Introduction

Lecture 1
October 11th, 2017

Jae W. Lee (jaewlee@snu.ac.kr)
Computer Science and Engineering
Seoul National University

***Slide credits**: Prof. Anthony Joseph (BerkeleyX CS100.1x, CS105x), Databricks, Simeon Yep (Splunk), John Phillips (Amazon AWS), Prof. DK Panda (Ohio State Univ.)*

Course information

- **Title: Big Data Engineering 3 (Parallel and Distributed Database)**

- **Schedule**
  - 2 – 5 PM [Wed, Thur]
  - Will be divided into two 75-minute sections with a 15-minute break
  - Lectures (by Professor) on Wednesdays, Labs (by Instructors) on Thursdays
  - Lecture Room: #38-B105

- **Lecturer**
  - Prof. Jae Wook Lee (이재욱, jaewlee@snu.ac.kr) @ SNU CSE
  - Office: Engineering Building #301-506
  - Phone: 880-1834
  - Office Hours: After class on Wednesdays

Course information

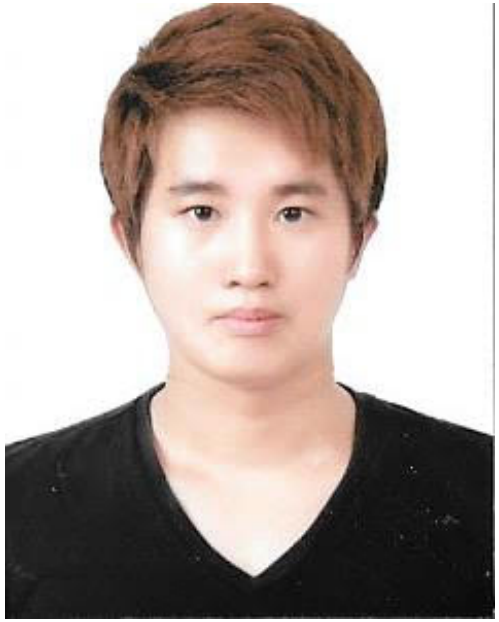
- **Course materials will be distributed through the web.**
  - Link: http://bdi.snu.ac.kr/academy/portal/index.php/bigdata-engineering-3-notice/

- **The main language for this course will be Python.**
  - Also, we assume you are familiar with a Linux programming environment.
  - All labs will be performed on Linux.

Lab instructors

- **Jonghyun Bae (배종현)**
  - E-mail: bnbbkr@gmail.com

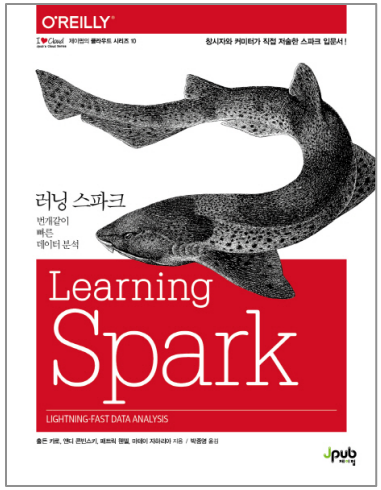
- **Jun Heo (허준)**
  - E-mail: heojun18@gmail.com

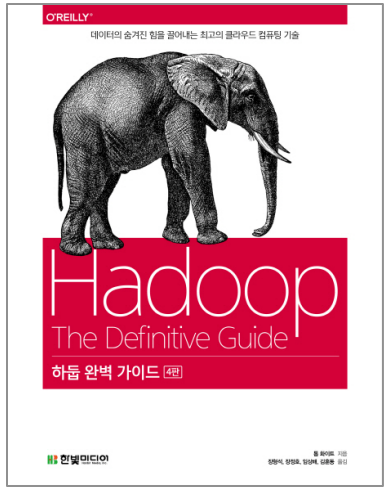


- **Office: Engineering Building #301-554-1**
- **Phone: 02-880-1836**
- **Office Hours: TBA**

Course information: Textbooks

- **No textbook is required.**
- **Reference books (not required).**
  - 러닝 스파크 (Learning Spark), 1st Ed. (2015)
    - Authors: Andy Konwinski, Holden Karau, Matei Zaharia, and Patrick Wendell (박종영 옮김)
  - 하둡 완벽 가이드 (Hadoop: The Definitive Guide), 4th Ed. (2015)
    - Authors: Tom White (장형석, 장정호, 임상배, 김훈동 옮김)





Course coverage and schedule*

	Contents	Remarks
Week 1	Introduction to the Course	
Week 2	Introduction to Apache Hadoop, HDFS, and Spark	
Week 3	Spark Programming Basics (1)	
Week 4	Spark Programming Basics (2)	
Week 5	SparkSQL and DataFrame	Office Hours (11/6, All day)
Week 6	Spark Libraries (MLlib, GraphX, Spark Streaming)	Office Hours (11/13, All day)
Week 7	Data Visualization and Profiling	
Week 8	Advanced Topics	

**** Tentative and subject to adjustment.***

Grading System*

- Attendance/Class Participation: 10%
- Lab Exercises: 40%
- Term Project: 25%
- Final Exam: 25%

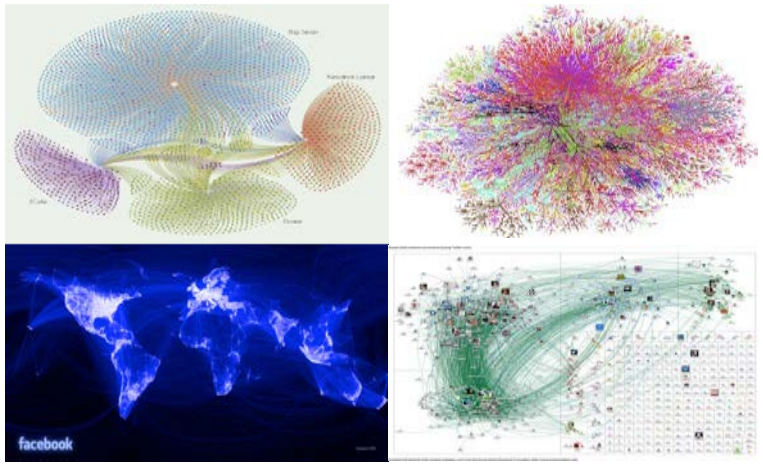
** Tentative and subject to adjustment.*

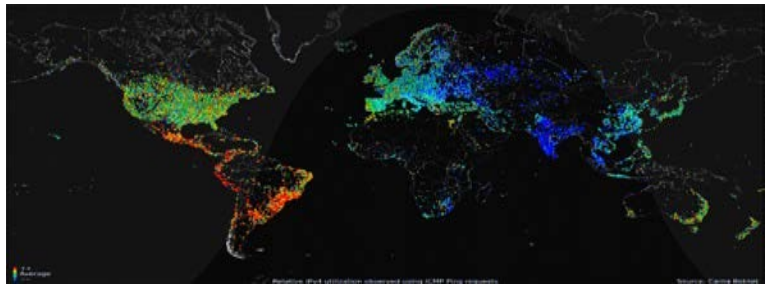
Outline

- **Big Data Trends**
- **The Big Data Problem**
- **Hardware for Big Data**
  - Distributing Work
  - Handling Failures and Slow Machines
- **MapReduce and Complex Jobs**
  - Apache Spark
- **Introduction to Amazon AWS**

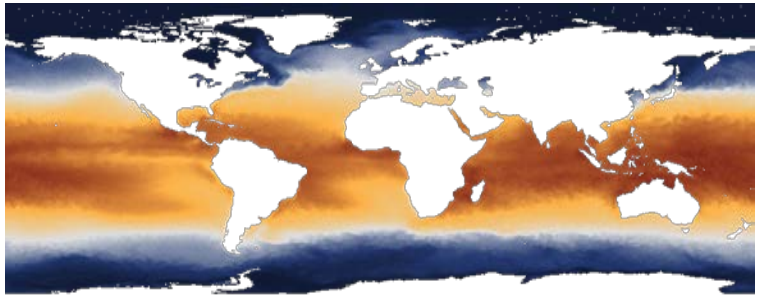
Introduction to Big Data Analytics and Trends

- Big Data has changed the way people understand and harness the power of data, both in the business and research domains
- Big Data has become one of the most important elements in business analytics
- Big Data and High Performance Computing (HPC) are converging to meet large scale data processing challenges
- Running High Performance Data Analysis (HPDA) workloads in the cloud is gaining popularity
  - According to the latest OpenStack survey, 27% of cloud deployments are running HPDA workloads



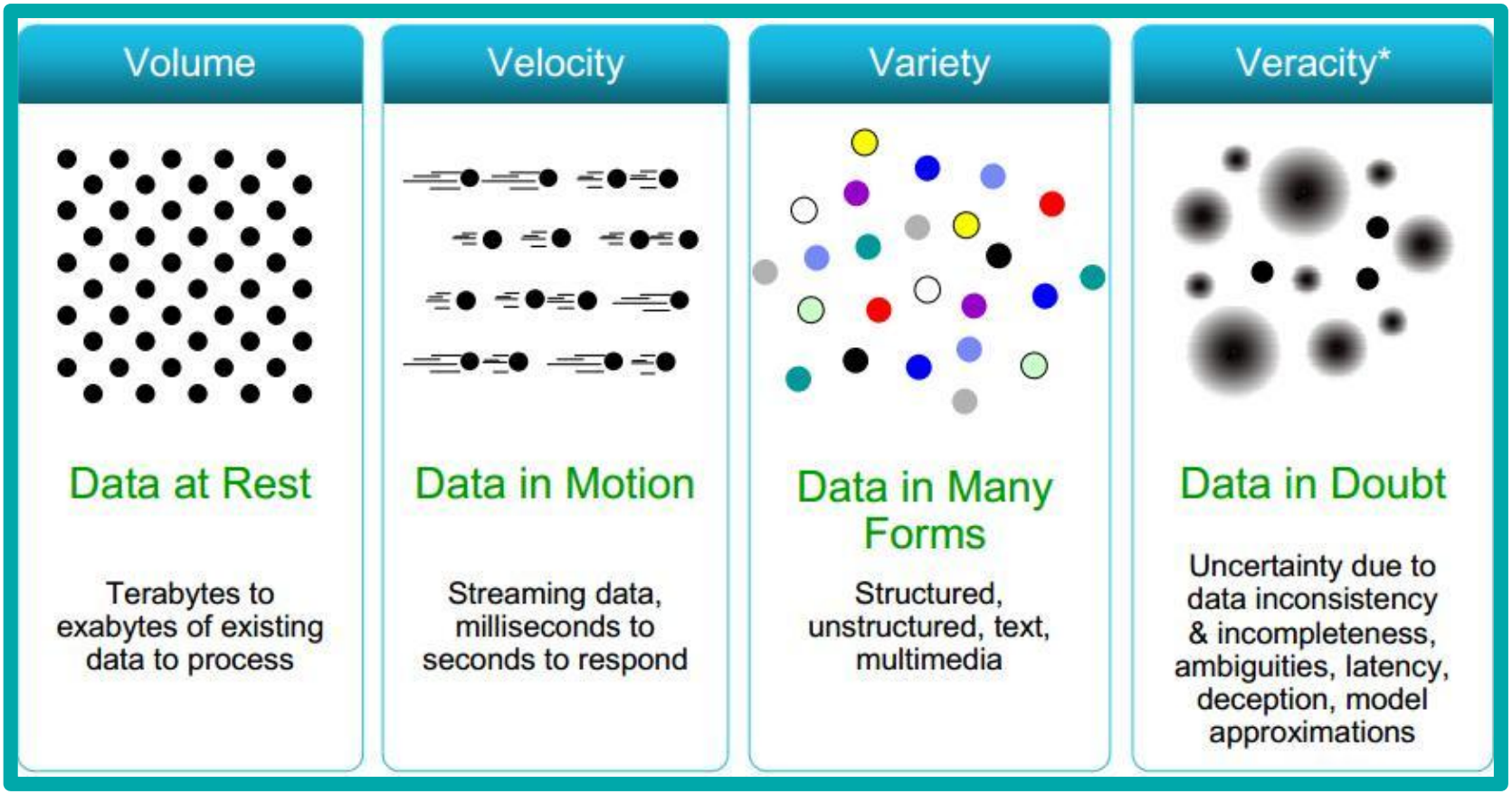


http://www.coolinfographics.com/blog/tag/data?currentPage=3



http://www.climatecentral.org/news/white-house-brings-together-big-data-and-climate-change-17194

4V Characteristics of Big Data



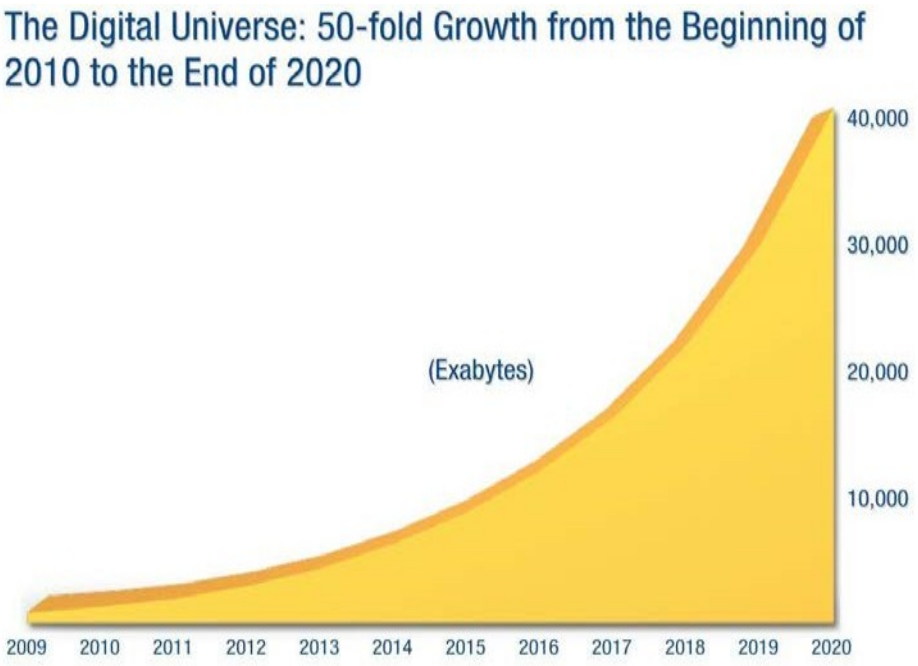
- **Commonly accepted 3V's of Big Data**
  - **Volume, Velocity, Variety**

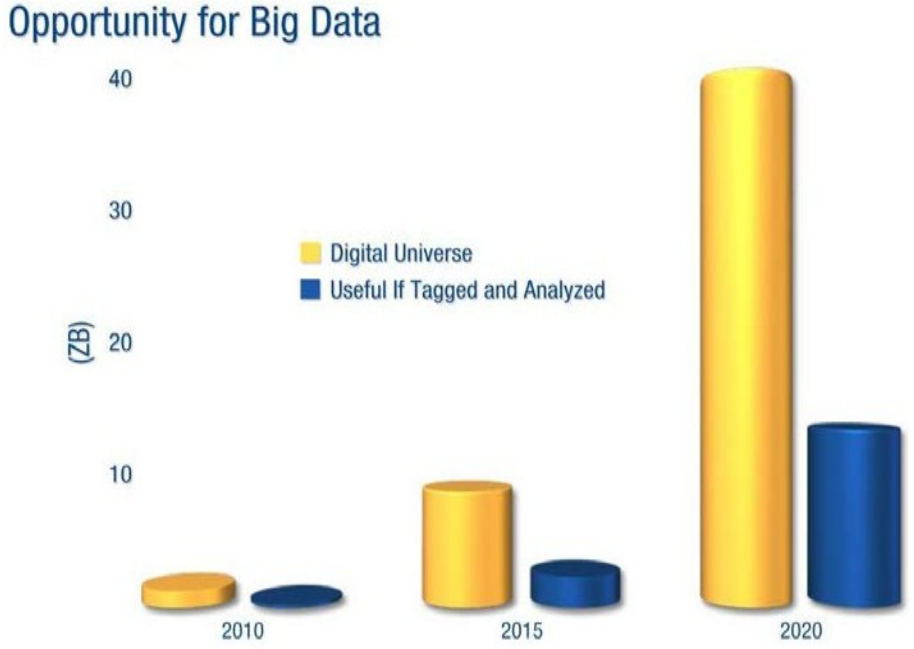
Source: http://api.ning.com/files/tRHkwQN7s-Xz5cxylXG004GLGJdjoPd6bVfVBwvgu*F5MwDDUCiHHdmBW-JTEz0cfJjGurJucBMTkIUNdL3jcZT8IPfNWfN9/dv1.jpg

Michael Stonebraker: Big Data Means at Least Three Different Things, http://www.nist.gov/itl/ssd/is/upload/NIST-stonebraker.pdf

- **4/5V's of Big Data – 3V + *Veracity, *Value**

Big Volume of Data by the End of this Decade



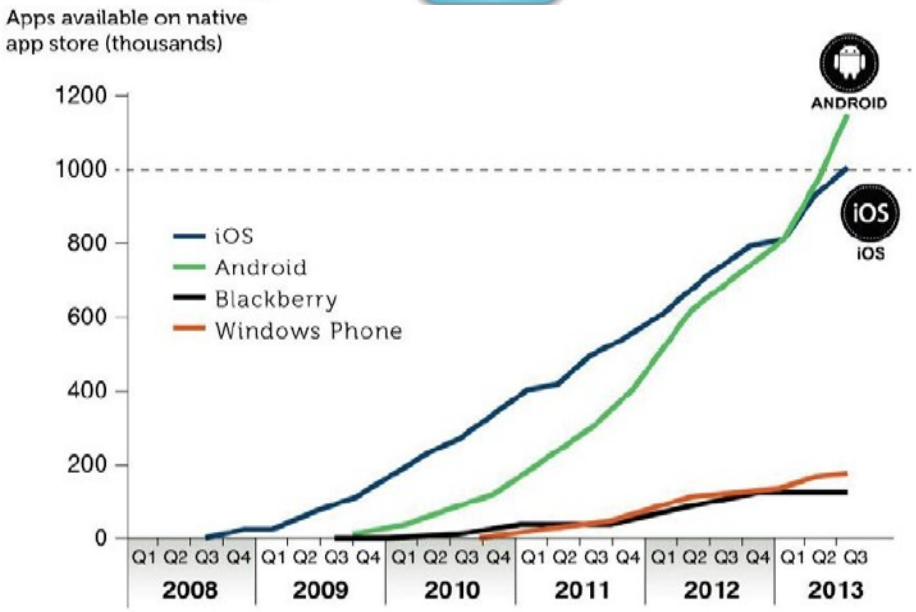


- From 2005 to 2020, the digital universe will grow by a factor of 300, from 130 exabytes to 40,000 exabytes.
- By 2020, a third of the data in the digital universe (more than 13,000 exabytes) will have Big Data Value, but only if it is tagged and analyzed.

Source: John Gantz and David Reinsel, "The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East", IDC's Digital Universe Study, sponsored by EMC, December 2012.

Data Generation in Internet Services and Applications

- Webpages (content, graph) Clicks (ad, page, social)
- Users (OpenID, FB Connect, etc.)
- e-mails (Hotmail, Y!Mail, Gmail, etc.) Photos, Movies (Flickr, YouTube, Video, etc.) Cookies / tracking info (see Ghostery)
- Installed apps (Android market, App Store, etc.)
- Location (Latitude, Loopt, Foursquared, Google Now, etc.) User generated content (Wikipedia & co, etc.)
- Ads (display, text, DoubleClick, Yahoo, etc.) Comments (Discuss, Facebook, etc.) Reviews (Yel Y!Local, etc.)
- Social connections (LinkedIn, Facebook, etc.) Purchase decisions (Netflix, Amazon, etc.) Instant Messages (YIM, Skype, Gtalk, etc.) Search terms (Google, Bing, etc.)
- News articles (BBC, NYTimes, Y!News, etc.) Blog posts (Tumblr, Wordpress, etc.) Microblogs (Twitter, Jaiku, Meme, etc.)
- Link sharing (Facebook, Delicious, Buzz, etc.)

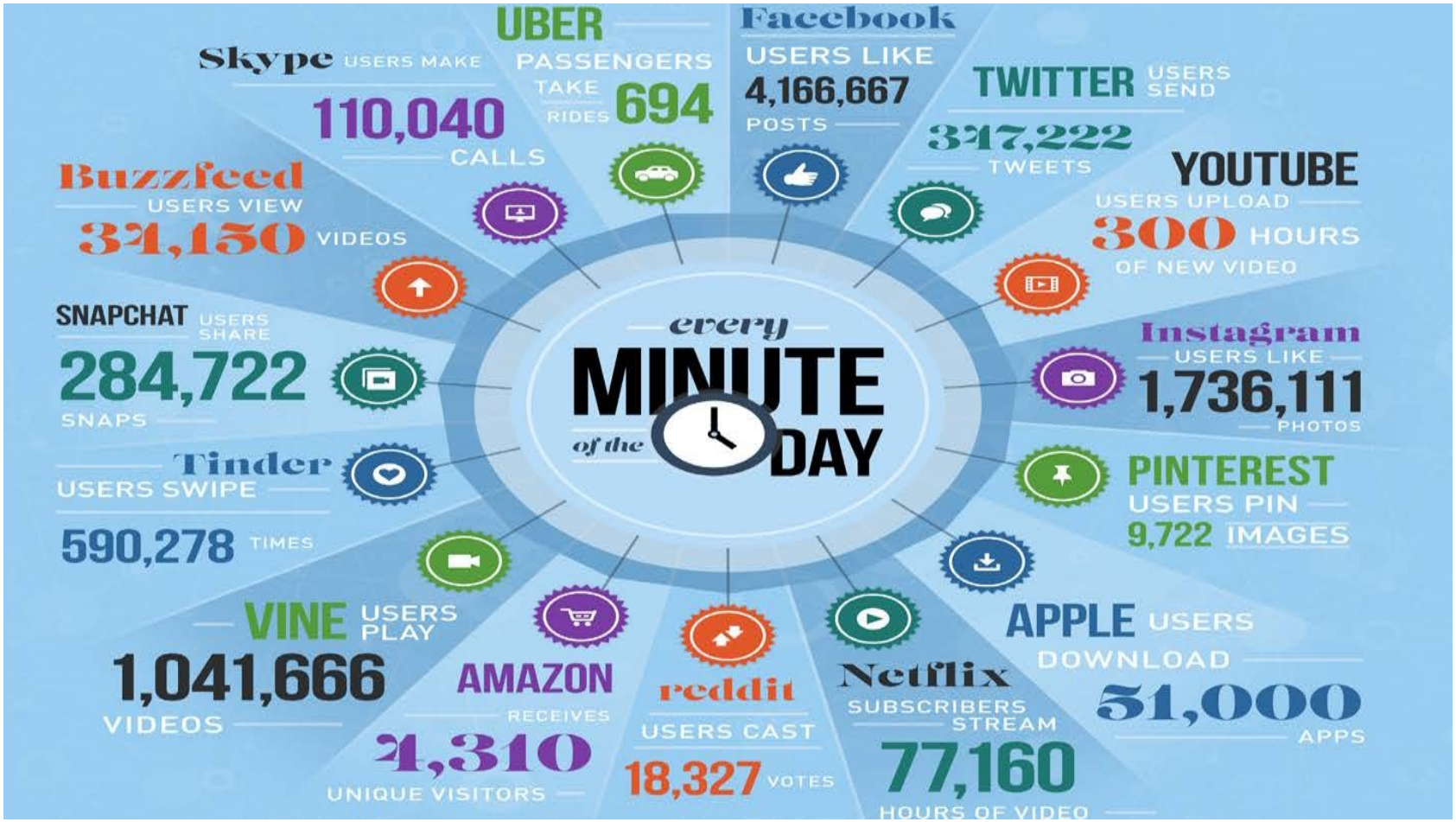


Number of Apps in the Apple App Store, Android Market, Blackberry, and Windows Phone (2013)

- **Android Market: <1200K**
- **Apple App Store: ~1000K**

Source: http://dazeinfo.com/2014/07/10/apple-inc-aapl-ios-google-inc-goog-android-growth-mobile-ecosystem-2014/

Velocity of Big Data – How Much Data Is Generated Every Minute on the Internet?

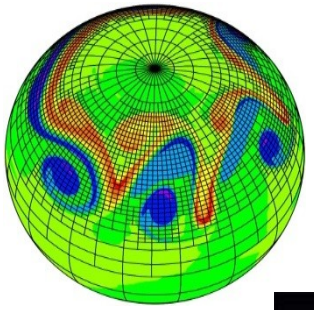


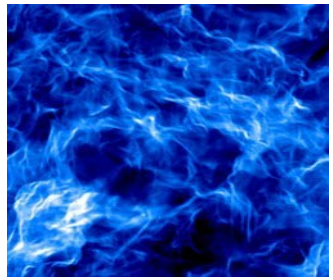
The global Internet population grew 18.5% from 2013 to 2015 and now represents 3.2 Billion People.

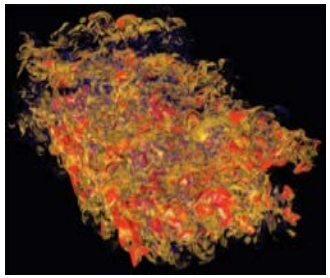
Courtesy: https://www.domo.com/blog/2015/08/data-never-sleeps-3-0/

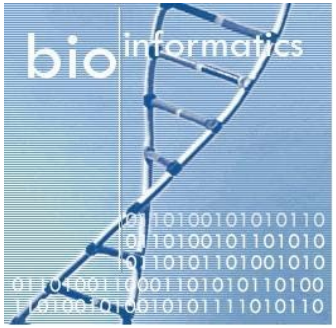
Not Only in Internet Services - Big Data in Scientific Domains

- **Scientific Data Management, Analysis, and Visualization**
- **Applications examples**
  - Climate modeling
  - Combustion
  - Fusion
  - Astrophysics
  - Bioinformatics
- **Data Intensive Tasks**
  - Runs large-scale simulations on supercomputers
  - Dump data on parallel storage systems
  - Collect experimental / observational data
  - Move experimental / observational data to analysis sites
  - Visual analytics – help understand data visually



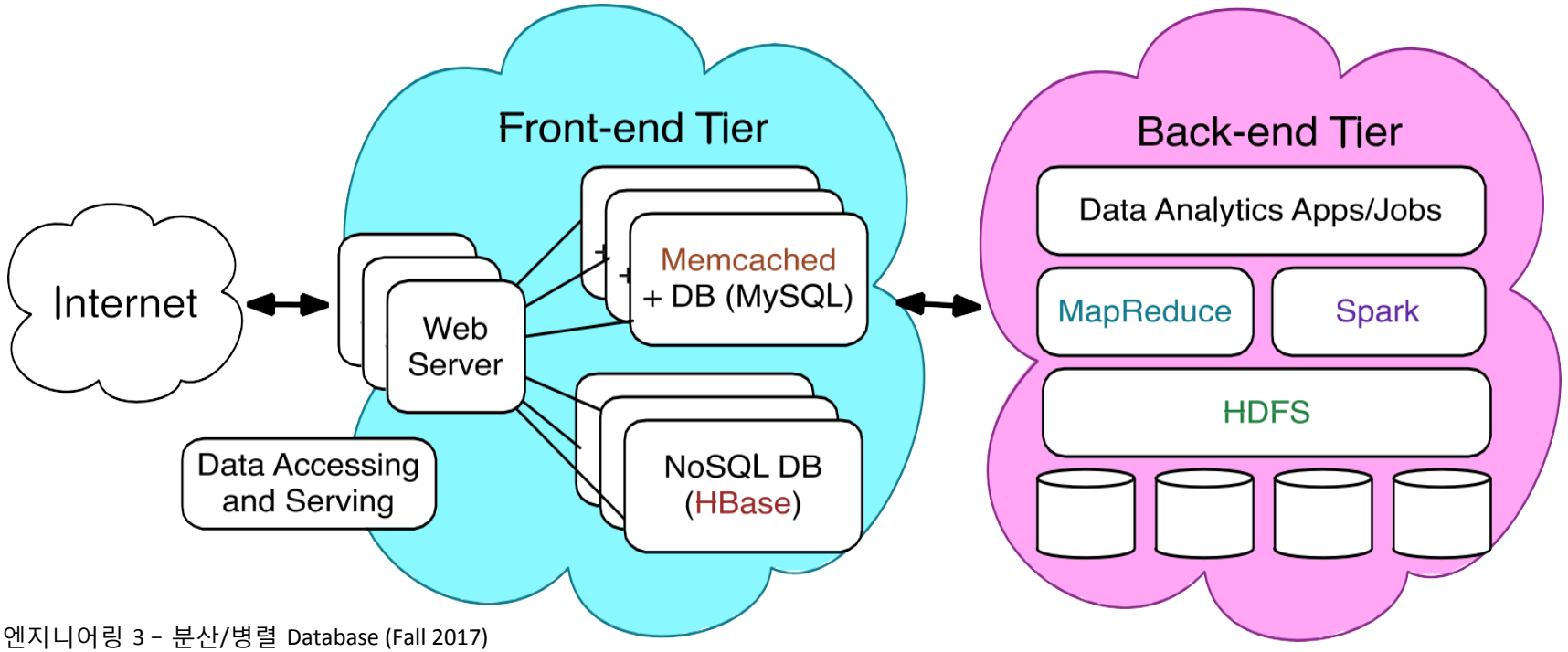






Data Management and Processing on Modern Clusters

- **Substantial impact on designing and utilizing data management and processing systems in multiple tiers**
  - Front-end data accessing and serving (Online)
    - Memcached + DB (e.g. MySQL), HBase
  - Back-end data analytics (Offline)
    - HDFS, MapReduce, Spark



Outline

- Big Data Trends
- **The Big Data Problem**
- Hardware for Big Data
  - Distributing Work
  - Handling Failures and Slow Machines
- MapReduce and Complex Jobs
  - Apache Spark
- Introduction to Amazon AWS

Some Traditional Analysis Tools

- Unix shell commands, Pandas, R, …

All run on a single machine!

The Big Data Problem

- Data growing faster than computation speeds
- Growing data sources
  - Web, mobile, scientific, …
- Storage getting cheaper
  - Size doubling every 18 months
- But, stalling CPU speeds and storage bottlenecks

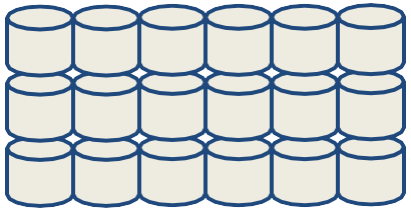
Big Data Examples

- Facebook's daily logs: 60 TB
- 1,000 genomes project: 200 TB
- Google web index: 10+ PB

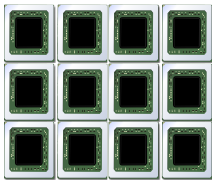
- Cost of 1 TB of disk: ~$35
- Time to read 1 TB from disk: 3 hours (100 MB/s)

The Big Data Problem

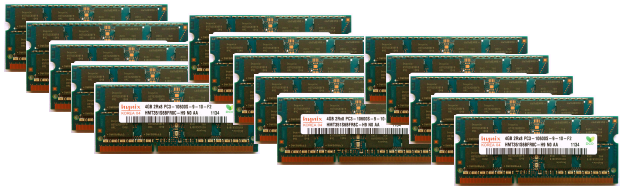
- A single machine can no longer process or even store all the data!
- Only solution is to distribute data over large clusters



Lots of hard drives



… and CPUs



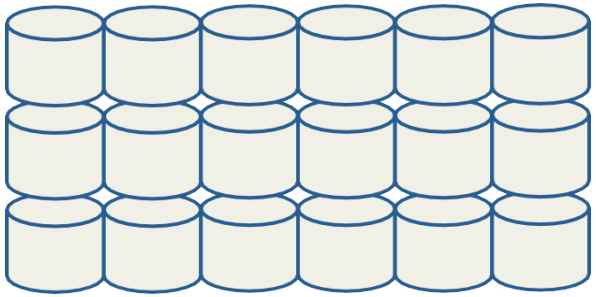
… and memory!

How do we program this thing?

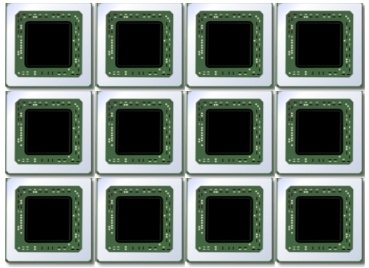
Outline

- Big Data Trends
- The Big Data Problem
- **Hardware for Big Data**
  - Distributing Work
  - Handling Failures and Slow Machines
- MapReduce and Complex Jobs
  - Apache Spark
- Introduction to Amazon AWS

Hardware for Big Data



Lots of hard drives (and SSDs)



… and CPUs

Hardware for Big Data

- **One big box? (1990's solution)**

- **But, expensive**
  - Low volume
  - All "premium" hardware

- **And, still not big enough!**

Image: Wikimedia Commons / User:Tonusamuel

Hardware for Big Data

- **Consumer-grade hardware**
  - Not “gold plated”
- **Many desktop-like servers**
  - Easy to add capacity
  - Cheaper per CPU/disk
- **Complexity in software**

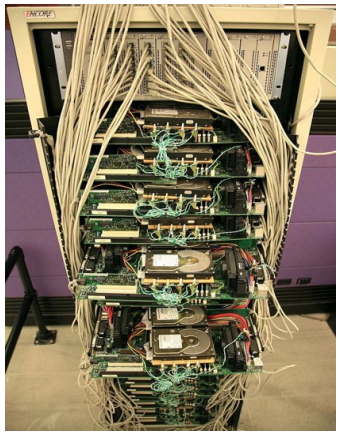


Image: Steve Jurvetson/Flickr

Problems with Cheap Hardware

- **Failures, Google's numbers:**
  - 1-5% hard drives/year
  - 0.2% DIMMs/year
- **Network speeds versus shared memory**
  - *Much* more latency
  - Network slower than storage
- **Uneven performance**

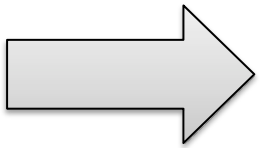
What's Hard About Cluster Computing?

- How do we split work across machines?

WordCount Example

- How do you count the number of occurrences of each word in a document?

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and
ham?"



I: 3
am: 3
Sam: 3
do: 1
you: 1
like: 1
…

WordCount Example

- One approach: Use a hash table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and
ham?"

{}

WordCount Example

- One approach: Use a hash table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and
ham?"

{I :1}

WordCount Example

- One approach: Use a hash table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and
ham?"

{I: 1,
am: 1}

WordCount Example

- One approach: Use a hash table

"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and
ham?"

{I: 1,
am: 1,
Sam: 1}

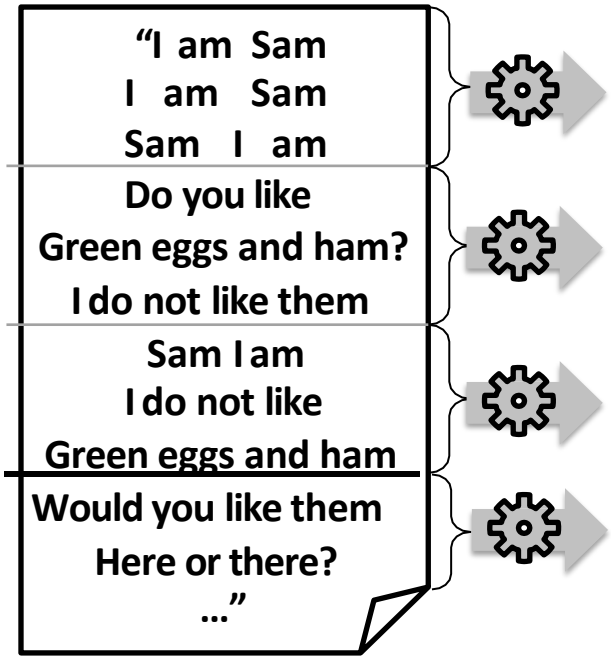
WordCount Example

- One approach: Use a hash table

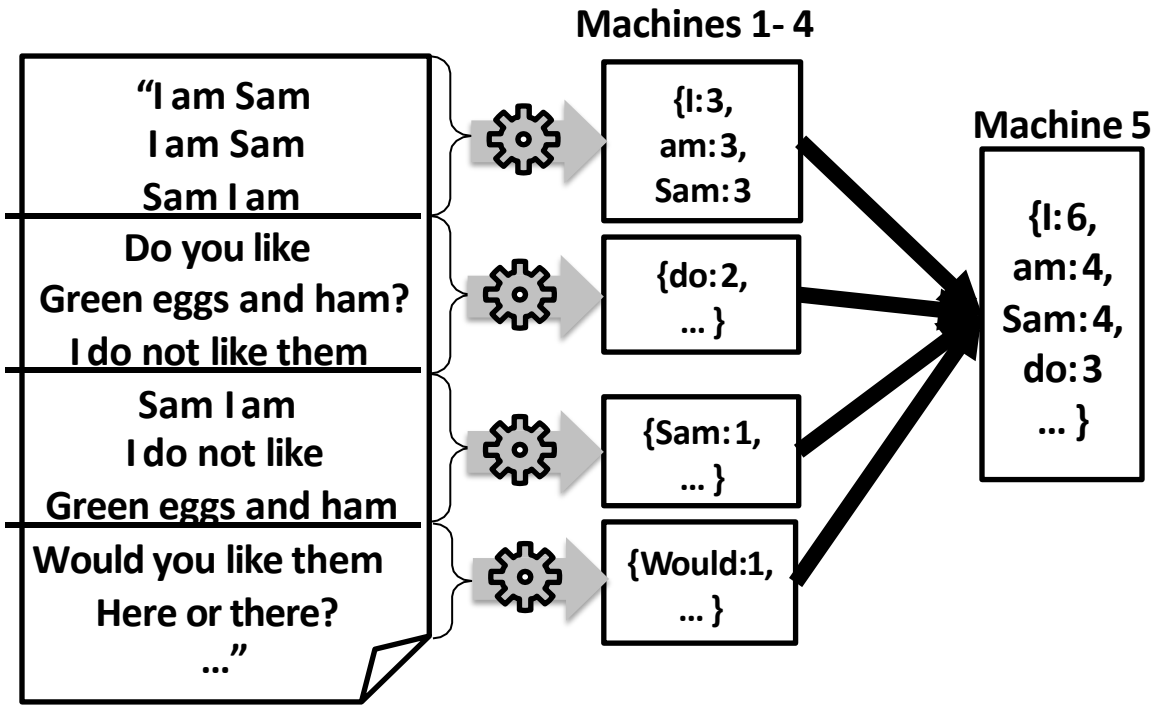
"I am Sam
I am Sam
Sam I am
Do you like
Green eggs and
ham?"

{I: 2,
am: 1,
Sam: 1}

What if the Document is Really Big?

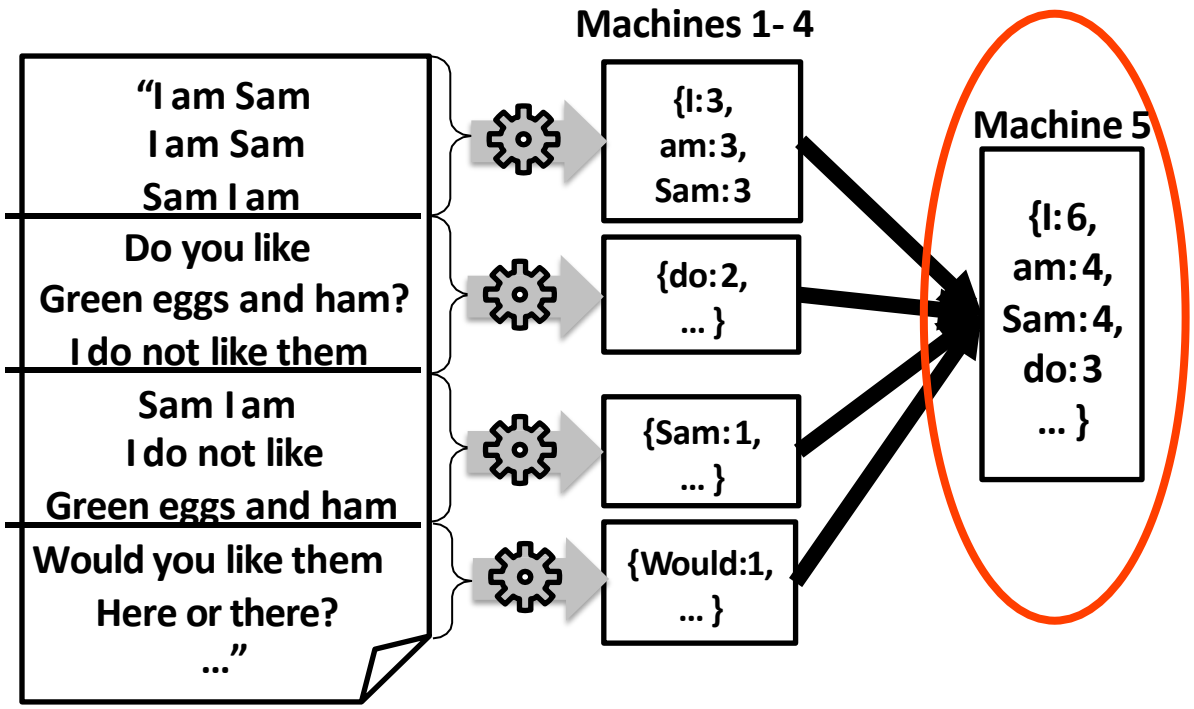


What if the Document is Really Big?



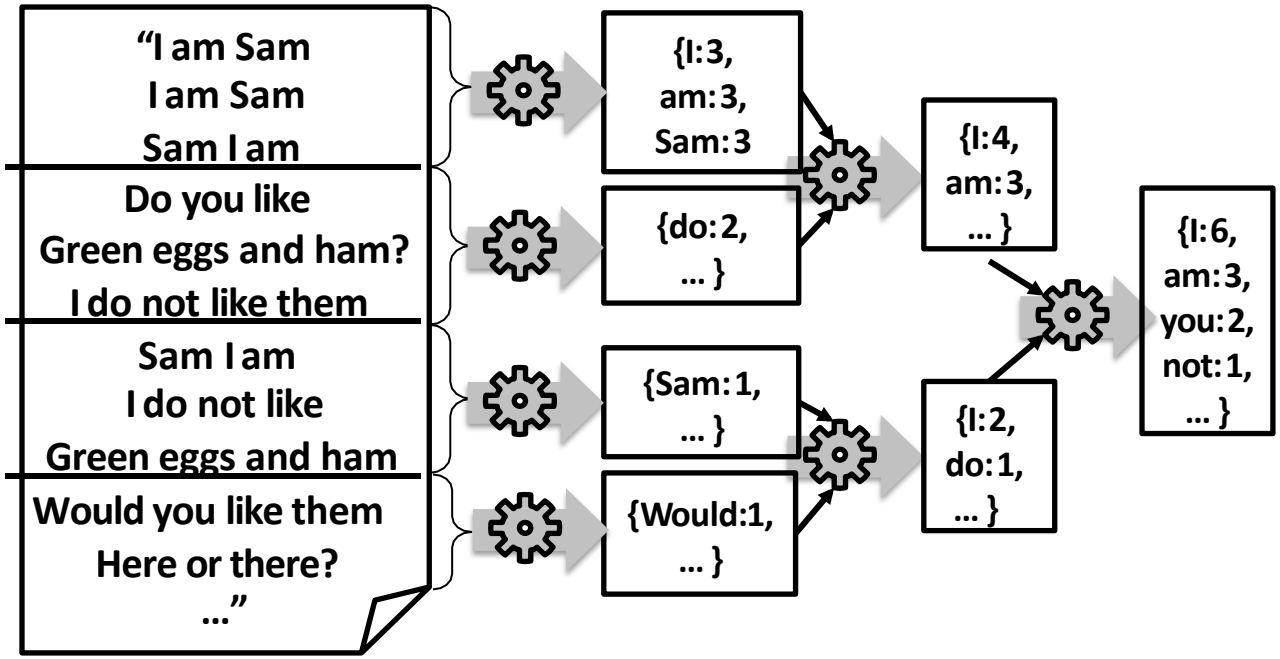
What's the problem with this approach?

What if the Document is Really Big?



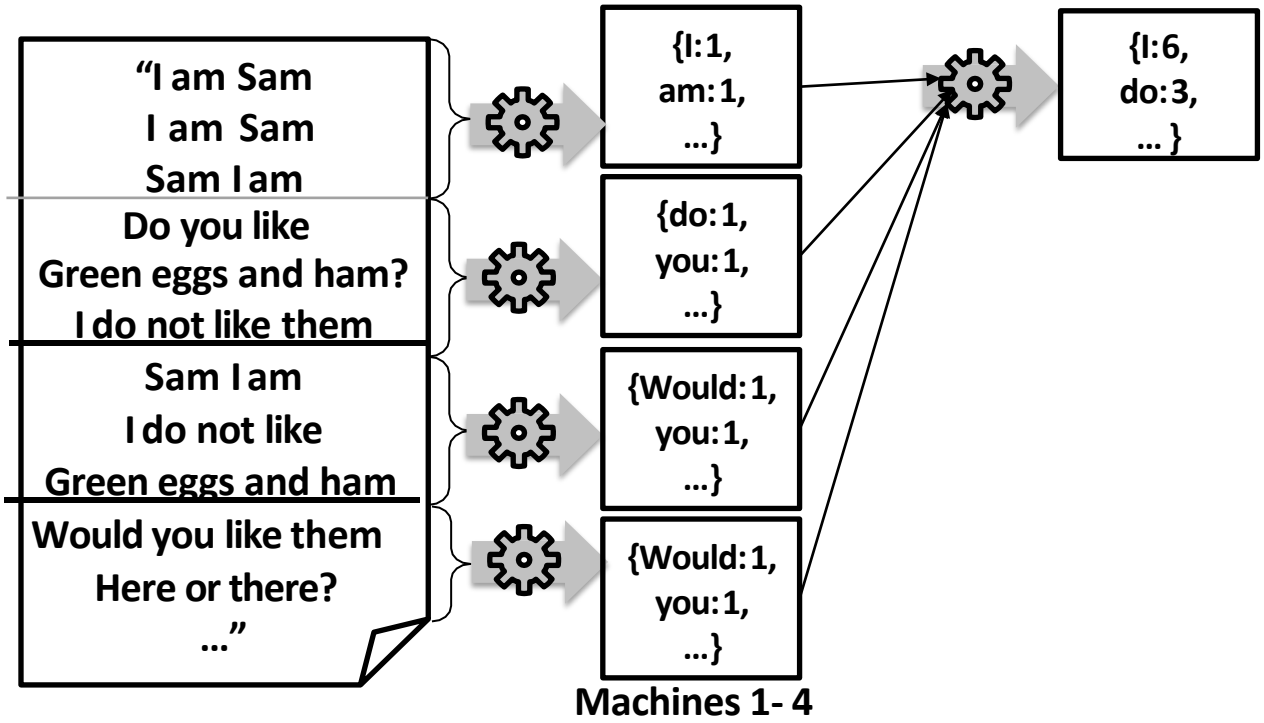
Results have to fit on one machine

What if the Document is Really Big?



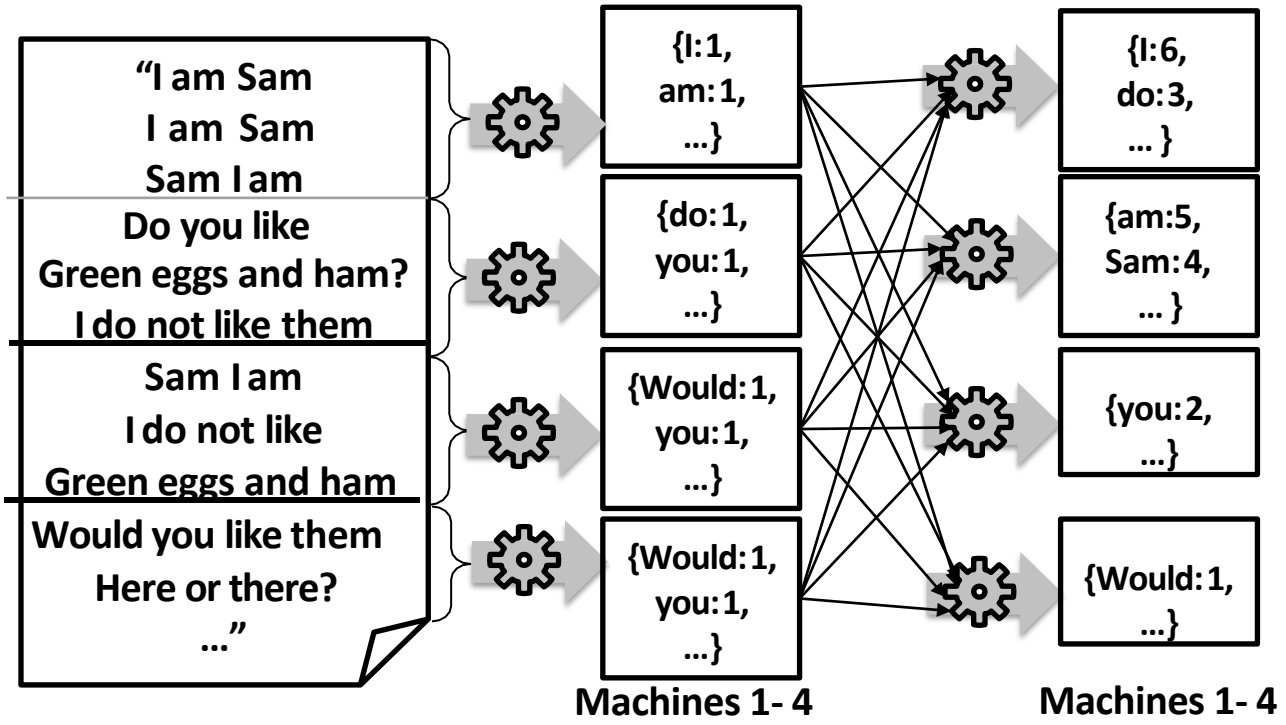
Can add aggregation layers but results still must fit on one machine

What if the Document is Really Big?



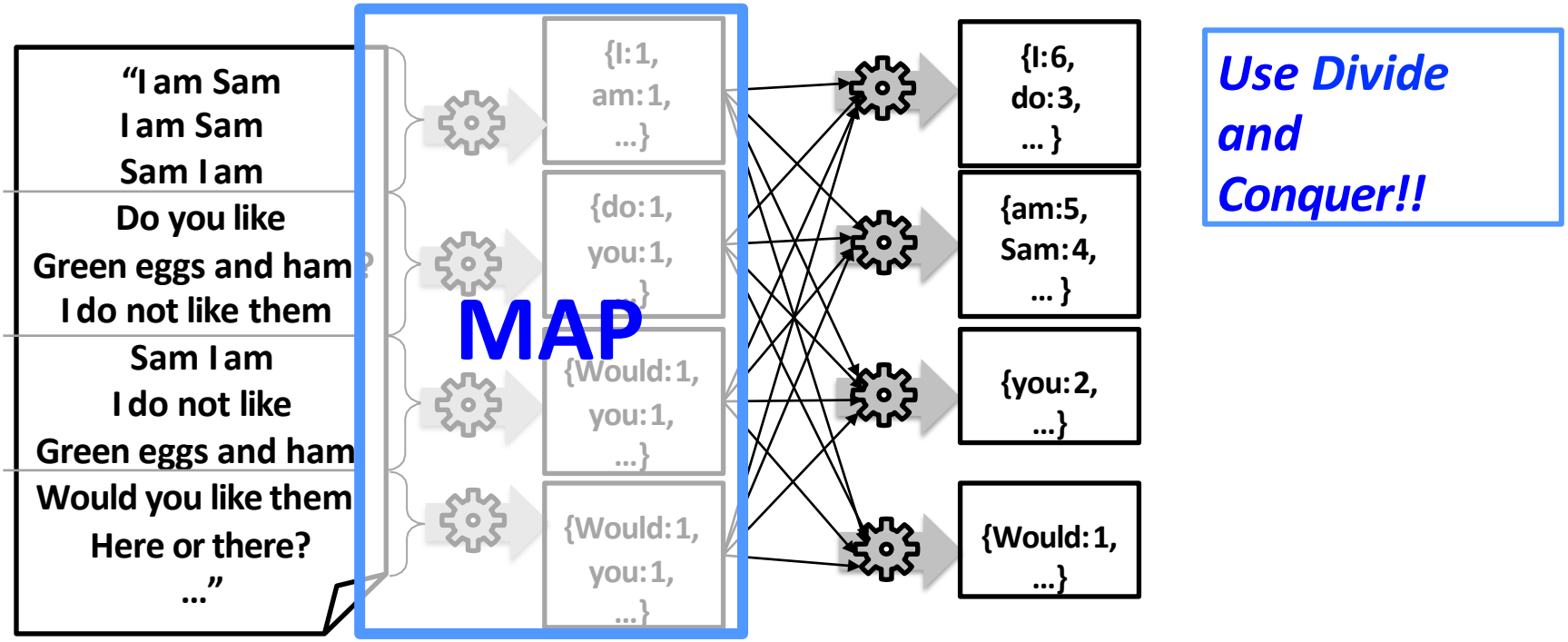
Use Divide and Conquer!!

What if the Document is Really Big?

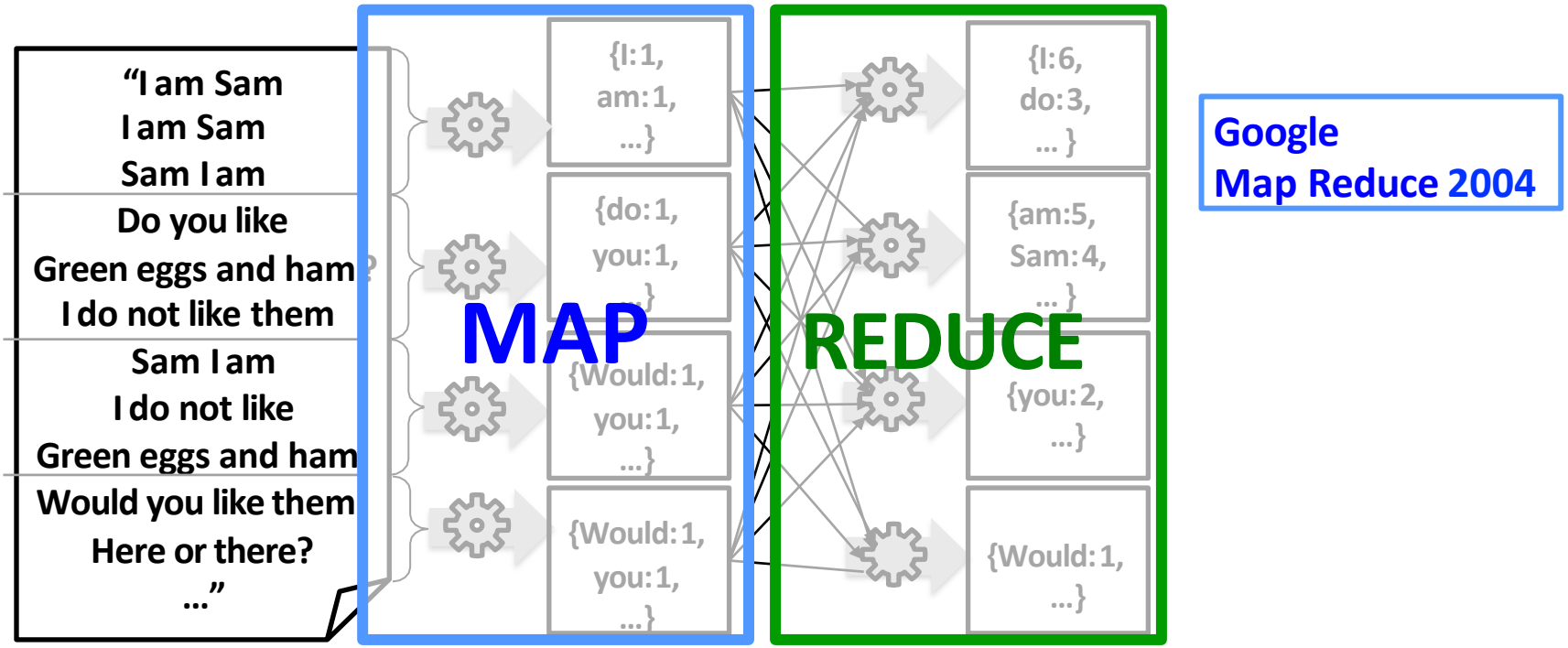


Use Divide and Conquer!!

What if the Document is Really Big?

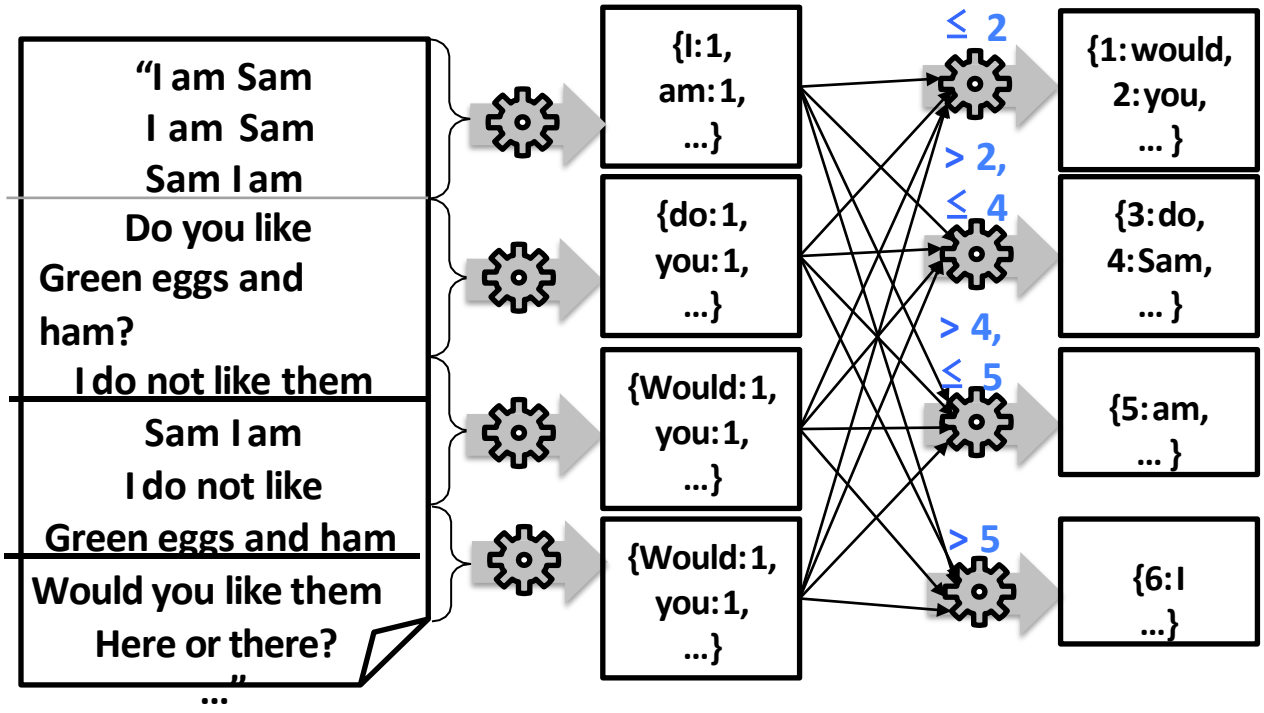


What if the Document is Really Big?



Source: http://research.google.com/archive/mapreduce.html

Map Reduce for Sorting



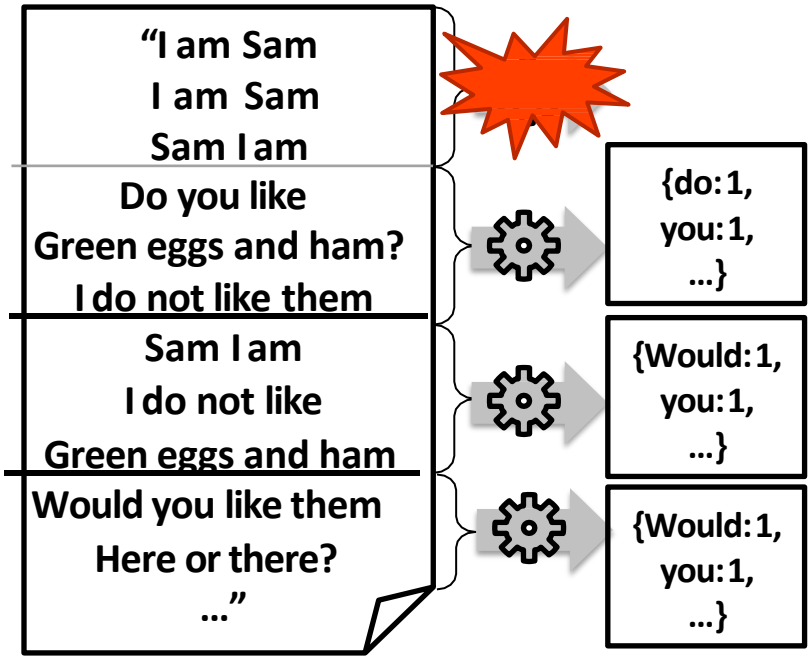
"What word is used most?"

What's Hard About Cluster Computing?

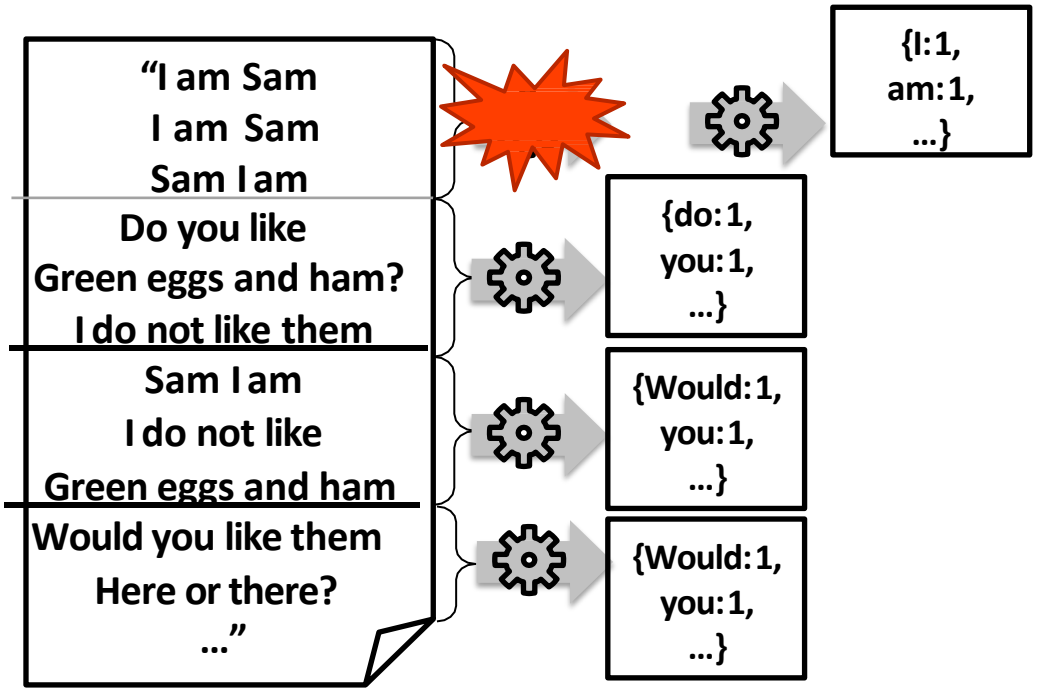
- **How to divide work across machines?**
  - Must consider network, data locality
  - Moving data may be very expensive

- **How to deal with failures?**
  - 1 server fails every 3 years → with 10,000 nodes see 10 faults/day
  - Even worse: stragglers (not failed, but slow nodes)

How Do We Deal with Machine Failures?

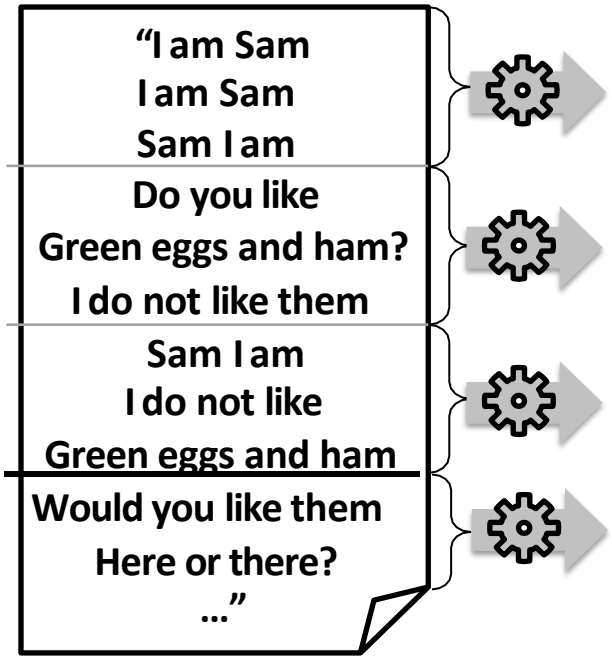


How Do We Deal with Machine Failures?

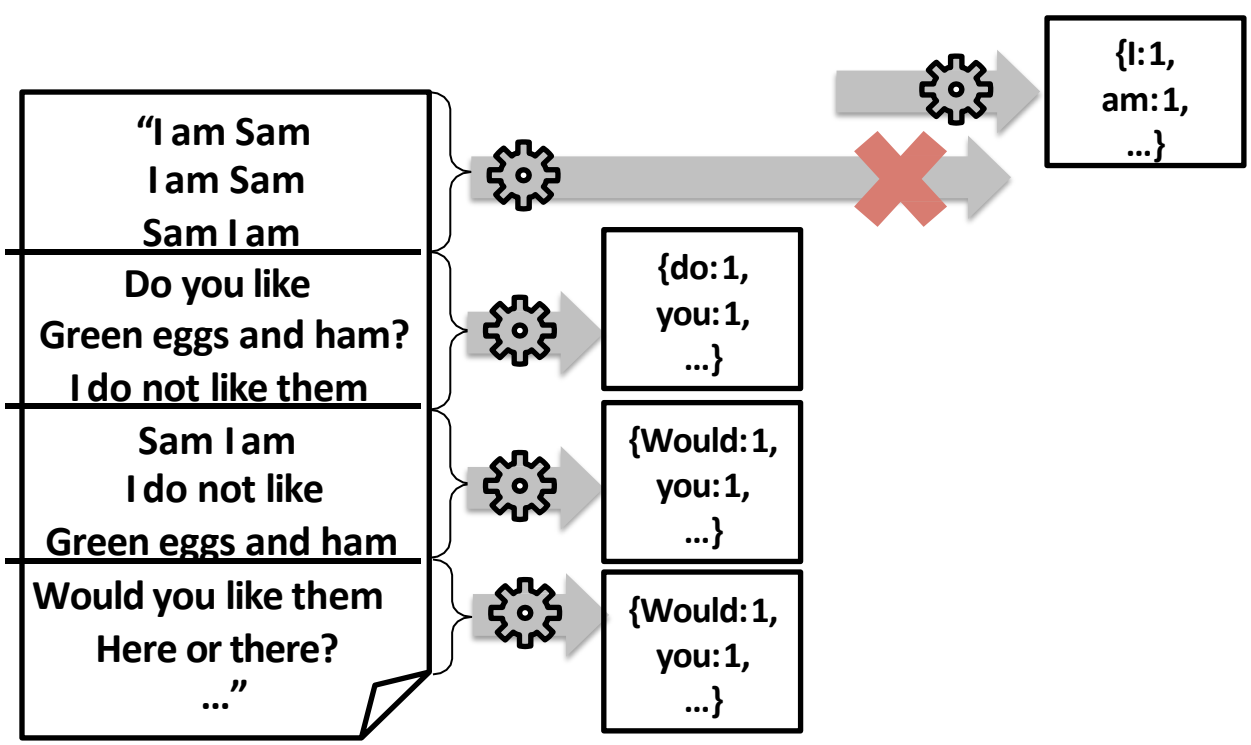


Launch another task!

How Do We Deal with Slow Tasks?



How Do We Deal with Slow Tasks?

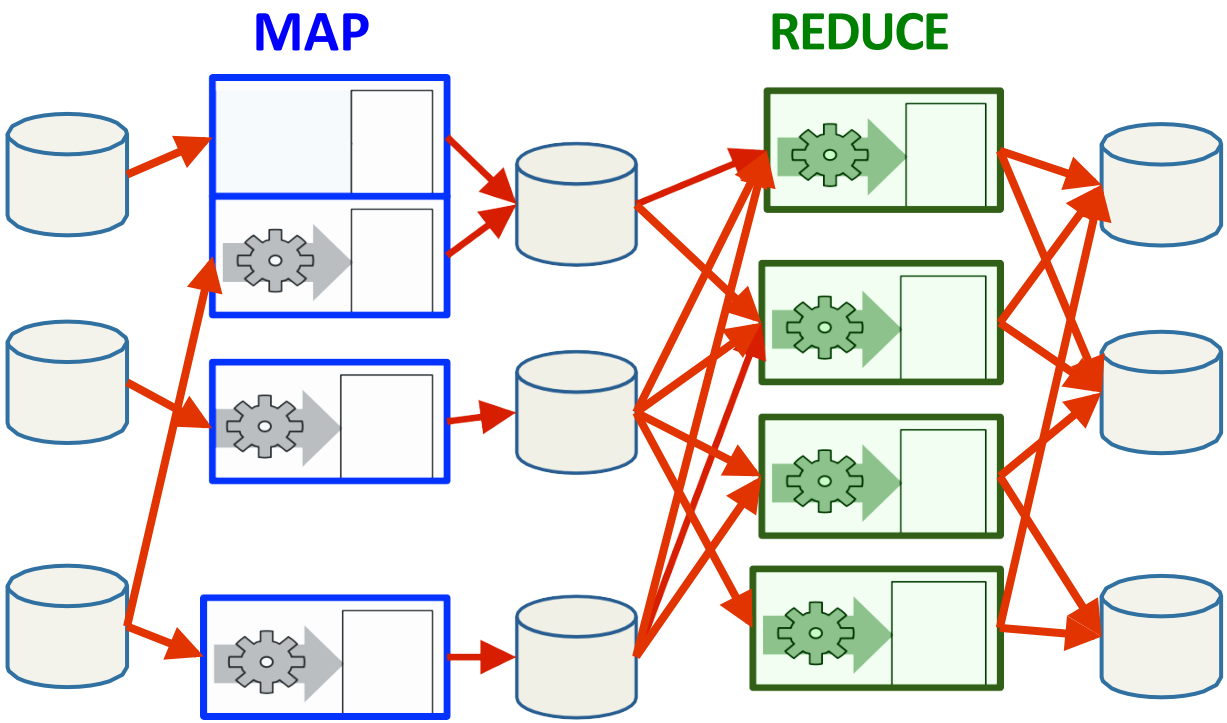


Launch another task!

Outline

- Big Data Trends
- The Big Data Problem
- Hardware for Big Data
  - Distributing Work
  - Handling Failures and Slow Machines
- **MapReduce and Complex Jobs**
  - Apache Spark
- Introduction to Amazon AWS

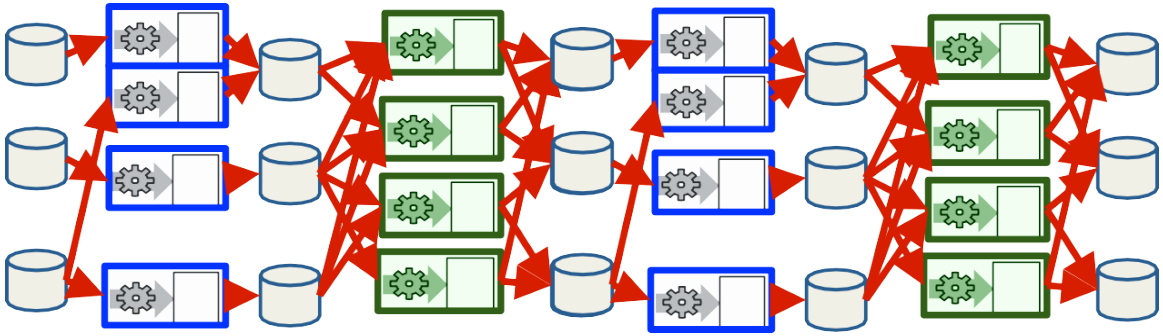
MapReduce: Distributed Execution

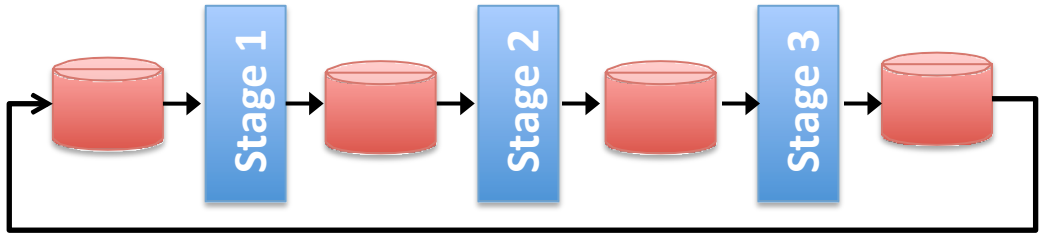


Each stage passes through the hard drives

MapReduce: Iterative Jobs

- Iterative jobs involve a lot of disk I/O for each repetition

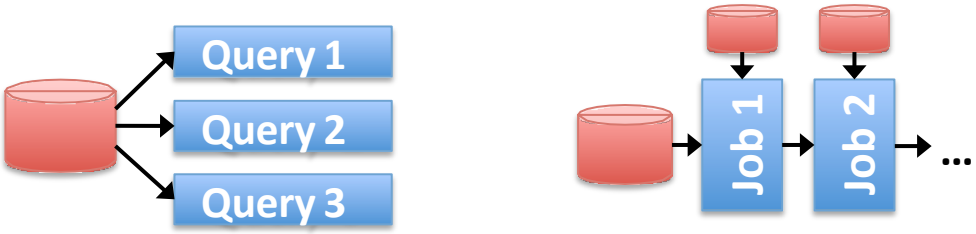




Disk I/O is very slow!

Apache Spark Motivation

- Using Map Reduce for complex jobs, interactive queries and online processing involves *lots of disk I/O*

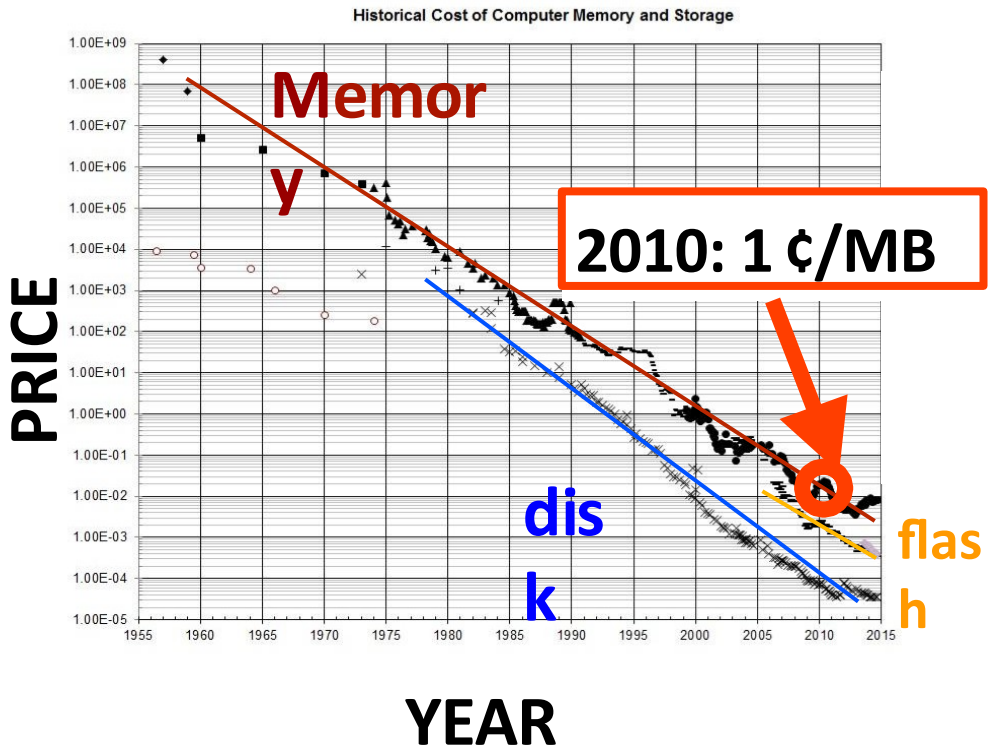


Interactive mining Stream processing

Also, iterative jobs

Disk I/O is very slow

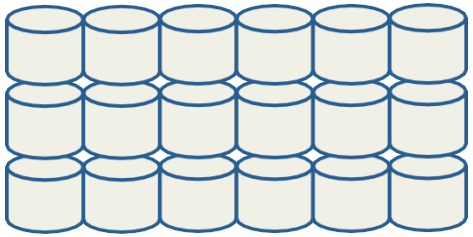
Tech Trend: Cost of Memory



Lower cost means can put more memory in each server

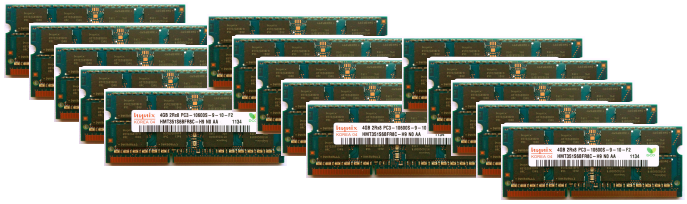
http://www.jcmit.com/mem2014.htm

Hardware for Big Data



Lots of hard drives (and SSDs)

… and CPUs



… and memory!

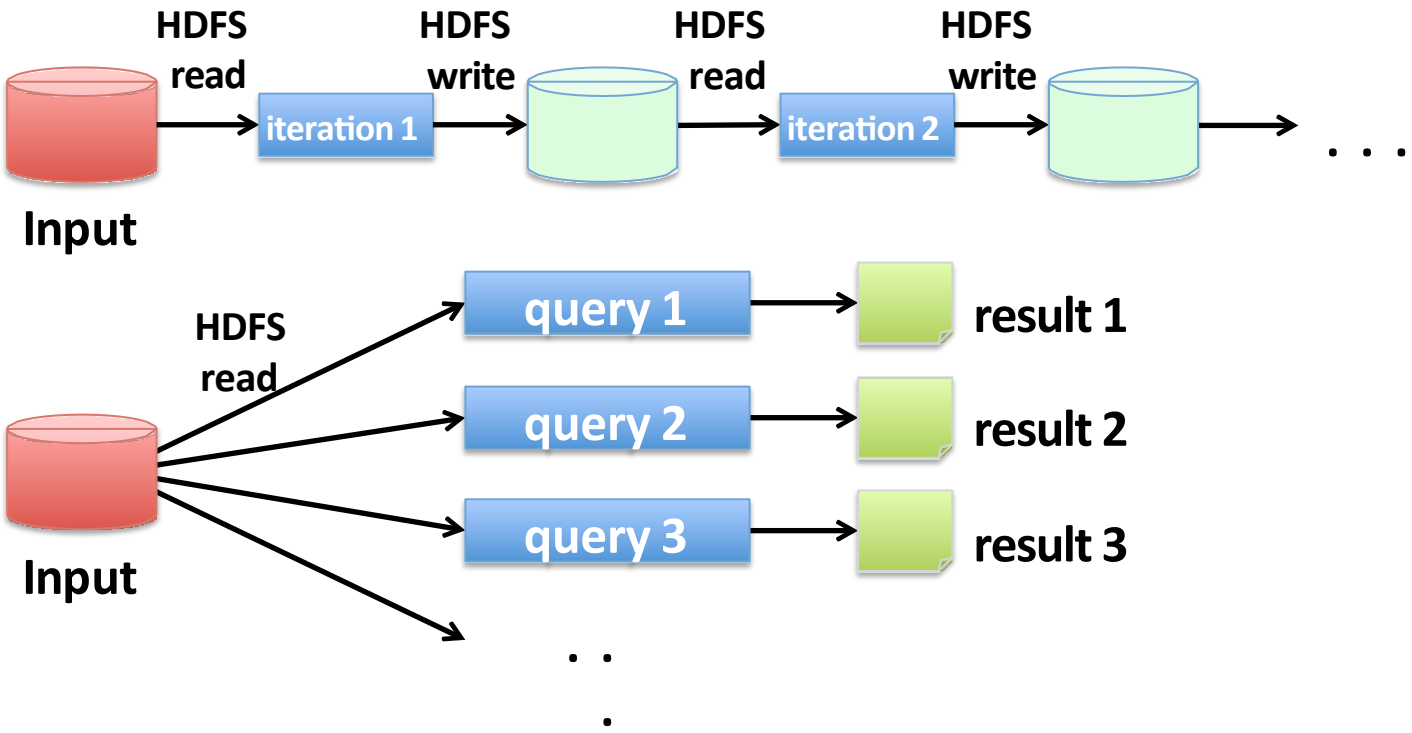
Opportunity

- Keep more data *in-memory*

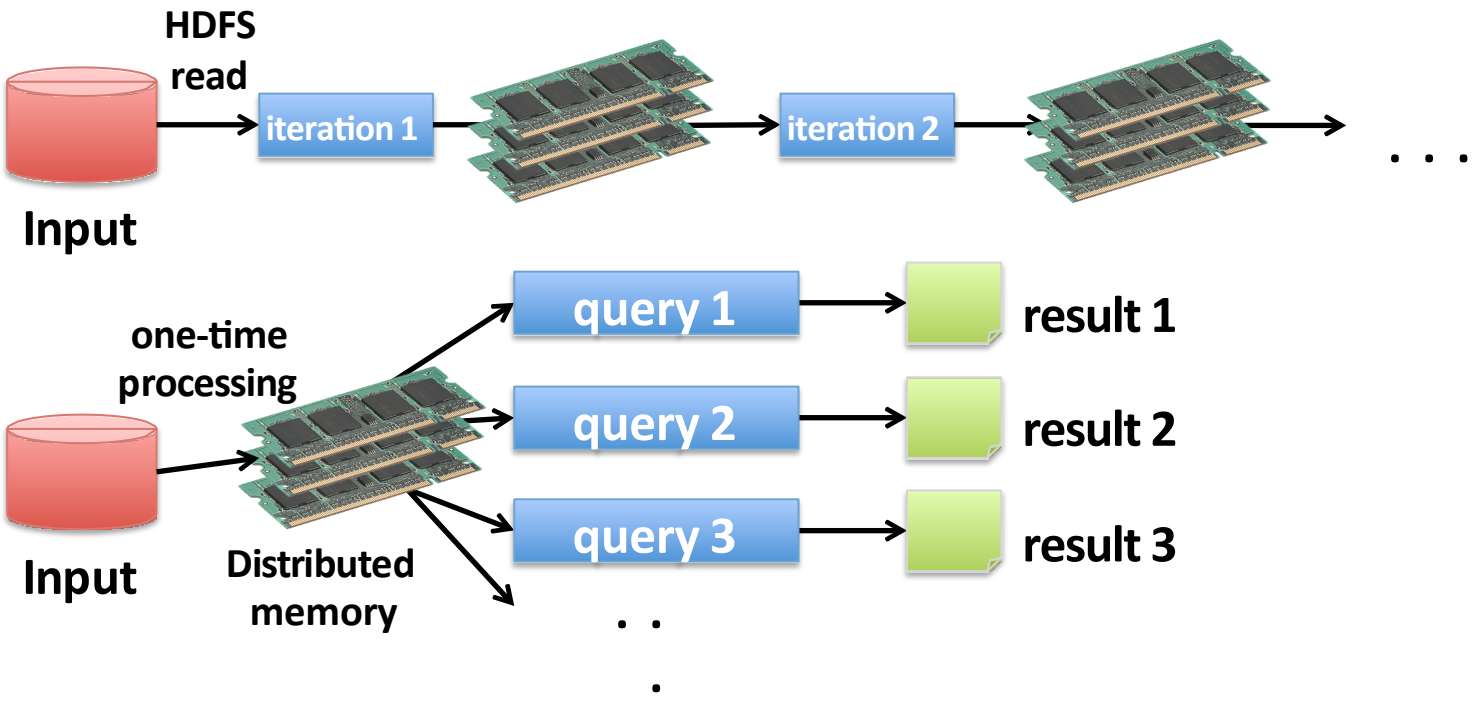
- Create new distributed execution engine:

http://people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf

Use Memory Instead of Disk



In-Memory Data Sharing



10-100x faster than network and disk

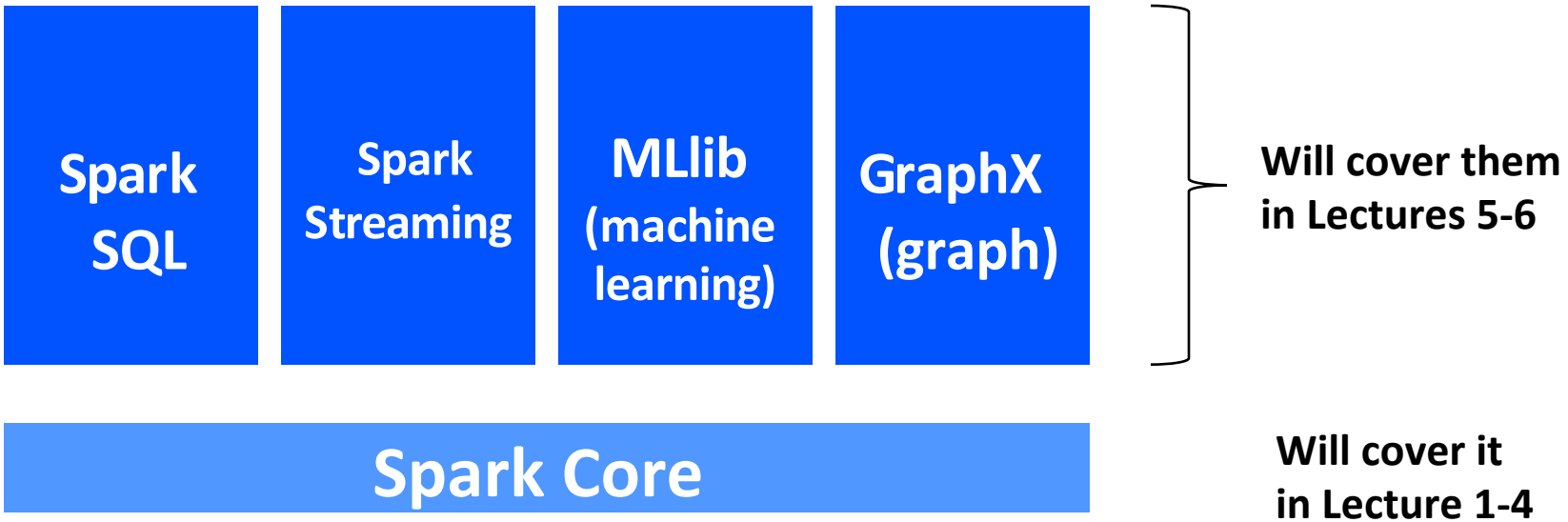
Resilient Distributed Datasets (RDDs)

- Write programs in terms of operations on distributed datasets
- Partitioned collections of objects spread across a cluster, stored in memory or on disk
- RDDs built and manipulated through a diverse set of parallel transformations (map, filter, join) and actions (count, collect, save)
- RDDs automatically rebuilt on machine failure

The Spark Computing Framework

- **Provides programming abstraction and parallel runtime to hide complexities of fault-tolerance and slow machines**
- **"Here's an operation, run it on all of the data"**
  - I don't care where it runs (you schedule that)
  - In fact, feel free to run it twice on different nodes

Spark Components



Spark and Map Reduce Differences

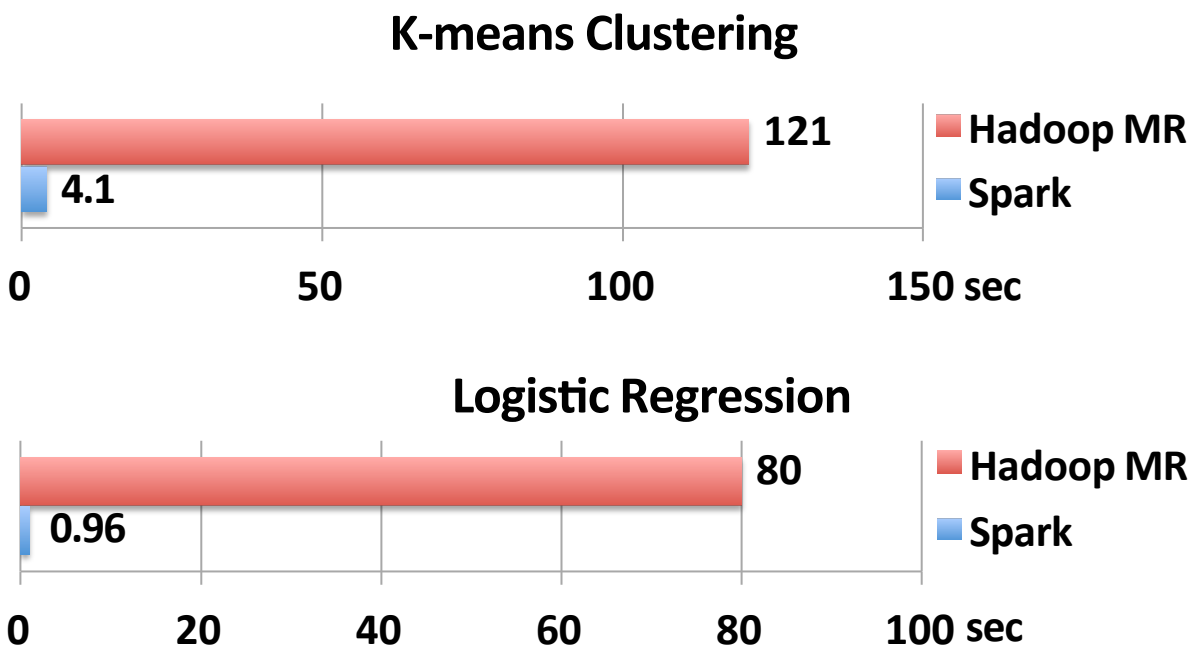
	Hadoop Map Reduce	Spark
Storage	Disk only	In-memory or on disk
Operations	Map and Reduce	Map, Reduce, Join, Sample, etc…
Execution model	Batch	Batch, interactive, streaming
Programming environments	Java	Scala, Java, R, and Python

Other Spark and MapReduce Differences

- **Generalized patterns**
  - unified engine for many use cases
- **Lazy evaluation of the lineage graph**
  - reduces wait states, better pipelining
- **Lower overhead for star ting jobs**
- **Less expensive shuffles**

In-Memory Can Make a Big Difference

- Two iterative machine learning algorithms:



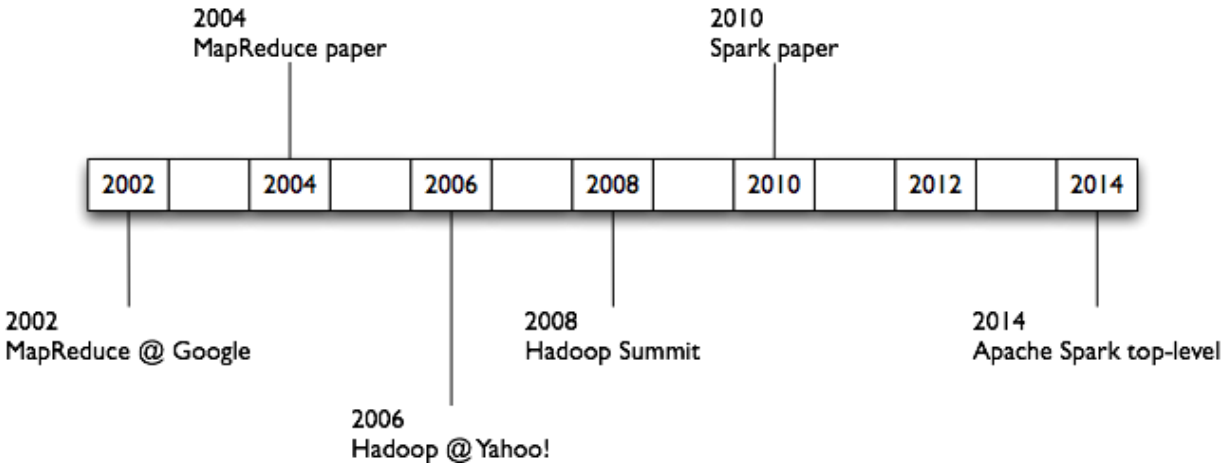
First Public Cloud Petabyte Sort

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Daytona Gray 100 TB
sort benchmark record (tied for 1st place)

http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

History Review



Historical References

- **circa 1979 – Stanford, MIT, CMU, etc.:** set/list operations in LISP, Prolog, etc., for parallel processing
  http://www-formal.stanford.edu/jmc/history/lisp/lisp.htm
- **circa 2004 – Google:** MapReduce: Simplified Data Processing on Large Clusters
  Jeffrey Dean and Sanjay Ghemawat
  http://research.google.com/archive/mapreduce.html
- **circa 2006 – Apache Hadoop:** originating from the Yahoo!'s Nutch Project
  Doug Cutting
  http://research.yahoo.com/files/cutting.pdf
- **circa 2008 – Yahoo!:** web scale search indexing
  Hadoop Summit, HUG, etc.
  http://developer.yahoo.com/hadoop/
- **circa 2009 – Amazon AWS:** Elastic MapReduce
  Hadoop modified for EC2/S3, plus support for Hive, Pig, Cascading, etc.
  http://aws.amazon.com/elasticmapreduce/

Spark Research Papers

- ***Spark: Cluster Computing with Working Sets***
  - Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica
  - USENIX HotCloud (2010)
  - people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf
- ***Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing***
  - Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica
  - NSDI (2012)
  - usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf

Outline

- Big Data Trends
- The Big Data Problem
- Hardware for Big Data
  - Distributing Work
  - Handling Failures and Slow Machines
- MapReduce and Complex Jobs
  - Apache Spark
- **Introduction to Amazon AWS**

Amazon Web Services (AWS)

- 아마존 닷컴이 제공하는 원격 컴퓨팅 서비스 (웹 서비스)
- 다양한 서비스 제공
  - **컴퓨팅**, 스토리지 및 콘텐츠 전송, 데이터베이스, 네트워킹, …
- **Amazon Elastic Compute Cloud (EC2)**
  - 안전하고 크기 조정이 가능한 컴퓨팅 파워를 클라우드에서 제공
  - 웹을 통해 간편하게 필요한 용량을 얻고 구성 가능
  - CPU, 메모리, 스토리지 및 네트워킹 용량의 조합으로 인스턴스 구성

인스턴스 유형	vCPU	메모리 (GiB)	스토리지(GB)	네트워킹 성능	물리적 프로세서	클록 속도 (GHz)	인텔 AVX†	인텔 AVX2†	인텔 Turbo	EBS OPT	향상된 네트워킹†
m4.large	2	8	EBS 전용	중간	인텔 제온 E5-2676 v3**	2.4	예	예	예	예	예
m4.xlarge	4	16	EBS 전용	높음	인텔 제온 E5-2676 v3**	2.4	예	예	예	예	예
m4.2xlarge	8	32	EBS 전용	높음	인텔 제온 E5-2676 v3**	2.4	예	예	예	예	예

* https://aws.amazon.com/ko/ec2/instance-types/

* https://ko.wikipedia.org/wiki/%EC%95%84%EB%A7%88%EC%A1%B4_%EC%9B%B9_%EC%84%9C%EB%B9%84%EC%8A%A4#/media/File:AmazonWebservices_Logo.svg

Amazon AWS Global Regions Locations

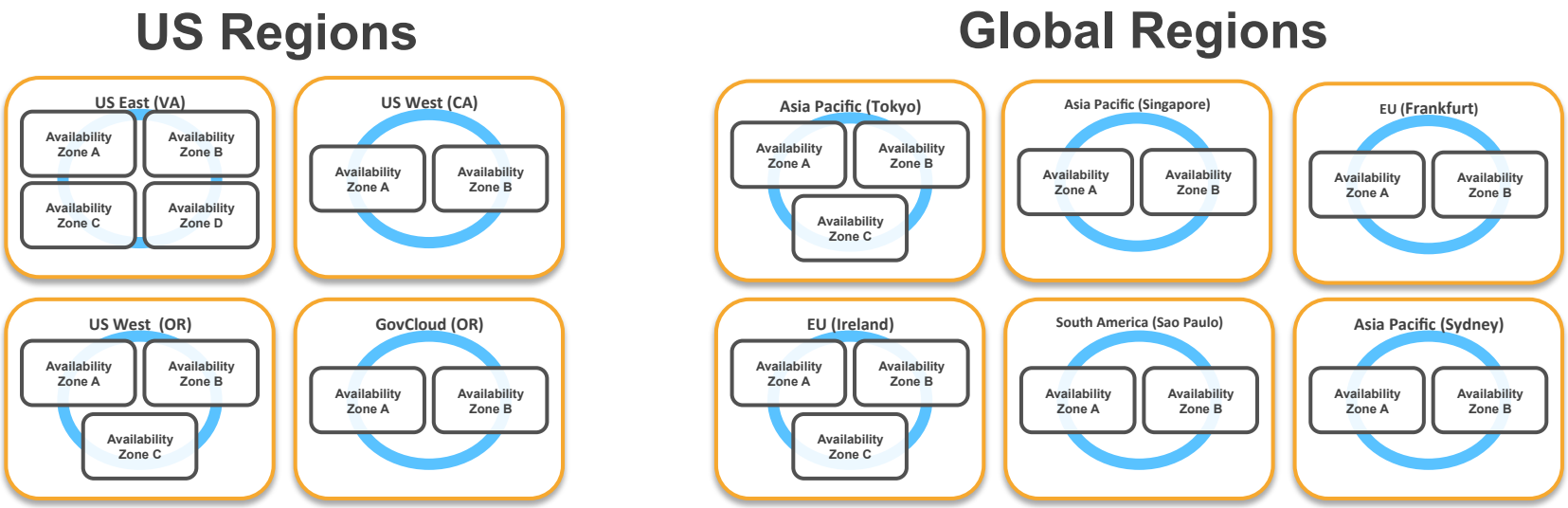
Source: Amazon AWS (2015)

Amazon AWS Global Regions Locations

Source: Amazon AWS (2015)

Amazon AWS Global Regions Locations

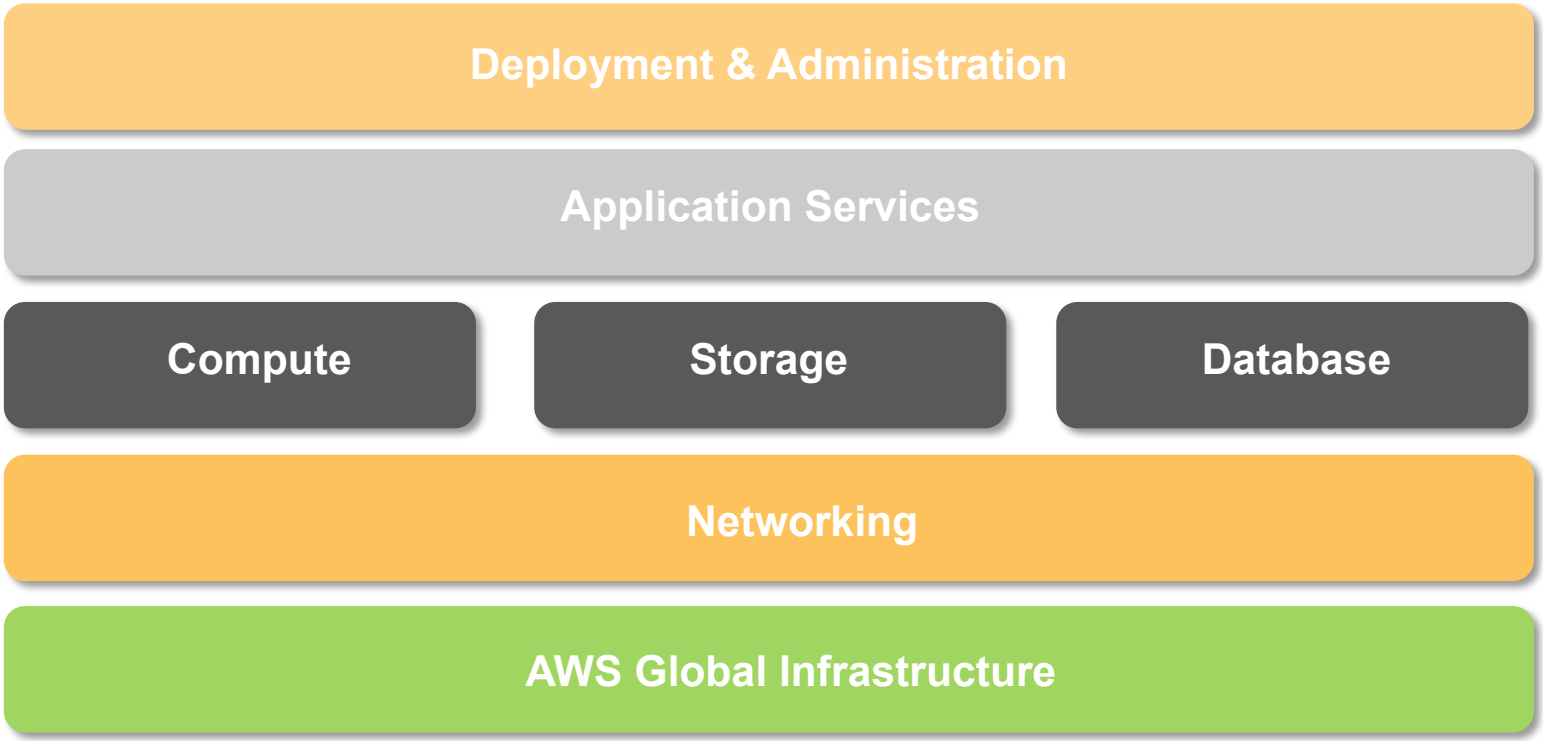
- **AWS Regions & Availability Zones**
  - Customer decides where applications and data reside



Note: Conceptual drawing only. The number of Availability Zones may vary.

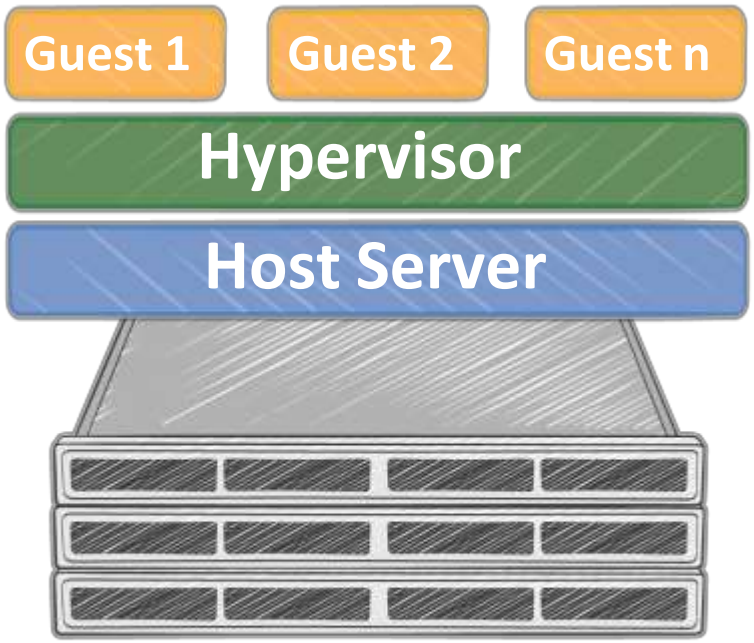
Services to Support Cloud Workloads

- AWS service stack



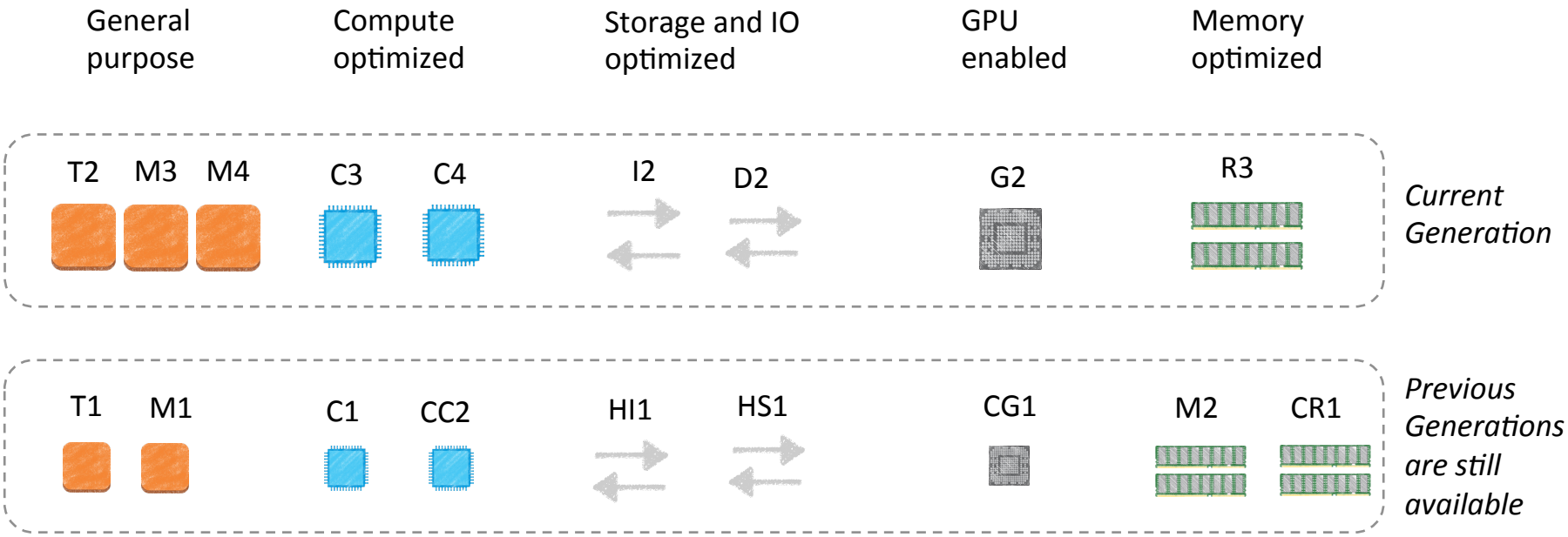
Amazon Web Services EC2

- **EC2 = Elastic Compute Cloud**
  - Web service that enables you to launch and manage Linux/ UNIX and Windows server in Amazon's data centers
- **EC2 instances are elastic virtual servers in the cloud**



Broad Set of Compute Instance Types

- As of 2015 (probably outdated a bit)



Broad Set of Compute Instance Types

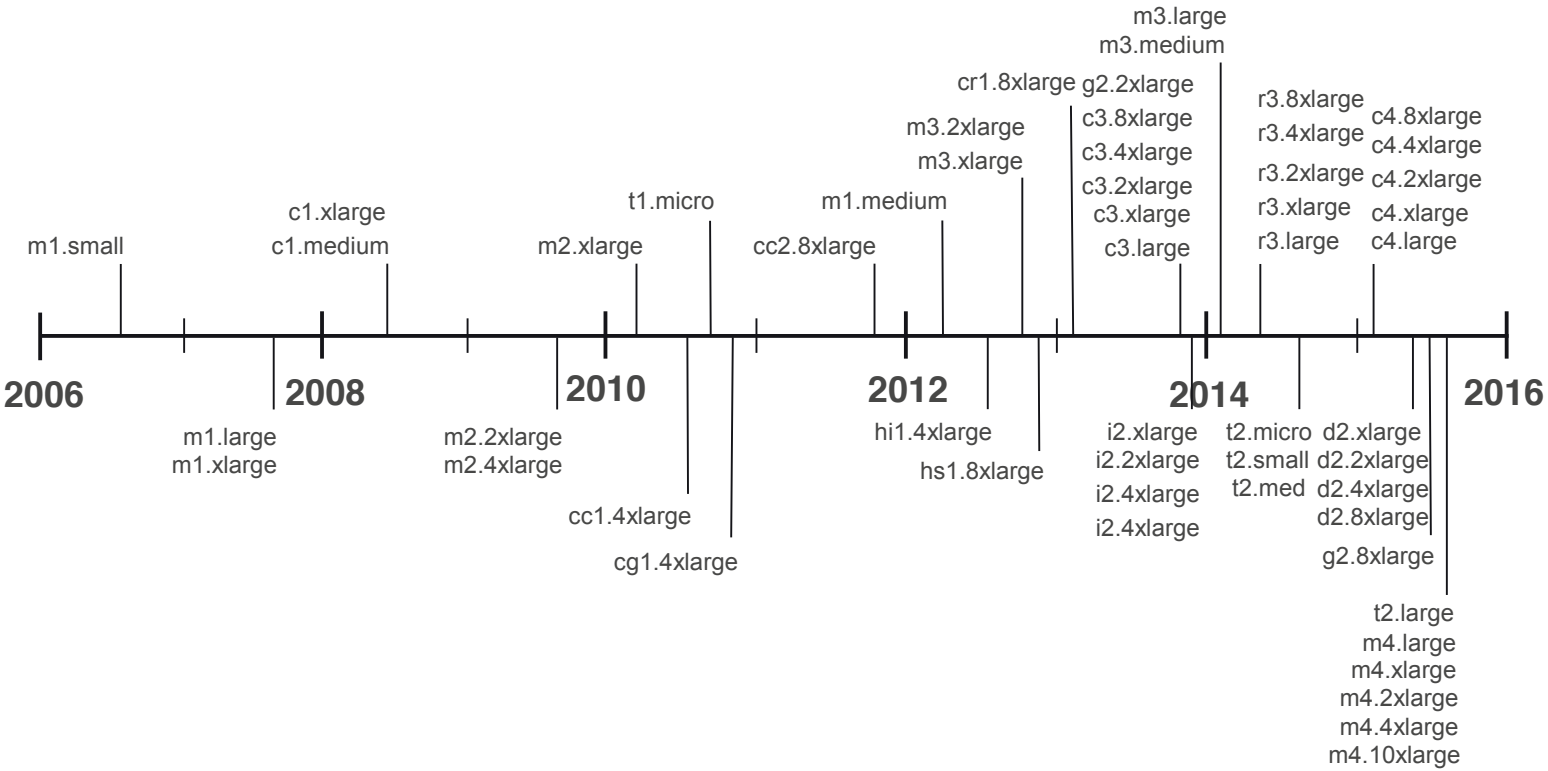
- **Instace selection: type & size**

- **Instance type**
  - Pricing: Spot vs. On-demand vs. Reserved
  - Family: Storage vs. Compute vs. GPU vs. Memory vs. General Purpose
  - Generation: Current vs. Previous

- **Instance size**
  - Determined by workload size: compute units, memory, storage
  - Micro, Small, Medium, Large, Extra Large (XL)
    - Multiple XL sizes: xlarge, 2xlarge, 4xlarge, 8xlarge
  - You may want to compare performance against a reference server

Broad Set of Compute Instance Types

- **Amazon EC2 instances history**



Typical User Scenario

1. **Sign-up for an AWS account**
   - Use AWS IAM – Identity and Access Management
2. **Launch an EC2 instance**
   - Via user chosen tool such as GUI, CLI, or external
3. **Use key credentials to access the EC2 instance**
   - You'll do this tomorow.
4. **Install your software on the EC2 instance**

We already did Step 1 & 2 for you. Every student will receive his/her own EC2 instance!

VM list: https://docs.google.com/spreadsheets/d/1X9Uavr2PACqgfLC3rOcQ7Gqo4-NQNKoBhvcaL_86g9E

Instance Storage

- **Instances have ephemeral storage (Current Gen has SSDs)**
  - General Purpose instances have GBs to TBs
  - Storage Optimized instances have up to 48TB
  - Data is lost when the instance dies
- **EBS – Elastic Block Storage**
  - Persistent block level storage volumes for use with EC2 instances
  - Cost associated – 1 TB costs $100/month
  - Data is not lost when instance dies – can be remounted with new instance
  - For storage needs larger than 16 TB, RAID required
  - Built-in resiliency – data is backed up
- **S3 – Simple Storage Service**
  - Online cloud storage service (files, data, etc…)
  - Need this for backup purposes (Snapshots)

Instance Storage

- **EFS – Elastic File Storage**
  - Scalable storage with relatively fast speed (faster than S3)
  - Automatically scalable (unlike EBS)
  - Especially helpful for running servers, shared volumes (like NAS devices)

AMAZON S3	AMAZON EBS	AMAZON EFS
Can be publicly accessible	Accessible only via the given EC2 Machine	Accessible via several EC2 machines and AWS services
Web interface	File System interface	Web and file system interface
Object Storage	Block Storage	Object storage
Scalable	Hardly scalable	Scalable
Slower than EBS and EFS	Faster than S3 and EFS	Faster than S3, slower than EBS
Good for storing backups	**Is meant to be EC2 drive**	**Good for shareable applications and workloads**

Source: https://www.cloudberrylab.com/blog/amazon-s3-vs-amazon-ebs/

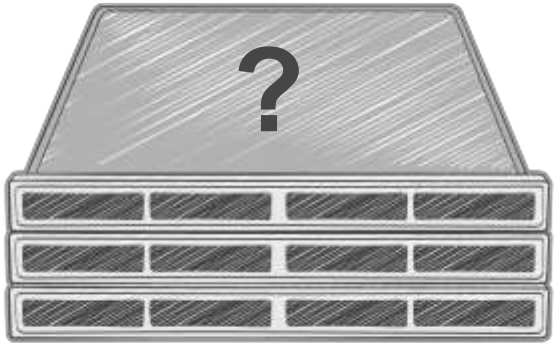
Instance Storage

- **Best practices for single instances (or non-replicated distributed deployments)**
  - Use EBS volumes for indexes and the OS/software
  - RAID can be an extra measure of reliability, but will consume CPU
  - Use snapshots to backup the instance (S3)
  - IOPS optimized provides benefits
  - XFS preferred (customer feedback)
  - c4 (compute optimized) instances will require storage

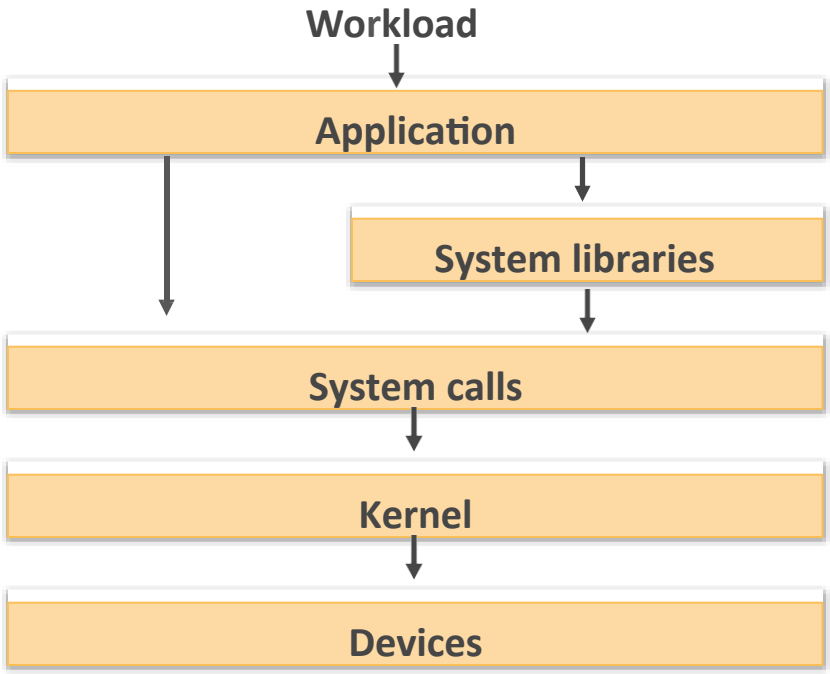
Source: Simeon Yep (Splunk)

Selecting Instances

- **Hiring a server**
  - Servers are hired to do jobs
  - Performance is measured differently depending on the job



- **Defining "performance"**
  - Response time
  - Throughput
  - Consistency
  - etc.



Selecting Instances

- **Performance factors**

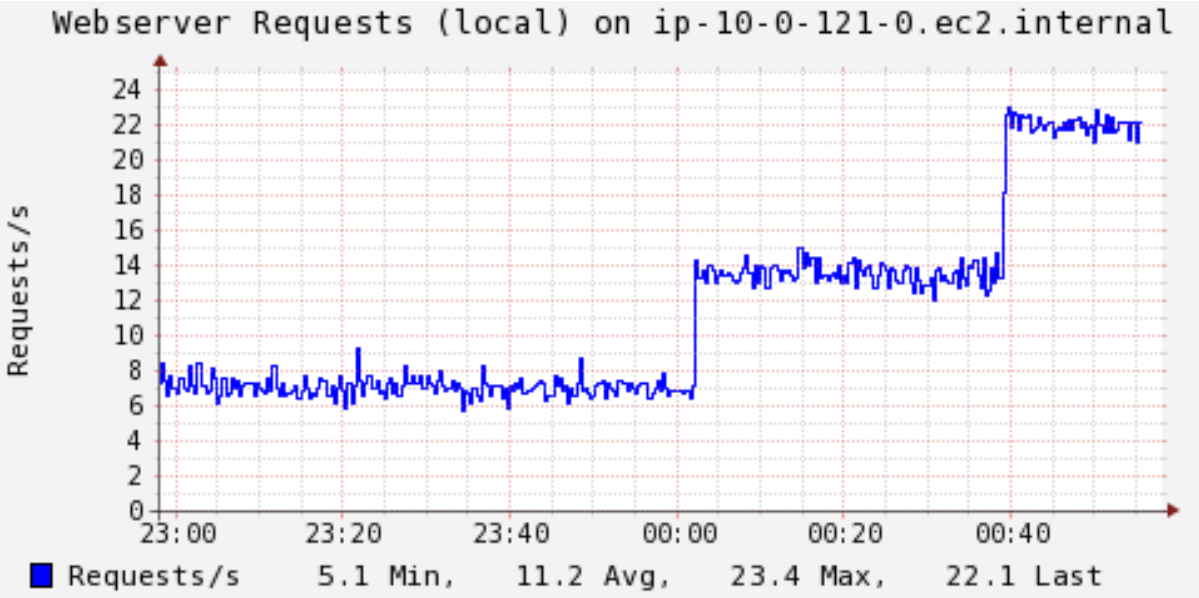
Resource	Performance factors	Key indicators
CPU	Sockets, number of cores, clock frequency, bursting capability	CPU utilization, run queue length
Memory	Memory capacity	Free memory, anonymous paging, thread swapping
Network interface	Max bandwidth, packet rate	Receive throughput, transmit throughput over max bandwidth
Disks	Input / output operations per second, throughput	Wait queue length, device utilization, device errors

Selecting Instances

- **Resource utilization**
  - For given performance, how efficiently are resources being used
  - Something at 100% utilization can't accept any more work
  - Low utilization can indicate more resource is being purchased than needed

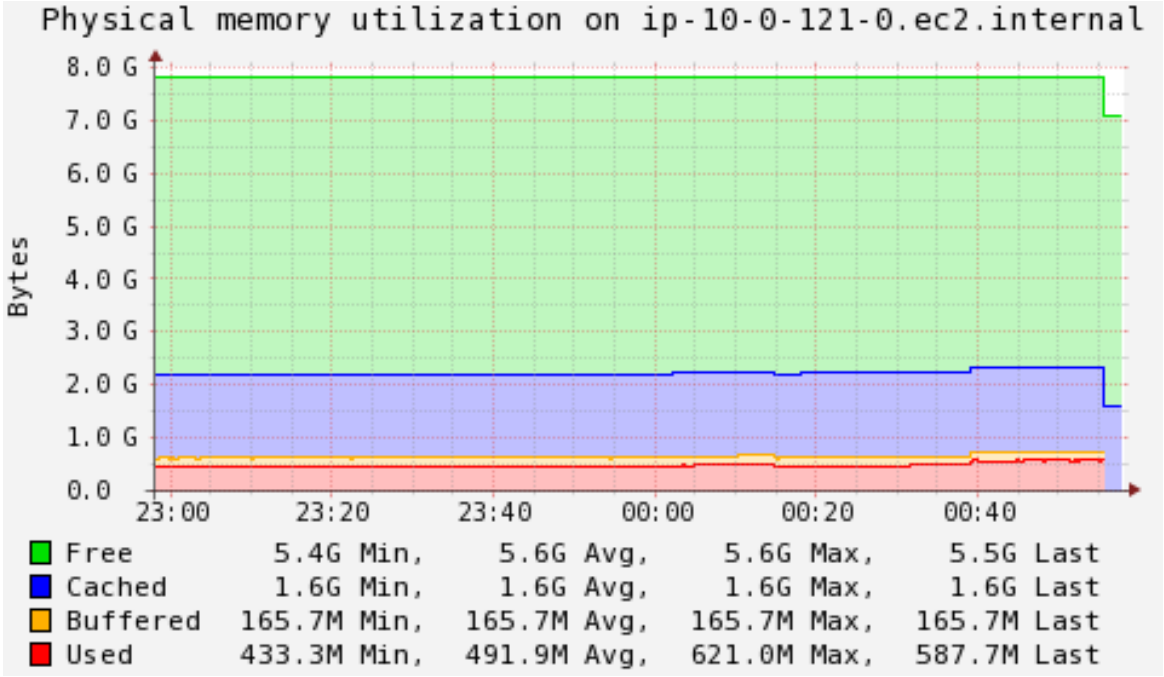
Selecting Instances

- **Example: Web application**
  - MediaWiki (like Wikipedia) installed on Apache with 140 pages of content
  - Load increased in intervals over time



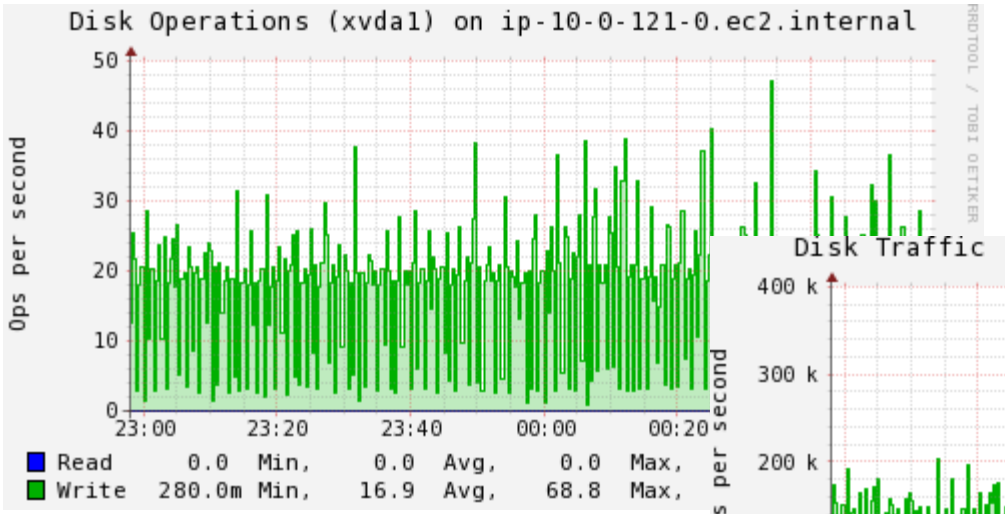
Selecting Instances

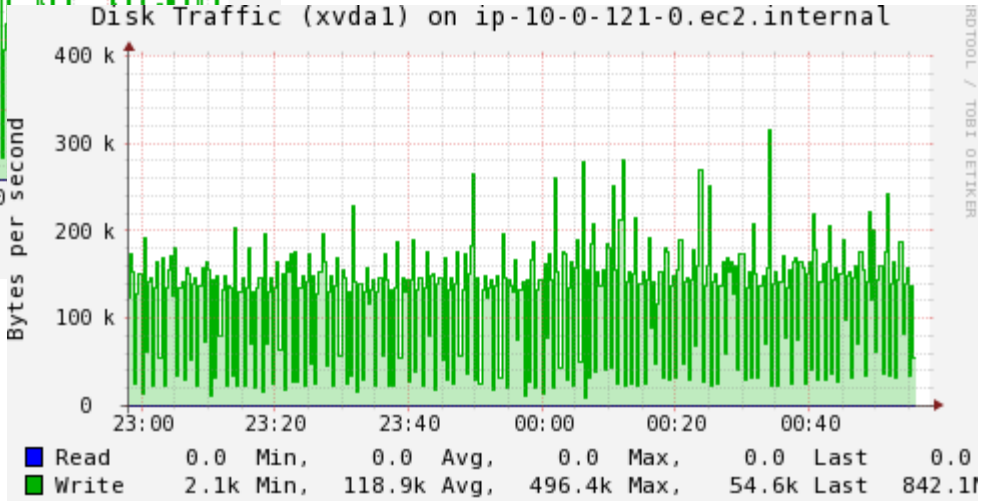
- **Example: Web Application**
  - Memory stats



Selecting Instances

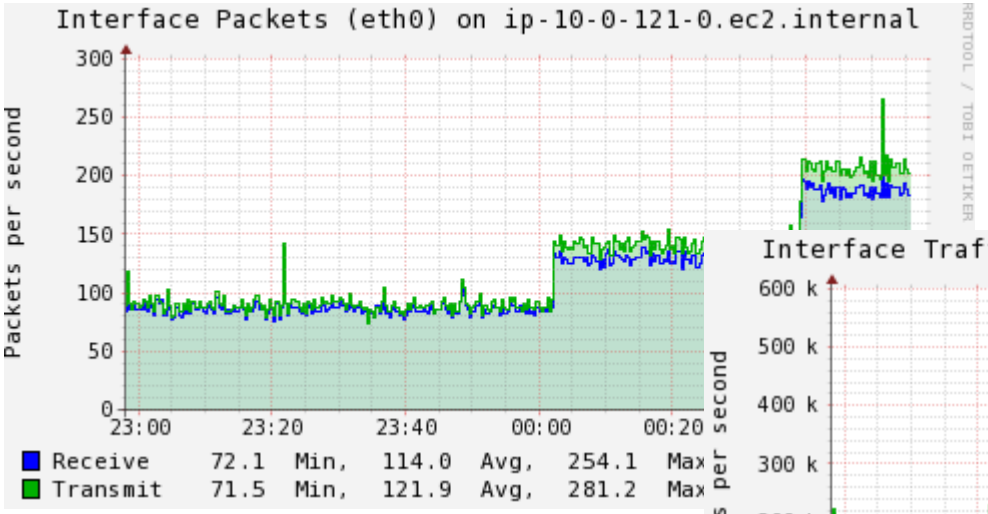
- **Example: Web Application**
  - Disk stats

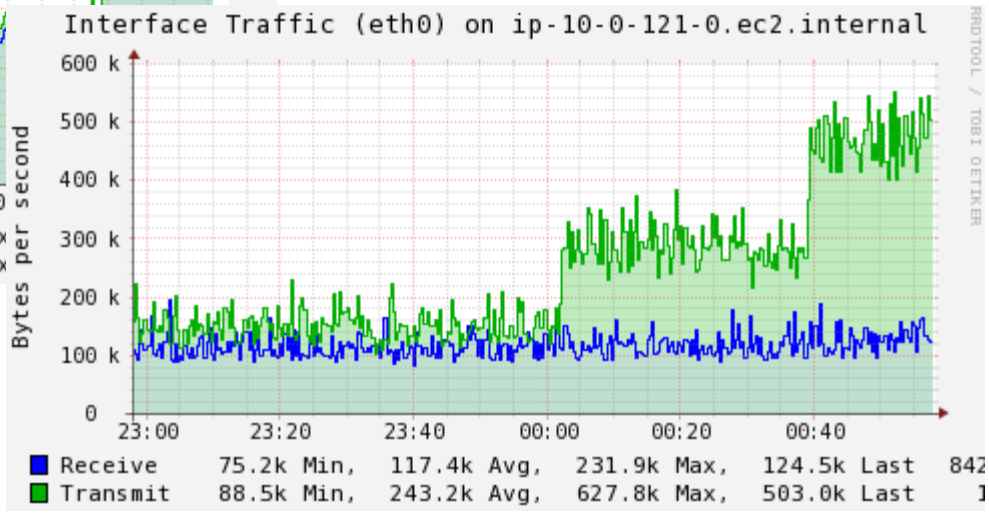




Selecting Instances

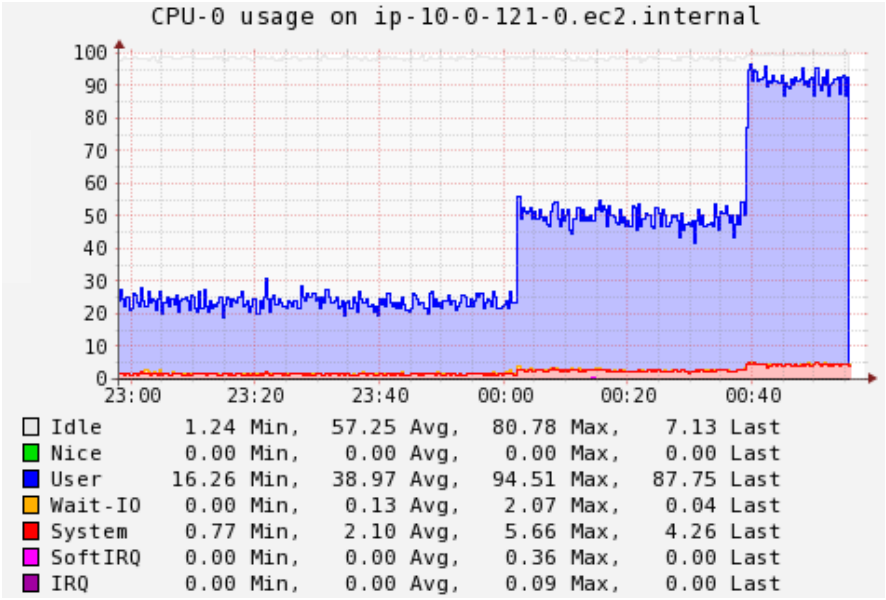
- **Example: Web Application**
  - Network stats

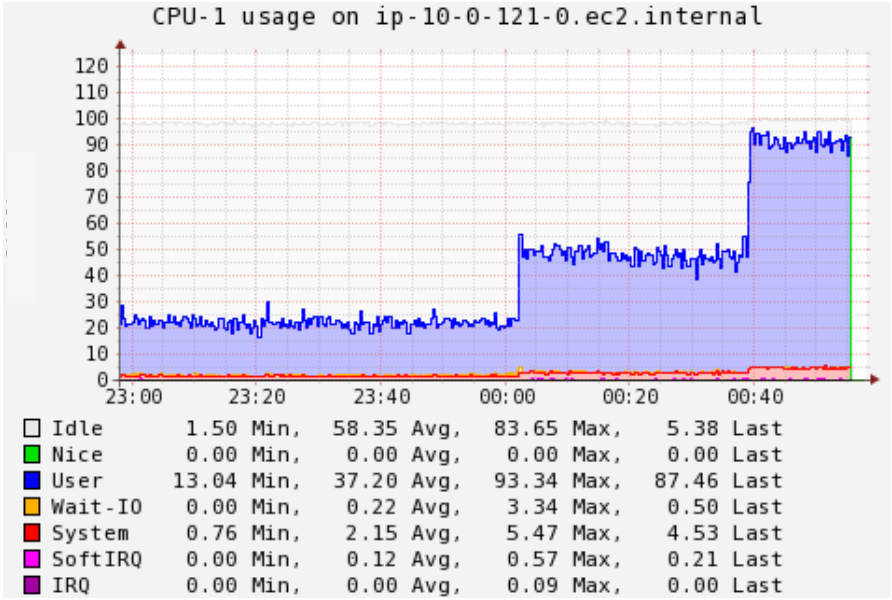




Selecting Instances

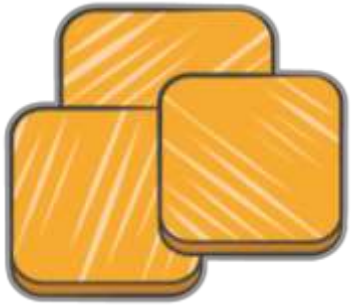
- **Example: Web Application**
  - CPU stats





Selecting Instances

- **Instance Selection = Performance Tuning**
  - Picking an instance is tantamount to resource performance tuning
  - Give back instances as easily as you can acquire new ones
  - Find an ideal instance type and workload combination



Summary

We have covered the following topics in this lecture:

- Course logistics
- Big data trends and problems
- How to distributed big data work to a cluster of machines
- MapReduce computation model
- Motivation for in-memory processing with Apache Spark
- Brief introduction to Amazon AWS and EC2