UNIVERSITY of WASHINGTON

BIG DATA 210: Introduction to Data Engineering

Autumn 2018

Module 1: Introduction to Big Data

Jerry Kuch

jkuch@uw.edu

Program and Course:

Schedule and Logistics

Logistics

- Wednesdays, 6-9PM from 10/3/2018 – 12/12/2018
  - Classroom:
    - Puget Sound Plaza Room 403
    - 1325 Fourth Ave., Suite 400, Seattle, WA, 98101
  - Online: Zoom Meetings
    - https://washington.zoom.us/my/bigdata
  - Course Website:
    - https://canvas.uw.edu/courses/1243280
- Typical Class:
  - Recap of previous class and review of assignments
  - Presentation of new material
  - Hands-on lab/assignment time
- Assignments
- Discussion Forums
- Final Project
- ***PLEASE GIVE FEEDBACK EARLY AND OFTEN!!!***

Courses of the Program

- BIGDATA 210
  - Introduction to Data Engineering
- BIGDATA 220
  - Building the Data Pipeline
- BIGDATA 230
  - Emerging Technologies in Big Data

Course Schedule

- Introduction to Big Data and Cloud Computing (1 week)
  - Course Logistics
  - Big Data Fundamentals
  - Cloud Computing Fundamentals, Course Sandbox Working Environment
- Scalable Computing: Hadoop and the Hadoop Ecosystem (1 week)
- Data Processing Using Apache Spark (3 weeks)
  - Spark architecture, functional programming ideas, RDDs
  - Spark partitioning/shuffling; persistence and shared variables, DataFrames, Datasets
  - Spark SQL; Notebooks: Jupyter and Zeppelin
- *A Possible Overflow Week*
- NoSQL Systems (2 weeks)
  - Big data management; Introduction to NoSQL; Redis; Hbase
  - Cassandra, Elasticsearch
- Beneath the Big Picture (or in-class project teamwork/collaboration session)
- Project Presentations (1 week)

Week 1

Introduction to Big Data
and Cloud Computing

Week 1 Agenda

- Class Introductions
- Course Objectives
- Introduction to Big Data Engineering
- Cloud Computing Basics
- Week 1 Homework

Introductions

Introductions: Me

- Who am I?
- Email: jkuch@uw.edu
- Office Hours:
  - The hour before class, either here in this room or down in the deli.
  - By arrangement: online or in-person.

Introductions: Who's the Other Guy?

https://www.manning.com/livevideo/spark-in-motion

Introductions: You!

- Introduce Yourselves!
  - Name
  - Brief summary of current work/experience
  - What you want to get out of this course
- Online Students
  - **Reminder:** it can be hard to see you so make your presence known!
- Discussion Forums

Course Objectives

Course Objectives

- To understand the building blocks and development process of modern analytics applications
- To be able to use a variety of available technologies and understand how to make choices between current and future options
- To prepare students for the next two courses
- To discriminate between Fact, Fiction and Myth
- To focus on real-world applications

Course Objectives

Introduction to Big Data Engineering

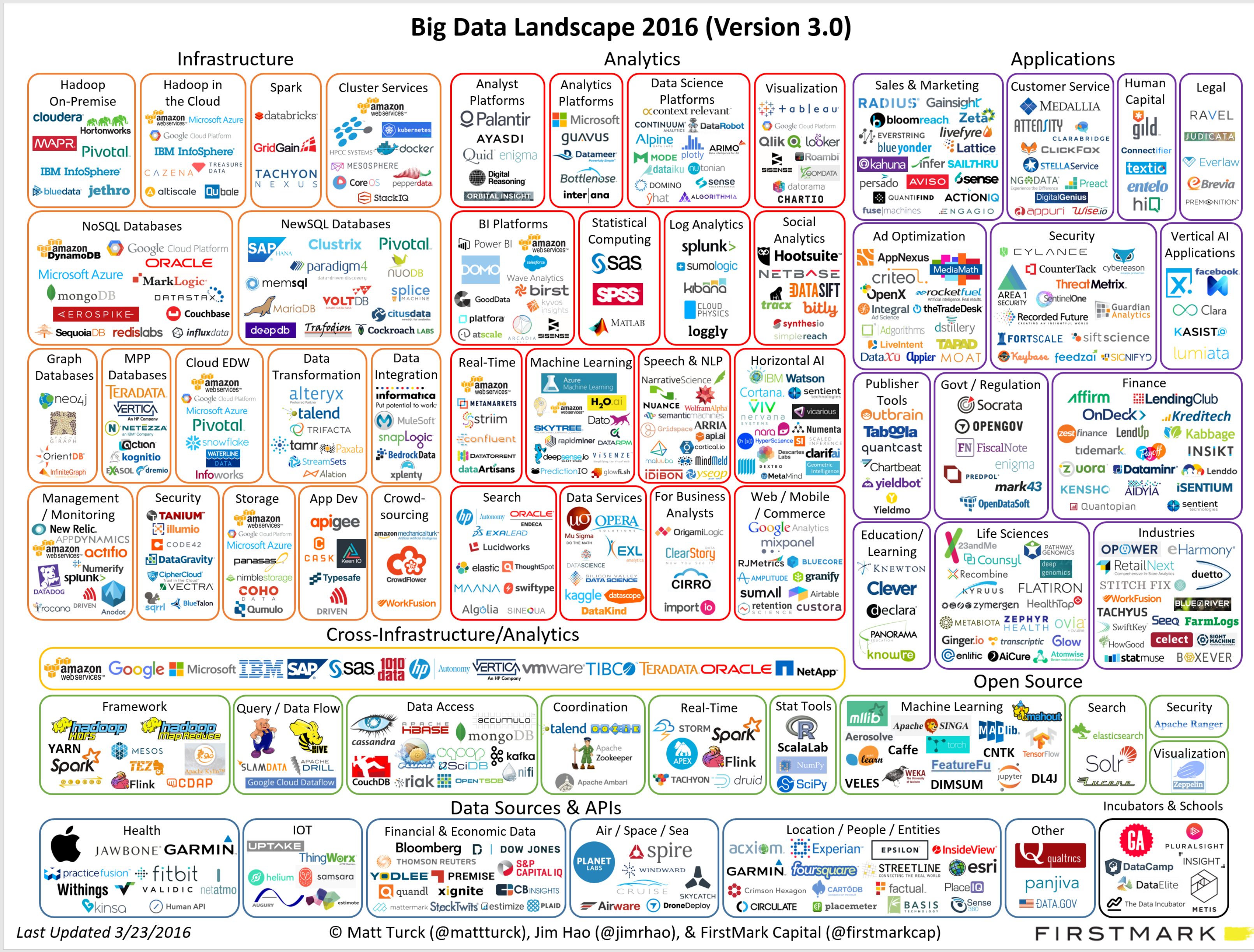
Introduction to Big Data

- ***Big Data*: data that is too large or complex to process on a single machine**
- ***The Traditional V's of Data*:**
    - *Volume*
    - *Velocity*
    - *Variety*
- ***Various New V's of Data:***
    - Variability, *Veracity*, Value, etc…
- (Big Data) != (Just Hadoop)

Introduction to Big Data

- ***Today we want:*** A collection of technologies that allow for processing any size or type of data to meet any requirements.
- ***Implications:***
  - Degree and type of structure in data may vary
  - How data will be used may be uncertain at first and change over time
  - “Big” often means roughly “too big for one machine” but this isn’t universal

Big Data Systems: Lay of the Land



Big Data Throughout Time

- Brief History
  - Analytics is not new
  - Enterprise Data Warehouse
    - Rigid Data Models
    - Expensive hardware/software
  - Extract-Transform-Load (ETL)
    - Complex time consuming processes to model data so it can be useful

Big Data: The Recent Past

- **Early 2000s:**
  - **Google, then Yahoo**
    - Size of data => *distribution* of storage and processing
    - Distributed computing is hard!
    - *Inevitability* of Commodity Hardware => *costs* & *opportunities.*
- **2006 and Beyond:**
  - **Amazon Web Services**
    - **Enabled by:** Virtualization and management of distributed systems at scale.
    - **Benefits:**
      - Pay as you go rental of storage and compute resources
      - Over time higher level services arrived with pay as you go rental model
      - Spend on what you need when you need it

Big Data: For the Masses

- Commercial Hadoop Distributions
  – Cloudera, Hortonworks, MapR…

- Still require complex engineering:
  - Many tools in the Hadoop and related ecosystems
  - Deployed at scale require management
  - Want data operations to run smoothly

Big Data (cont.)

Problems with Hadoop

Big Data: Hadoop's Ugly Early Days

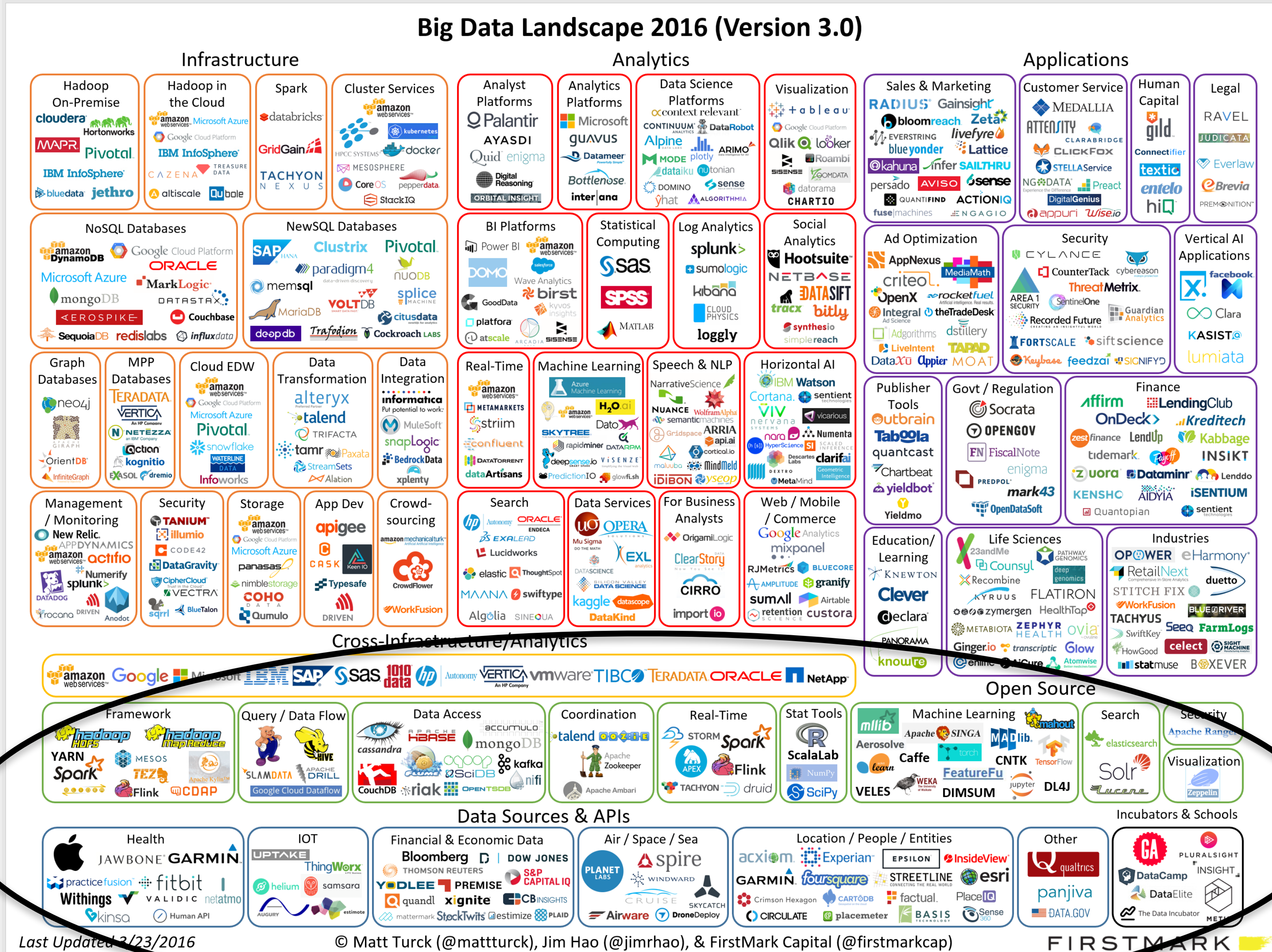
```
// mapper
private IntWritable one = new IntWritable(1);
private IntWritable output = new IntWritable();
protected void map(LongWritable key, Text value, Context context) {
    String[] fields = value.split("\t");
    output.set(Integer.parseInt(fields[1]));
    context.write(one, output);
}


// reducer
IntWritable one = new IntWritable(1);
DoubleWritable average = new DoubleWritable();
protected void reduce(IntWritable key, Iterable<IntWritable> values,
Context context) {
    int sum = 0;
    int count = 0;
    for(IntWritable value : values) {
        sum += value.get();
        count++;
    }
    average.set(sum / (double) count);
    context.Write(key, average);
}
```

Big Data: Hadoop Seeks Usability

Big Data: Hadoop's Discontents

- **Hadoop Speed (or lack thereof…)**
  - All I/O and lifecycle dependent on HDFS
  - Every job begins and ends with HDFS read/ write
  - Real-world applications require multiple stages of transformations
- Just one part of what is needed for a complete solution
- Still need to ETL

Big Data: Where we are today



Big Data: The Operational Problem

Data is useless if it isn't in a usable state and accessible by the consumer in a timely manner

Big Data: Where Things Are Going?

CIO Magazine Data Trends for 2017:

The emergence of the data engineer: *The term, "data scientist," will become less relevant, and will be replaced by "data engineers," DataStax says.*

Data scientists *focus on applying data science and analytic results to critical business issues.*

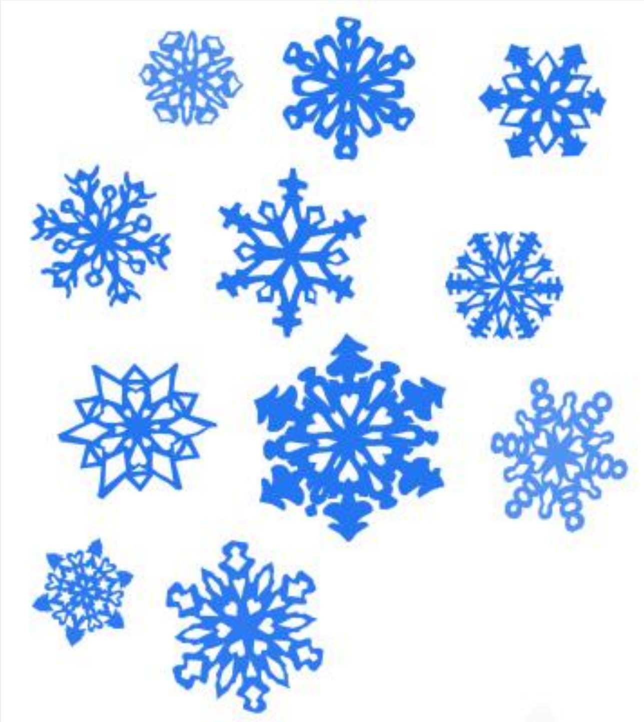
Data engineers*, on the other hand, design, build and manage big data infrastructure. They focus on the architecture and keeping systems performing."*

Big Data: Data Science and Data Engineering

- Data Engineer vs Data Scientist
  - What is science? What is engineering?
  - Is it a distinction without a difference?
- Possible Analogy:
  - Data Scientist : Industrial Lab Chemist
  - …as…
  - Data Engineer : Chemical or Process Engineer designing and running a plant

Big Data: Data Engineering

- Data Engineering



- Batch
- Real-time/Streaming
- Machine Learning
- Predictive
- Graphs
- Etc…

Big Data: Data Engineering

- **What is a well engineered solution?**
  - PURPOSEFUL
  - The “correct” component(s)
  - To solve customer requirements
    - Both explicit and implicit
  - With ~~good~~ GREAT end to end performance
- **Desirable properties:**
  - *Robustness*, *reliability*: the safety of work and data
  - *Repeatability:* to ease experiments, modeling, and work in production
  - Easy and efficient *experimentation*, *adaptation*, *enhancement* and *repair*

Introduction to Cloud Computing

What is Cloud Computing?

Cloud Computing

- **What the !?#*? is "the Cloud?"**
  - Most basically: running stuff and/or storing stuff, *somewhere else*
  - Perhaps operated and kept up, in whole or in part, *by someone else*
- **A model for providing (and paying for) computing resources and services**
  - In some ways recalls the mainframe era notion of *utility computing*

Cloud Computing

- **Public Cloud vs Private Cloud**
  - *Public:* e.g. AWS, Azure, GCP
  - *Private:* most places this means "machine room"
  - *Hybrid:* a weird, awkward, muddy area.
- **Virtual *Servers* vs Hosted *Services***
  - Do you get physical HW? A VM? Containers?
  - Or do you get a running, maintained system you can use? (e.g. Amazon EMR, SQS, etc.)

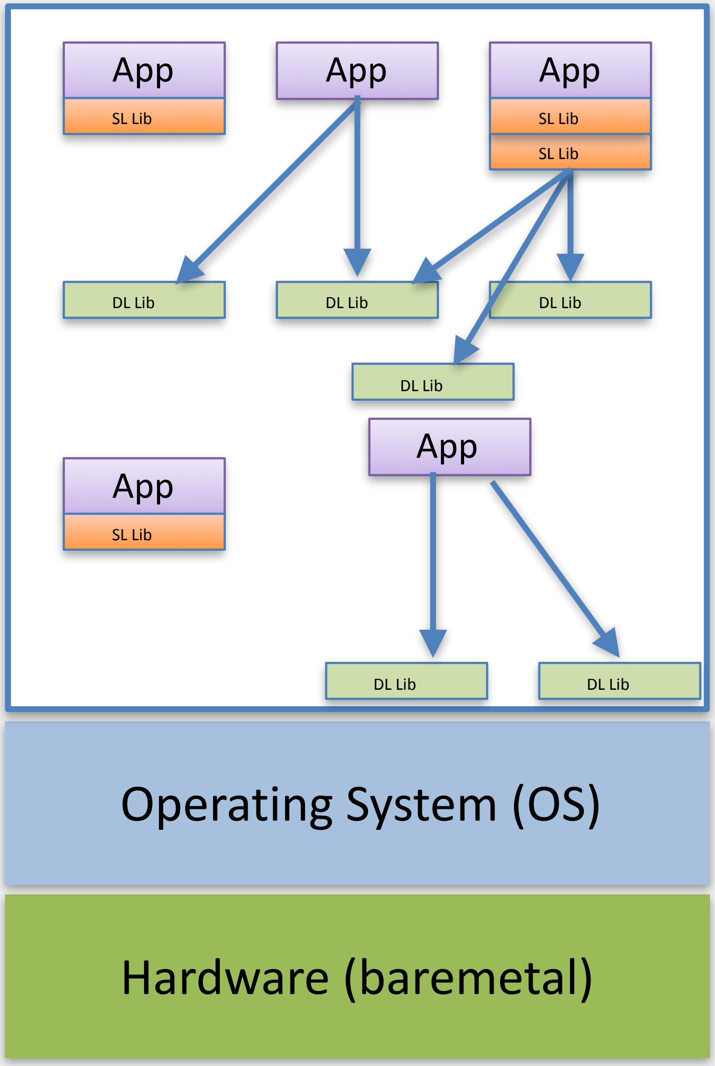
Cloud Computing: Enabling Technologies

Enabling Technologies for the Modern Cloud

- Raw hardware resources are divided, shared and managed through:
  - Operating Systems
  - Virtualization and *Virtual Machines*
  - *Containers*
- We look at each in turn and see:
  - What they provide us
  - What they demand of us

Your Own Physical Server and OS: The Old Days

- What are the pieces from the bottom up?
  - HW, OS, apps, libraries
- What does each do?
- Things to note:
  - You're paying when not using, one way or the other
  - What all gets shared?
  - Noisy neighbor problem.
  - Polluted systems.
  - System upgrades => probably lost availability
  - "DLL Hell"



(Re)-enter Virtualization

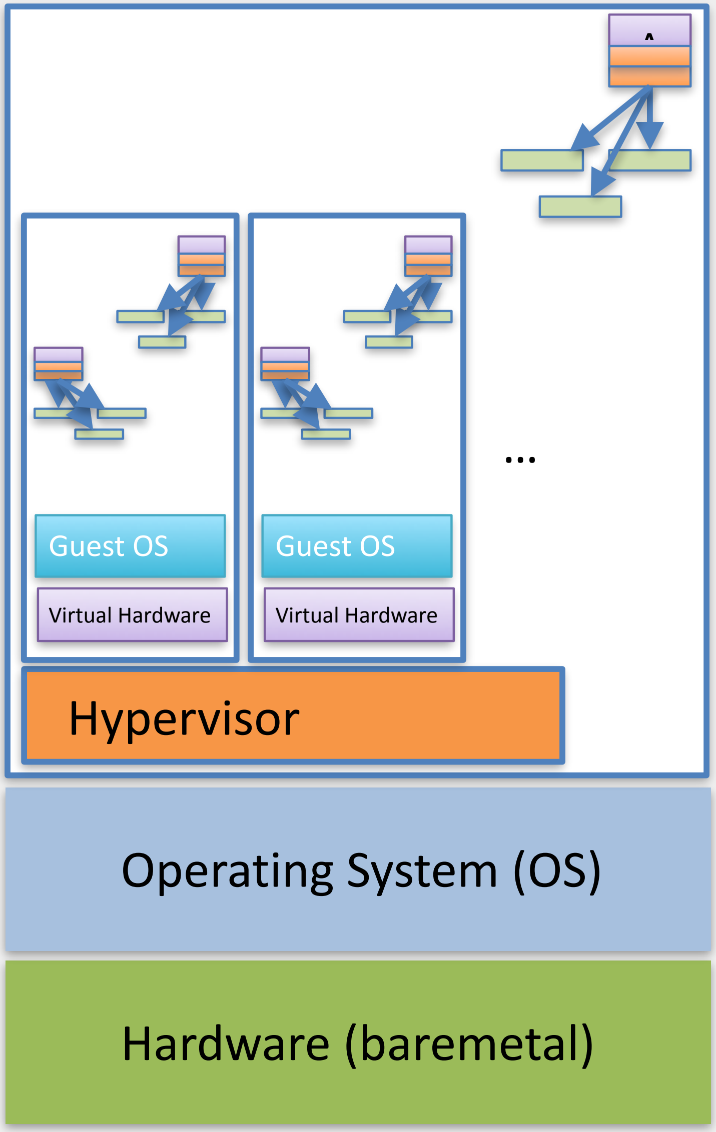
- **Virtualization** - creating a virtual (rather than an actual) version of something
- In some sense *operating systems* virtualize the resources of the hardware:
  - Making each app think it has the run of the CPU
  - Virtual memory
  - Virtualized I/O devices
- With *hypervisors* we virtualize the entire machine to create *virtual machines*

(Re)-enter Virtualization

- Virtualization has a long history…
  - From IBM mainframes in the 1960s to present day
- Virtualizing ubiquitous x86 was long impossible:
  - Byzantine architecture
  - Ill-behaved instructions
- Late 1990s: 32-bit full virtualization of x86 by VMware
- Many imitators followed (Xen, KVM etc.)
- AMD (then) Intel added v12n hardware support
- Key artifact: a *hypervisor* or *virtual machine monitor*
- Hypervisors come in two flavors

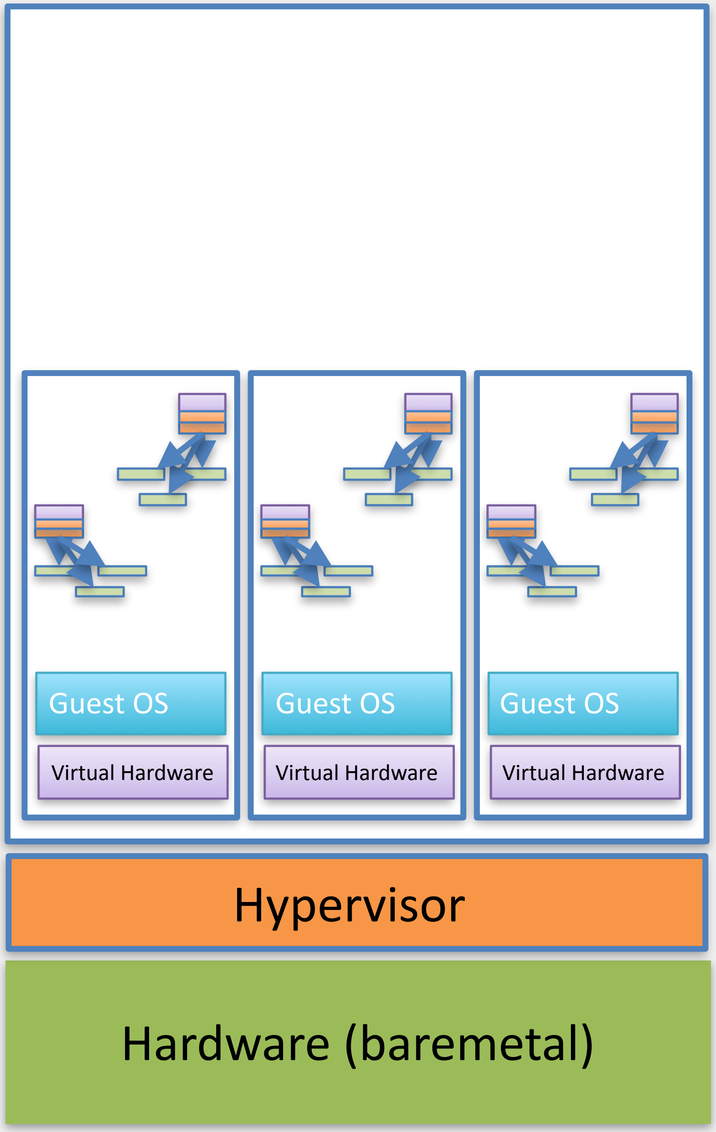
Virtualization: One Type

- Here a hypervisor runs atop the OS…
- It creates virtual machines inside of which:
    - You can run an OS as a guest
    - That OS can run apps
    - All separated from one another
- When a guest OS tries to access hardware, the hypervisor traps the access and delegates through to the host OS and ultimately bare hardware
- Virtualization makes one *physical* machine into many *virtual machines*



Virtualization: Another Type

- Key difference: No “host OS” this time.
- Bare metal hardware boots the hypervisor
- The hypervisor, as before, creates and manages VMs
- Actions against hardware in guest OS’s are trapped by the hypervisor and translated through to the bare metal hardware
- With director control of hardware, hypervisor has advantages
- Still: one *physical machine* made into many *virtual machines*



Virtualization is Great, but…

- Bare metal physical server often not a granular enough unit
- VMs gave us smaller distribution units
- But VMs still:
  - Have a whole guest OS
  - Take a while to boot, depending on OS
  - Within a single VM all the old woes of the filthy OS deployment remain
- How to complement what we have with VMs with greater granularity?

Containers

- Containers: an OS feature where OS kernel allows:
  - Multiple isolated user-space instances
  - Programs running in a container can only see the container’s contents:
    - Filesystems
    - Devices assigned to container
- Like virtualization has a long history
- Today’s most visible examples is Docker

Containers: Docker

- Docker on Linux:
  - Packages software in *containers*
- Containers:
  - Bundle own programs, tools, libraries and configuration files (avoids DLL Hell, config stomping)
  - Run on same OS kernel (lighter weight than VMs)
  - Segregation relies on services supported by the OS (e.g. cgroups, overlay file systems, kernel namespaces)

Containers: Advantages

- Docker is now a standardized format:
  - Make a container
  - Bundle your code, its dependencies, configuration, etc.
  - Distribute it via public or private repositories
- Efficiency:
  - Can be pruned to bare essentials
  - Can start quickly (just running programs, not a full OS boot cycle)
  - Containers can be kept small and simple and evolved
- We'll use Docker later in the course to quickly deploy things

Summary: Cloud Computing and Enabling Technologies

- ***The Cloud:*** running stuff somewhere else, maybe rented, maybe on demand
  - e.g. AWS, GCP, Azure
- Enabled by:
  - Distributed Systems / Networking
  - Virtualization
    - e.g. VMware, Zen, KVM, QEMU, Parallels
  - Containers
    - e.g. Docker

Cloud Computing: A Few Final Concerns

- **Virtual vs Physical servers**
  - **Storage**
    - Data Locality. Where is the data? How expensive is it to access or move?
  - **Regions / Availability Zones**
    - How to keep data and systems safe in event of failure?
  - **Public / Private DNS**
    - How to make data sufficiently visible/accessible?

Cloud Computing

- **Security**
  – Firewalls: to block unwanted access
  – Keep locked down as much as you can
  – Open ports as needed

– All the enabling technologies can help with this if used prudently

Week 1 Assignment

Week 1: Reading and Getting Oriented

- **Reading, *Kleppmann* book:**
  - Chapter 1: *Reliable, Scalable and Maintainable Applications* (22 pages)
  - Chapter 2: *Data Models and Query Languages* (42 pages)
- **Motivation:** read briskly, keep
- **Assignment:**
  - A few qualitative questions to keep in mind
  - Request for feedback on Lecture 1

Next Week

- Introduction to Scalable Computing
- Distributed Computing:
  - Challenges Presented
  - Opportunities Created
  - Compromises Forced
- The Hadoop Ecosystem
  - HDFS
  - MapReduce
  - Hive
- Meet the Course Sandbox Environment