

CS162 Operating Systems and Systems Programming Lecture 24

Capstone: Cloud Computing

April 29, 2013
Anthony D. Joseph
<http://inst.eecs.berkeley.edu/~cs162>

Goals for Today

- Distributed systems
- Cloud Computing programming paradigms
- Cloud Computing OS

Note: Some slides and/or pictures in the following are adapted from slides Ali Ghodsi.

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

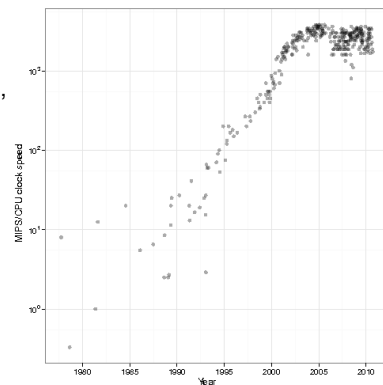
24.2

Background of Cloud Computing

- 1990: Heyday of parallel computing, multi-processors
 - 52% growth in performance per year!

- 2002: The thermal wall
 - Speed (frequency) peaks, but transistors keep shrinking

- The Multicore revolution
 - 15-20 years later than predicted, we have hit the performance wall



4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.3

At the same time...

- Amount of stored data is exploding...



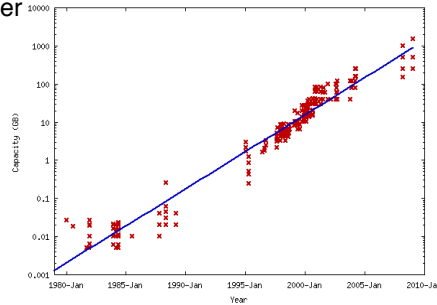
4/2

Anthony D. Joseph CS162 ©UCB Spring 2013

24.4

Data Deluge

- Billions of users connected through the net
 - WWW, FB, twitter, cell phones, ...
 - 80% of the data on FB was produced last year
- Storage getting cheaper
 - Store more data!



4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.5

Solving the Impedance Mismatch

- Computers not getting faster, and we are drowning in data
 - How to resolve the dilemma?
- Solution adopted by web-scale companies
 - Go massively *distributed* and *parallel*



4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.6

Enter the World of Distributed Systems

- Distributed Systems/Computing
 - *Loosely coupled* set of computers, communicating through *message passing*, solving a common goal
- Distributed computing is *challenging*
 - Dealing with *partial failures* (examples?)
 - Dealing with *asynchrony* (examples?)
- Distributed Computing versus Parallel Computing?
 - distributed computing=parallel computing + partial failures

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.7

Dealing with Distribution

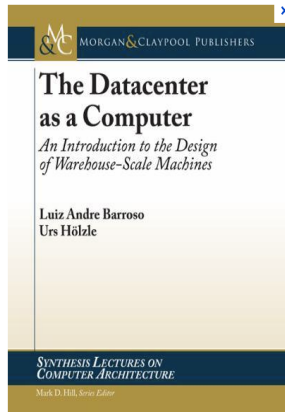
- We have seen several of the tools that help with distributed programming
 - Message Passing Interface (MPI)
 - Distributed Shared Memory (DSM)
 - Remote Procedure Calls (RPC)
- But, distributed programming is still very hard
 - Programming for scale, fault-tolerance, consistency, ...

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.8

The Datacenter is the new Computer



- “*Program*” == Web search, email, map/GIS, ...
- “*Computer*” == 10,000’s computers, storage, network
- Warehouse-sized facilities and workloads
- *Built from less reliable components than traditional datacenters*

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.9

Datacenter/Cloud Computing OS

- If the datacenter/cloud is the new computer
 - What is its **Operating System**?
 - Note that we are not talking about a host OS

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.10

Classical Operating Systems

- Data sharing
 - Inter-Process Communication, RPC, files, pipes, ...
- Programming Abstractions
 - Libraries (libc), system calls, ...
- Multiplexing of resources
 - Scheduling, virtual memory, file allocation/protection, ...

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.11

Datacenter/Cloud Operating System

- Data sharing
 - Google File System, [key/value stores](#)
- Programming Abstractions
 - Google MapReduce, PIG, Hive, Spark
- Multiplexing of resources
 - Apache projects: Mesos, [YARN \(MRv2\)](#), [ZooKeeper](#), [BookKeeper](#), ...

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.12

Google Cloud Infrastructure

- Google File System (GFS), 2003
 - Distributed File System for entire cluster
 - Single namespace
- Google MapReduce (MR), 2004
 - Runs queries/jobs on data
 - Manages work distribution & fault-tolerance
 - Colocated with file system
- Apache open source versions Hadoop DFS and Hadoop MR



4/29/2013

Anthony D. Joseph CS162 @UCB Spring 2013

24.13

GFS/HDFS Insights

- *Petabyte* storage
 - Files split into large blocks (128 MB) and replicated across several nodes
 - Big blocks allow high throughput sequential reads/writes
- Data *striped* on hundreds/thousands of servers
 - Scan 100 TB on 1 node @ 50 MB/s = 24 days
 - Scan on 1000-node cluster = 35 minutes

4/29/2013

Anthony D. Joseph CS162 @UCB Spring 2013

24.14

GFS/HDFS Insights (2)

- *Failures* will be the norm
 - Mean time between failures for 1 node = 3 years
 - Mean time between failures for 1000 nodes = 1 day
- Use *commodity* hardware
 - Failures are the norm anyway, buy cheaper hardware
- No complicated consistency models
 - Single writer, append-only data

4/29/2013

Anthony D. Joseph CS162 @UCB Spring 2013

24.15

MapReduce Insights

- Restricted key-value model
 - Same *fine-grained operation* (Map & Reduce) repeated on big data
 - Operations must be *deterministic*
 - Operations must be *idempotent/no side effects*
 - Only communication is through the shuffle
 - Operation (Map & Reduce) output saved (on disk)

4/29/2013

Anthony D. Joseph CS162 @UCB Spring 2013

24.16

What is MapReduce Used For?

- At **Google**:
 - Index building for Google Search
 - Article clustering for Google News
 - Statistical machine translation
- At **Yahoo!**:
 - Index building for Yahoo! Search
 - Spam detection for Yahoo! Mail
- At **Facebook**:
 - Data mining
 - Ad optimization
 - Spam detection

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.17

MapReduce Pros

- Distribution is completely **transparent**
 - Not a single line of distributed programming (ease, correctness)
- Automatic **fault-tolerance**
 - Determinism enables running failed tasks somewhere else again
 - Saved intermediate data enables just re-running failed reducers
- Automatic **scaling**
 - As operations as side-effect free, they can be distributed to any number of machines dynamically
- Automatic **load-balancing**
 - Move tasks and speculatively execute duplicate copies of slow tasks (*stragglers*)

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.18

MapReduce Cons

- Restricted programming model
 - Not always natural to express problems in this model
 - Low-level coding necessary
 - Little support for iterative jobs (lots of disk access)
 - High-latency (batch processing)
- Addressed by follow-up research
 - **Pig** and **Hive** for high-level coding
 - **Spark** for iterative and low-latency jobs

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.19

Pig

- High-level language:
 - Expresses sequences of MapReduce jobs
 - Provides relational (SQL) operators (JOIN, GROUP BY, etc)
 - Easy to plug in Java functions
- Started at Yahoo! Research
 - Runs about 50% of Yahoo!'s jobs

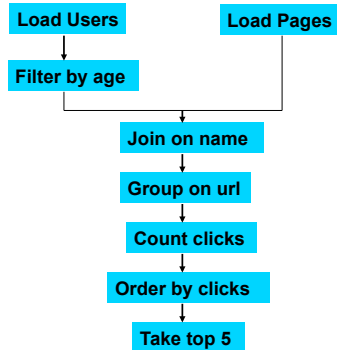


4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

Example Problem

Given *user data* in one file, and *website data* in another, find the *top 5 most visited pages* by users aged 18-25



Example from <http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt>
 4/29/2013 Anthony D. Joseph CS162 ©UCB Spring 2013 24.21

In MapReduce

```

// User class
public class User {
    String name;
    int age;
    User(String name, int age) {
        this.name = name;
        this.age = age;
    }
}

// Page class
public class Page {
    String url;
    int clicks;
    Page(String url, int clicks) {
        this.url = url;
        this.clicks = clicks;
    }
}

// Mapper class
public class Mapper {
    public void map(String line, Context context) {
        // Parse user data
        if (line.startsWith("user:")) {
            String[] parts = line.split(",");
            User user = new User(parts[1], Integer.parseInt(parts[2]));
            context.write(user);
        }
        // Parse page data
        if (line.startsWith("page:")) {
            String[] parts = line.split(",");
            Page page = new Page(parts[1], Integer.parseInt(parts[2]));
            context.write(page);
        }
    }
}

// Reducer class
public class Reducer {
    public void reduce(List<Writable> values, Context context) {
        // Join users and pages
        // Group by url
        // Count clicks
        // Order by clicks
        // Take top 5
    }
}
  
```

4/29/2013 Anthony D. Joseph CS162 ©UCB Spring 2013 24.22
 Example from <http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt>

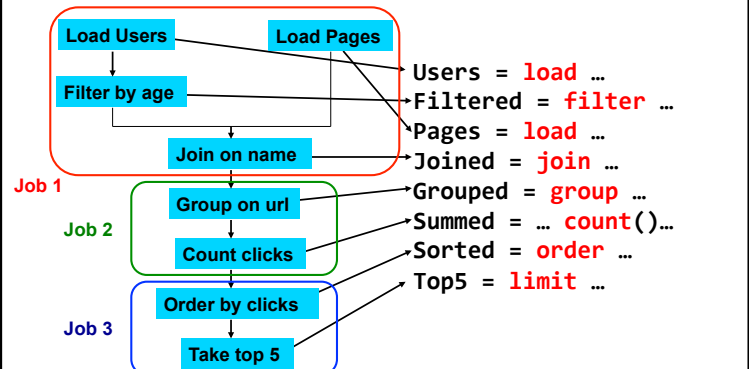
In Pig Latin

Users = load 'users' as (name, age);
 Filtered = filter Users by age >= 18 and age <= 25;
 Pages = load 'pages' as (user, url);
 Joined = join Filtered by name, Pages by user;
 Grouped = group Joined by url;
 Summed = foreach Grouped generate group, count(Joined) as clicks;
 Sorted = order Summed by clicks desc;
 Top5 = limit Sorted 5;
 store Top5 into 'top5sites';

Example from <http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt>
 4/29/2013 Anthony D. Joseph CS162 ©UCB Spring 2013 24.23

Translation to MapReduce

Notice how naturally the components of the job translate into Pig Latin.



4/29/2013 Anthony D. Joseph CS162 ©UCB Spring 2013 24.24
 Example from <http://wiki.apache.org/pig-data/attachments/PigTalksPapers/attachments/ApacheConEurope09.ppt>

Hive

- Relational database built on Hadoop
 - Maintains table schemas
 - SQL-like query language (which can also call Hadoop Streaming scripts)
 - Supports table partitioning, complex data types, sampling, some query optimization
- Developed at Facebook
 - Used for many Facebook jobs



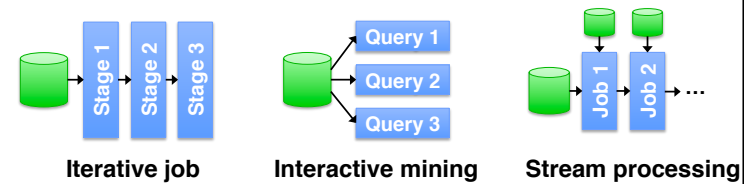
4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

Spark Motivation

Complex jobs, interactive queries and online processing all need one thing that MR lacks:

Efficient primitives for **data sharing**



4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.26

Spark Motivation

Complex jobs, interactive queries and online processing all need one thing that MR lacks:

Efficient primitives for **data sharing**

Problem: in MR, the only way to share data across jobs is using stable storage (e.g. file system) → slow!

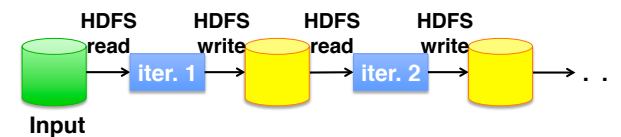
Iterative job Interactive mining Stream processing

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.27

Examples



HDFS read → query 1 → result 1

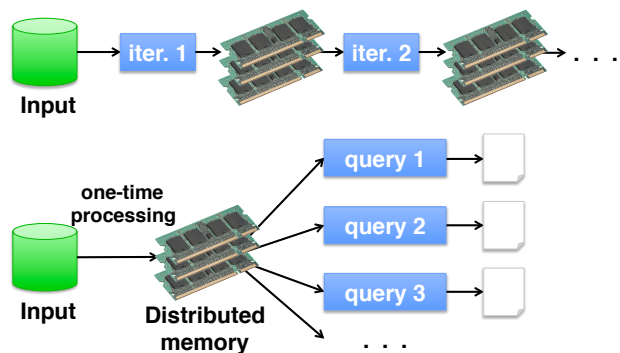
Opportunity: DRAM is getting cheaper → use main memory for intermediate results instead of disks

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.28

Goal: In-Memory Data Sharing



4/2 **10-100x faster than network and disk**

Solution: Resilient Distributed Datasets (RDDs)

- Partitioned collections of records that can be stored in memory across the cluster
- Manipulated through a diverse set of transformations (map, filter, join, etc)
- Fault recovery without costly replication
 - Remember the series of transformations that built an RDD (its lineage) to recompute lost data
- <http://www.spark-project.org/>

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.30

Administrivia

- Project 4
 - Design Doc due **tonight (4/29) by 11:59pm**, reviews Wed-Fri
 - Code due next week **Thu 4/9 by 11:59pm**
- Final Exam Review
 - **Monday 5/6, 2-5pm in 100 Lewis Hall**
- My RRR week office hours
 - **Monday 5/6, 1-2pm and Wednesday 5/8, 2-3pm**
- CyberBunker.com 300Gb/s DDoS attack against Spamhaus
 - 35 yr old Dutchman “S.K.” arrested in Spain on 4/26
 - Was using van with “various antennas” as mobile office

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.31

5min Break

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.32

Datacenter Scheduling Problem

- Rapid innovation in datacenter computing frameworks
- **No single framework optimal for all applications**
- Want to run multiple frameworks in a single datacenter
 - ...to maximize utilization
 - ...to share data between frameworks

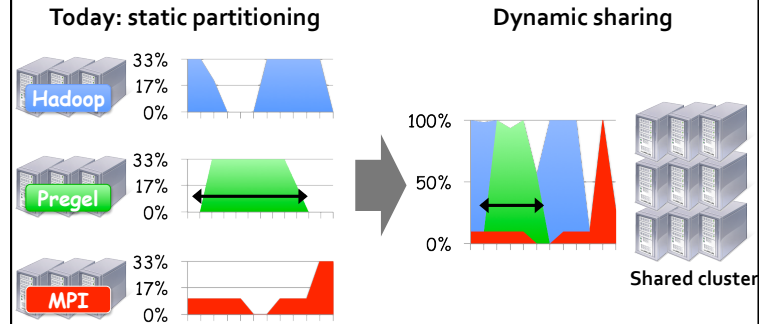


4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.33

Where We Want to Go



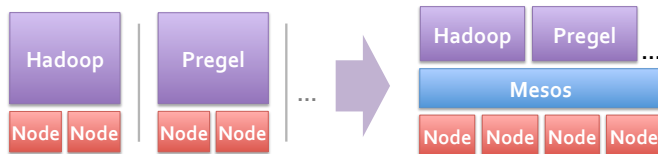
4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.34

Solution: Apache Mesos

- Mesos is a common resource sharing layer over which diverse frameworks can run



- Run multiple instances of the *same* framework
 - Isolate production and experimental jobs
 - Run multiple versions of the framework concurrently
- Build *specialized frameworks* targeting particular problem domains
 - Better performance than general-purpose abstractions

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.35

Mesos Goals

- **High utilization** of resources
- **Support diverse frameworks** (current & future)
- **Scalability** to 10,000's of nodes
- **Reliability** in face of failures

<http://incubator.apache.org/mesos/>

Resulting design: Small microkernel-like core that pushes scheduling logic to frameworks

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.36

Mesos Design Elements

- Fine-grained sharing:
 - Allocation at the level of *tasks* within a job
 - Improves utilization, latency, and data locality
- Resource offers:
 - Simple, scalable application-controlled scheduling mechanism

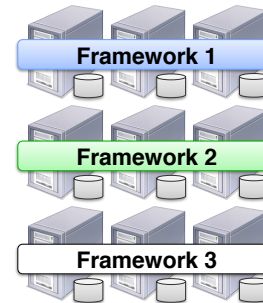
4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.37

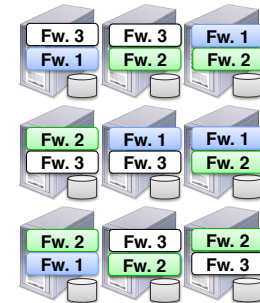
Element 1: Fine-Grained Sharing

Coarse-Grained Sharing (HPC):



Storage System (e.g. HDFS)

Fine-Grained Sharing (Mesos):



Storage System (e.g. HDFS)

+ Improved utilization, responsiveness, data locality

4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.38

Element 2: Resource Offers

- Option: Global scheduler
 - Frameworks express needs in a specification language, global scheduler matches them to resources
- + Can make optimal decisions
 - Complex: language must support all framework needs
 - Difficult to scale and to make robust
 - Future frameworks may have unanticipated needs

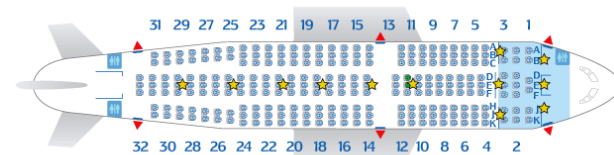
4/29/2013

Anthony D. Joseph CS162 ©UCB Spring 2013

24.39

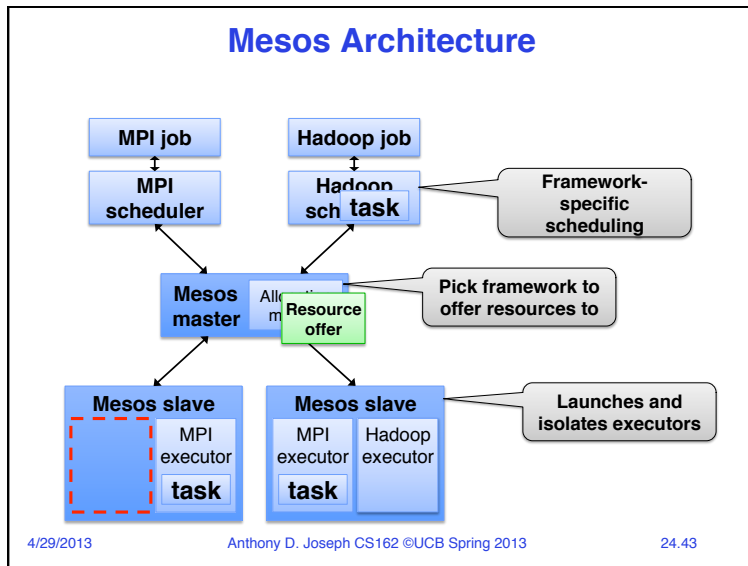
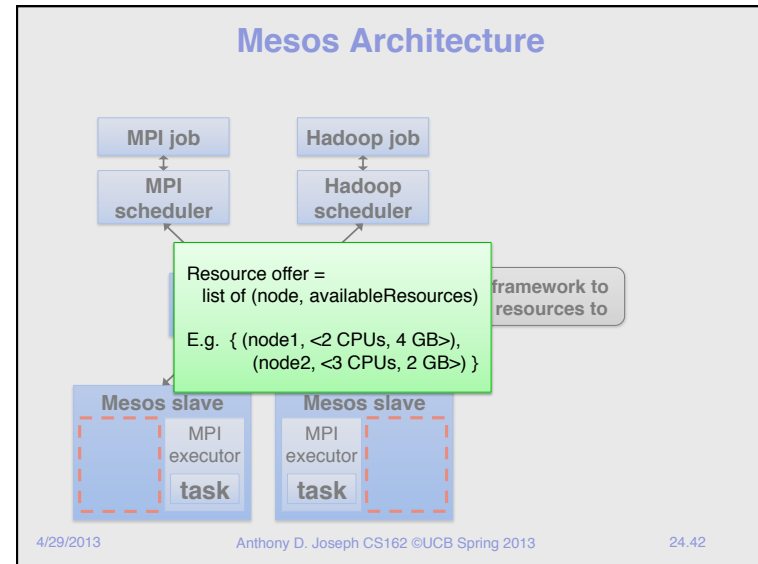
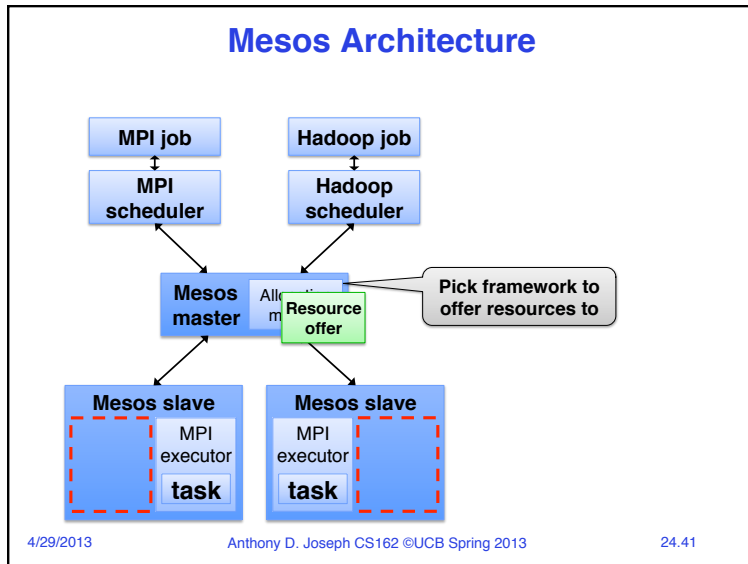
Element 2: Resource Offers

- Mesos: Resource offers
 - Offer available resources to frameworks, let them pick which resources to use and which tasks to launch
- + Keeps Mesos simple, lets it support future frameworks
 - Decentralized decisions might not be optimal








4/29/2

★ Video Screen ▲ Exit □ Club .40



Deployments

-  1,000's of nodes running over a dozen production services
-  Genomics researchers using Hadoop and Spark on Mesos
-  Spark in use by Yahoo! Research
-  Spark for analytics
-  Hadoop and Spark used by machine learning researchers

The diagram is dated 4/29/2013 and is attributed to Anthony D. Joseph CS162 ©UCB Spring 2013, slide 24.44.

Summary

- Cloud computing/datacenters are the new computer
 - Emerging “Datacenter/Cloud Operating System” appearing
- Pieces of the DC/Cloud OS
 - High-throughput filesystems (GFS/HDFS)
 - Job frameworks (MapReduce, Spark, Pregel)
 - High-level query languages (Pig, Hive)
 - Cluster scheduling (Apache Mesos)