

Introduction

DSCI 551

Wensheng Wu

Logistics

- Instructor email: wenshenw@usc.edu
 - please indicate section: 551 in the subject line.
- Class meeting times:
 - see syllabus
- Office hours:
 - After class
 - 4-5pm Monday, by appointment, online

Logistics

- Graders
 - check out announcements
- Class materials
 - Posted on course web site
- AWS EC2 (elastic compute cloud)

Piazza

- 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
- You should have received invitation to sign up

Prerequisites

- Programming skills:
 - Python (homework, Spark), Java (e.g., for Hadoop only)
 - Scala
- Unix-like environment & shell commands (ls?)
 - E.g., Amazon EC2
- Basic knowledge of algorithms and data structures
 - Sorting, hashing, etc. (CS 570)//merge sort?
 - 3, 2, 1, 4, 6, 5
 - => 1, 2, 3 4, 5, 6 (runs)
 - merge => 1, 2, 3, 4, 5, 6
 - $h(k) \Rightarrow$ if k is even, send (k,v) to R_0 ; otherwise, send to R_1
 - $3\%2 = 1$
 - $2\%2 = 0$
 - I/O
 - 1TB (data on SSD) 1GB main memory) => runs
 - I/O
- Basic probability and statistics

notes

- $h(x) = x \% 2$
- $h(3) = 1$
- $h(4) = 0$
- $h('john') = 0$
- $h('bill') = 1$

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>
- See four more books in syllabus

Additional readings

- Links can be found in Syllabus
 - Check out the schedule

Grading structure

• Homework	20%
• Midterm 1	15%
• Midterm 2	15%
• Comprehensive	25%
• Lab tasks	5%
• Group project	20%

Grading scale

- [94, 100] = A
- [90, 94) = A-
- [87, 90) = B+
- [83, 87) = B
- [80, 83) = B-
- [77, 80) = C+
- [73, 77) = C
- ... (see Syllabus for complete breakdown)

Lab tasks

- Four tasks:
 - EC2, HDFS, MongoDB, DynamoDB

Exams

- Closed-notes & book
- Midterms:
 - See syllabus for time
- Final:
 - See syllabus for time

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

Course project

- Details to be posted
- Done in phases
 - Proposal
 - Midterm report
 - Final report

Late Policy

- No LATE submissions will be accepted
- Make up for tests are permitted only when
 - You have an emergency, typically medical
 - Let me know at least two weeks in advance
 - You may be required to contact Campus Support and Intervention office for verification of emergency

Grading Corrections

- All coursework's grades are final one week after grades are posted
- Final exam grades (& all grades) are final after final exam regrading deadline (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

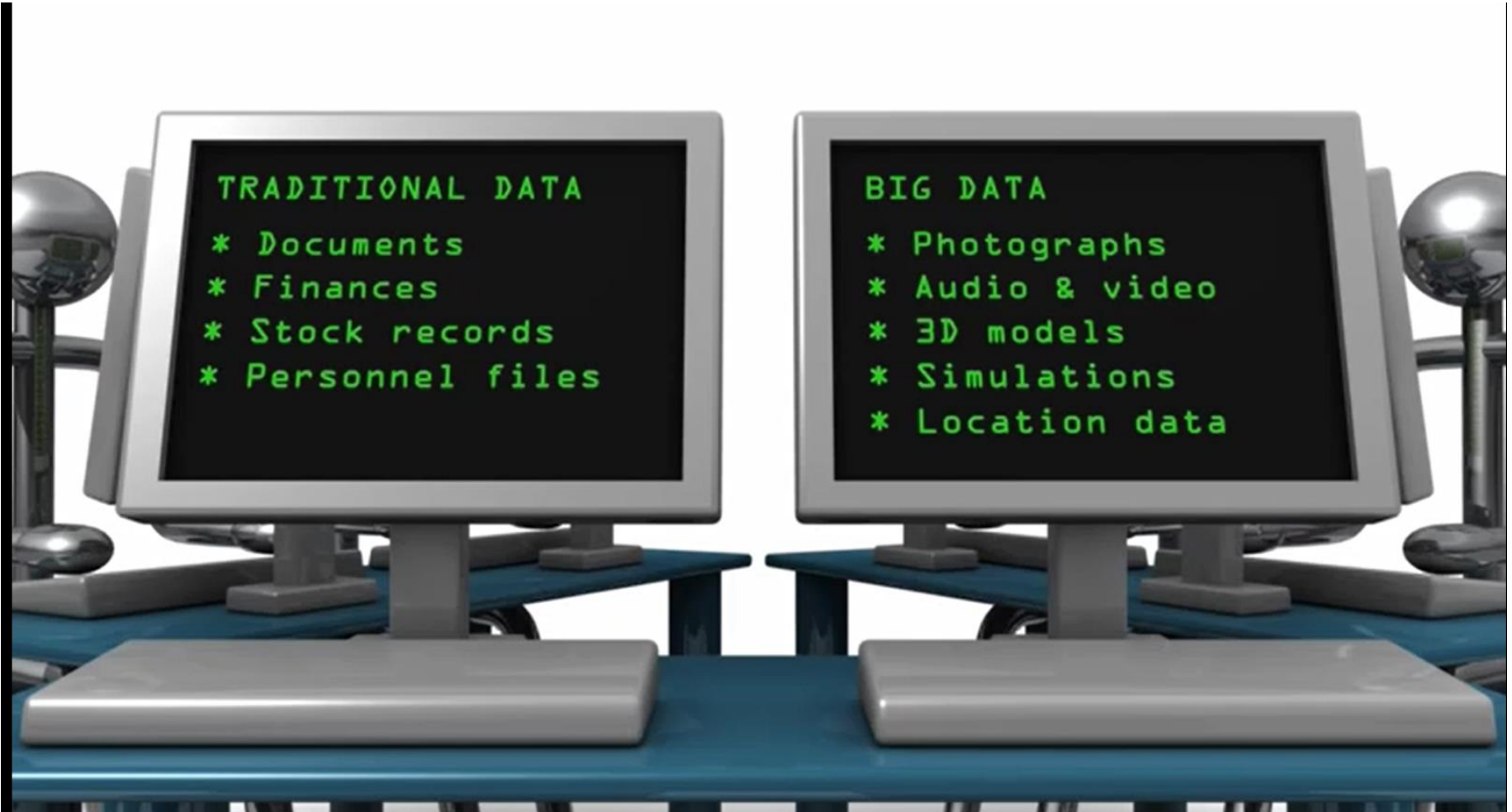
Academic Integrity

- **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 does it have? **3Vs?**
 - What big data technologies were mentioned?
 - Hadoop: HDFS and MapReduce

Variety



Internet Traffic in 2012

- 4.8 zettabyte = 4.8 billion terabytes
- Zettabyte (1000 exabytes)
- Exabyte
- Petabyte
- Terabyte = 2^{40} (storage)
 - 1TB = 1024 (2^{10}) GB
 - $2^2 = 4$, $2^3 = 8$, $2^7 = 128$
- Gigabyte = 2^{30} (memory)
- Megabyte (128MB, HDFS)
 - 1MB = $2^{20} = 2^{10} * 2^{10}$
- Kilobyte = 2^{10} (1KB) = 1024B // $2^5 = 32$

Main memory:

12GB

SSD:

1TB

123 (decimal) = $1 * 10^2 + 2 * 10^1 + 3 * 10^0$

111 (binary) = $1 * 2^2 + 1 * 2^1 + 1 * 2^0$

111 + 1 (binary) = 1000 = 8 (decimal)

001

==

1000

11 = $1 * 2^1 + 1 * 2^0 = 3$

100 (binary) = $1 * 2^2 = 4$

100 (decimal) = $1 * 10^2$

Notes

- main memory volatile
- export data from MySQL into CSV/JSON format
- apache Hive – HQL
- very structured – relations (data in MySQL)
- semi-structured (JSON/XML)
- unstructured (texts) NLP

Major topics

- Storage systems



- File systems & file formats
- Database management systems (RDBMS)
 - R = relational
- Big data solution stack

Storage Systems

- Hard disk
- SSD (Solid state drive)

4KB = block size for HDD

128MB = block size in HDFS

$128\text{MB}/4\text{KB} = 32\text{K}$



Internal of hard disk

Actuator

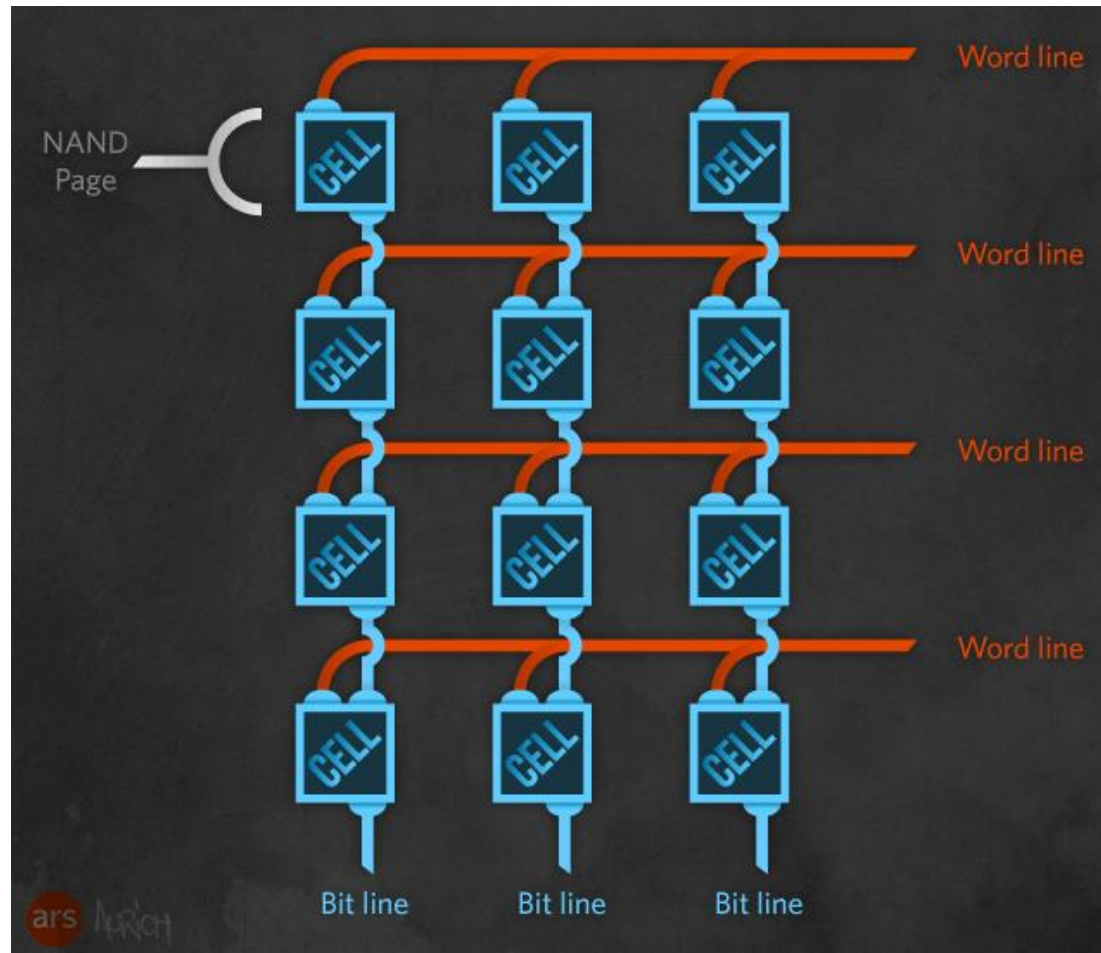
Spindle



Platter

Disk head


NAND flash



Latencies: read, write, and erase

	SLC	MLC	TLC	HDD	RAM
P/E cycles	100k	10k	5k	*	*
Bits per cell	1	2	3	*	*
Seek latency (μ s)	*	*	*	9000	*
Read latency (μ s)	25	50	100	2000-7000	0.04-0.1
Write latency (μ s)	250	900	1500	2000-7000	0.04-0.1
Erase latency (μ s)	1500	3000	5000	*	*
<i>Notes</i>	* metric is not applicable for that type of memory				
<i>Sources</i>	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

File Systems

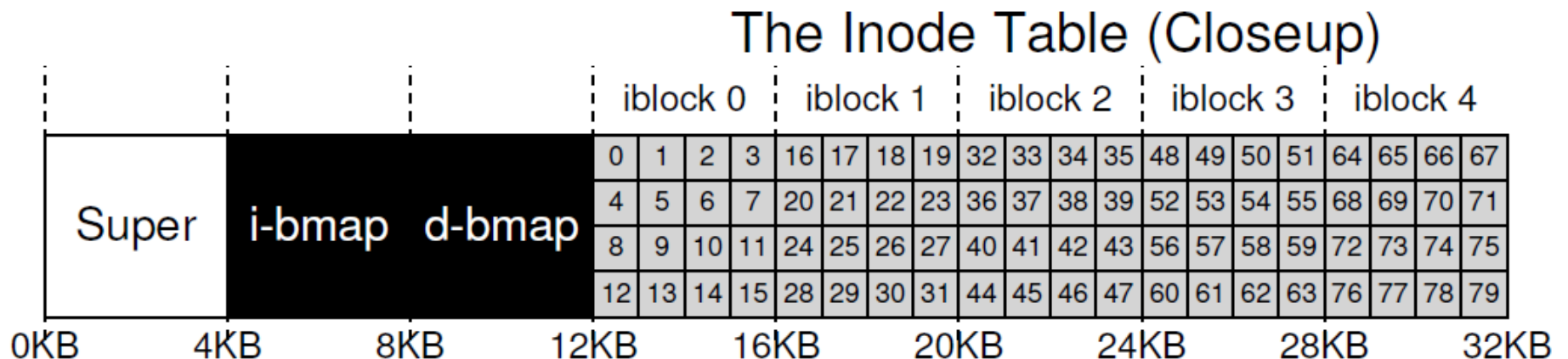
- Standalone
 - Single machine
- Distributed (e.g., Hadoop)
 - A number of data servers

Standalone file systems

- Data structures
 - Data blocks
 - Metadata blocks (Inodes)
 - Bitmap blocks (for space allocation)
- Access paths
 - Read a file
 - Write a file

Inode (index node)

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



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

NameNode:

Stores metadata only

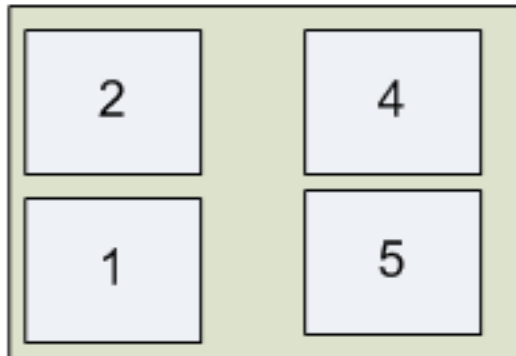
METADATA:

/user/aaron/foo → 1, 2, 4

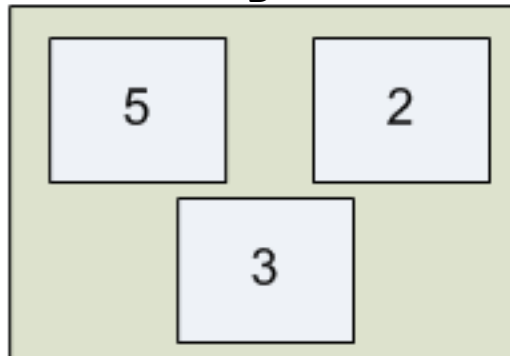
/user/aaron/bar → 3, 5

DataNodes: Store blocks from files

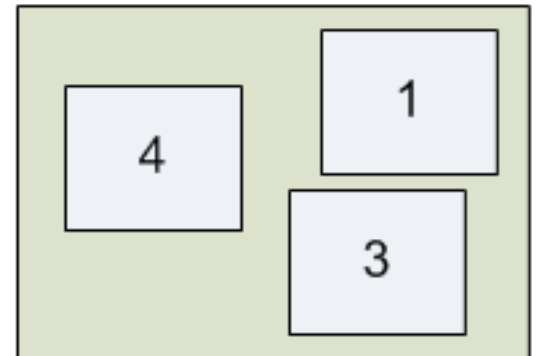
A



B



C



File system image in namenode

```
</LastEncodingSection>
▼<INodeSection>
  <lastInodeId>16422</lastInodeId>
  <numInodes>38</numInodes>
  ▼<inode>
    <id>16385</id>
    <type>DIRECTORY</type>
    <name/>
    <mtime>1581231015982</mtime>
    <permission>ec2-user:supergroup:0755</permission>
    <nsquota>9223372036854775807</nsquota>
    <dsquota>-1</dsquota>
  </inode>
  ▼<inode>
    <id>16386</id>
    <type>DIRECTORY</type>
    <name>user</name>
    <mtime>1581231034866</mtime>
    <permission>ec2-user:supergroup:0755</permission>
    <nsquota>-1</nsquota>
    <dsquota>-1</dsquota>
  </inode>
```

Directory section

```
</SnapshotSection>
▼<INodeDirectorySection>
  ▼<directory>
    <parent>16385</parent>
    <child>16386</child>
  </directory>
  ▼<directory>
    <parent>16386</parent>
    <child>16387</child>
  </directory>
  ▼<directory>
    <parent>16387</parent>
    <child>16390</child>
    <child>16412</child>
    <child>16401</child>
    <child>16391</child>
    <child>16388</child>
  </directory>
  ▼<directory>
    <parent>16388</parent>
    <child>16389</child>
  </directory>
```

Major topics

- Storage systems
- File systems & **file formats**
- Database management systems
- 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

 Addison Wesley, 1995

<p> <i> Data on the Web </i>

Abiteoul, Buneman, Suciu

 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 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>
        </activity>
    </application>
</manifest>
```

<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>

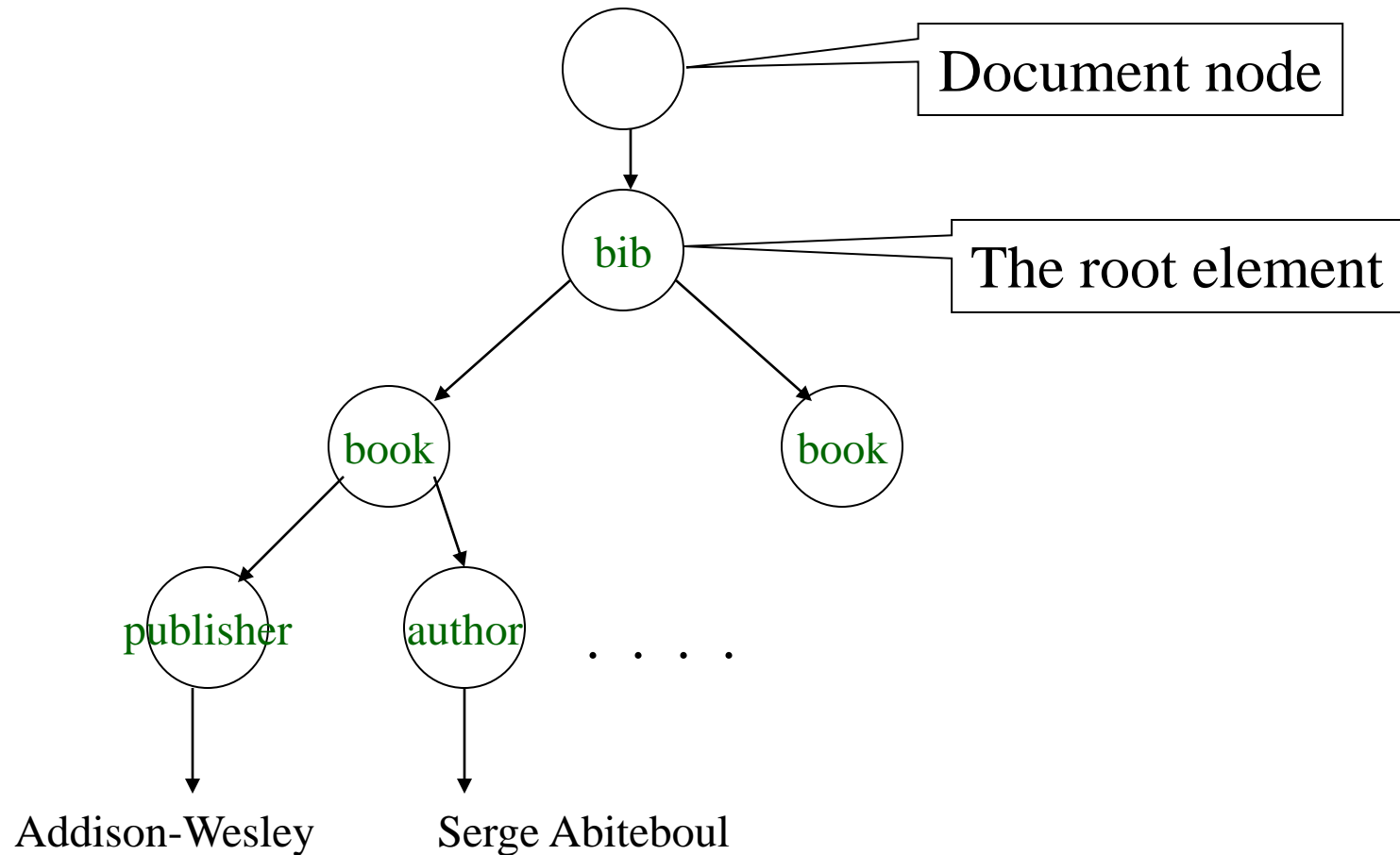
<year>1998</year>

</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



Relational DBMS

- Data models
 - E (entity set) R
 - Relational (redundancy => update anomaly)
- Schema
 - describes the structure of data
 - including constraints

RDBMS

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

RDBMS

- Query execution algorithms
 - External sorting
 - One-pass algorithms
 - Nested-loop join, sorting, hashing-based
 - 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
 - ACID vs CAP

RDBMS







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

RDBMS Examples

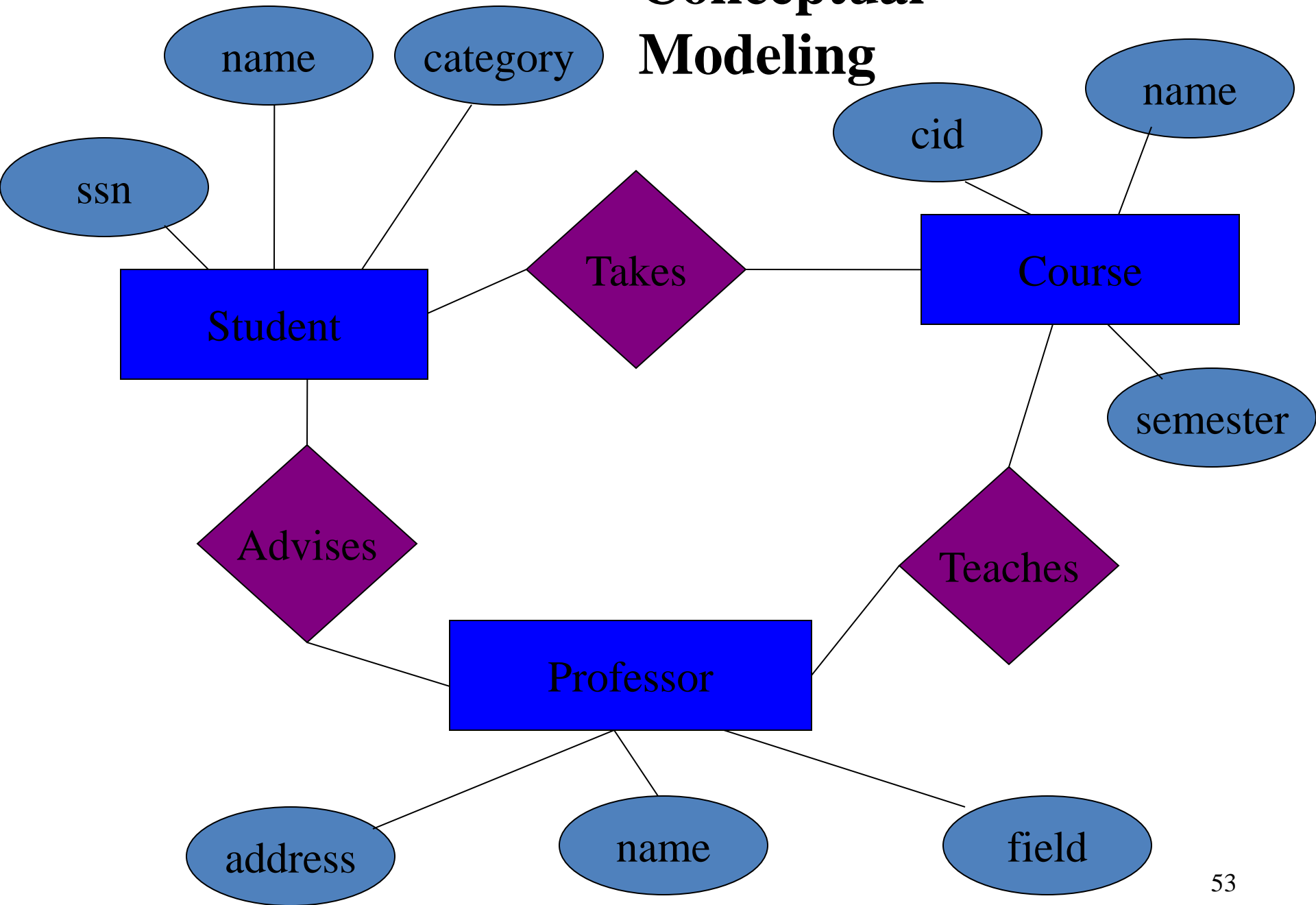
- MySQL (can be installed in Amazon AWS EC2)
- Amazon RDS (Relational database as a service)
 - DBMS in the cloud
 - Database as a service
- Data warehouse on RDBMS
 - OLAP

Amazon RDS: Database-as-a-service

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

	<h2>MySQL</h2> <p>MySQL Community Edition</p> <p>MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.</p> <ul style="list-style-type: none">• Supports database size up to 6 TB.• Instances offer up to 32 vCPUs and 244 GiB Memory.• Supports automated backup and point-in-time recovery.• Supports cross-region read replicas.	Select
		
		
		
		
		

Conceptual Modeling



Schema Design and Implementation

- Tables (relations):

Students:

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

Takes:

SSN	CID
123-45-6789	CSE444
123-45-6789	CSE541
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(uey) L(anguage)
 - clause

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

Select A's ,agg
From R's
Where C's
Group by A's
Having
Order by
Limit ?
Offset ?
(pagination)

===
Insert
Update
Delete

Declarative (what)

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

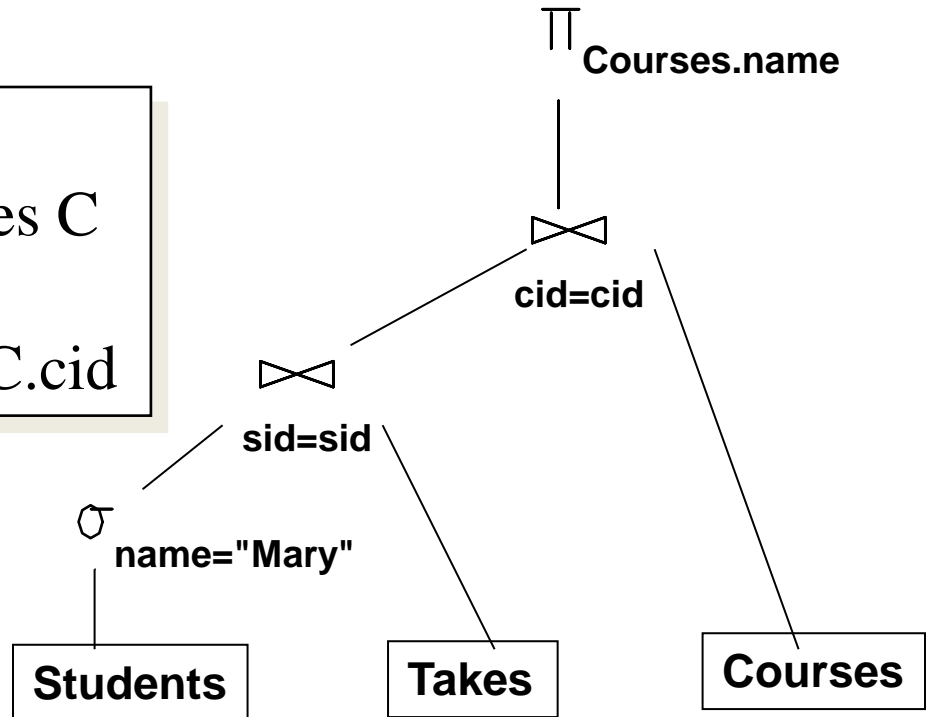
Query Optimization

Goal:

Declarative SQL query • \longrightarrow *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
```

filtering
projection



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 

Topics

- Big data management & analytics
 - Cloud data storage (Amazon S3)
 - NoSQL
 - Google **Firestore** (real-time database, ...)
 - **MongoDB** (shell, mongo)
 - Amazon **DynamoDB** (row store, key-value)
 - Cassandra (not required)
 - 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

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

The screenshot shows a file management interface with the following elements:

- Buttons:** 'Upload' (blue), 'Create Folder' (grey), and 'Actions' (grey with a dropdown arrow).
- Search:** A search bar with a magnifying glass icon and the text 'Search by'.
- Breadcrumbs:** 'All Buckets / inf551'.
- Table:** A table with columns 'Name' and 'Size'. It contains two rows:
 - Desert.jpg:** A file icon, the name 'Desert.jpg', and a size of '826.1 KB'.
 - s3:** A folder icon and the name 's3', with a size of '--'.
- Context Menu:** A dark grey menu is open over the 'Desert.jpg' row, listing the following actions:
 - Open
 - Download
 - Create Folder...
 - Upload** (highlighted in orange)
 - Make Public
 - Rename
 - Delete
 - Initiate Restore
 - Cut
 - Copy
 - Paste
 - Properties



<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 (CAP)
- Weaker (e.g., eventual) consistency model

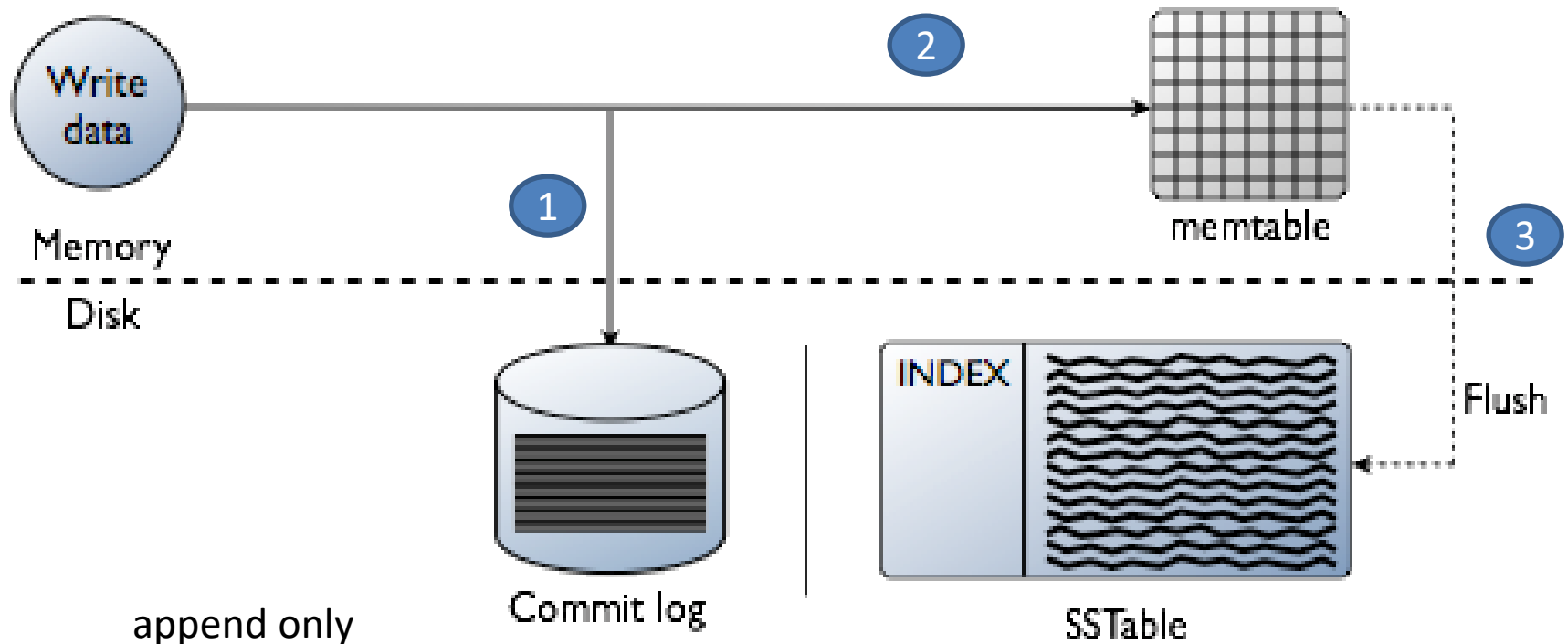
Example NoSQL databases

- MongoDB, Firebase, etc.
 - Manage JSON documents
- Amazon DynamoDB
 - Row store
 - row = item = a collection of key-value pairs
- Apache Cassandra (not required)
 - Wide column store
 - Google's Bigtable clone
- Neo4J...

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>:

Amazon DynamoDB

Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, IoT, and many other applications.

Create table

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

DynamoDB

Dashboard

Tables

Reserved capacity

Create tableActions ▾

Filter by table name X

	Name
<input checked="" type="radio"/>	Books

OverviewItemsMetricsAlarms

Create itemActions ▾

Scan: [Table] Books: Author, Title ^

Scan ▾

[Table] Books: Author, Title

+ Add filter

Start search

<

<input type="checkbox"/>	Author	Title	
--------------------------	--------	-------	--

May add new attributes

Tree ▾

Item {3}

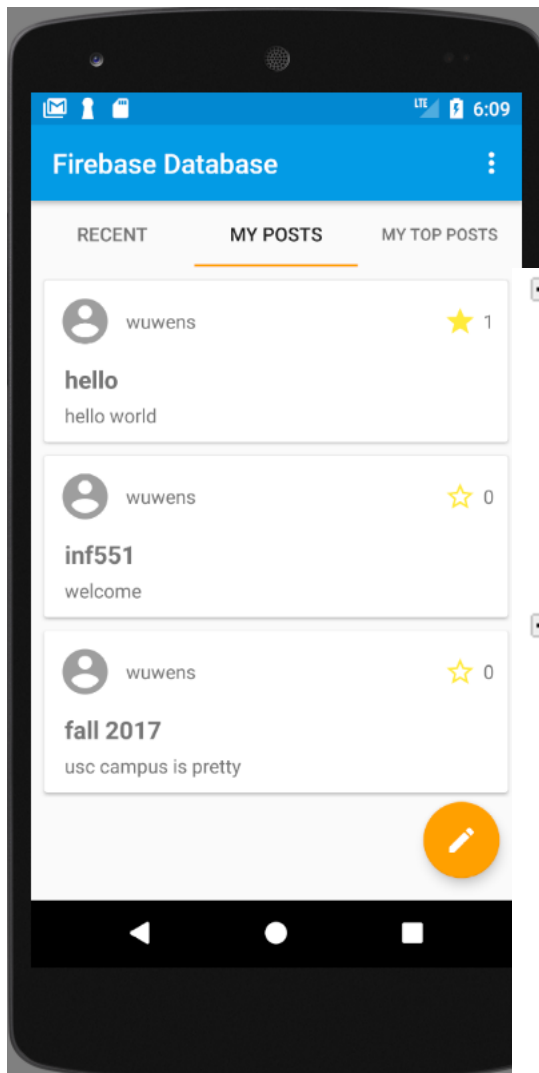
- Author String : Bill Clinton
- Title String : My Life
- ISBN String : 1234567890

+ Append ▾

- String
- Binary
- Number
- StringSet
- NumberSet
- BinarySet
- Map
- List

<input type="checkbox"/>	Author	Title	ISBN
<input type="checkbox"/>	Bill Clinton	My Life	1234567890

Firestore: a cloud database



post-comments

-Ks-oZBfVttBkx_Ocfrq

-Ks-otimnHiahFzpzgvY

author: "uwens"

text: "hello hello"

uid: "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2"

posts

+ -Ks-oNK4isG3XjvO0ym2 + -

+ -Ks-oPc7HLjKN8Z5p9j7

-Ks-oZBfVttBkx_Ocfrq

author: "uwens"

body: "hello world"

starCount: 1

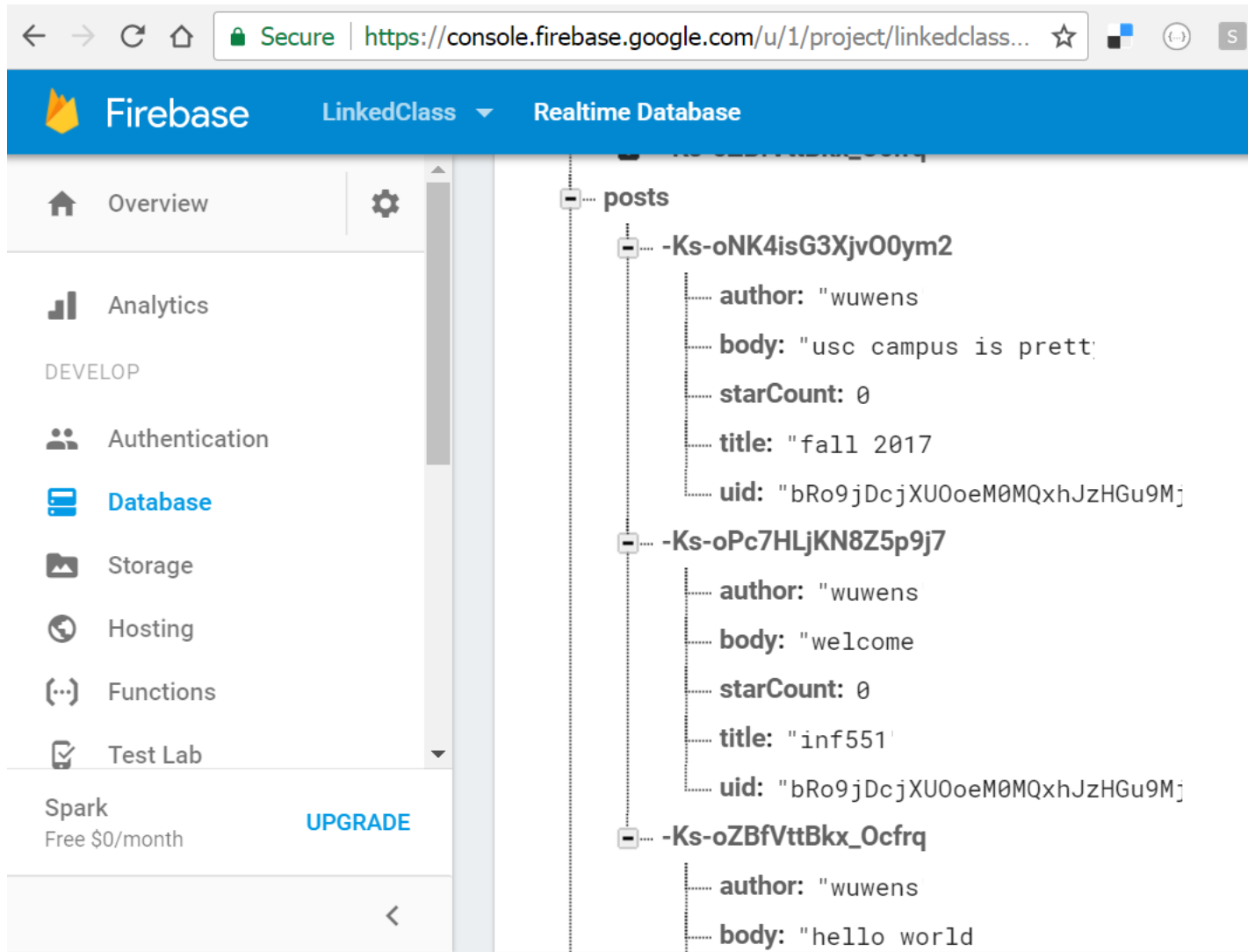
+ stars

title: "hello"

uid: "bRo9jDcjXUOoeM0MQxhJzHGu9Mj2"

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


Firebase



The screenshot shows the Firebase Realtime Database console in a web browser. The address bar indicates the URL is <https://console.firebase.google.com/u/1/project/linkedclass...>. The page header features the Firebase logo, the project name "LinkedClass", and the selected database "Realtime Database".

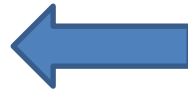
The left sidebar contains navigation links: Overview, Analytics, DEVELOP, Authentication, Database (highlighted), Storage, Hosting, Functions, and Test Lab. At the bottom of the sidebar, there is a "Spark" section indicating "Free \$0/month" and an "UPGRADE" button.

The main content area displays a JSON tree structure for the "posts" node. It contains three entries, each with a unique key and the following fields: author, body, starCount, title, and uid.

```
posts
├── -Ks-oNK4isG3Xjv00ym2
│   ├── author: "wuwens"
│   ├── body: "usc campus is prett"
│   ├── starCount: 0
│   ├── title: "fa11 2017"
│   └── uid: "bRo9jDcjXU0oeM0MQxhJzHGu9Mj"
├── -Ks-oPc7HLjKN8Z5p9j7
│   ├── author: "wuwens"
│   ├── body: "welcome"
│   ├── starCount: 0
│   ├── title: "inf551"
│   └── uid: "bRo9jDcjXU0oeM0MQxhJzHGu9Mj"
└── -Ks-oZBfVtBkx_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



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)`

`reduce()` is in `functools` module of Python 3

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)`

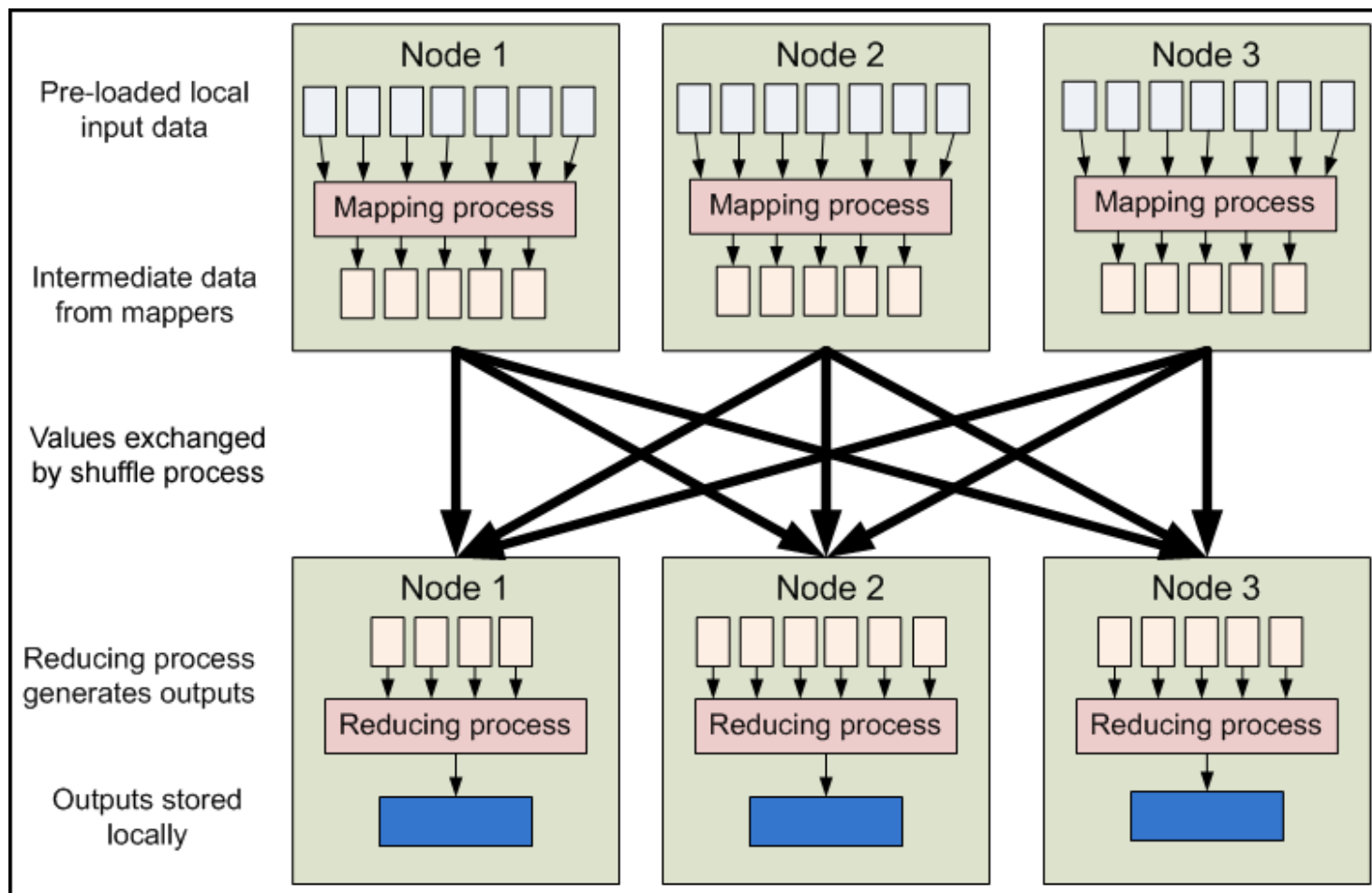
How is reduce() in Python evaluated?

- $z = \text{reduce}(f, \text{list})$ where f is add function
- Initially, z (an accumulator) is set to $\text{list}[0]$
- Next, repeat $z = \text{add}(z, \text{list}[i])$ for each $i > 0$
- Return final z
- Example: $z = \text{reduce}(\text{add}, [1, 2, 3])$
 - $i = 0, z = 1; i = 1, z = 3; i = 2, z = 6$

Hadoop MapReduce

- Map
 - $\langle k, v \rangle \Rightarrow$ list of $\langle k', v' \rangle$
- Reduce:
 - $\langle k', \text{list of } v' \rangle \Rightarrow$ list of $\langle k'', v'' \rangle$
- Write MapReduce programs on Hadoop
 - Using Java

MapReduce



WordCount: mapper

Object can be replaced with LongWritable

```
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);  
            }  
        }  
    }  
}
```

Data types of input key-value

Data types of output key-value

Key-value pairs with specified data types

WordCount: reducer

```
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);
    }
}
```

Data types of input key-value

Data types of output key-value

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

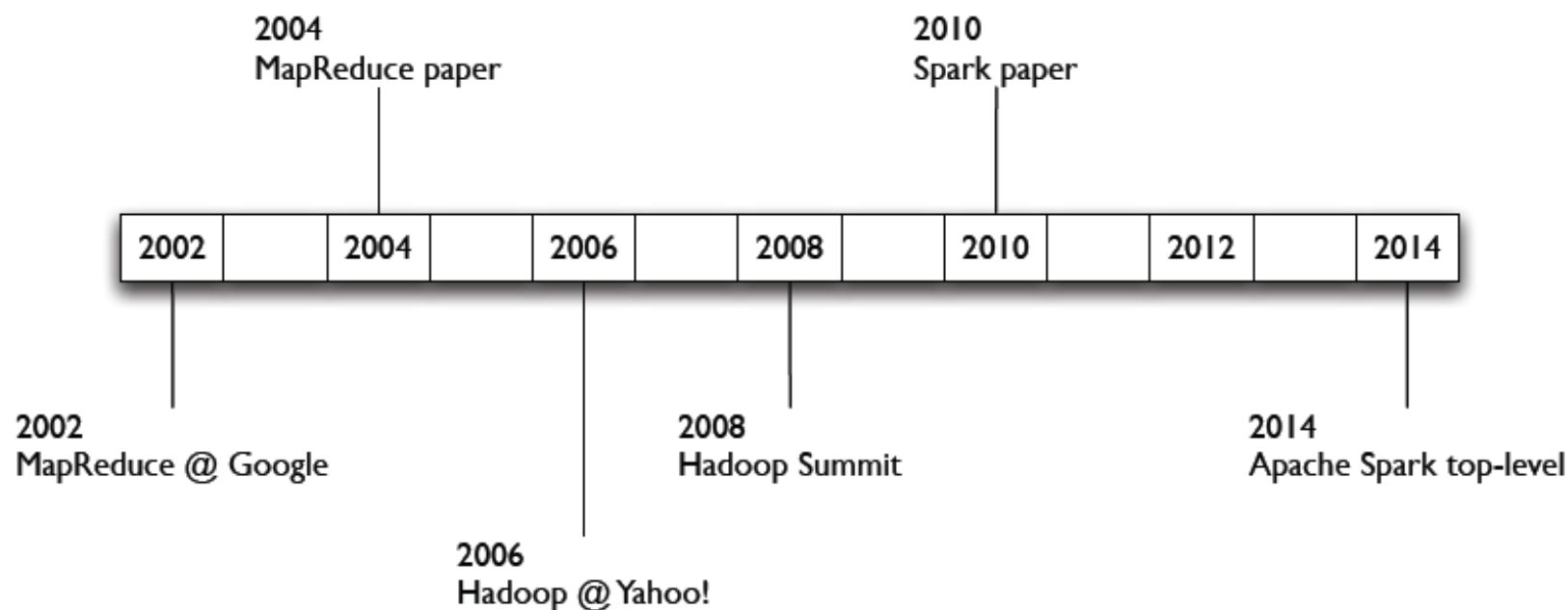
Problems

- 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

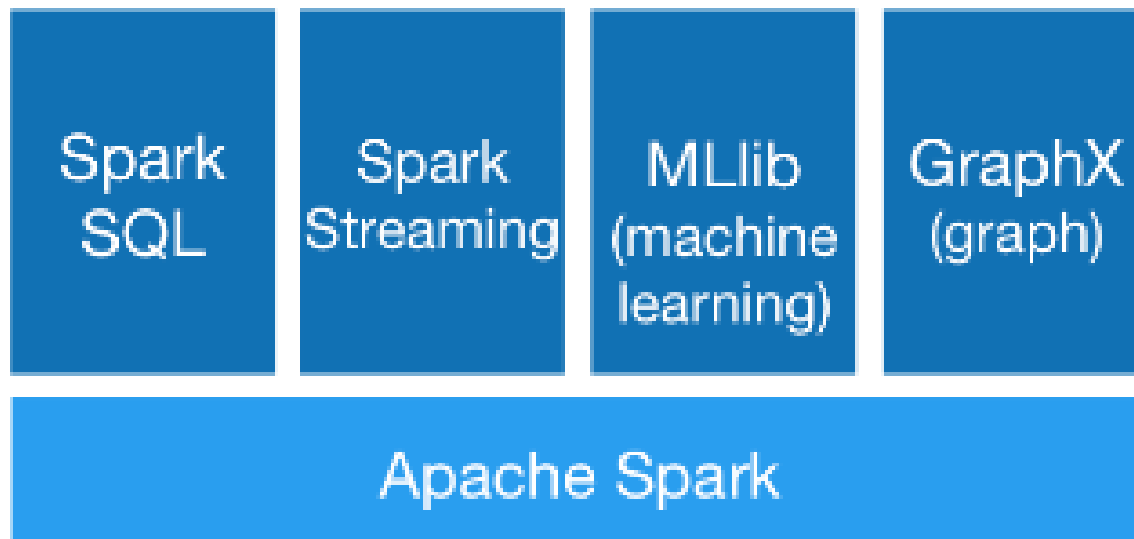


Spark

- 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
- We will see **DataFrame** in Spark too



Spark

- Run on Hadoop, Cassandra, HBase, etc.



wc.py

```
from pyspark import SparkContext
from operator import add

sc = SparkContext(appName="dsci551")

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(v[0], v[1])
```

Coming up...

- Task: Setting up an EC2 instance
- Details:
 - see posted instructions and come to class!

Resources

- Merge sort:
 - <https://www.interviewbit.com/tutorial/merge-sort-algorithm/>
 - <https://www.youtube.com/watch?v=Nso25TkBsYI>
- Hashing
 - https://www.tutorialspoint.com/python_data_structure/python_hash_table.htm
 - <https://www.programiz.com/python-programming/methods/built-in/hash>