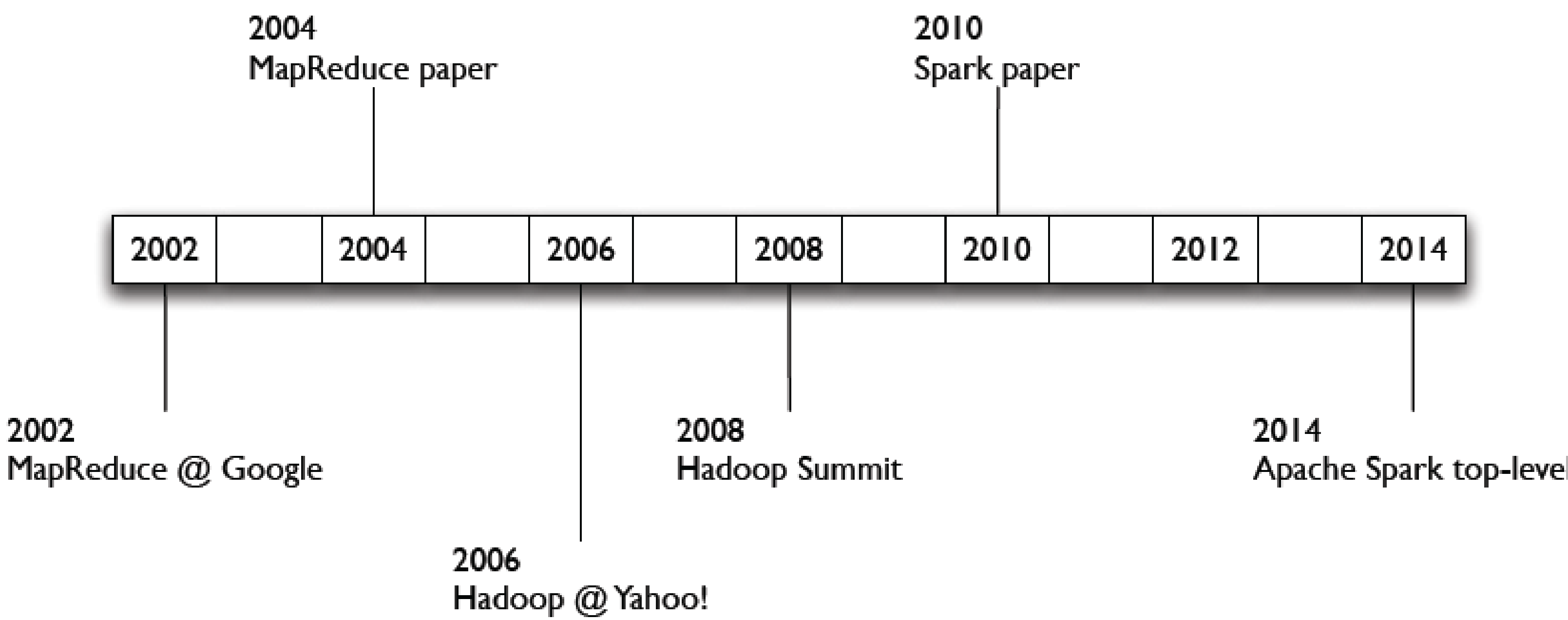
Problems

- Ill-suited for iterative algorithms that requires repeated reuse of data
  - E.g., machine learning and data mining algorithms such as k-means, PageRank, logistic regression
- Ill-suited for interactive exploration of data
  - E.g., OLAP on big data

In-memory MapReduce (Spark)

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

Apache Spark: history



Spark

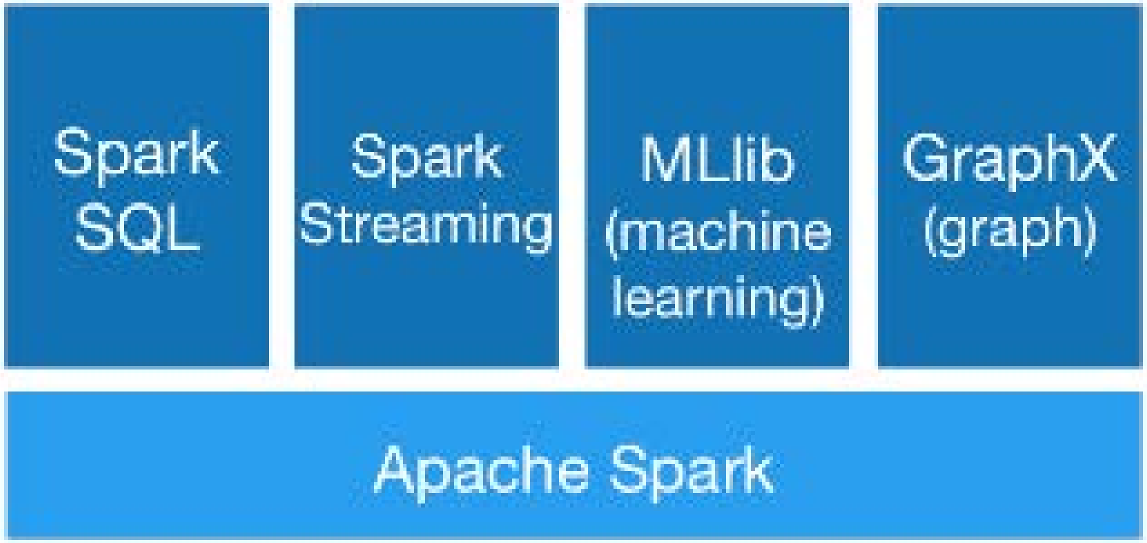
- Support working sets through RDD
  - Enabling reuse & fault-tolerance

- 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 (if time permits)

Data warehousing

- Multidimensional data model
  - Star vs snowflake schema

- OLAP operations: rollup, drill-down, etc.

- Materialized views

- Index (we will cover this if time permits)
  - Bitmap
  - Run-length encoding
  - Join index

What is a Warehouse?

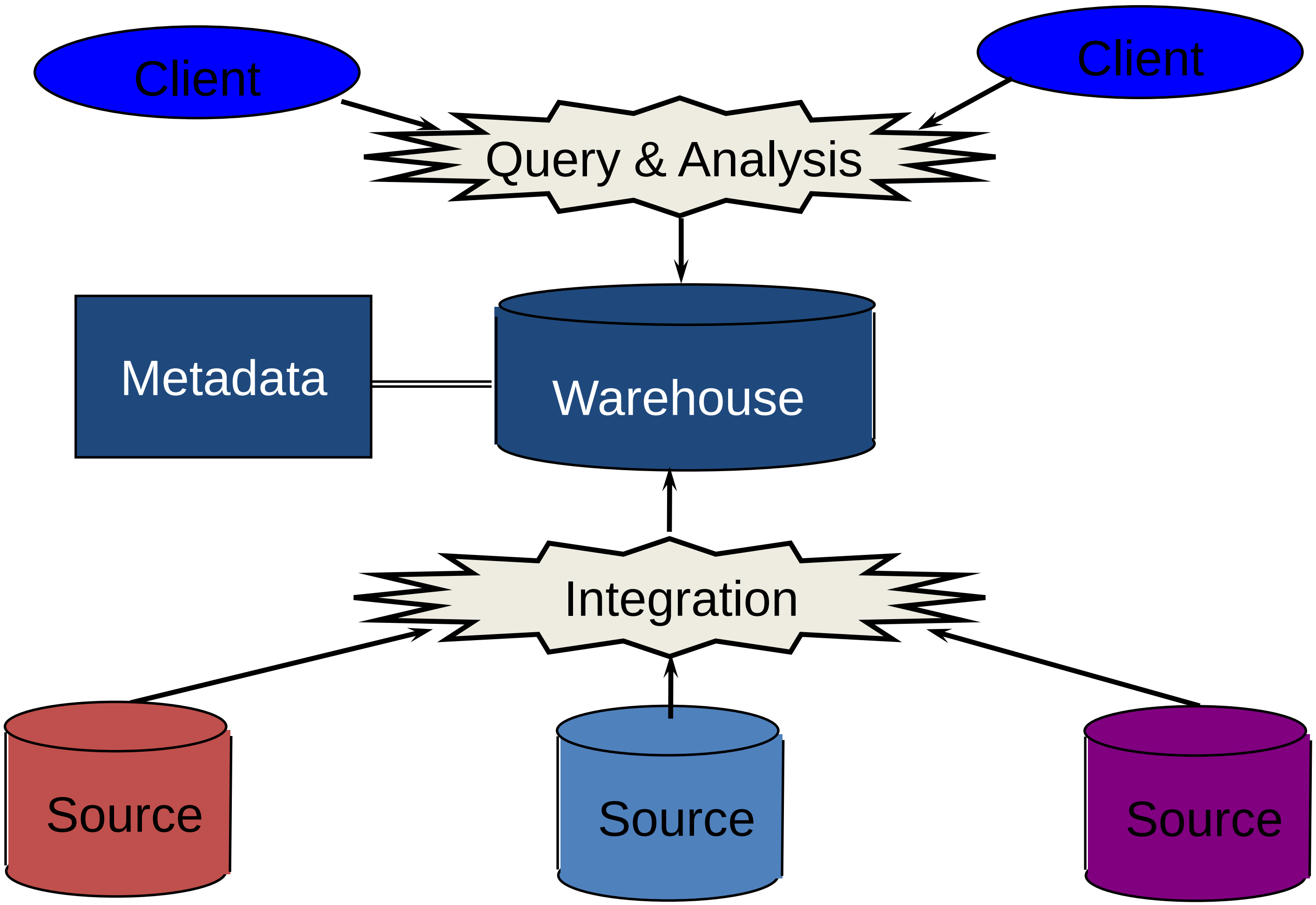
- Collection of diverse data
  - subject oriented, e.g., sales
  - aimed at executive, decision maker
  - often a copy of operational data
  - with value-added data (e.g., summaries, history)
  - integrated
  - time-varying: historical data, discovering trend
  - non-volatile: once in warehouse, data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

more

What is a Warehouse?

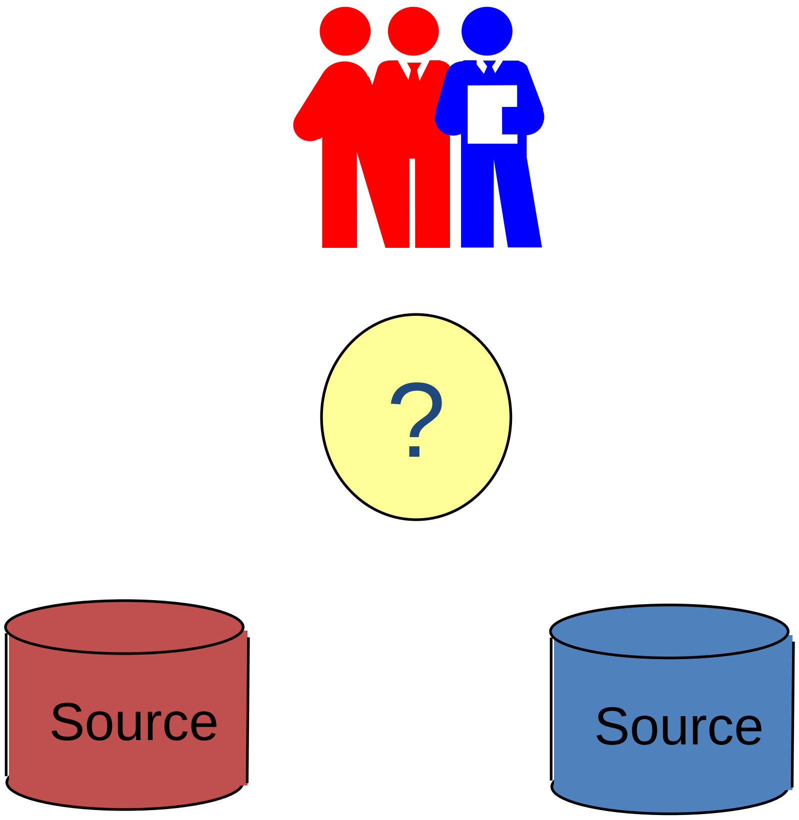
- Collection of tools
  - gathering data
  - cleansing, integrating, ...
  - querying, reporting, analysis
  - data mining
  - monitoring, administering warehouse

Warehouse Architecture

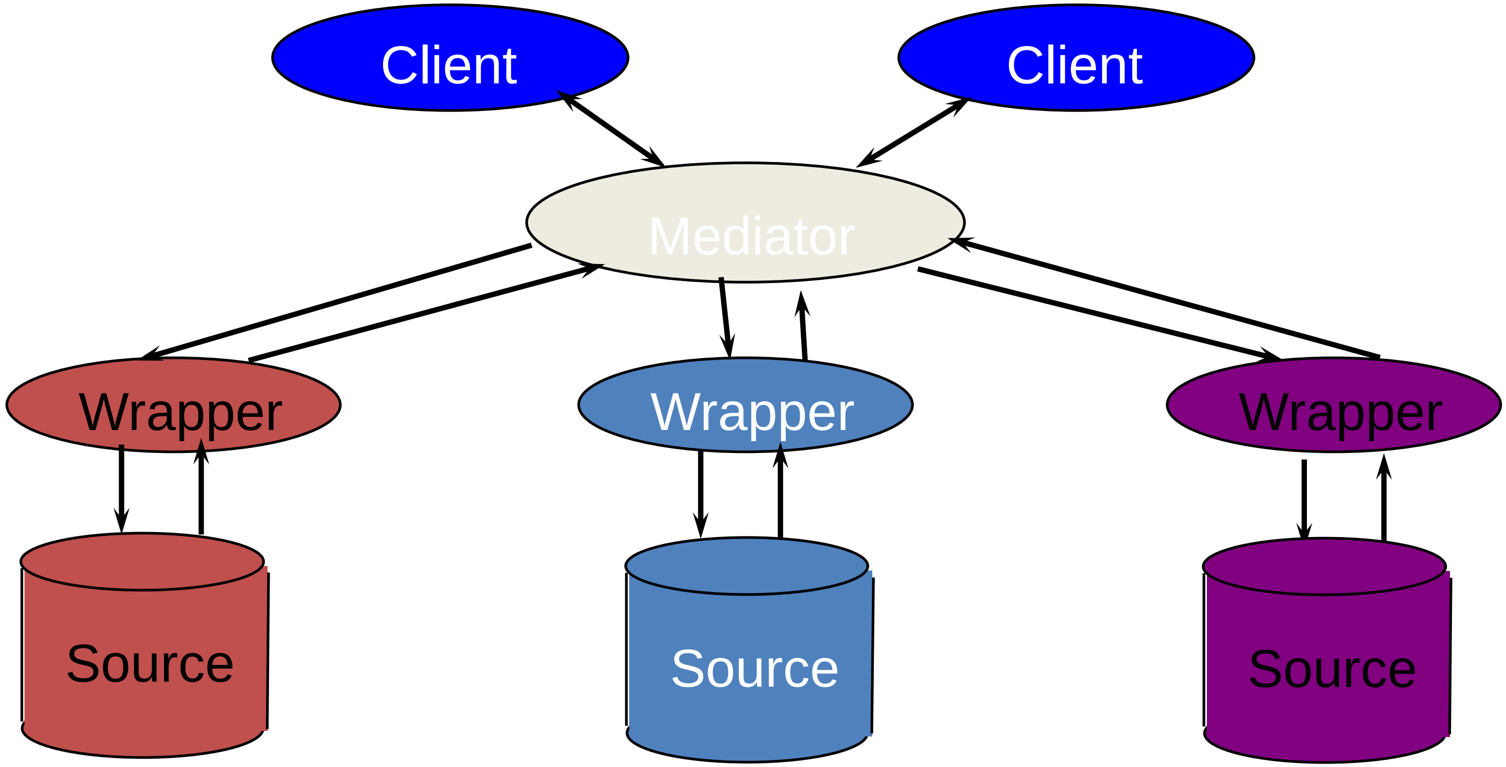


Why a Warehouse?

- 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

- May require hefty storage space
- Need to 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

Lab session

- Task: Setting up an EC2 instance

- Details: see lab session slides to be posted…