IEMS5730
Big Data Systems and Information Processing

Overview of Big Data Processing
via
Datacenter-scale Computing

Prof. Wing C. Lau
Department of Information Engineering
wclau@ie.cuhk.edu.hk

Acknowledgements

- The slides used in this chapter are adapted from the following sources:
  - "Data-Intensive Information Processing Applications," by Jimmy Lin, University of Maryland.

This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States. See http://creativecommons.org/licenses/by-nc-sa/3.0/us/ for details

  - CS246 Mining Massive Data-sets, by Jure Leskovec, Stanford University.
  - Stat 260 Scalable Machine Learning of UC Berkeley, by Alex Smola, CMU.
  - 10-605 Machine Learning from Big Datasets, by William Cohen, CMU.
  - "Intro to Hadoop" in UCBerkeley i291 - Analyzing BigData with Twitter, by Bill Graham, Twitter.
- All copyrights belong to the original authors of the material.

What is this course about?

- Computing Infrastructure for Data-intensive Information Processing and Analytics
- Focus on the System Aspects of Big Data Processing:
  - Big Data Computing Infrastructure
  - The Big Data Processing Software Stack
  - Mainstream Parallel and Distributed Programming Models,
  - Their corresponding Platforms/ Frameworks and
  - Additional Operational/Programming Tools

  for Big Data Processing/Analytics in the Real World
- This course is NOT about the Design/ Analysis of Machine Learning or Data Mining algorithms
  - For ML/DM, you should consider to take IERG4300, Web-scale Information Analysis, instead of (or in parallel with) this course

Course Administrivia

Course Pre-requisites

- Strong Programming and Hands-on Software Development skills ;
- Some Operating System Management/Configuration Skills
- No previous experience necessary in
  - MapReduce/ Hadoop or other
  - Parallel and distributed programming models

  BUT we expect you to pick them up quickly

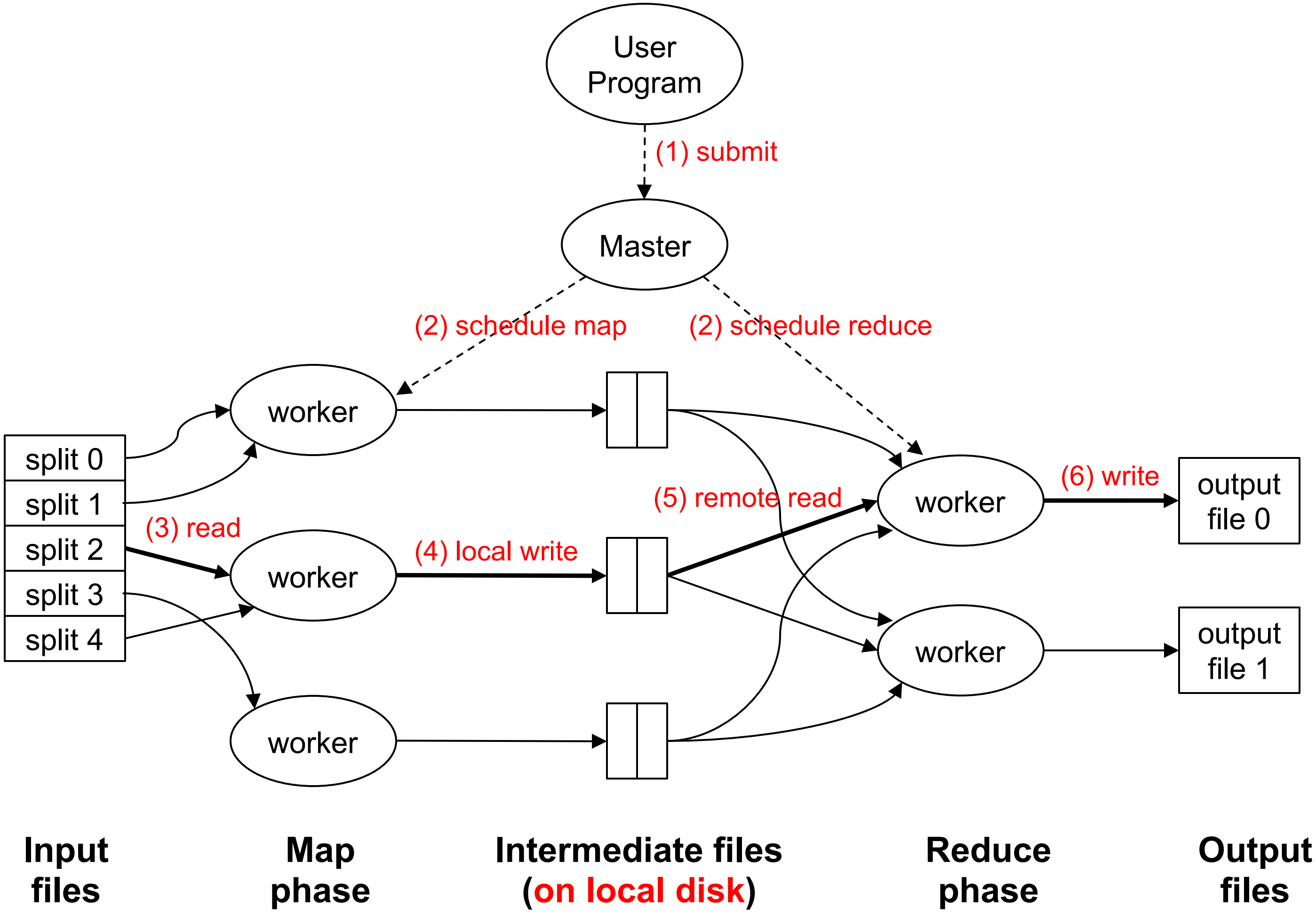
What is MapReduce?

- The 1st widely-deployed (successful) Programming model for expressing distributed computations at a massive scale
- Execution framework (actual software system) for organizing and performing such computations
- Open-source implementation called Hadoop

HW#0

Set up your own
Hadoop 2.x Cluster +
run a sample MapReduce program
using
a Free Public Cloud Infrastructure

Due in 10 days:
Due Date: Jan 21, 2023 11:59am
(noon-time on CNY Eve !!)



User Program

(1) submit

Master

(2) schedule map

(2) schedule reduce

split 0

split 1

split 2

split 3

split 4

(3) read

worker

worker

worker

(4) local write

(5) remote read

worker

worker

(6) write

output file 0

output file 1

Input files

Map phase

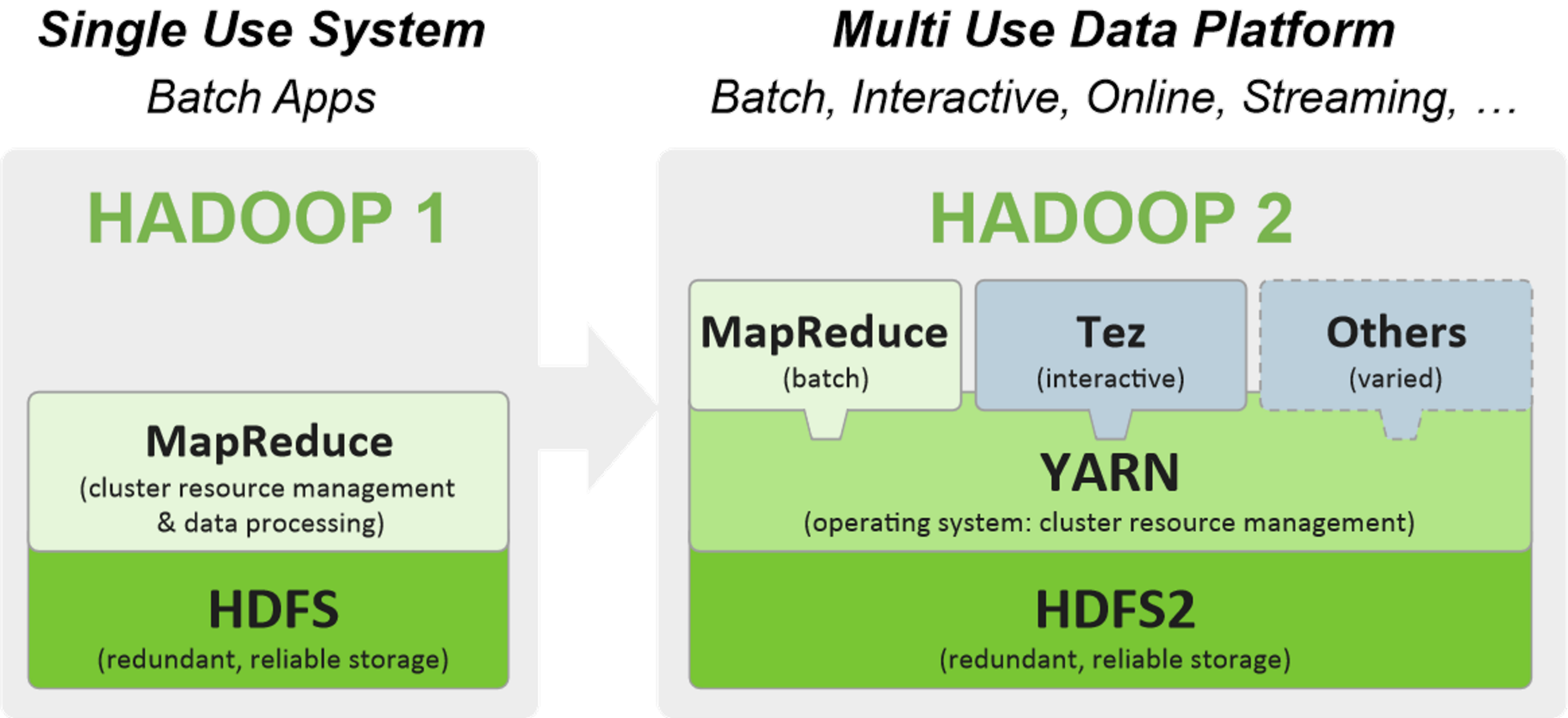
Intermediate files (on local disk)

Reduce phase

Output files

Adapted from (Dean and Ghemawat, OSDI 2004)

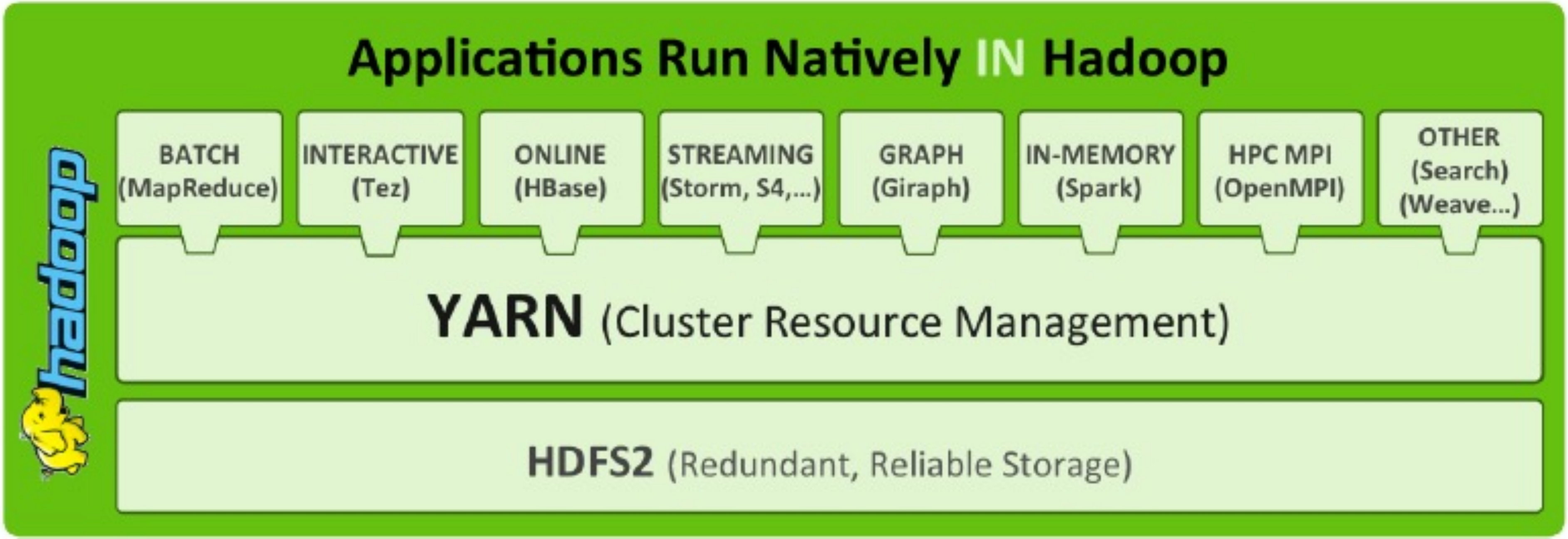
YARN for Hadoop 2.0


Single Use System
Batch Apps
Multi Use Data Platform
Batch, Interactive, Online, Streaming, …
HADOOP 1
MapReduce
(cluster resource management & data processing)
HDFS
(redundant, reliable storage)
HADOOP 2
MapReduce
(batch)
Tez
(interactive)
Others
(varied)
YARN
(operating system: cluster resource management)
HDFS2
(redundant, reliable storage)

- YARN (Yet Another Resource Negotiator)
  - Like an Operating System (OS) for a Data-center-scale Computing Cluster
  - Serve as the resource management platform for Cluster of Computers to support general Distributed/Parallel Applications/Frameworks beyond the MapReduce computational model.

V. K. Vavilapalli, A. C. Murthy, "Apache Hadoop YARN: Yet Another Resource Negotiator", in ACM Symposium on Cloud Computing (SoCC) 2013.

A Big Data Processing Stack with YARN

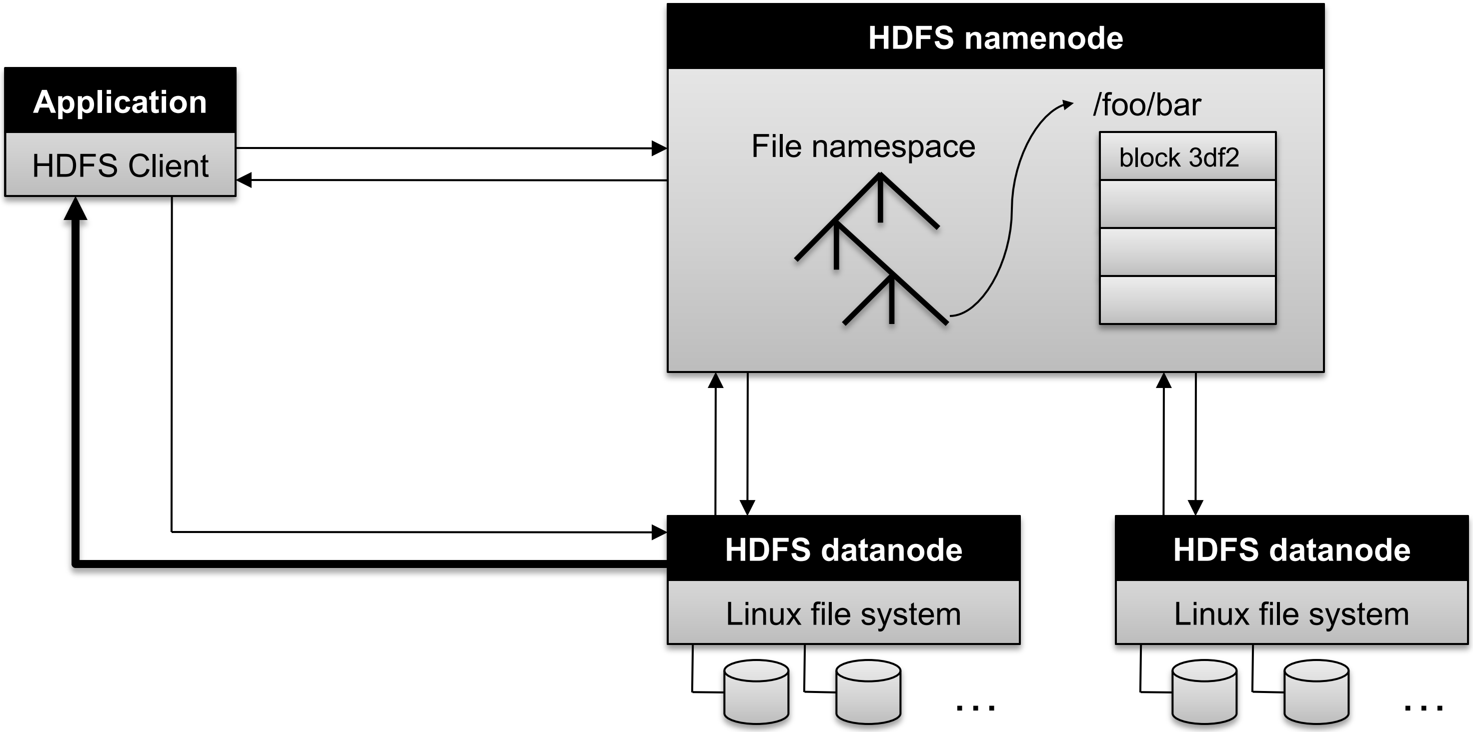


Applications Run Natively IN Hadoop
hadoop
BATCH (MapReduce)
INTERACTIVE (Tez)
ONLINE (HBase)
STREAMING (Storm, S4,...)
GRAPH (Giraph)
IN-MEMORY (Spark)
HPC MPI (OpenMPI)
OTHER (Search) (Weave...)
YARN (Cluster Resource Management)
HDFS2 (Redundant, Reliable Storage)

What is MapReduce or Hadoop (cont'd) ?

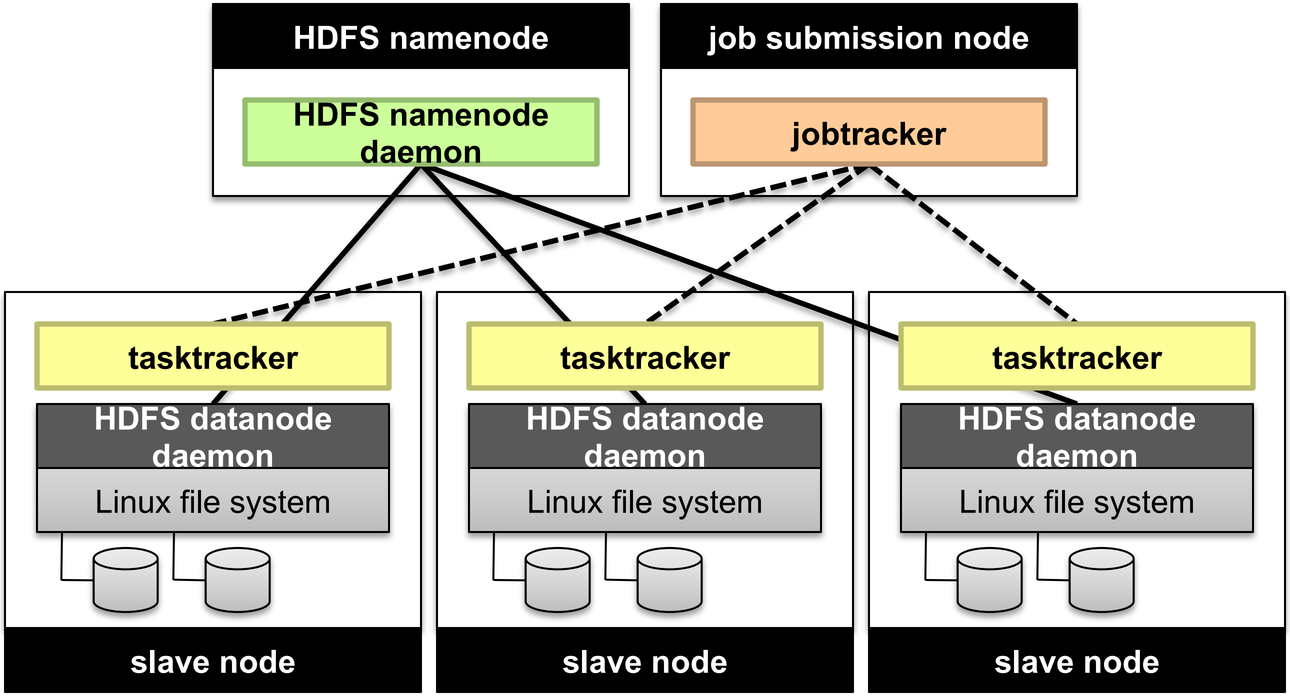
- Hadoop version 1.0 supports mainly MapReduce and a large-scale Distributed File System called Hadoop Distributed File System (HDFS)
  - HDFS is the open-source version of Google File System (GFS)
- Since version 2.0, Hadoop has added YARN to become a general resource management platform (i.e. OS for a Datacenter-scale Cluster) which supports not only MapReduce, HDFS, but also other datacenter-scale computing models/ frameworks, e.g.
  - Giraph for Graph processing,
  - Storm for stream processing,
  - TeZ for DAG-based parallel-processing
  - Spark, etc.

Hadoop Distributed File System (HDFS) Architecture



Adapted from (Ghemawat et al., SOSP 2003)

Putting everything together under Hadoop 1.0



HDFS namenode

HDFS namenode daemon

job submission node

jobtracker

tasktracker

HDFS datanode daemon

Linux file system

slave node

tasktracker

HDFS datanode daemon

Linux file system

slave node

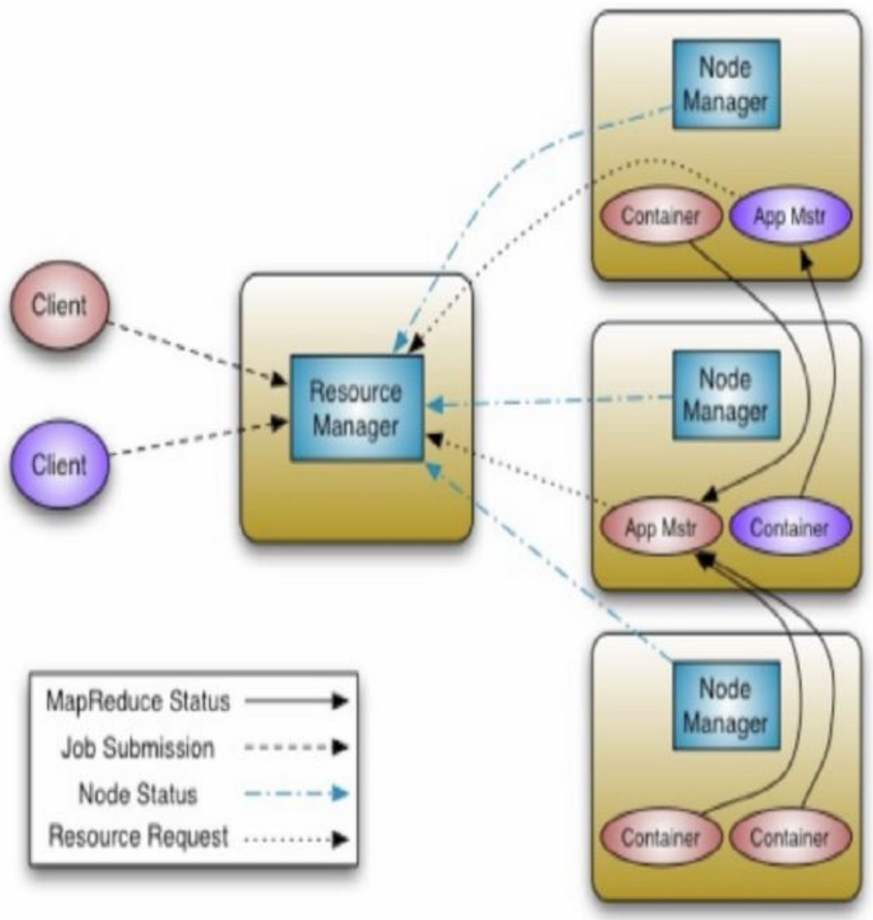
tasktracker

HDFS datanode daemon

Linux file system

slave node

Different Terminologies for Job and Task Tracking under Hadoop2.0 / YARN



- Scalability - Clusters of 6,000-10,000 machines
  - Each machine with 16 cores, 48G/96G RAM, 24TB/36TB disks
  - 100,000+ concurrent tasks
  - 10,000 concurrent jobs

- HDFS architecture and terminologies largely remain unchanged w.r.t. Hadoop 1.0 as shown in previous 2 slides

YARN Framework

RescourceManager:

Arbitrates resources among all the applications in the system

NodeManager:

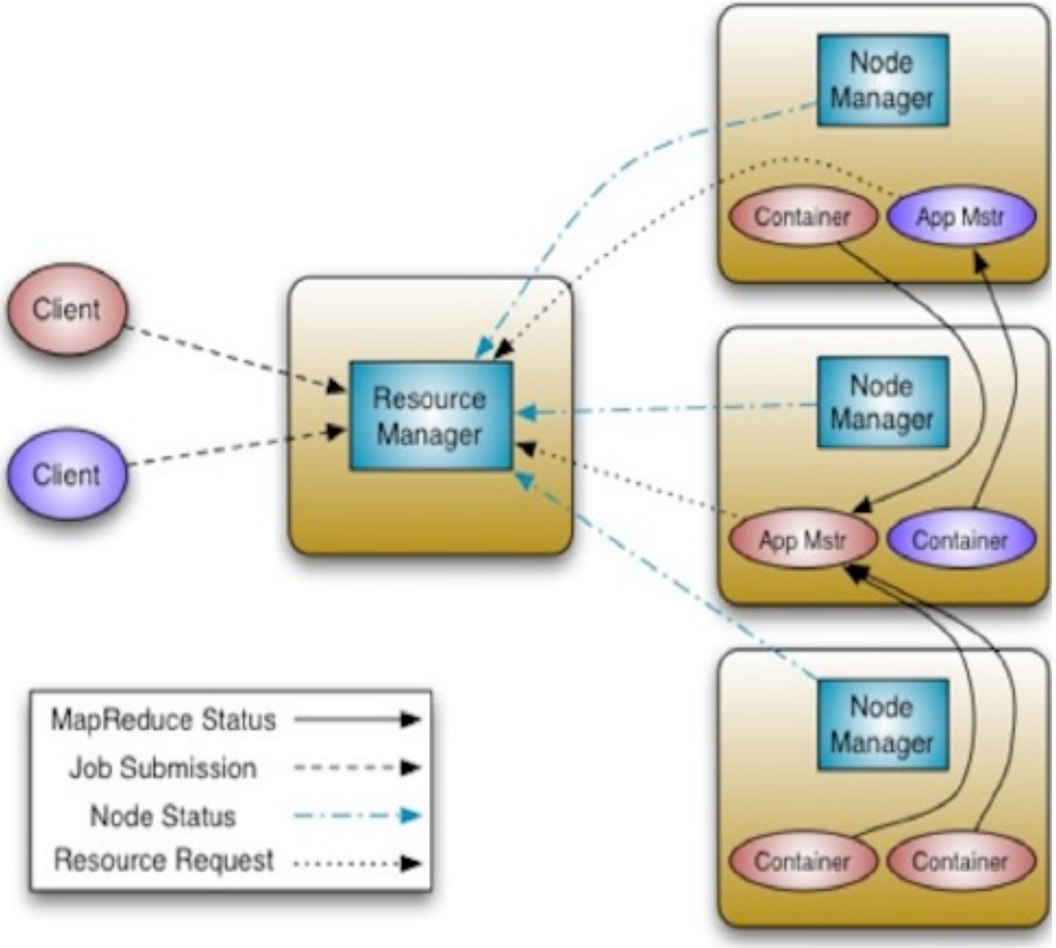
the per-machine slave, which is responsible for launching the applications' containers, monitoring their resource usage

ApplicationMaster:

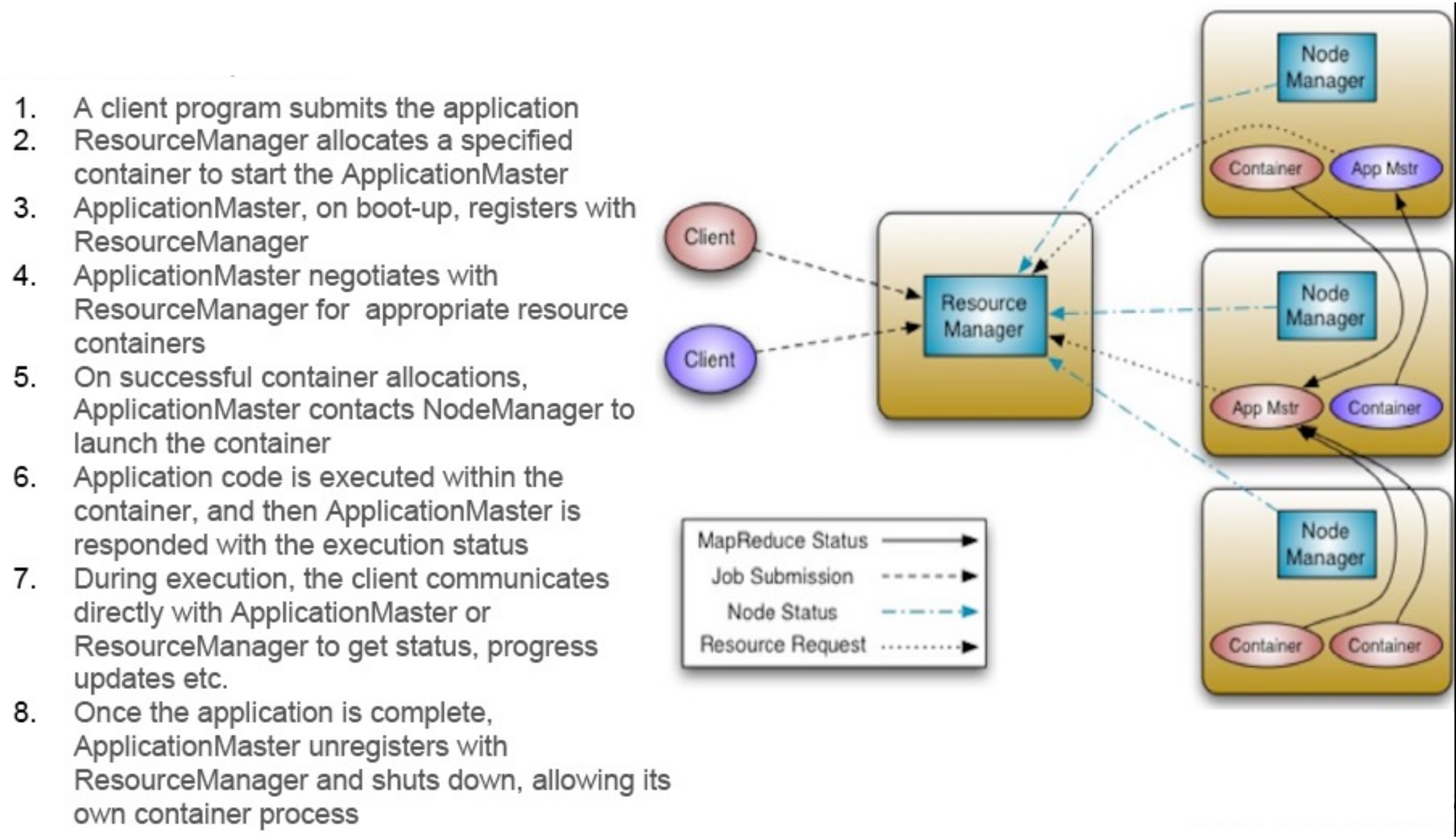
Negatiate appropriate resource containers from the Scheduler, tracking their status and monitoring for progress

Container:

Unit of allocation incorporating resource elements such as memory, cpu, disk, network etc, to execute a specific task of the application (similar to map/reduce slots in MRv1)

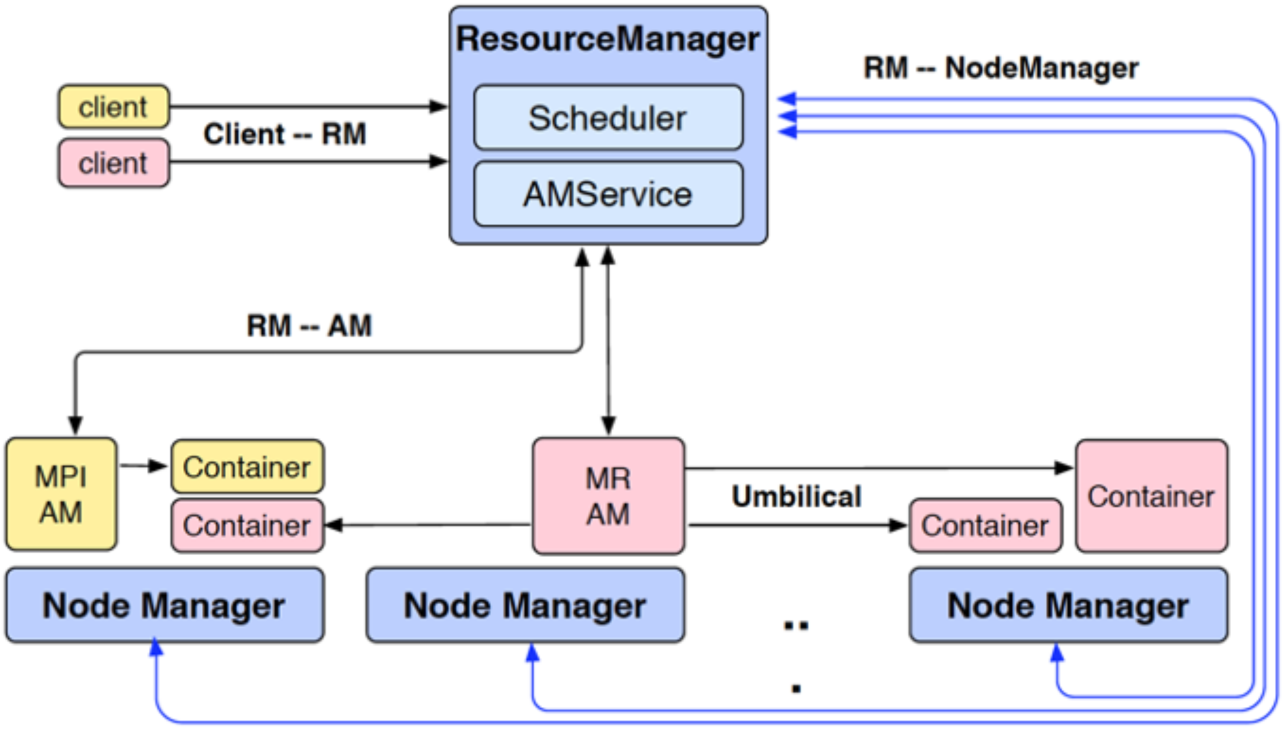


YARN Execution Sequence



1. A client program submits the application
2. ResourceManager allocates a specified container to start the ApplicationMaster
3. ApplicationMaster, on boot-up, registers with ResourceManager
4. ApplicationMaster negotiates with ResourceManager for appropriate resource containers
5. On successful container allocations, ApplicationMaster contacts NodeManager to launch the container
6. Application code is executed within the container, and then ApplicationMaster is responded with the execution status
7. During execution, the client communicates directly with ApplicationMaster or ResourceManager to get status, progress updates etc.
8. Once the application is complete, ApplicationMaster unregisters with ResourceManager and shuts down, allowing its own container process

Cluster Resource Management w/ YARN in Hadoop2.0



- Multiple frameworks (Applications) can run on top of YARN to share a Cluster, e.g. MapReduce is one framework (Application), MPI, or Storm are other ones.
- YARN splits the functions of JobTracker into 2 components: resource allocation and job-management (e.g. task-tracking/ recovery):
  - Upon launching, each Application will have its own Application Master (AM), e.g. MR-AM in the figure above is the AM for MapReduce, to track its own tasks and perform failure recovery if needed
  - Each AM will request resources from the YARN Resource Manager (RM) to launch the Application's jobs/tasks (Containers in the figure above) ;
  - The YARN RM determines resource allocation across the entire cluster by communicating with/ controlling the Node Managers (NM), one NM per each machine.

Computing/ Cloud Resources

- Hadoop on your local machine
- Hadoop in a Virtual Machine on your local machine
- Sign-up for Freebie (limited-time) Trial accounts from Commercial Cloud Computing Services:
  - Amazon Web Service (AWS), Google Compute Engine
  - Homework sets will require each student to setup various Big Data Processing systems using these Public Cloud Services
- The IE DIC (Data-Intensive Cluster):
  - Already Setup with Hadoop 2.0/ YARN, MapReduce, Hadoop Distributed File System (HDFS), Pig, Hive, Spark, Kubernetes, etc
- You may use your own cluster installed over the free public cloud service or learn to use the IE DIC to run different Parallel/ Distributed Programming tasks.

This course is not for you…

- If you’re not genuinely interested in the topic
- If you can’t put in the time
- If you’re not ready to do a lot of work
- If you’re not open to thinking about computing in new ways
- If you can’t cope with the uncertainty, unpredictability, etc. that comes with bleeding edge software

Otherwise, this will be a richly rewarding course!

- If you do not have any Operating System Internals background, e.g., from
  - An undergrad course in O.S. and/ or
  - Managing/Running a Linux-based computer

You would need to pick-up such skills yourself promptly but this can be VERY time-consuming, ADDITIONAL 30-40 hours per Homework !!

Source: Wikipedia (Japanese rock garden)

Zen

- We will be using bleeding edge technologies (= immature!)
  - Bugs, undocumented features, inexplicable behavior
  - Data loss(!)
- Don’t get frustrated (take a deep breath)…
  - Those W$*#T@F! moments
- Be patient…
  - We will inevitably encounter “situations” along the way
- Be flexible…
  - We will have to be creative in workarounds
- Be constructive…
  - Tell me how I can make everyone’s experience better

Web-Scale, Big Data

Source: Wikipedia (Everest)

How many users and objects?

- Flickr has >6 billion photos
- Facebook has 1.15 billion active users
- Google is serving >1.2 billion queries/day on more than 27 billion items
- >2 billion videos/day watched on YouTube

How much data?

- Modern applications use massive data:
  - Rendering 'Avatar' movie required >1 petabyte of storage
  - eBay has >6.5 petabytes of user data
  - CERN's LHC will produce about 15 petabytes of data per year
  - In 2008, Google processed 20 petabytes per day
  - German Climate computing center dimensioned for 60 petabytes of climate data
  - Someone estimated in 2013 that Google had

    10 exabytes on disk and ~ 5 exabytes on tape backup
  - NSA Utah Data Center is said to have 5 zettabyte (!)
- How much is a zettabyte?
  - 1,000,000,000,000,000,000,000 bytes
  - A stack of 1TB hard disks that is 25,400 km high

Who cares?

- Ready-made large-data problems
  - Lots of user-generated content
  - Even more user behavior data
  - Examples: Facebook friend suggestions, Google ad placement
  - Business intelligence: gather everything in a data warehouse and run analytics to generate insight
- Utility computing
  - Provision Hadoop clusters on-demand in the cloud
  - Lower barrier to entry for tackling large-data problem
  - Commoditization and democratization of large-data capabilities

What to do with More Data?

POPULAR SCIENCE
THE FUTURE NOW
THE CONTROL CENTERS
Using Data to Feed the World, Solve Cold Cases, Battle Malware, Predict Our Fate
OFFICER ALGORITHM
Can a Crime Be Prevented Before It Begins?
NEW WAYS OF SEEING
A Gallery of Extraordinary Infographics
PLUS
Juan Enriquez Reprograms Life
James Gleick Unsplits the Bit
AND Lawrence Weschler Questions the Cloud
SPECIAL ISSUE
DATA IS POWER
HOW INFORMATION IS DRIVING THE FUTURE

- User Behavior Analysis
- AB Test Analysis
- Ad Targetting
- Trending Topics
- User and Topic Modeling
- Recommendations (Collaborative Filtering)
- Predictions
- Novel/ Abnormality Detection
- Training/Building AI, e.g. Alpha Go, Apple Siri, Google Translate

- Following Theory, Experiment and Simulation,

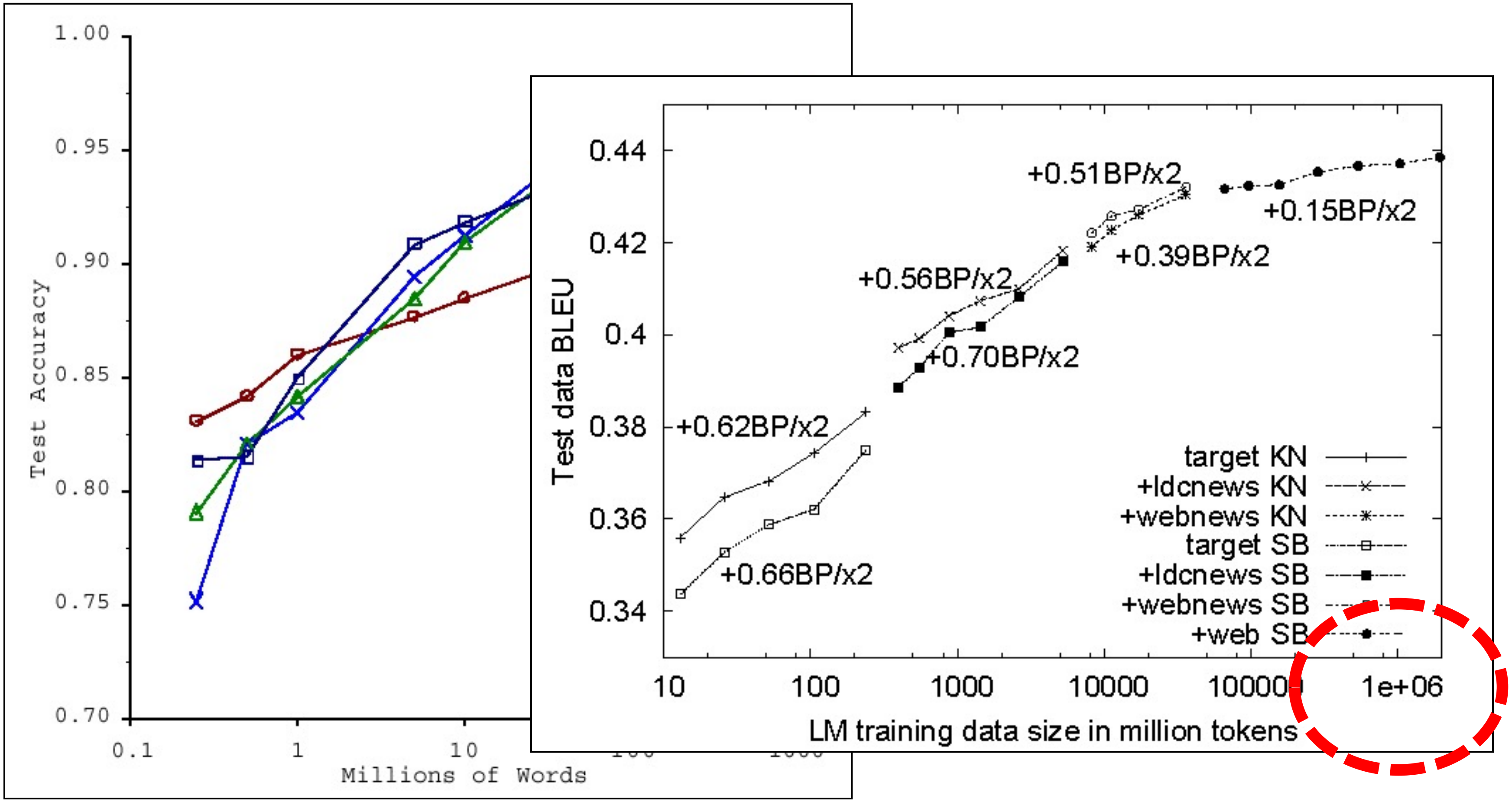
  Big Data has become the 4th-Paradigm of Science

`s/knowledge/data/g`

Knowledge Discovery via **Scalable** Information Analytics, e.g.

Scalable Data Mining, Statistical Modeling, Machine Learning

There’s no Data like more Data!



How do we get here if we’re not Google?

(Banko and Brill, ACL 2001)
(Brants et al., EMNLP 2007)

By 2001, we have learned that, for many tasks, there’s no real *substitute* for using lots of data

…and in 2009

Eugene Wigner's article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences" examines why so much of physics can be neatly explained with simple mathematical formulas such as $f = ma$ or $e = mc^2$. Meanwhile, sciences that involve human beings rather than elementary particles have proven more resistant to elegant mathematics. Economists suffer from physics envy over their inability to neatly model human behavior. An informal, incomplete grammar of the English language runs over 1,700 pages.

Perhaps when it comes to natural language processing and related fields, we're doomed to complex theories that will never have the elegance of physics equations. But if that's so, we should stop acting as if our goal is to author extremely elegant theories, and instead embrace complexity and make use of the best ally we have: the unreasonable effectiveness of data.

Norvig, Pereira, Halevy, "The Unreasonable Effectiveness of Data", 2009

Emphasis of this course

- **We will stress on:**
  - Parallel/ Distributed Programming Models and Architectures for processing Massive Datasets !
  - Open-source de facto Standard Frameworks
  - Scalability

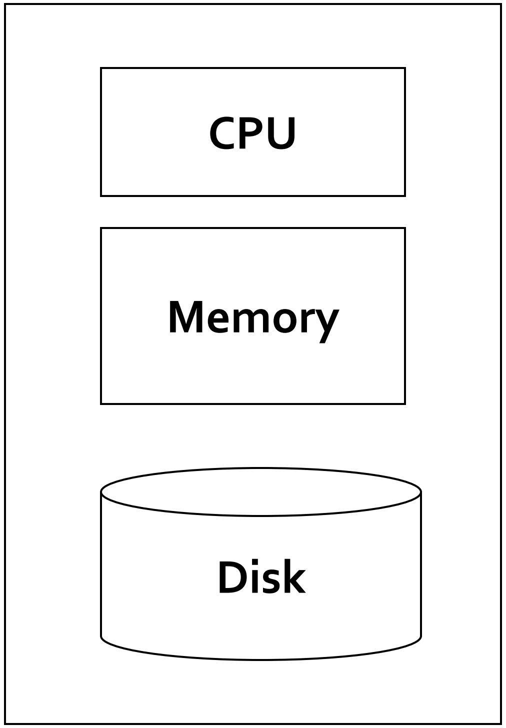
What will we learn?

- **We will learn to process/ compute with different types of Big Data:**
  - Data is of Large Volume (Terabyte-sized files)
  - Data from a Large Graph
  - Data is Infinite/ never-ending (Stream)
  - Data comes in Batches and can afford offline processing
  - Data comes in Fast and requires Low-latency processing
- **We will learn to use computation models beyond single machine:**
  - Programming and Running Datacenter-scale Computing Clusters

How do you want that data?

How do we scale up processing for Big Data ?

Single Node Architecture



“Classical”
Machine Learning, Statistics,
Data Mining

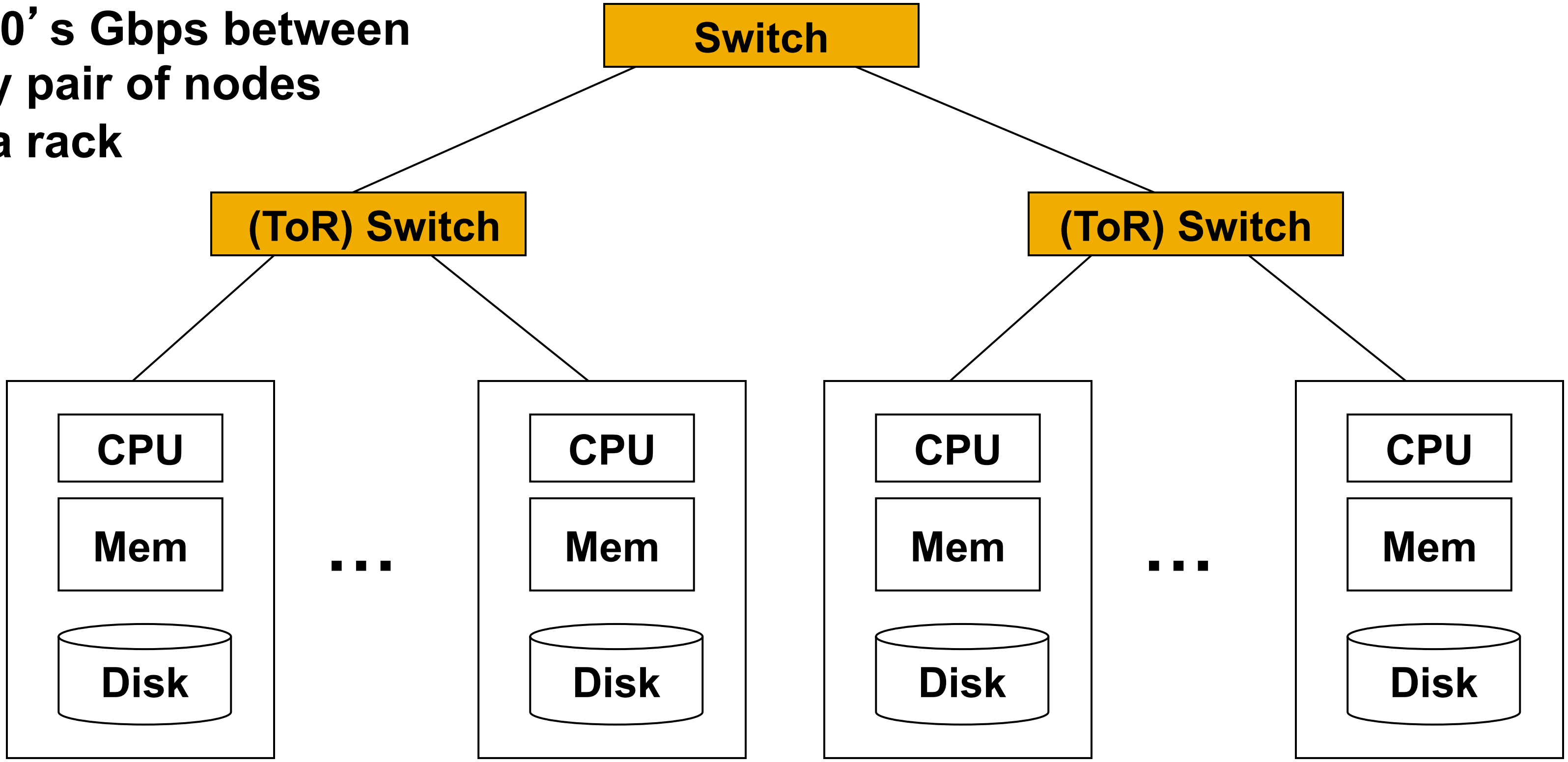
Motivation: Google Example

- 20+ billion web pages x 20KB = 400+ TB
- 1 computer reads 30-35 MB/sec from disk
  - ~4 months to read the web
- ~1,000 hard drives to store the web
- Take even more to **do** something useful with the data!
- **Today, a standard architecture for such problems is emerging:**
  - Cluster of commodity Linux nodes
  - Commodity network (Ethernet) to connect them

Cluster Architecture

10’s to 100 Gbps backbone between racks

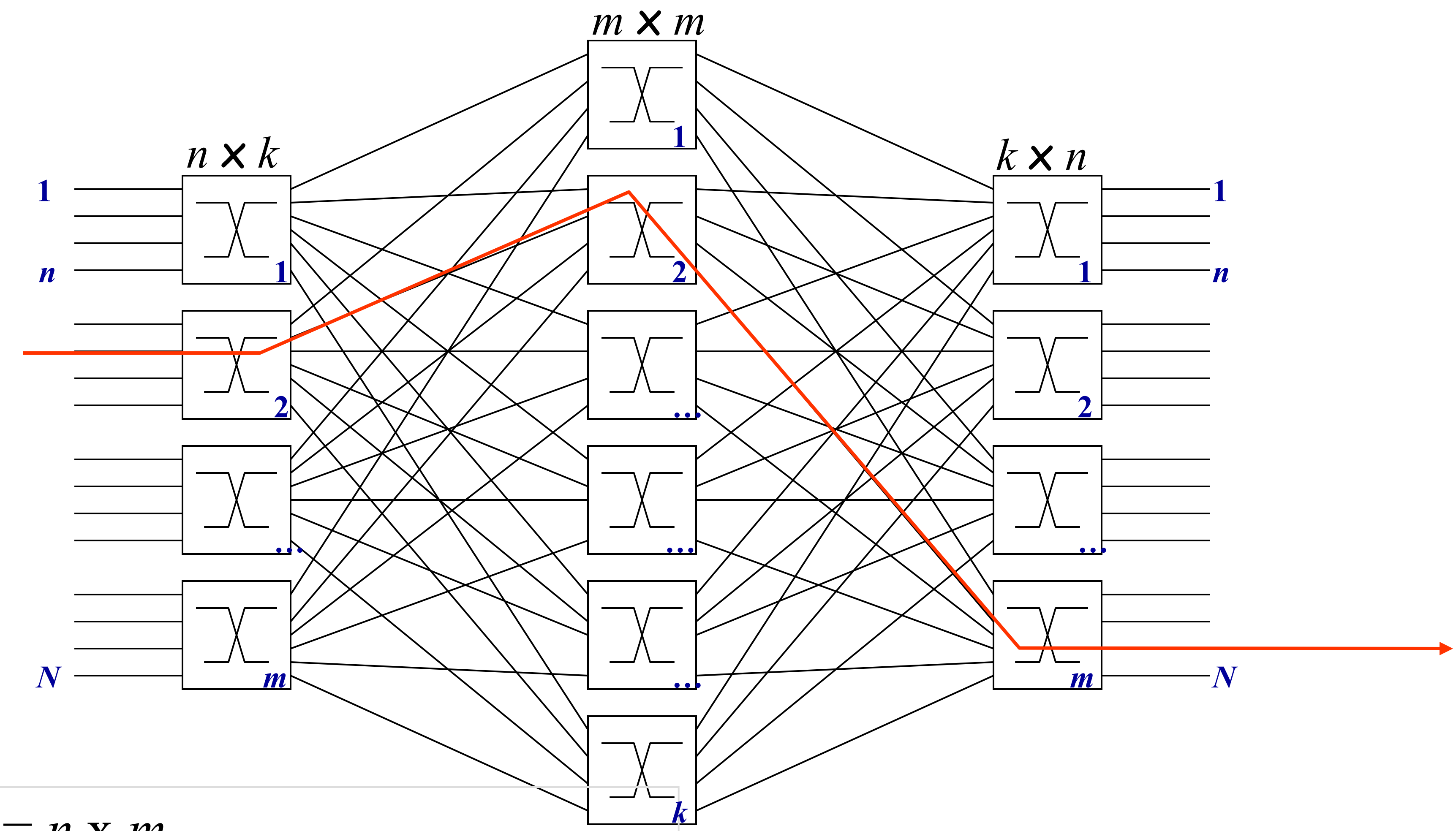
1-10’s Gbps between
any pair of nodes
in a rack



Each rack contains 16-64 nodes

In 2011, it was guestimated that Google had 1M machines, http://bit.ly/Shh0RO

Interconnection via a 3-stage Clos Network (instead of a Tree)

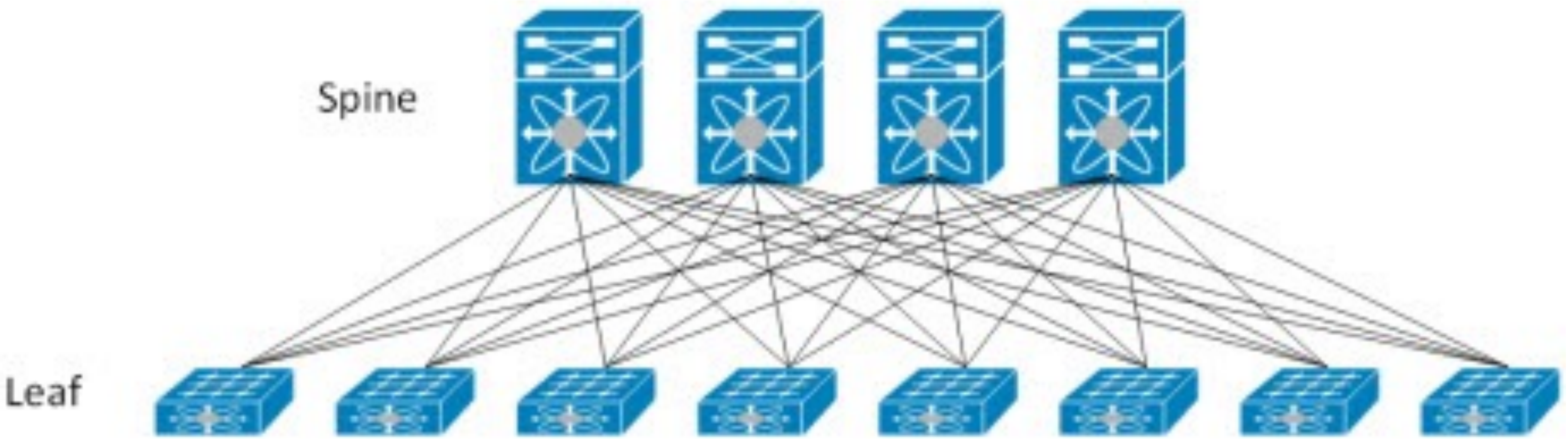


$N = n \text{ x } m$

$k >= n$

$k = 16, n = 16, m = 4,$

Clos Networks' Reappearance in Datacenter Networks (aka the Spine and Leaf Topology, or Folded Clos, or Fat-Trees)



The Top of Rack (ToR) switches are the Leaf switches

Each ToR is connected to multiple Core switches which represent the Spine.

# of Uplinks (of each ToR) = # of Spine switches

# of Downlinks (of each Spine switch) = # of Leaf switches

Multiple ECMP exists for every pair of Leaf switches

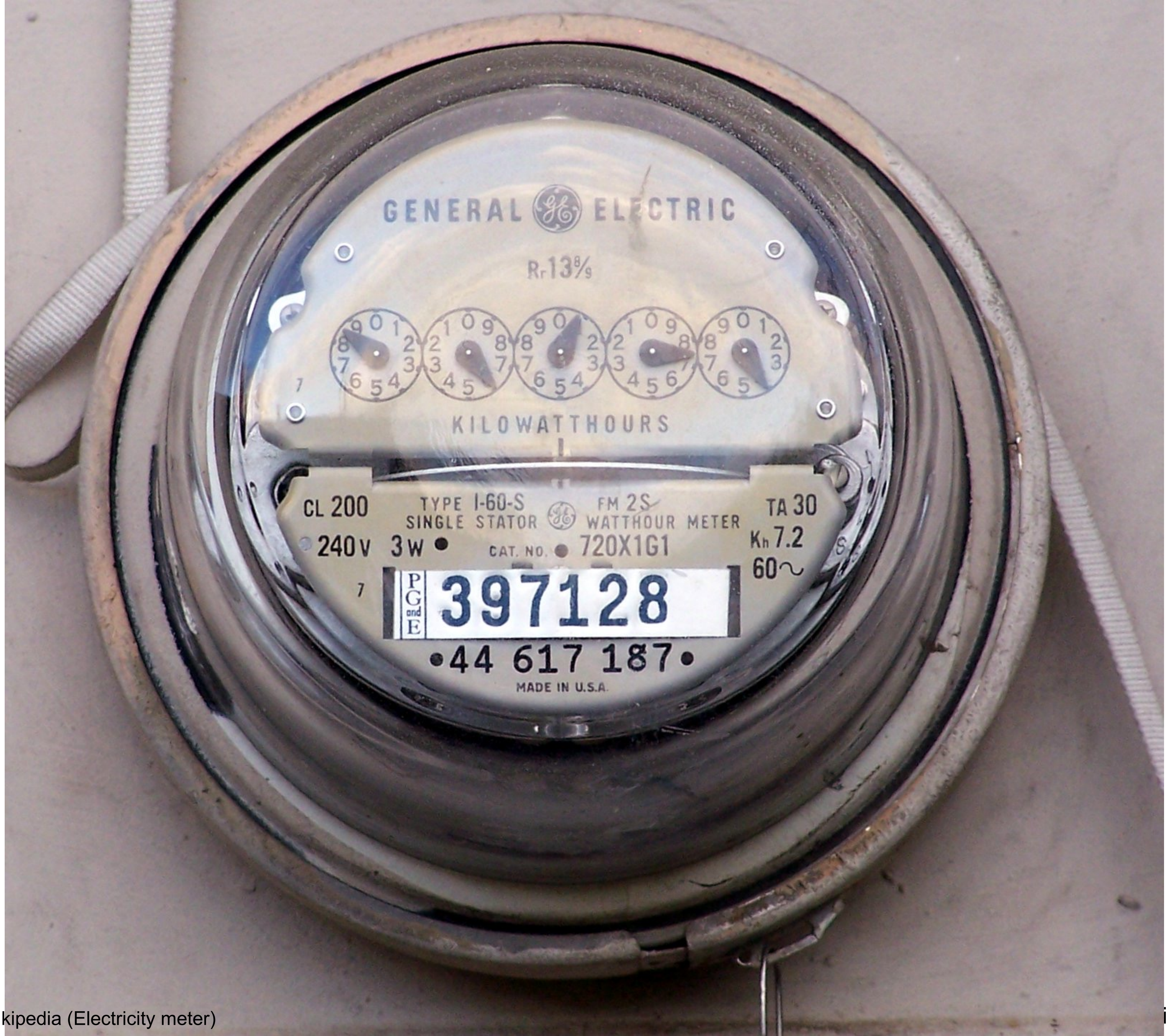
Support Incrementally "Scale-out" by adding more Leaf and Spine switches

"Jupiter Rising: A Decade of Clos Topologies and Central Control in Google's Datacenter Networks," ACM Sigcomm 2015.

Commoditized Big Data Processing via Cloud Computing?

The best thing since sliced bread?

- Before clouds…
  - Grids
  - Vector supercomputers
  - …
- Cloud computing means many different things:
  - Large-data processing
  - Rebranding of web 2.0
  - Utility computing
  - Everything as a service



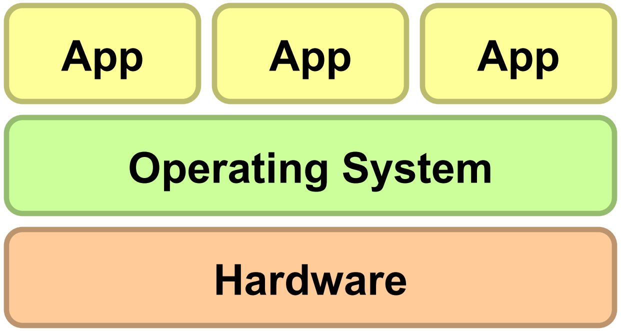
Source: Wikipedia (Electricity meter)

Utility Computing

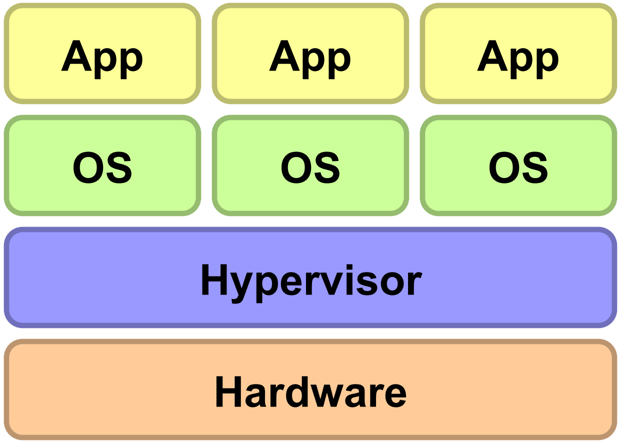
- What?
  - Computing resources as a metered service (“pay as you go”)
  - Ability to dynamically provision virtual machines
- Why?
  - Cost: capital vs. operating expenses
  - Scalability: “infinite” capacity
  - Elasticity: scale up or down on demand
- Does it make sense?
  - Benefits to cloud users
  - Business case for cloud providers

I think there is a world market for about five computers.
– Thomas J Watson of IBM, 1943

Enabling Technology: Virtualization



Traditional Stack



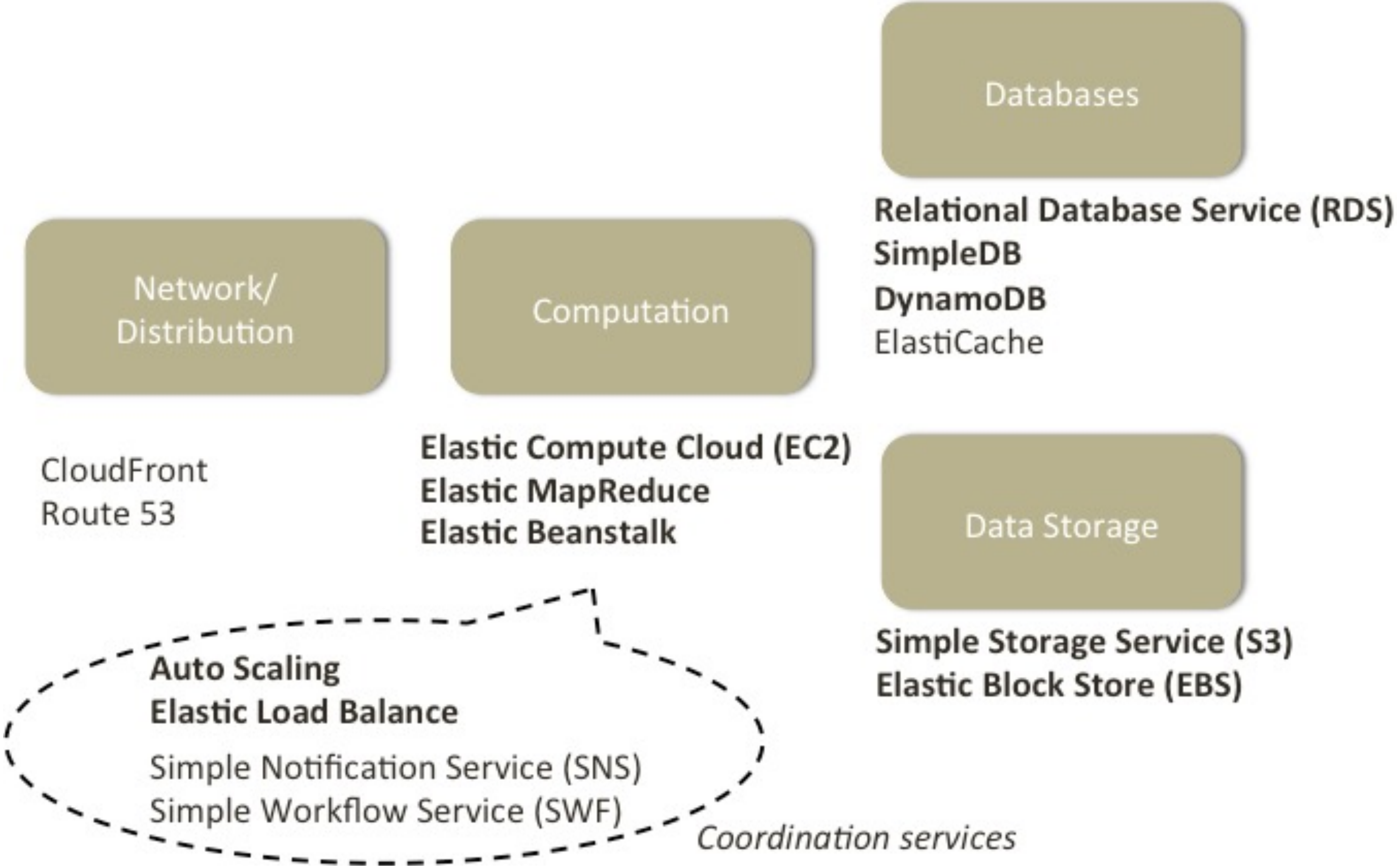
Virtualized Stack

Everything as a Service

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon's EC2, Google Compute Engine, Rackspace
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades, …
  - Example: Google App Engine, Google Kubernetes
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail, Office 360, Salesforce.com

Offerings of a Leading Cloud Computing Service Provider: Amazon Web Services (AWS)

Overview of AWS Services



Network/ Distribution

CloudFront
Route 53

Computation

Elastic Compute Cloud (EC2)
Elastic MapReduce
Elastic Beanstalk

Databases

Relational Database Service (RDS)
SimpleDB
DynamoDB
ElastiCache

Data Storage

Simple Storage Service (S3)
Elastic Block Store (EBS)

Auto Scaling
Elastic Load Balance
Simple Notification Service (SNS)
Simple Workflow Service (SWF)

Coordination services

What is Amazon Web Services (AWS) ?

- AWS provides a collection of services for building cloud applications

- Services for:
  - **Storage**: S3, EBS
  - **Computation**: Elastic Cloud Computing (EC2), scaling/load balancer, Elastic Map/Reduce, Elastic Beanstalk
  - **Databases**: RDS, DynamoDB, ElastiCache
  - **Coordination**: Simple Notification Service, Simple Workflow Framework
  - Content delivery network
  - Amazon CloudFront
  - Amazon Mechanical Turk (MTurk)
    A 'marketplace for work'
  - …

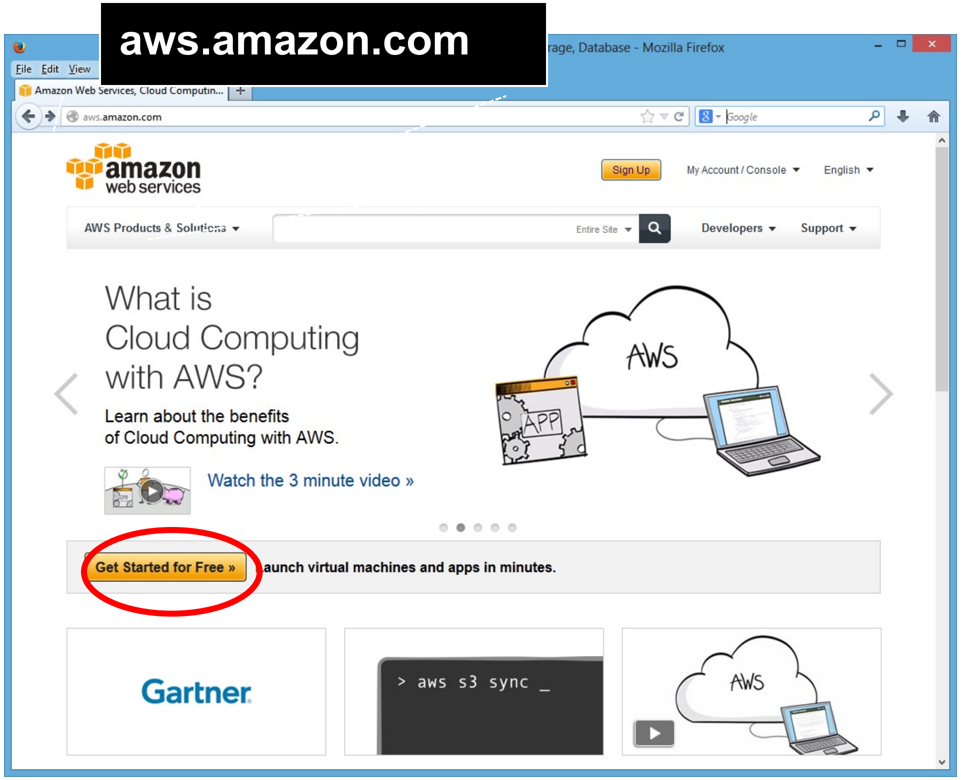
- All services are paid depending on use

http://phx.corporate-ir.net/phoenix.zhtml?c=176060&p=irol-corporateTimeline

Using AWS Services

- AWS Management Console
  - Easy to use, great for manual configurations
  - Use **username** / **password** provided
- Command line tools
  - For writing scripts
    - e.g., create a set of machines to analyze data every day
  - Use **access key ID** and **secret access key,** or **certificates for EC2**
- AWS API
  - Integrating cloud services into your applications
    - e.g., storing data on the cloud, running computation in the background
  - Use **access key ID** and **secret access key, or certificates for EC2**
- SSH into EC2 instances is performed using a different keypair

Setting up an AWS account


aws.amazon.com

What is Cloud Computing with AWS?

Learn about the benefits of Cloud Computing with AWS.

Watch the 3 minute video »

Launch virtual machines and apps in minutes.

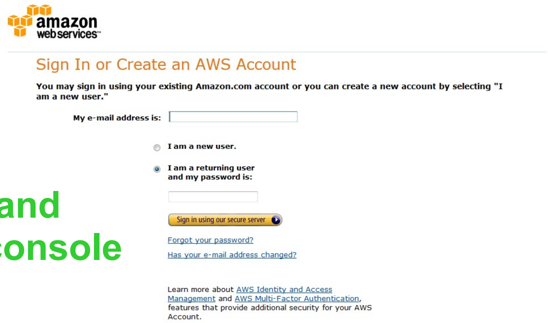
```
> aws s3 sync _
```

- Sign up for an account on aws.amazon.com
  - **You need to choose an username and a password**
  - These are for the management interface only
  - **Your programs will use other credentials (RSA keypairs, access keys, ...) to interact with AWS**

AWS credentials

Command-line tools
SOAP APIs

AWS web site and management console



Sign In or Create an AWS Account

You may sign in using your existing Amazon.com account or you can create a new account by selecting "I am a new user."

My e-mail address is:

○ **I am a new user.**

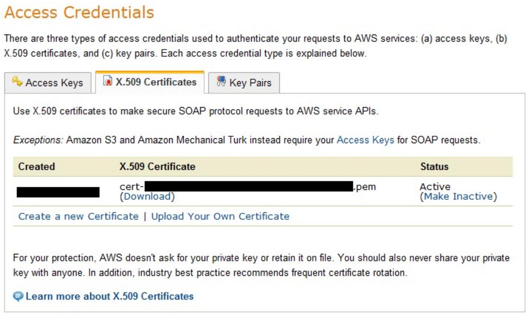
● **I am a returning user and my password is:**

Forgot your password?

Has your e-mail address changed?

Learn more about AWS Identity and Access Management and AWS Multi-Factor Authentication, features that provide additional security for your AWS Account.

Sign-in credentials



Access Credentials

There are three types of access credentials used to authenticate your requests to AWS services: (a) access keys, (b) X.509 certificates, and (c) key pairs. Each access credential type is explained below.

Use X.509 certificates to make secure SOAP protocol requests to AWS service APIs.

Exceptions: Amazon S3 and Amazon Mechanical Turk instead require your Access Keys for SOAP requests.

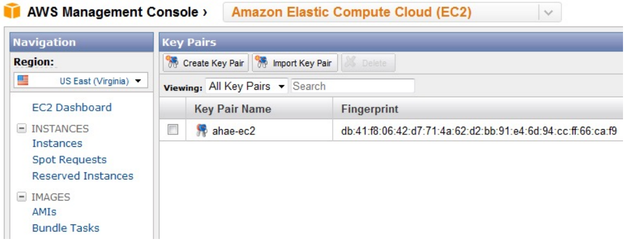
Created	X.509 Certificate	Status
[redacted]	cert-[redacted].pem (Download)	Active (Make Inactive)

Create a new Certificate | Upload Your Own Certificate

For your protection, AWS doesn't ask for your private key or retain it on file. You should also never share your private key with anyone. In addition, industry best practice recommends frequent certificate rotation.

Learn more about X.509 Certificates

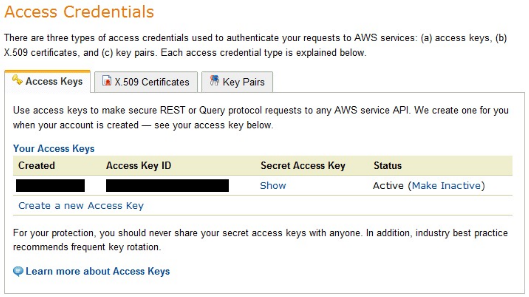
X.509 certificates



AWS Management Console › Amazon Elastic Compute Cloud (EC2)

Viewing: All Key Pairs Search

	Key Pair Name	Fingerprint
□	ahae-ec2	db:41:f8:06:42:d7:71:4a:62:d2:bb:91:e4:6d:94:cc:ff:66:ca:f9



Access Credentials

There are three types of access credentials used to authenticate your requests to AWS services: (a) access keys, (b) X.509 certificates, and (c) key pairs. Each access credential type is explained below.

Use access keys to make secure REST or Query protocol requests to any AWS service API. We create one for you when your account is created — see your access key below.

Your Access Keys

Created	Access Key ID	Secret Access Key	Status
[redacted]	[redacted]	Show	Active (Make Inactive)

Create a new Access Key

For your protection, you should never share your secret access keys with anyone. In addition, industry best practice recommends frequent key rotation.

Learn more about Access Keys

Connecting to an instance (e.g., via ssh)

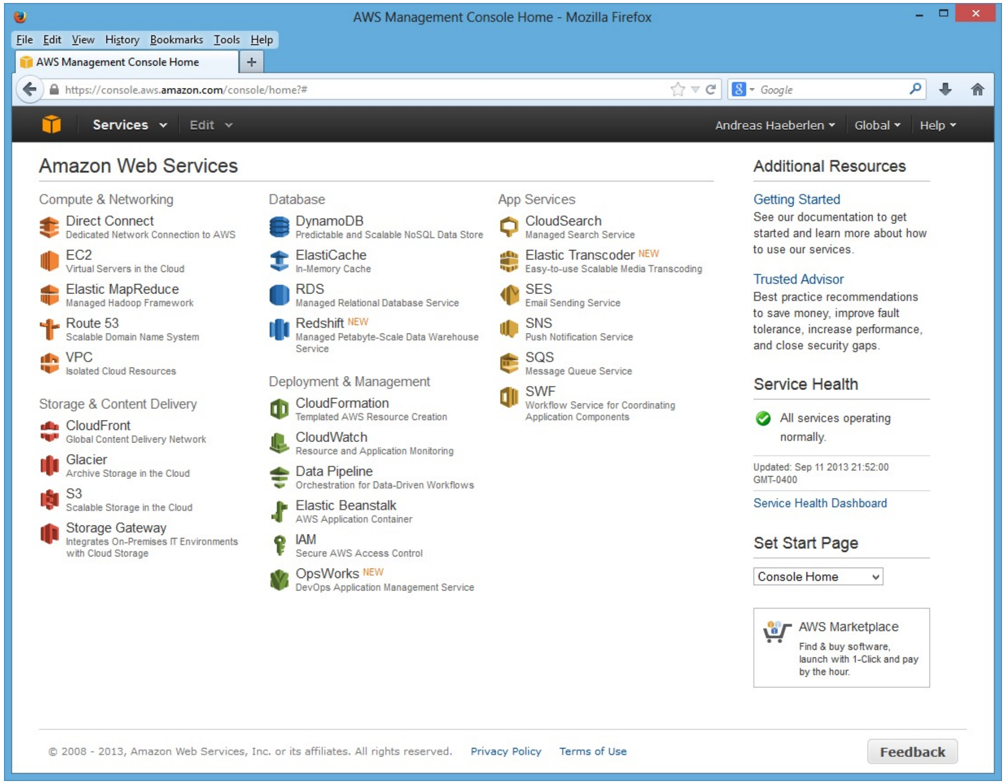
EC2 key pairs

Access keys

REST APIs

- Why so many different types of credentials?

The AWS management console


AWS Management Console Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
AWS Management Console Home
https://console.aws.amazon.com/console/home?#
Google
Services
Edit
Andreas Haeberlen
Global
Help

Amazon Web Services

Compute & Networking

- Direct Connect — Dedicated Network Connection to AWS
- EC2 — Virtual Servers in the Cloud
- Elastic MapReduce — Managed Hadoop Framework
- Route 53 — Scalable Domain Name System
- VPC — Isolated Cloud Resources

Storage & Content Delivery

- CloudFront — Global Content Delivery Network
- Glacier — Archive Storage in the Cloud
- S3 — Scalable Storage in the Cloud
- Storage Gateway — Integrates On-Premises IT Environments with Cloud Storage

Database

- DynamoDB — Predictable and Scalable NoSQL Data Store
- ElastiCache — In-Memory Cache
- RDS — Managed Relational Database Service
- Redshift NEW — Managed Petabyte-Scale Data Warehouse Service

Deployment & Management

- CloudFormation — Templated AWS Resource Creation
- CloudWatch — Resource and Application Monitoring
- Data Pipeline — Orchestration for Data-Driven Workflows
- Elastic Beanstalk — AWS Application Container
- IAM — Secure AWS Access Control
- OpsWorks NEW — DevOps Application Management Service

App Services

- CloudSearch — Managed Search Service
- Elastic Transcoder NEW — Easy-to-use Scalable Media Transcoding
- SES — Email Sending Service
- SNS — Push Notification Service
- SQS — Message Queue Service
- SWF — Workflow Service for Coordinating Application Components

Additional Resources

Getting Started
See our documentation to get started and learn more about how to use our services.

Trusted Advisor
Best practice recommendations to save money, improve fault tolerance, increase performance, and close security gaps.

Service Health

All services operating normally.

Updated: Sep 11 2013 21:52:00 GMT-0400

Service Health Dashboard

Set Start Page

Console Home

AWS Marketplace
Find & buy software, launch with 1-Click and pay by the hour.

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

- Used to control many AWS services:
  - For example, start/stop EC2 instances, create S3 buckets...

REST and SOAP

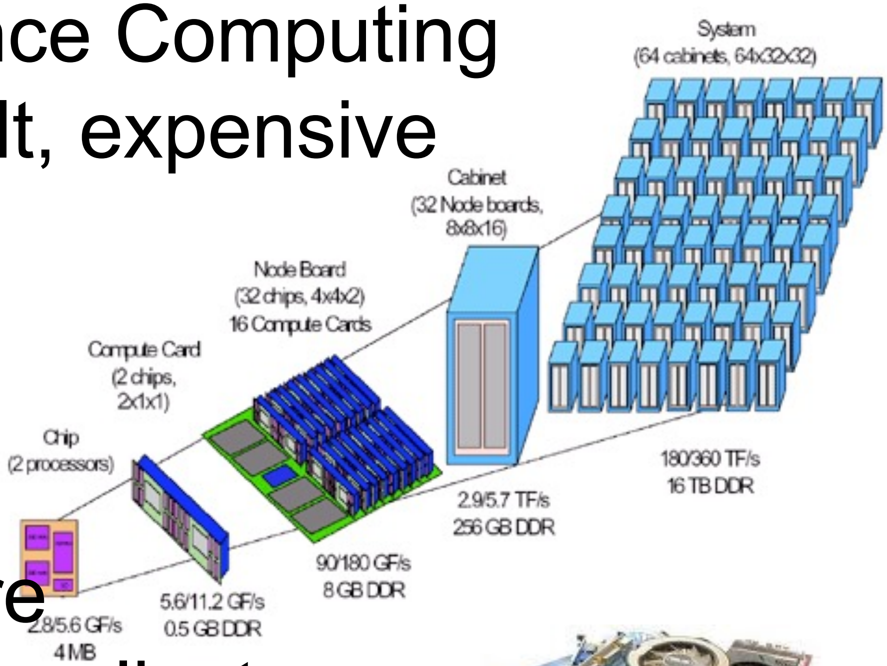
- How do your programs access AWS?
  - Via the REST or SOAP protocols
  - Example: Launch an EC2 instance, store a value in S3, ...
- Simple Object Access protocol (SOAP)
  - Not as simple as the name suggests
  - XML-based, extensible, general, standardized, but also somewhat heavyweight and verbose
  - Increasingly deprecated (e.g., for SimpleDB and EC2)
- Representational State Transfer (REST)
  - Much simpler to develop than SOAP
  - Web-specific; lack of standards

How do we scale up processing for Big Data ?

Or: How to run a Job on MANY potentially FAULTY boxes ?

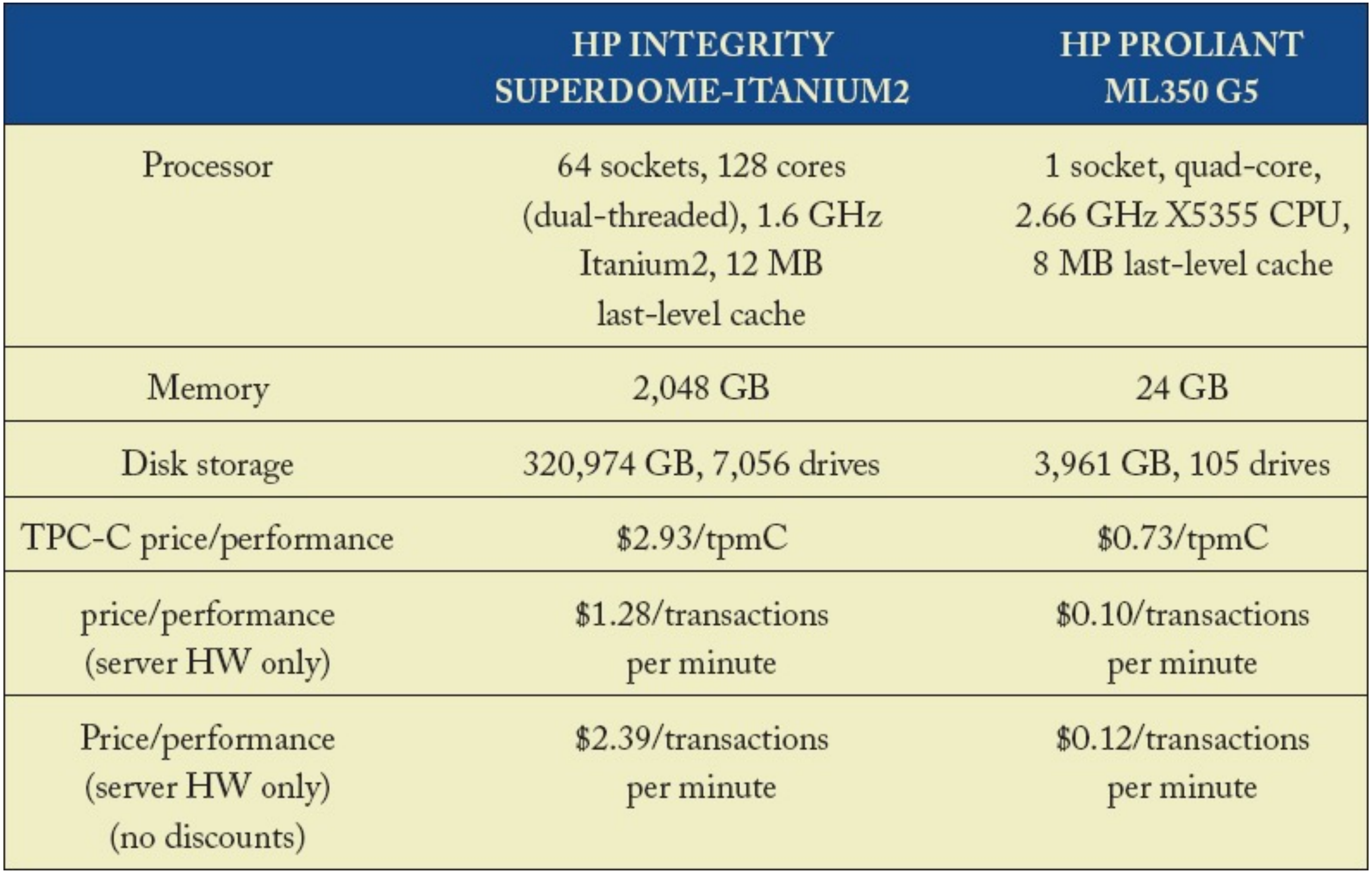
Using Commodity Hardware

- 80-90’s: High Performance Computing
  Very reliable, custom built, expensive



- Now: Consumer hardware
  Cheap, efficient, easy to replicate,
  BUT not very reliable,
- MUST deal with it!

Why commodity machines?



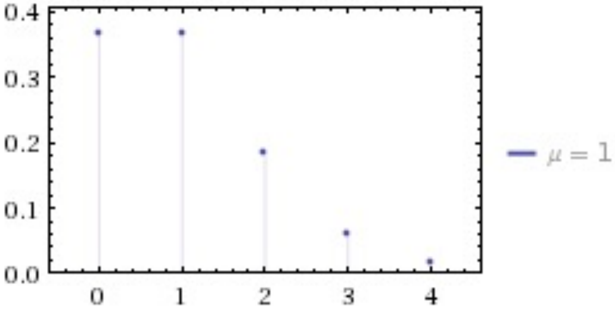
	HP INTEGRITY SUPERDOME-ITANIUM2	HP PROLIANT ML350 G5
Processor	64 sockets, 128 cores (dual-threaded), 1.6 GHz Itanium2, 12 MB last-level cache	1 socket, quad-core, 2.66 GHz X5355 CPU, 8 MB last-level cache
Memory	2,048 GB	24 GB
Disk storage	320,974 GB, 7,056 drives	3,961 GB, 105 drives
TPC-C price/performance	$2.93/tpmC	$0.73/tpmC
price/performance (server HW only)	$1.28/transactions per minute	$0.10/transactions per minute
Price/performance (server HW only) (no discounts)	$2.39/transactions per minute	$0.12/transactions per minute

Source: Barroso and Urs Hölzle (2009); performance figures from late 2007

Fault Tolerance

- Performance goal
  - 1 failure per year
  - for a 1000-machine Cluster
- Poisson approximation

$$\Pr(n) = \frac{1}{n!} e^{-\mu} \mu^n$$



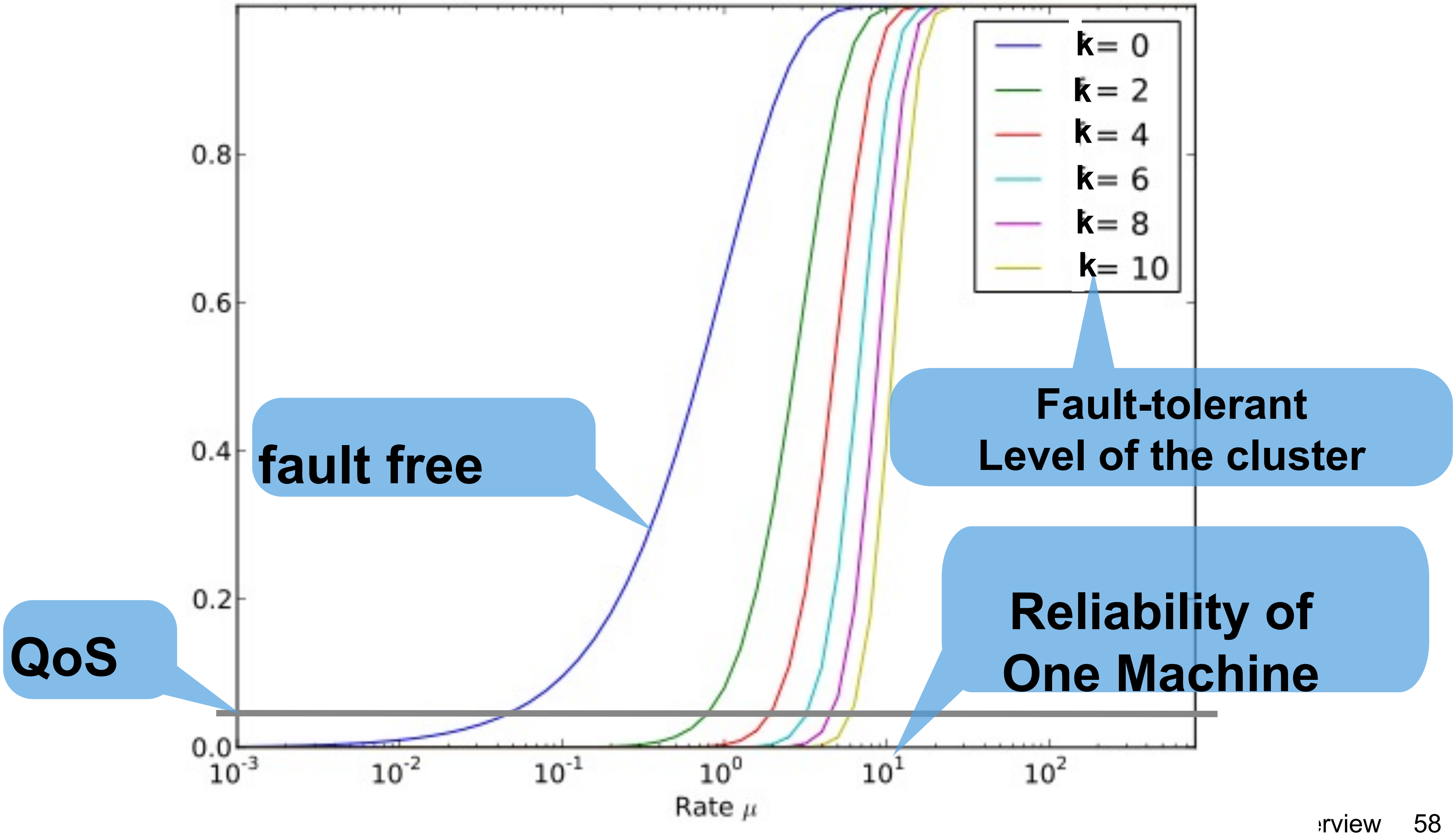
  - Assume failure rate μ per machine
  - Poisson rates of independent random variables are additive, so we can combine

=> With Fault Intolerant Engineering
We need a rate of 1 failure per 1000 years per machine

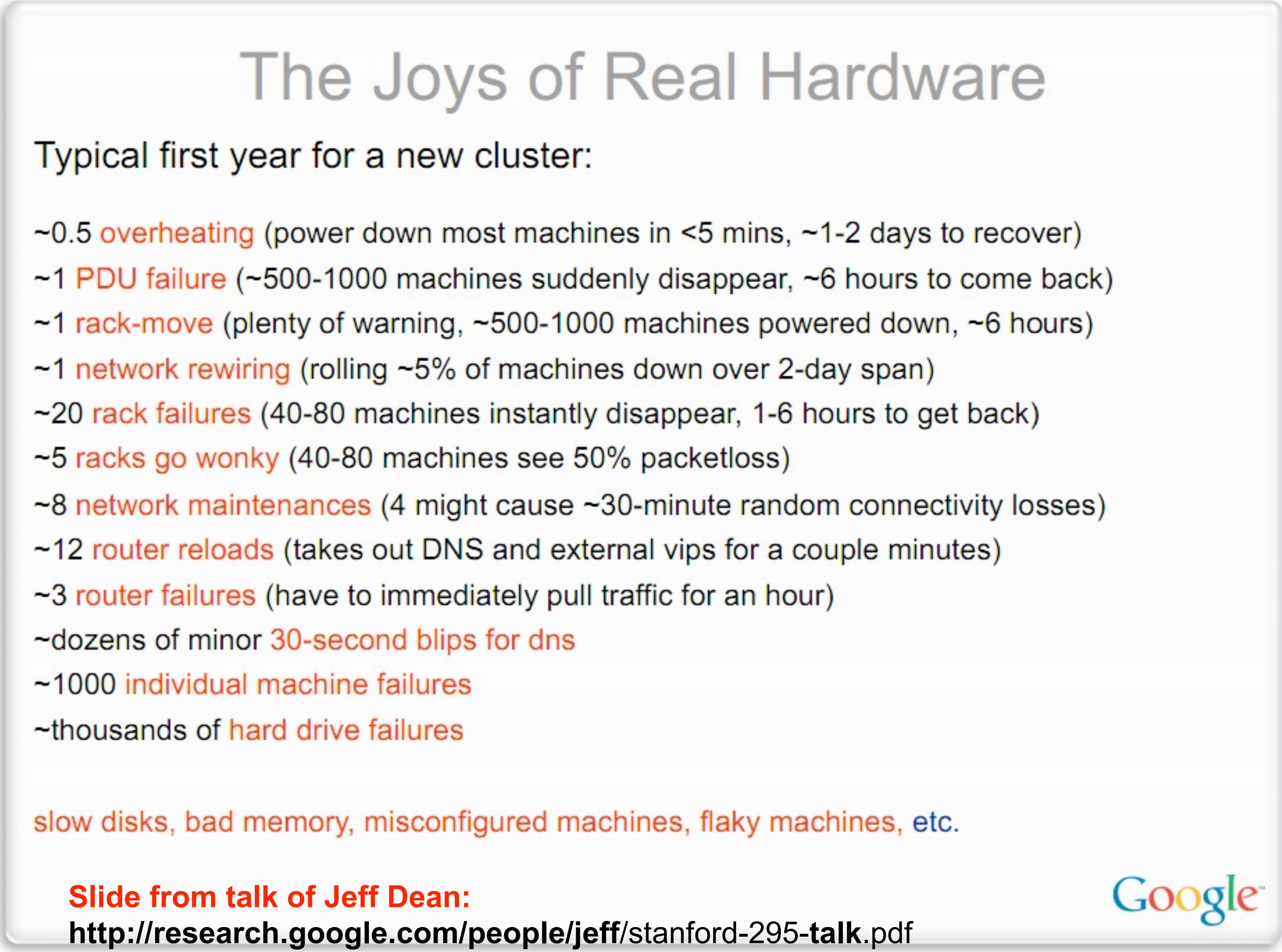
- Fault tolerance
Assume we can tolerate k faults among m machines in t time units

$$\Pr(f > k) = 1 - \sum_{n=0}^{k} \frac{1}{n!} e^{-\lambda t} (\lambda t)^n$$

Fault tolerance



Performance Characteristics of Hardware in a Datacenter-scale Computer



The Joys of Real Hardware

Typical first year for a new cluster:

~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)

~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hours to come back)

~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hours)

~1 network rewiring (rolling ~5% of machines down over 2-day span)

~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)

~5 racks go wonky (40-80 machines see 50% packetloss)

~8 network maintenances (4 might cause ~30-minute random connectivity losses)

~12 router reloads (takes out DNS and external vips for a couple minutes)

~3 router failures (have to immediately pull traffic for an hour)

~dozens of minor 30-second blips for dns

~1000 individual machine failures

~thousands of hard drive failures

slow disks, bad memory, misconfigured machines, flaky machines, etc.

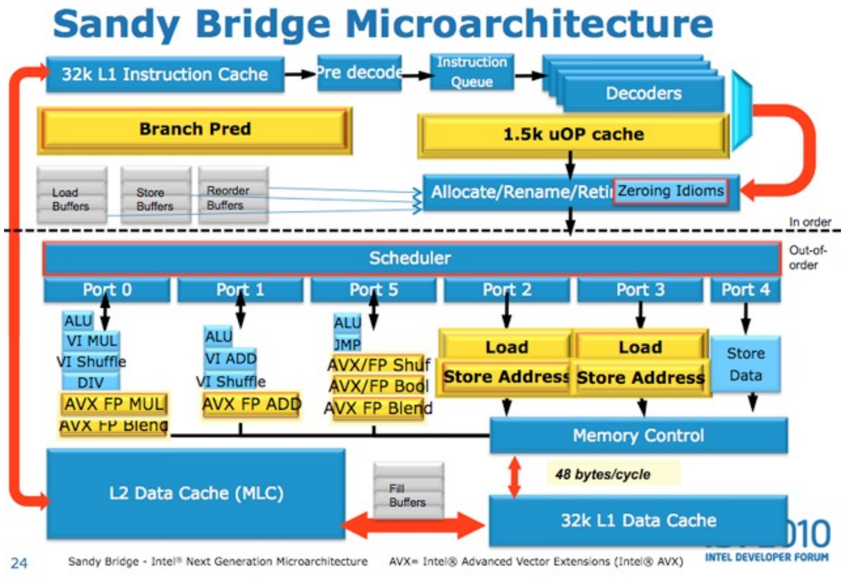
Slide from talk of Jeff Dean:
http://research.google.com/people/jeff/stanford-295-**talk**.pdf

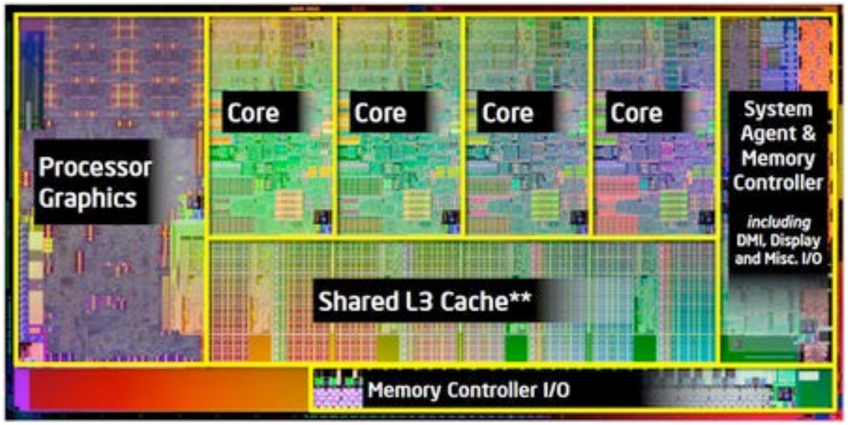
“Facts” about Jeff Dean

- Compilers don’t warn Jeff Dean. Jeff Dean warns compilers.
- Jeff Dean builds his code before committing it, but only to check for compiler and linker bugs.
- Jeff Dean writes directly in binary. He then writes the source code as a documentation for other developers.
- Jeff Dean once shifted a bit so hard, it ended up on another computer.
- When Jeff Dean has an ergonomic evaluation, it is for the protection of his keyboard.
- gcc -O4 emails your code to Jeff Dean for a rewrite.
- When he heard that Jeff Dean's autobiography would be exclusive to the platform, Richard Stallman bought a Kindle.
- Jeff Dean puts his pants on one leg at a time, but if he had more legs, you’d realize the algorithm is actually only O(log n)

CPU

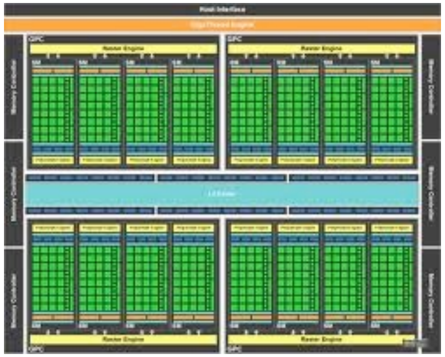
- Multiple cores (e.g. Intel Xeon E7 series has 4-24 cores per CPU @2016)
- Multiple sockets (1-4) per board
- 2-4 GHz clock
- 10-100W power
- Several cache levels (hierarchical, 8-16MB total)
- Vector processing units (SSE4, AVX) http://software.intel.com/en-us/avx
- Perform several operations at once
- Use this for fast linear algebra (4-8 multiply adds in one operation)
- Memory interface 20-40GB/s
- Internal bandwidth >100GB/s
- 100+ GFlops for matrix matrix multiply
- Integrated low end GPU

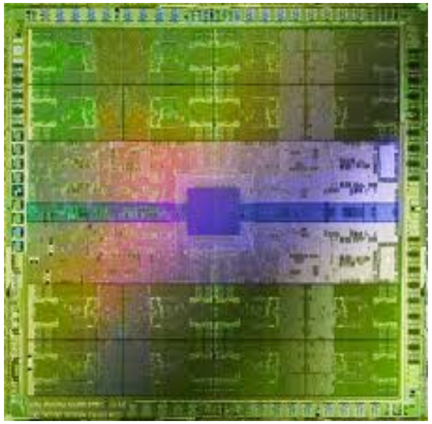




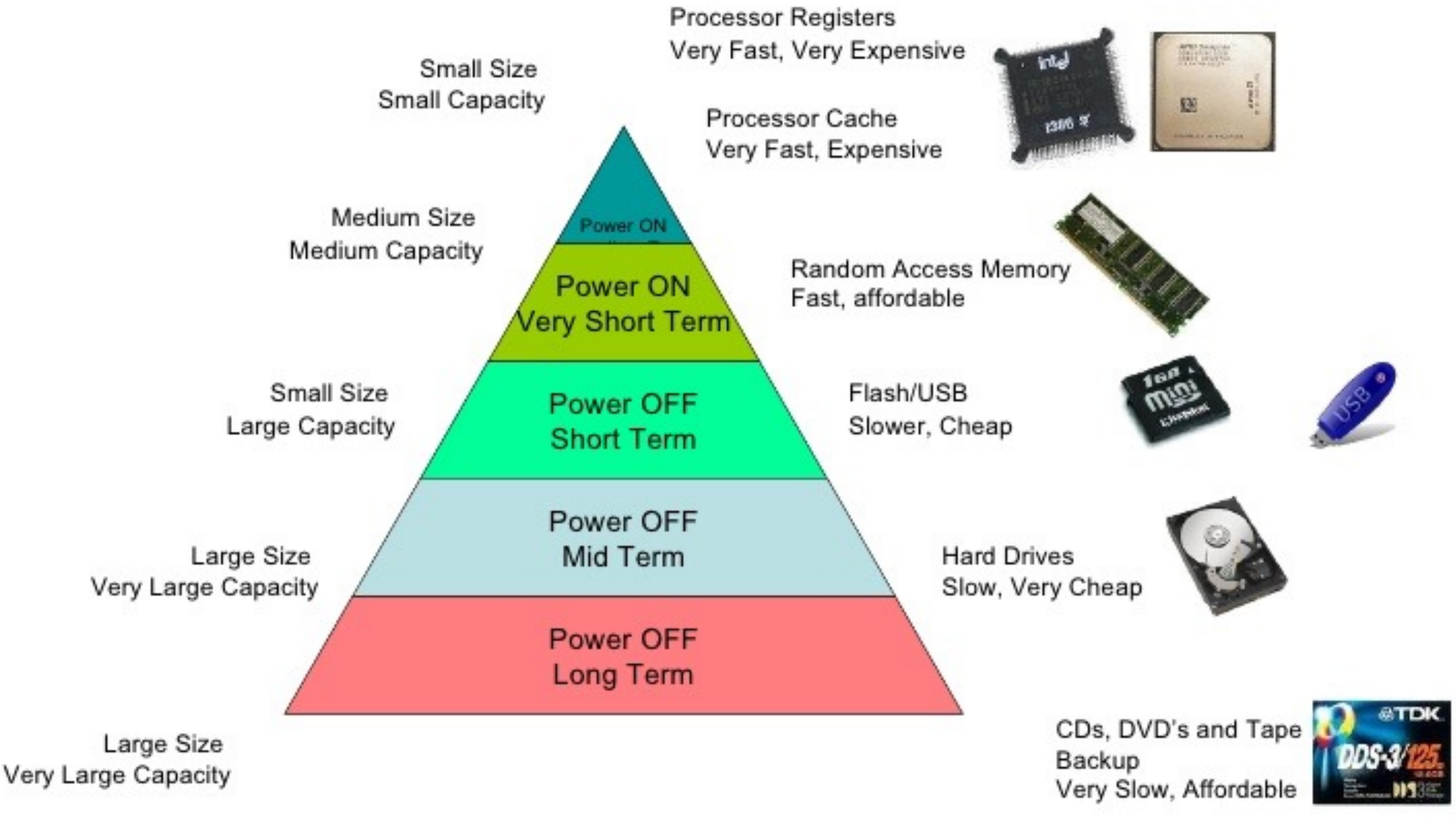
GPU

- nVidia GeForce10 has 400 to 4000 cores / drawing many 100's of Watt
- Cores have hierarchical structure tricky to synchronize threads (interrupts, semaphores, etc.)
- Upto 10's of GB memory (Tesla V100 has 32GB)
- 1 TFlop (single precision)
- Max. Memory Bandwidth ~1000GB/s
- 192GB/s PCIe 4.0 bus bottleneck?





Computer Memory Hierarchy



Processor Registers
Very Fast, Very Expensive

Small Size
Small Capacity

Processor Cache
Very Fast, Expensive

Power ON

Medium Size
Medium Capacity

Power ON
Very Short Term

Random Access Memory
Fast, affordable

Small Size
Large Capacity

Power OFF
Short Term

Flash/USB
Slower, Cheap

Power OFF
Mid Term

Large Size
Very Large Capacity

Hard Drives
Slow, Very Cheap

Power OFF
Long Term

Large Size
Very Large Capacity

CDs, DVD's and Tape
Backup
Very Slow, Affordable

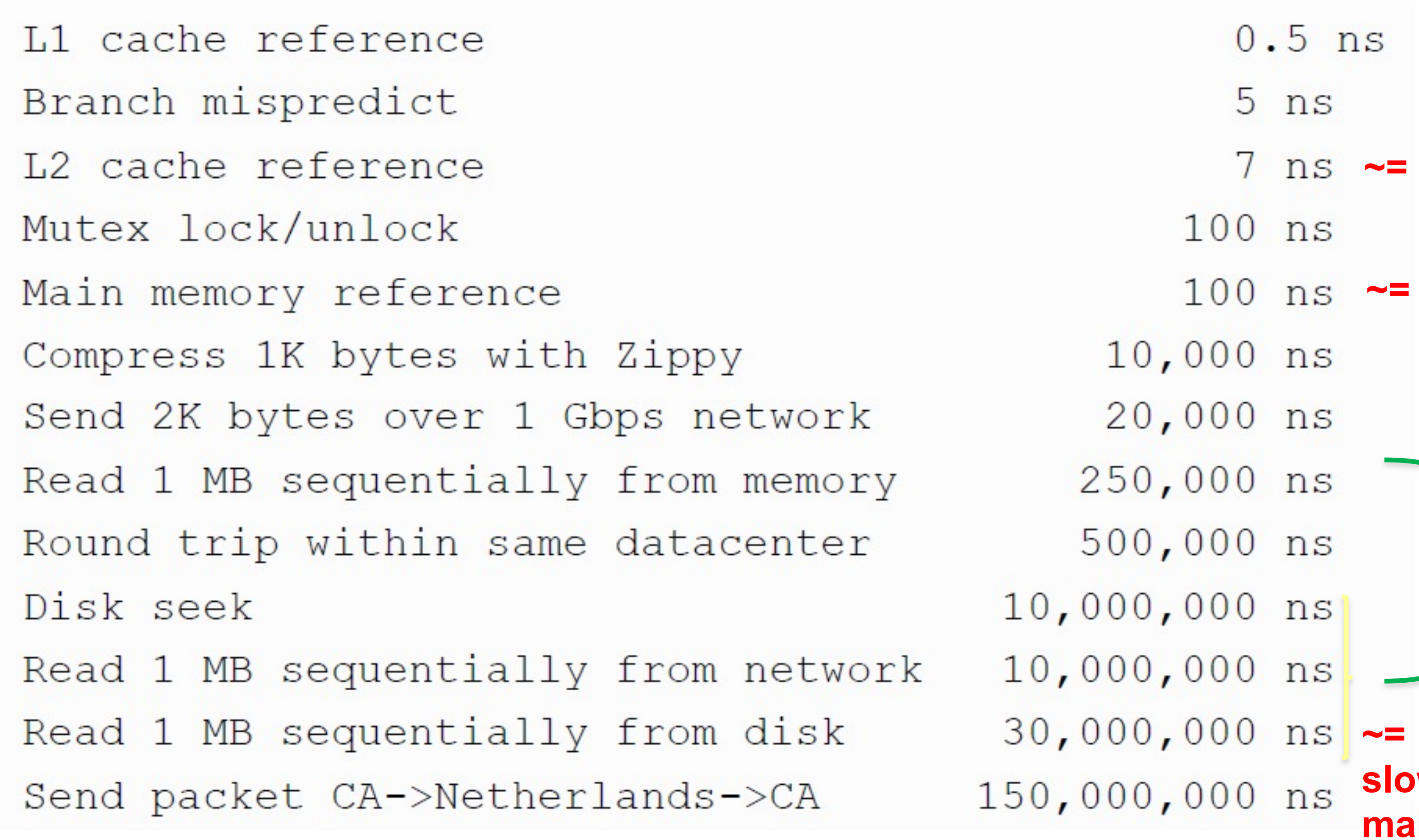
DRAM

- 2-4 channels (32 bit wide)
- 1GHz speed
- High latency (~10ns for DDR4)
- High burst data rate (>10 GB/s)
- Avoid random access in code if possible.
- Memory align variables
- Know your platform (FBDIMM vs. DDR)
  (code may run faster on old MacBookPro than a Xeon)

Storage

- Harddisks (SATA3 – 6 Gbps) circa 2018 -
  - 4-8 TB of storage (30GB/$)
  - 150 MB/s bandwidth (sequential)
  - 5 ms seek (200 IOPS)
  - cheap
- SSD (SATA3 – 6Gbps) circa 2018 -
  - 128-4096 GB storage (3-5GB/$)
  - 500 MB/s bandwidth (sequential read/write)
  - 100,000 IOPS / < 1 ms seek (queueing)
  - Reads a little faster than writes
    - e.g. 550 vs. 520 MB/s for Samsung 850Pro
  - reliable (but limited lifetime - NAND)
- NVMe (M.2 port) /PCIe SSD circa 2018 -
  - 128-2048GB storage
  - (3D XPoint: 0.7GB/$ - NAND-based 2 GB/$)
  - 1500 – 3500 MB/s (sequential read/write)
  - 150,000 – 500,000 IOPS

Numbers (Jeff Dean says) Everyone Should Know



L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from network	10,000,000 ns
Read 1 MB sequentially from disk	30,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

~= 10x

~= 15x

40x diff

~= 100,000x slower than main mem access

A typical disk

What do we count?

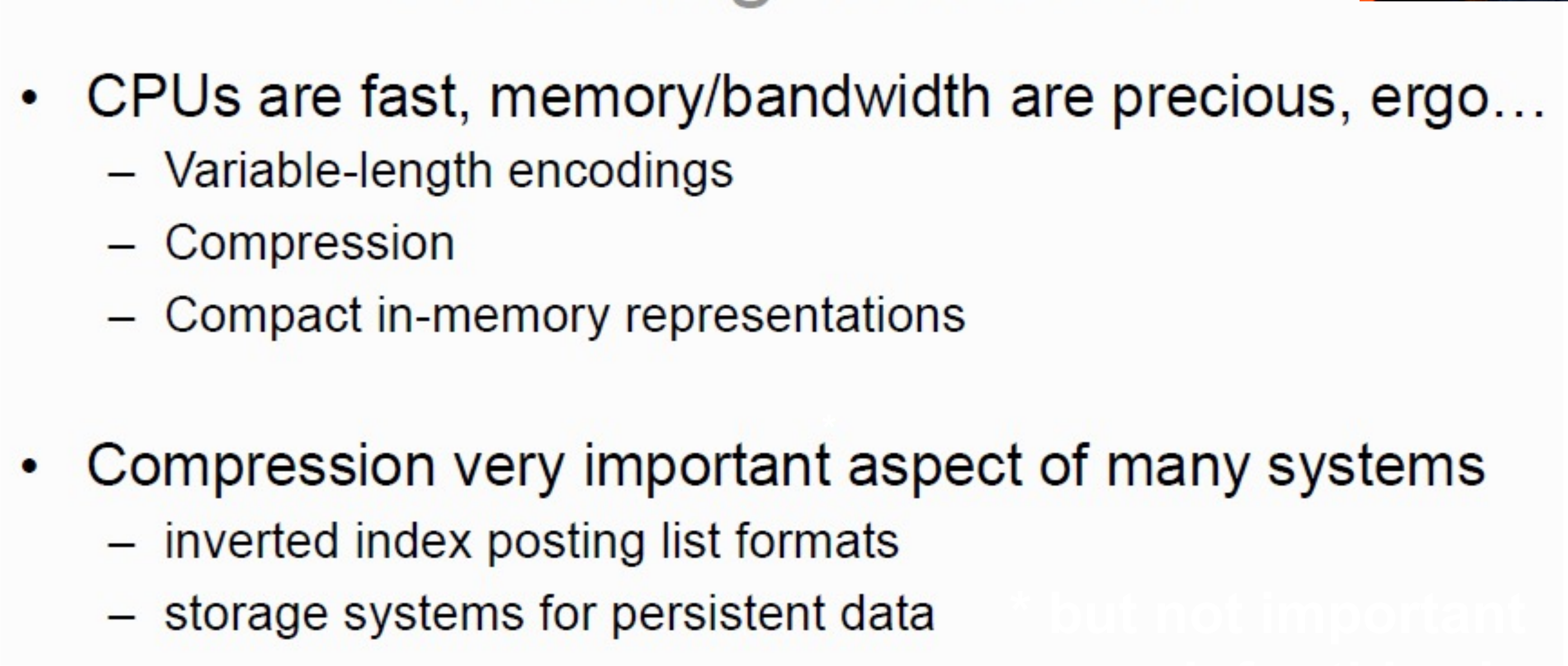
- Compilers don't warn Jeff Dean. Jeff Dean warns compilers.
- ….
- Memory access/instructions are *qualitatively different* from disk access
- Seeks are *qualitatively different* from sequential reads on disk
- Cache, disk fetches, etc work best when you stream through data *sequentially*
- **Best case for data processing: stream through the data *once* in *sequential order,* as it's found on disk.**

Seeks vs. Scans

- Consider a 1 TB database with 100 byte records
  - We want to update 1 percent of the records
- Scenario 1: random access
  - Each update takes ~30 ms (seek, read, write)
  - 10^8 updates = ~35 days
- Scenario 2: rewrite all records
  - Assume 100 MB/s throughput
  - Time = 5.6 hours(!)
- Lesson: avoid random seeks!

Source: Ted Dunning, on Hadoop mailing list

Other lessons (circa 2007)



Encoding Your Data

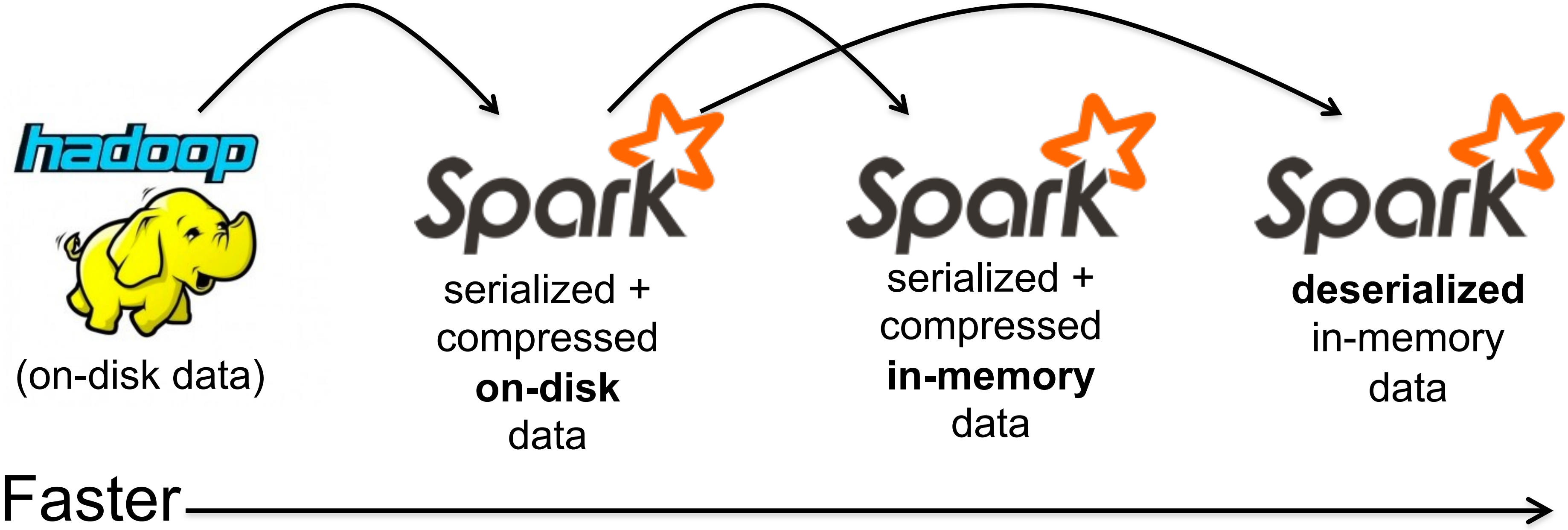
- CPUs are fast, memory/bandwidth are precious, ergo…
  - Variable-length encodings
  - Compression
  - Compact in-memory representations
- Compression very important aspect of many systems
  - inverted index posting list formats
  - storage systems for persistent data

* but not important

- This “conventional” wisdom may become out-dated already !

More recent Observations on Spark Performance Analysis

[K. Ousterhout et al, NSDI 2015]



up to 10x

spark.apache.org

I/O related

19%

6x or more

amplab.cs.berkeley.edu/benchmark/

(on-disk data)

serialized + compressed **on-disk** data

serialized + compressed **in-memory** data

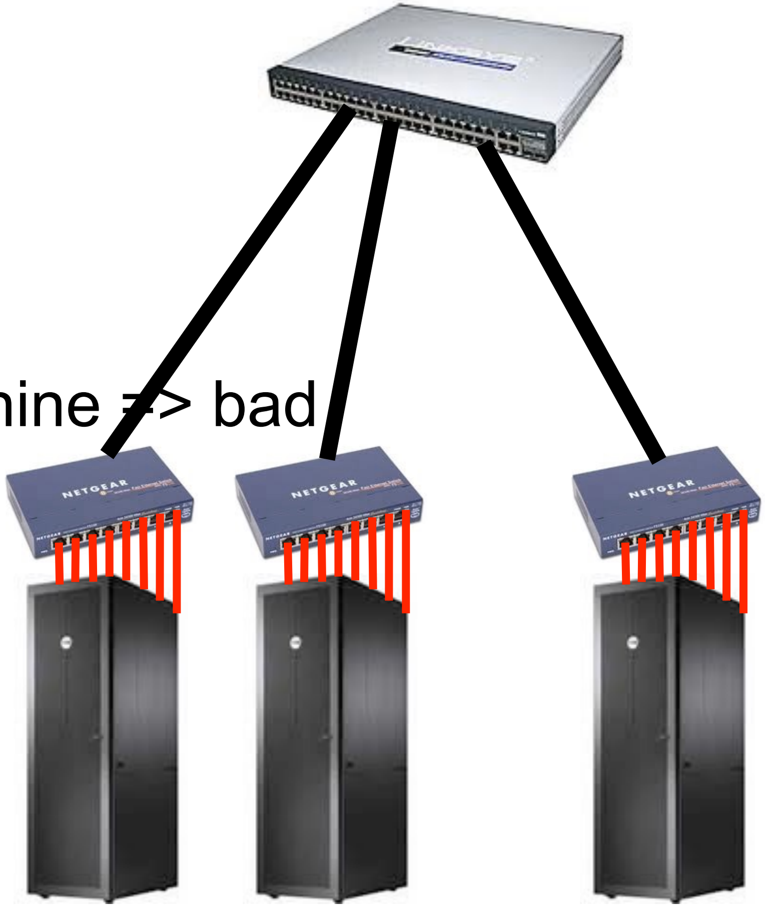
deserialized in-memory data

Faster

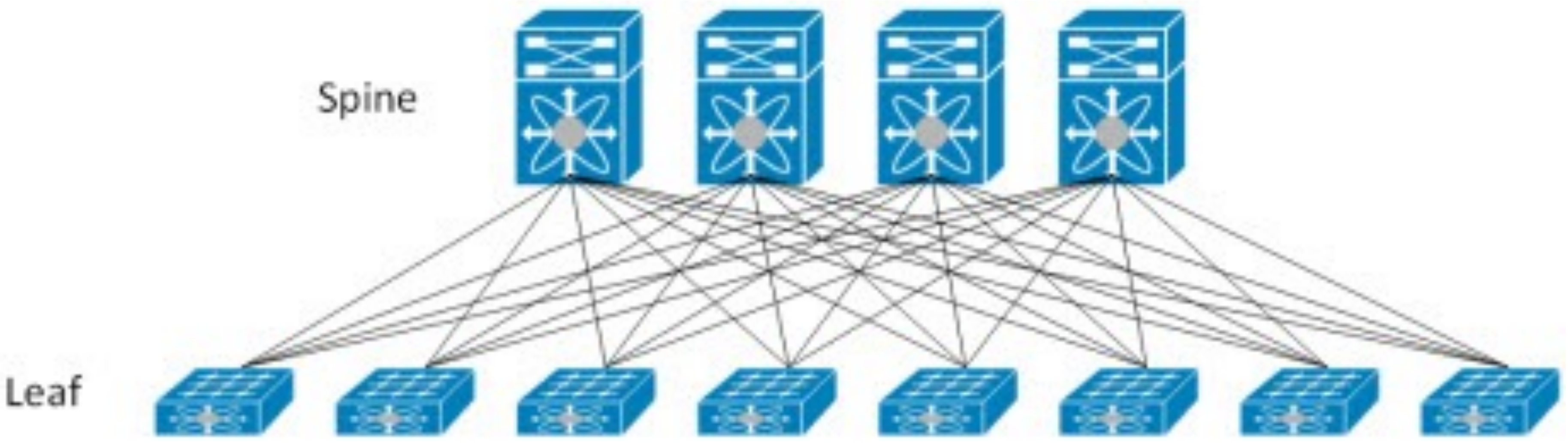
Switches & Colos

- In theory perfect point to point bandwidth (e.g. 10Gb Ethernet)
- Big switches are expensive
  crossbar bandwidth linear in #ports,

  BUT price superlinear
- Real switches have finite buffers
  - many connections to a single machine => bad
  - buffer overflow / dropped packets / collision avoidance
- Hierarchical structure
  - more bandwidth within rack
  - lower latency within rack
  - lots of latency between Colos
- Hadoop gives you machines where the data is (not necessarily on same rack!)



Clos Networks' Reappearance in Datacenter Networks (aka the Spine and Leaf Topology, or Folded Clos, or Fat-Trees)



The Top of Rack (ToR) switches are the Leaf switches

Each ToR is connected to multiple Core switches which represent the Spine.

# of Uplinks (of each ToR) = # of Spine switches

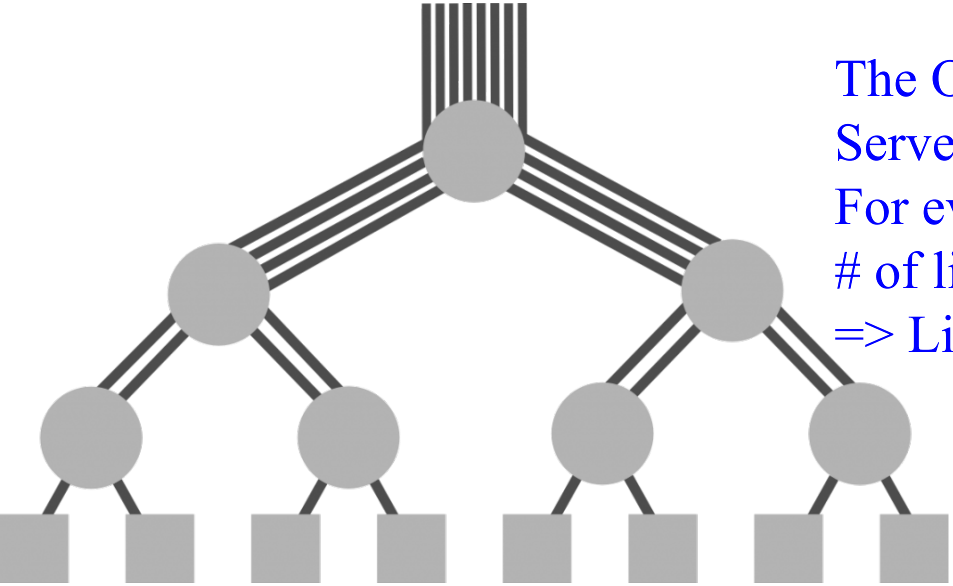
# of Downlinks (of each Spine switch) = # of Leaf switches

Multiple ECMP exists for every pair of Leaf switches

Support Incrementally "Scale-out" by adding more Leaf and Spine switches

"Jupiter Rising: A Decade of Clos Topologies and Central Control in Google's Datacenter Networks," ACM Sigcomm 2015.

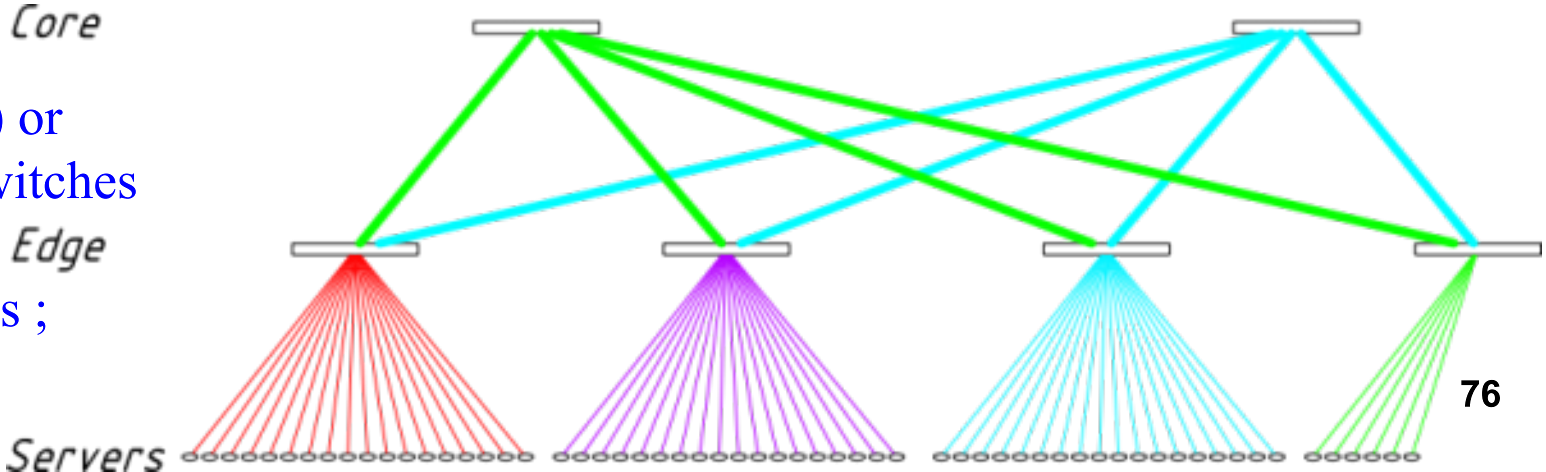
Clos Networks' Reappearance in Datacenter Networks (aka the Spine and Leaf Topology, or Folded Clos, or Fat-Trees)



The Original Fat-Tree Topology [Leiserson 85]:
Servers (Processors) are the leafs ;
For every non-leaf node (Switch) in the tree,
# of links to its Parent = # of links to its Children
=> Links at "Fatter" towards the top of the tree

Source: http://clusterdesign.org/fat-trees/

Example:
All Leaf (Edge) or
Spine (Core) switches
are identical
36-port switches ;



Communication Cost ?

- Nodes need to talk to each other!
  - SMP (Symmetric Multi-Processor machine): latencies ~100 ns
  - LAN: latencies ~100 υs
- Scaling “up” vs. scaling “out”
  - Smaller cluster of SMP machines vs. larger cluster of commodity machines
  - E.g., 8 128-core machines vs. 128 8-core machines
  - Note: no single SMP machine is big enough
- Let’s model communication overhead…

Source: analysis on this and subsequent slides from Barroso and Urs Hölzle, “The Datacenter as a Computer,” 2009.

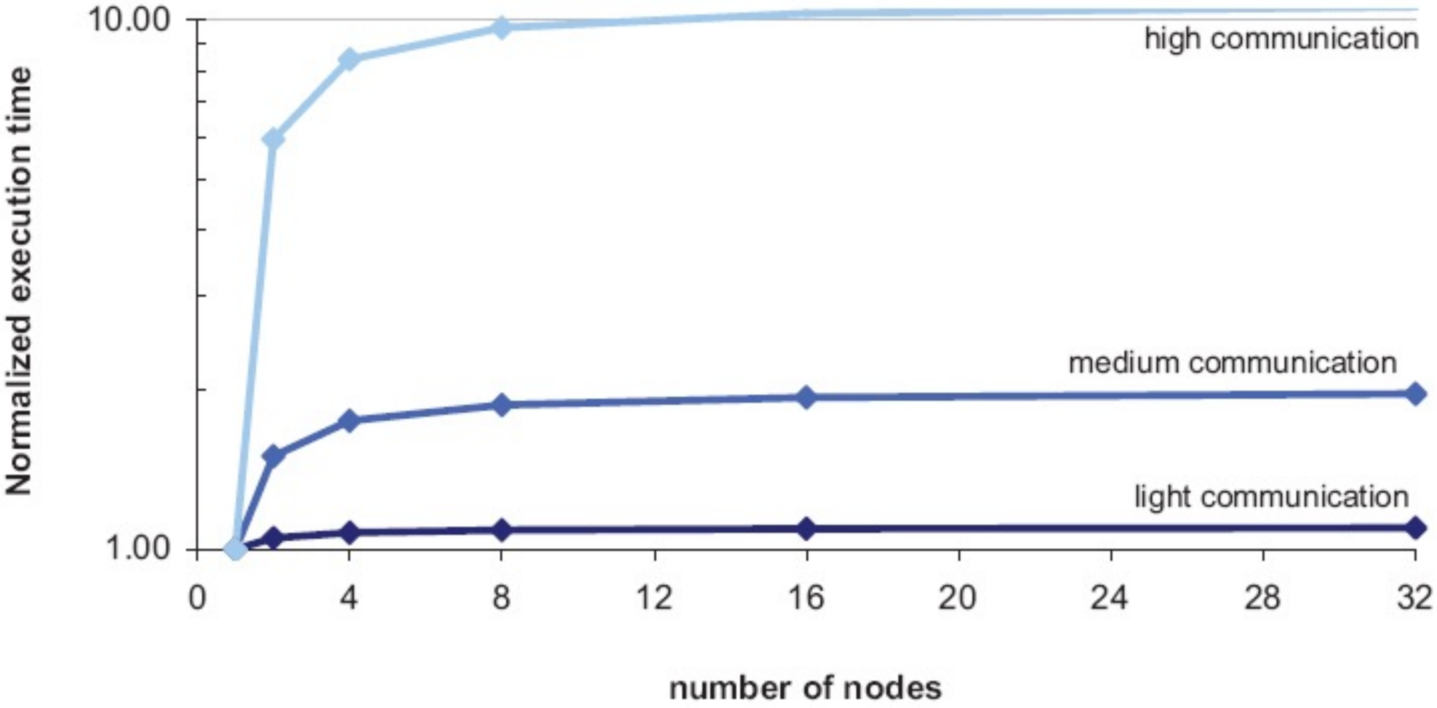
Modeling Communication Costs

- Simple execution cost model:
  - Fraction of local access inversely proportional to size of cluster
  - *n* nodes (each node is a shared-memory SMP domain, ignoring cores for now)
  - Total cost = cost of computation + cost to access global data

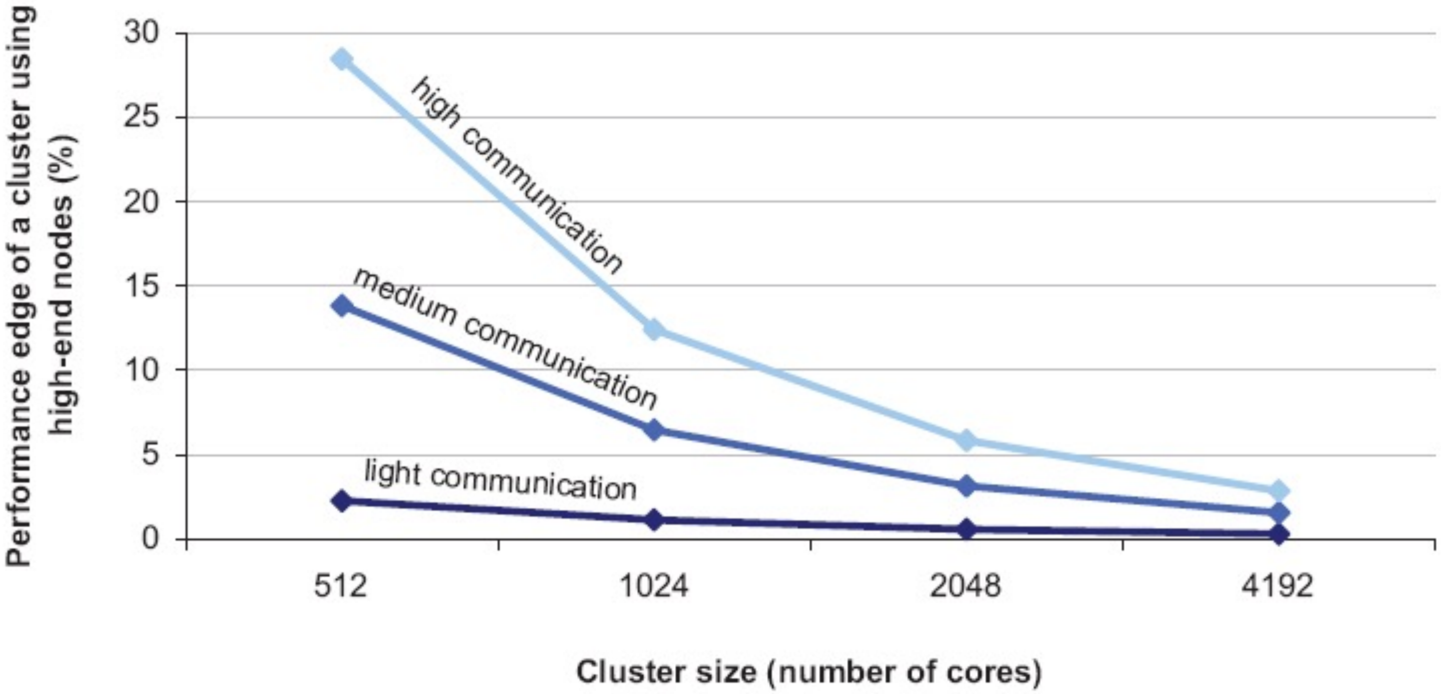
    = 1 ms + f x [100 ns / n + 100 υs x (1 - 1/n)]

    - Light communication: $f=1$
    - Medium communication: $f=10$
    - Heavy communication: $f=100$
- What are the costs in parallelization?

Cost of Parallelization



Advantages of scaling “out”



So why not?

Data Intensive Computing

- Data collection too large to transmit economically over Internet --- Petabyte data collections
- Computation produces small data output containing a high density of information
- Implemented in “Clouds”
- Easy to write programs, fast turn around.
- MapReduce.
  - Map(k1, v1) -> list (k2, v2)
  - Reduce(k2,list(v2)) -> list(v3)
- Hadoop (Yarn), PIG, HDFS, Hbase
- Sawzall, Google File System, BigTable

The datacenter *is* the computer

“Big Ideas”

- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour

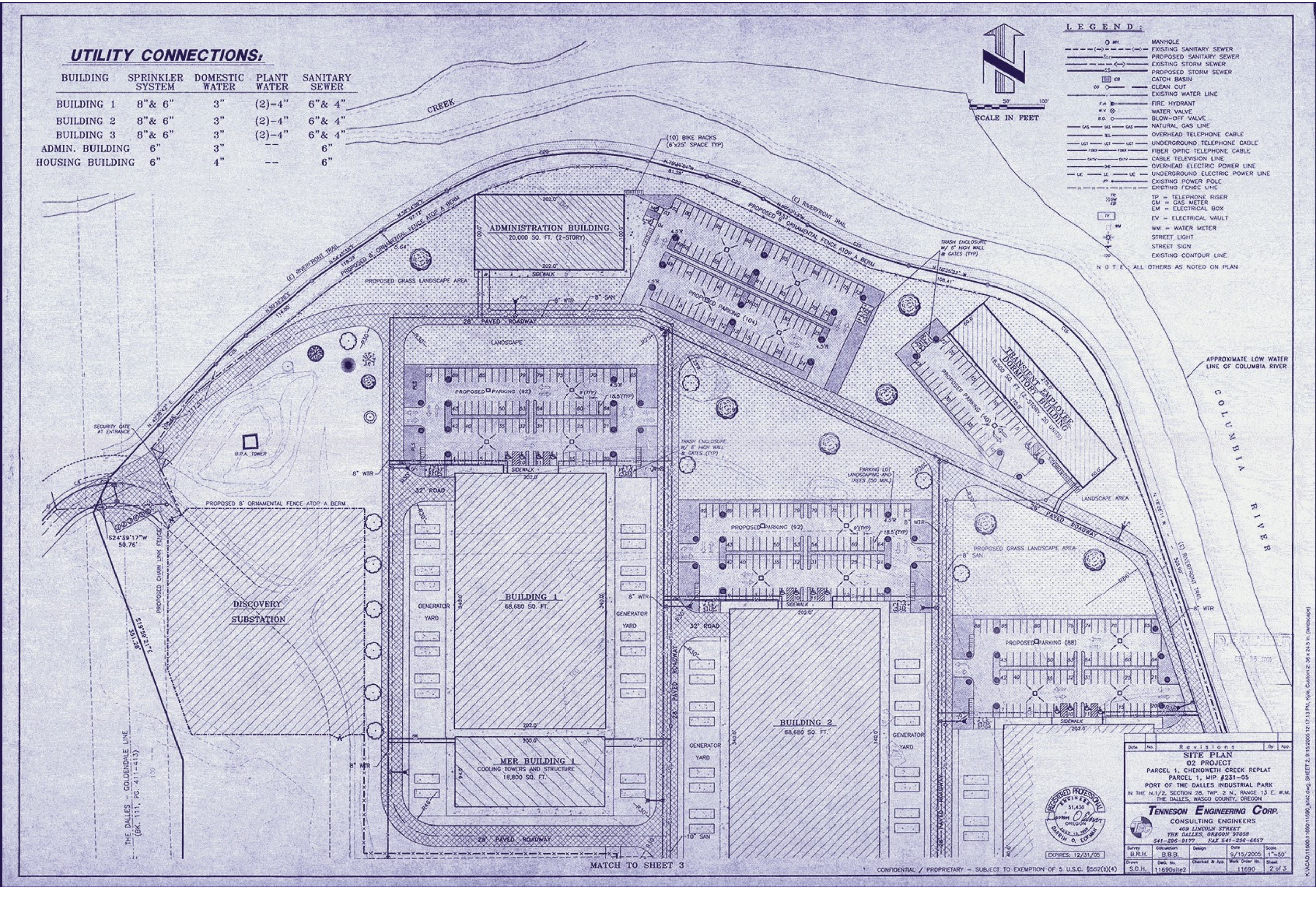
“Big Ideas”

- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour

Source: Wikipedia (The Dalles, Oregon)

Source: NY Times (6/14/2006)

Source: www.robinmajumdar.com



CREEK
SCALE IN FEET
0
50'
100'
(E) RIVERFRONT TRAIL
PROPOSED 8' ORNAMENTAL FENCE ATOP A BERM
(10) BIKE RACKS (6'x25' SPACE TYP)
ADMINISTRATION BUILDING
20,000 SQ. FT. (2-STORY)
SIDEWALK
PROPOSED GRASS LANDSCAPE AREA
PROPOSED PARKING (104)
TRASH ENCLOSURE W/ 6' HIGH WALL & GATES (TYP)
TRANSIENT EMPLOYEE DORMITORY BUILDING
16,500 SQ. FT. (2-STORY, 20 UNITS)
PROPOSED PARKING (40)
APPROXIMATE LOW WATER LINE OF COLUMBIA RIVER
COLUMBIA RIVER
28' PAVED ROADWAY
LANDSCAPE
PROPOSED PARKING (92)
SECURITY GATE AT ENTRANCE
B.P.A. TOWER
8" WTR
8" SAN
32' ROAD
PARKING LOT LANDSCAPING AND TREES (50 MIN.)
LANDSCAPE AREA
PROPOSED 8' ORNAMENTAL FENCE ATOP A BERM
S24°39'17"W
50.76'
PROPOSED CHAIN LINK FENCE
DISCOVERY SUBSTATION
GENERATOR YARD
BUILDING 1
68,680 SQ. FT.
202.0'
340.0'
MER BUILDING 1
COOLING TOWERS AND STRUCTURE
18,800 SQ. FT.
BUILDING 2
68,680 SQ. FT.
PROPOSED PARKING (88)
PROPOSED GRASS LANDSCAPE AREA
THE DALLES - GOLDENDALE LINE (BK. 111, PG. 411-413)
10" SAN
MATCH TO SHEET 3

UTILITY CONNECTIONS:

BUILDING	SPRINKLER SYSTEM	DOMESTIC WATER	PLANT WATER	SANITARY SEWER
BUILDING 1	8"& 6"	3"	(2)-4"	6"& 4"
BUILDING 2	8"& 6"	3"	(2)-4"	6"& 4"
BUILDING 3	8"& 6"	3"	(2)-4"	6"& 4"
ADMIN. BUILDING	6"	3"	--	6"
HOUSING BUILDING	6"	4"	--	6"

LEGEND:

MANHOLE
EXISTING SANITARY SEWER
PROPOSED SANITARY SEWER
EXISTING STORM SEWER
PROPOSED STORM SEWER
CATCH BASIN
CLEAN OUT
EXISTING WATER LINE
FIRE HYDRANT
WATER VALVE
BLOW-OFF VALVE
NATURAL GAS LINE
OVERHEAD TELEPHONE CABLE
UNDERGROUND TELEPHONE CABLE
FIBER OPTIC TELEPHONE CABLE
CABLE TELEVISION LINE
OVERHEAD ELECTRIC POWER LINE
UNDERGROUND ELECTRIC POWER LINE
EXISTING POWER POLE
EXISTING FENCE LINE
TP = TELEPHONE RISER
GM = GAS METER
EM = ELECTRICAL BOX
EV = ELECTRICAL VAULT
WM = WATER METER
STREET LIGHT
STREET SIGN
EXISTING CONTOUR LINE

NOTE: ALL OTHERS AS NOTED ON PLAN

REGISTERED PROFESSIONAL ENGINEER 51,430 OREGON DARRIN O. ECKMAN

EXPIRES: 12/31/05

CONFIDENTIAL / PROPRIETARY - SUBJECT TO EXEMPTION OF 5 U.S.C. §552(b)(4)

Date	No.	Revisions	By	App.

SITE PLAN
02 PROJECT
PARCEL 1, CHENOWETH CREEK REPLAT
PARCEL 1, MIP #231-05
PORT OF THE DALLES INDUSTRIAL PARK
IN THE N.1/2, SECTION 28, TWP. 2 N., RANGE 13 E. W.M.
THE DALLES, WASCO COUNTY, OREGON

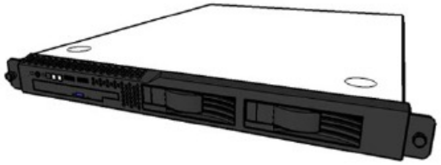
TENNESON ENGINEERING CORP.
CONSULTING ENGINEERS
409 LINCOLN STREET
THE DALLES, OREGON 97058
541-296-9177 FAX 541-296-6657

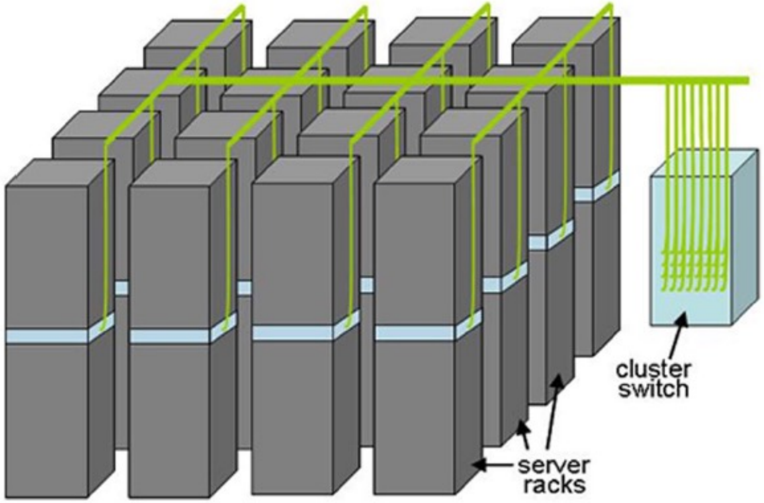
Survey	Calculation	Design	Date	Scale
B.R.H.	B.B.B.		9/15/2005	1"=50'
Drawn	Dwg. No.	Checked & App.	Work Order No.	Sheet
S.D.H.	11690site2		11690	2 of 3

Source: Harper's (Feb, 2008)

Source: Bonneville Power Administration

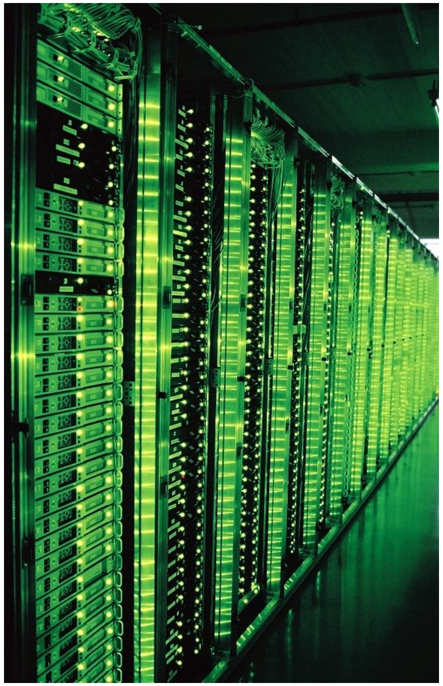
Building Blocks





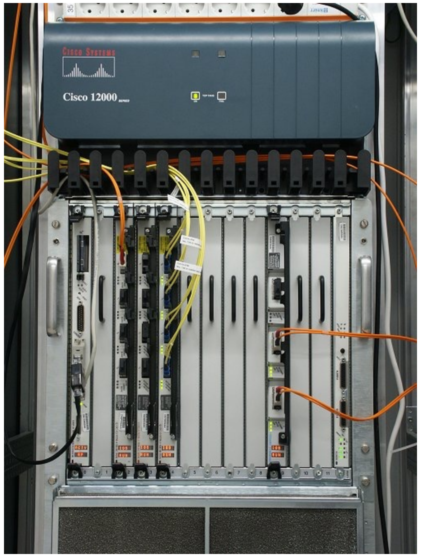
Source: Barroso and Urs Hölzle (2009)

What's in a data center?



- Hundreds or thousands of racks

What's in a data center?

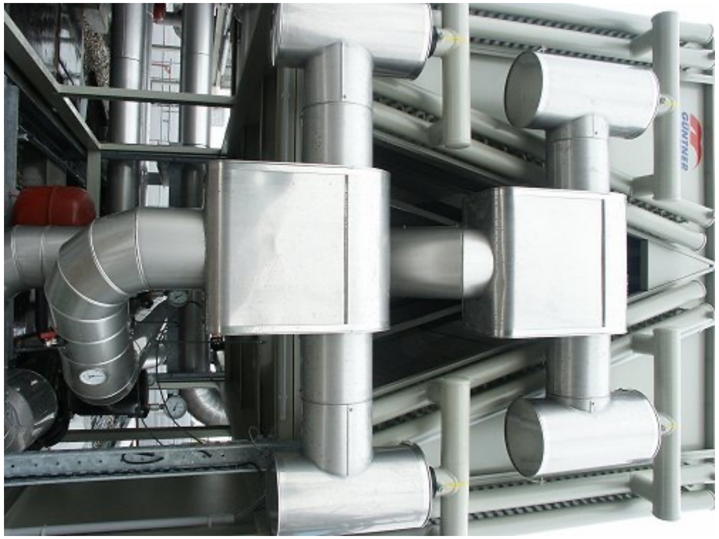


- Massive networking

What's in a data center?

- Emergency power supplies

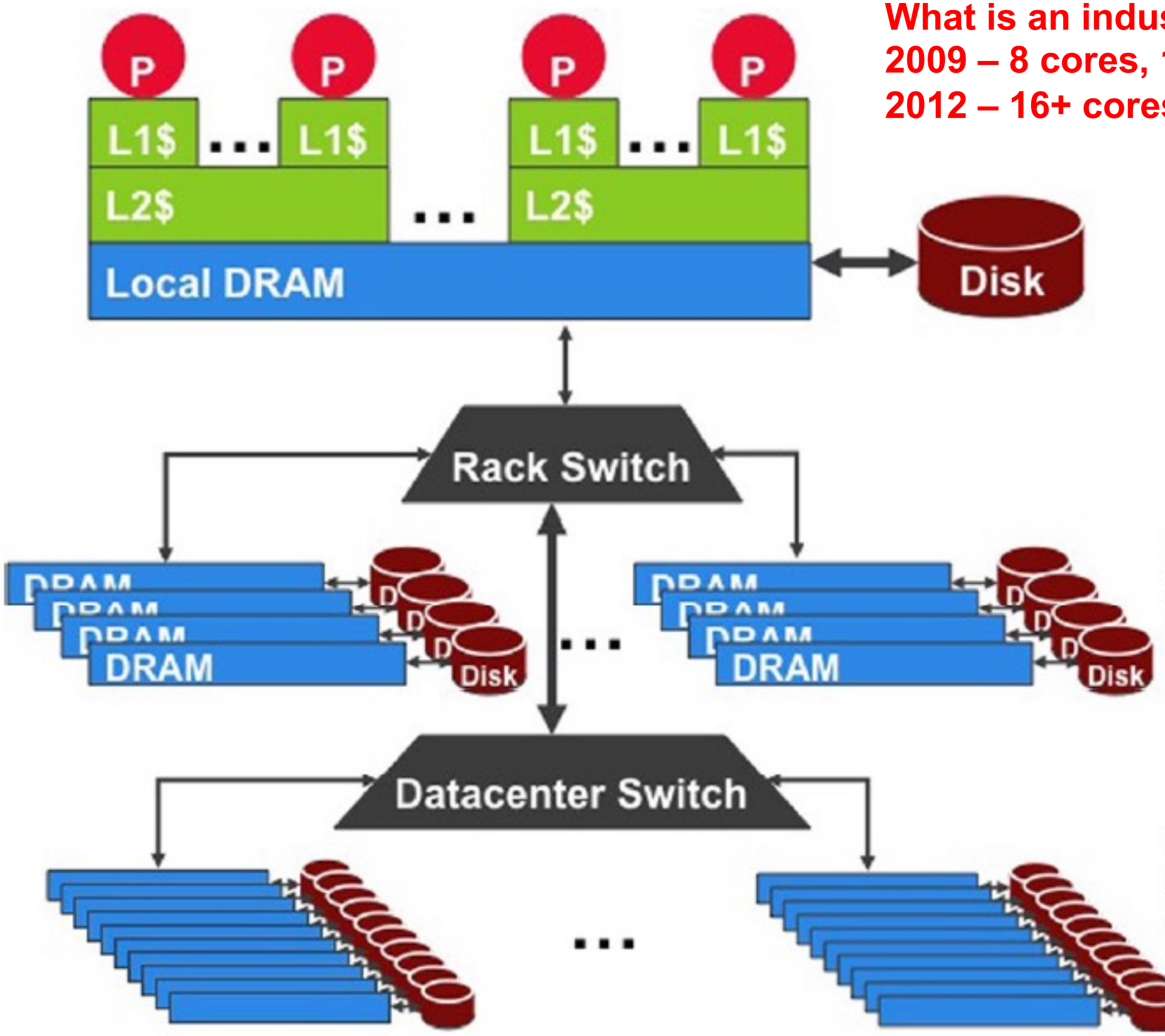
What's in a data center?



- Massive cooling

Storage Hierarchy

What is an industrial “Commodity Machine” ?
2009 – 8 cores, 16GB RAM, 4x1TB Disks
2012 – 16+ cores, 48-96GB RAM, 12x(2~3)TB Disks



One server
DRAM: 16GB, 100ns, 20GB/s
Disk: 2TB, 10ms, 200MB/s

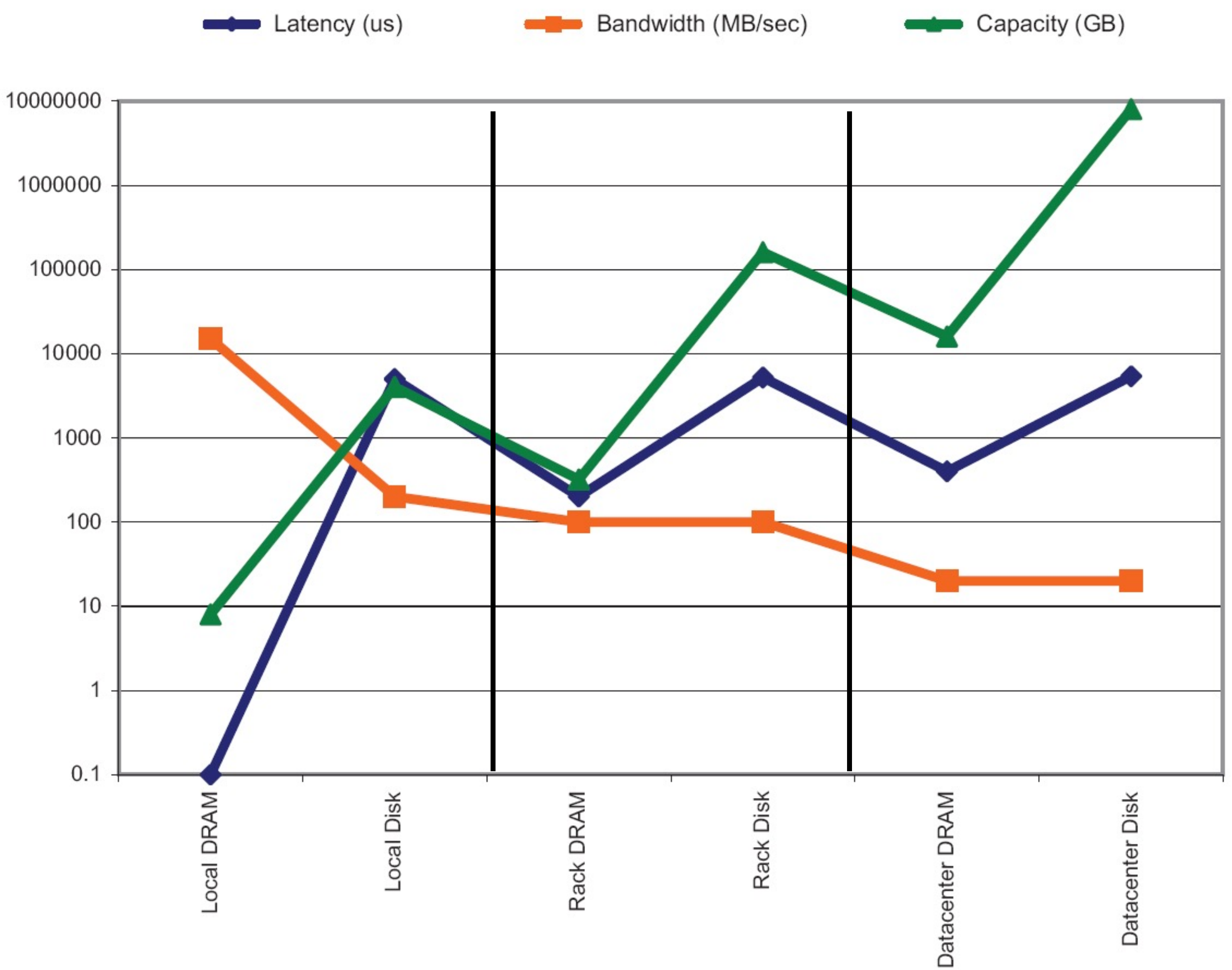
Local rack (80 servers)
DRAM: 1TB, 300us, 100MB/s
Disk: 160TB, 11ms, 100MB/s

Cluster (30 racks)
DRAM: 30TB, 500us, 10MB/s
Disk: 4.80PB, 12ms, 10MB/s

The sense of scale…

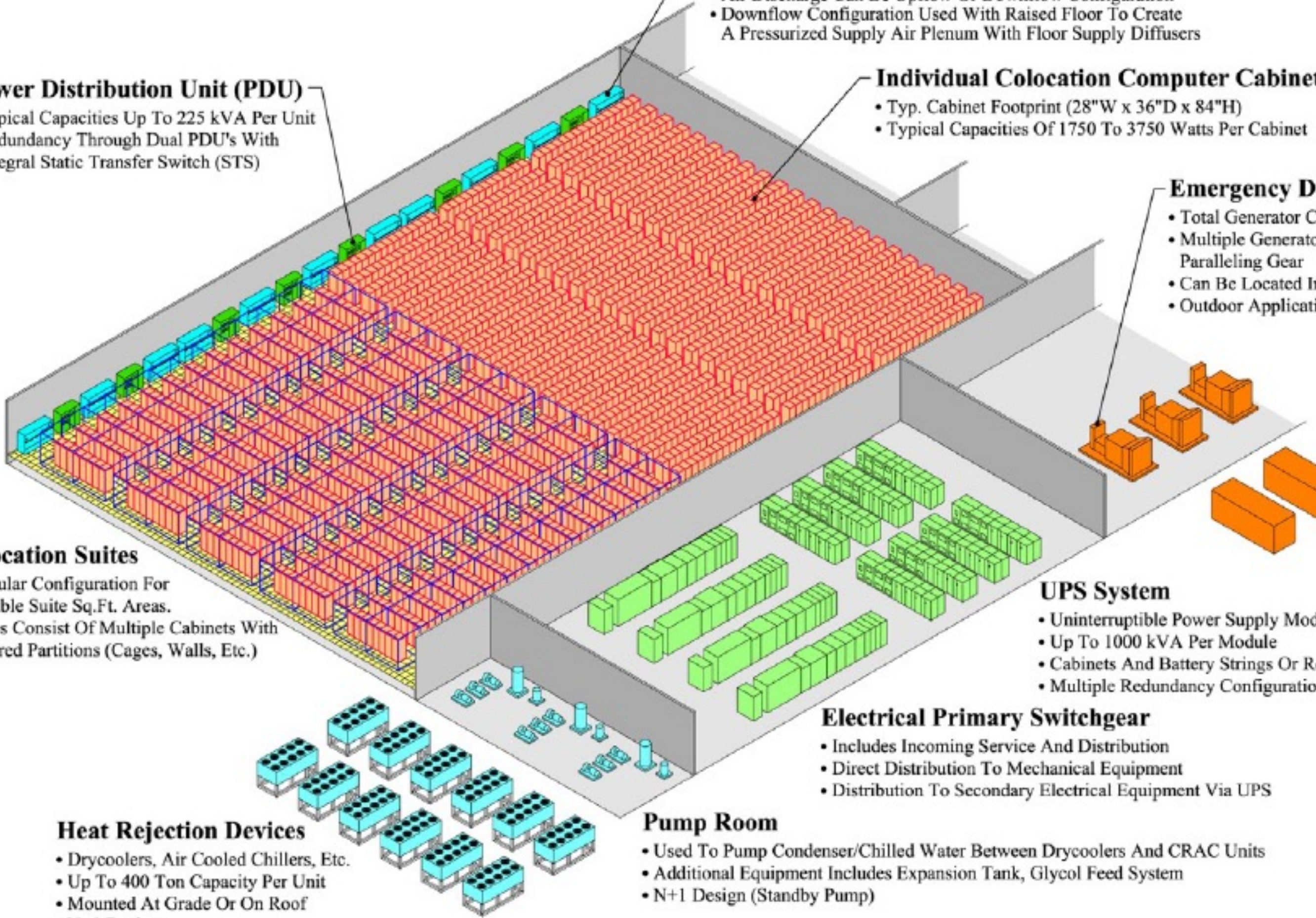
Source: Barroso and Urs Hölzle (2009)

Storage Hierarchy



Source: Barroso and Urs Hölzle (2009)

Anatomy of a Datacenter



Computer Air Handling Unit (CRAC)

- Up To 30 Ton Sensible Capacity Per Unit
- Air Discharge Can Be Upflow Or Downflow Configuration
- Downflow Configuration Used With Raised Floor To Create A Pressurized Supply Air Plenum With Floor Supply Diffusers

Power Distribution Unit (PDU)

- Typical Capacities Up To 225 kVA Per Unit
- Redundancy Through Dual PDU's With Integral Static Transfer Switch (STS)

Individual Colocation Computer Cabinets

- Typ. Cabinet Footprint (28"W x 36"D x 84"H)
- Typical Capacities Of 1750 To 3750 Watts Per Cabinet

Emergency Diesel Generators

- Total Generator Capacity = Total Electrical Load To Building
- Multiple Generators Can Be Electrically Combined With Paralleling Gear
- Can Be Located Indoors Or Outdoors At Grade Or On Roof.
- Outdoor Applications Require Sound Attenuating Enclosures

Fuel Oil Storage Tanks

- Tank Capacity Dependant On Length Of Generator Operation
- Can Be Located Underground Or At Grade Or Indoors

Colocation Suites

- Modular Configuration For Flexible Suite Sq.Ft. Areas.
- Suites Consist Of Multiple Cabinets With Secured Partitions (Cages, Walls, Etc.)

UPS System

- Uninterruptible Power Supply Modules
- Up To 1000 kVA Per Module
- Cabinets And Battery Strings Or Rotary Flywheels
- Multiple Redundancy Configurations Can Be Designed

Electrical Primary Switchgear

- Includes Incoming Service And Distribution
- Direct Distribution To Mechanical Equipment
- Distribution To Secondary Electrical Equipment Via UPS

Heat Rejection Devices

- Drycoolers, Air Cooled Chillers, Etc.
- Up To 400 Ton Capacity Per Unit
- Mounted At Grade Or On Roof
- N+1 Design

Pump Room

- Used To Pump Condenser/Chilled Water Between Drycoolers And CRAC Units
- Additional Equipment Includes Expansion Tank, Glycol Feed System
- N+1 Design (Standby Pump)

Source: Barroso and Urs Hölzle (2009)

Energy matters!

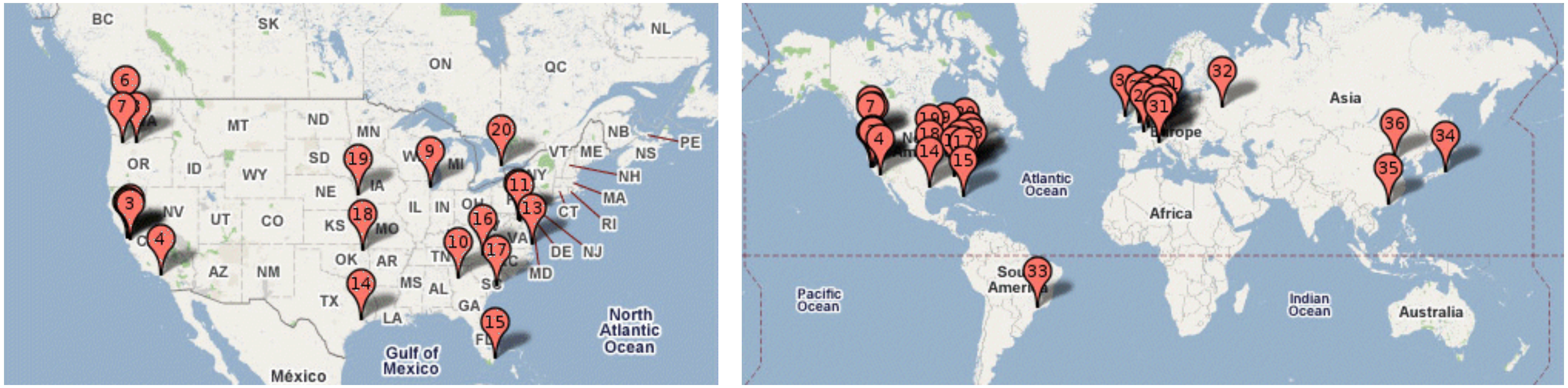
Company	**Servers**	**Electricity**	**Cost**
eBay	16K	~$0.6*10^5$ MWh	~$3.7M/yr
Akamai	40K	~$1.7*10^5$ MWh	~$10M/yr
Rackspace	50K	~$2*10^5$ MWh	~$12M/yr
Microsoft	>200K	>$6*10^5$ MWh	>$36M/yr
Google	>500K	>$6.3*10^5$ MWh	>$38M/yr
USA (2006)	**10.9M**	**$610*10^5$ MWh**	**$4.5B/yr**

- Data centers consume a lot of energy
  - Makes sense to build them near sources of cheap electricity
  - Example: Price per KWh is 3.6ct in Idaho (near hydroelectric power), 10ct in California (long distance transmission), 18ct in Hawaii (must ship fuel)
  - Most of this is converted into heat → Cooling is a big issue!

Scaling up

- What if even a data center is not big enough?
  - Build additional data centers
  - Where? How many?

Global distribution



- Data centers are often globally distributed
  - Example above: Google data center locations (inferred)
- Why?
  - Need to be close to users (physics!)
  - Cheaper resources
  - Protection against failures

Trend: Modular data center

- Need more capacity? Just deploy another container!

Justifying the “Big Ideas”

- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour

Recap

- Web-Scale Data – their sources and uses
- What is Web-Scale Information Analytics:
  - Data Mining, Statistical Modeling, Machine Learning, and…
- What is this Course about ?
- Computing Infrastructure for Web-scale Data Processing
- How to scale out hardware for Web-scale Data processing ?