CSE 487/587
Data Intensive Computing

Lecture 1: Introduction

Vipin Chaudhary

vipin@buffalo.edu

716.645.4740
305 Davis Hall

Overview of Today's Lecture

- Introduction
- Administrative Matters
- More Introduction

DIC: What is it about?

- BigData
  - Structured and Unstructured Data
  - Various technologies like Mapreduce, R, and more
  - Have some hands-on work in class (get your laptops)
- Introductory course
  - Programming experience with these technologies
  - Some hands-on work in class (get your laptops)
- But Getting you ready for next level
  - Encourage self study
  - Reading research papers

Processes 20 PB a day (2008)
Crawls 20B web pages a day (2012)
Search index is 100+ PB (5/2014)
Bigtable serves 2+ EB, 600M QPS (5/2014)

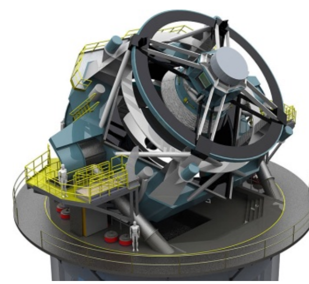
150 PB on 50k+ servers running 15k apps (6/2011)

Wayback Machine: 400B web pages archived, 10+ PB (2/2014)

Hadoop: 10K nodes, 150K cores, 150 PB (4/2014)

LHC: ~15 PB a year

300 PB data in Hive + 600 TB/day (4/2014)



LSST: 6-10 PB a year (~2015)

S3: 2T objects, 1.1M request/second (4/2013)

SKA: 0.3 – 1.5 EB per year (~2020)

Data is growing fast

Science/Engineering

- Data bases from astronomy, genomics, natural languages, seismic modeling, fluid dynamics, …

Humanities

- Scanned books, historic documents, …

Commerce

- Corporate sales, stock market transactions, census, airline traffic, …

Entertainment

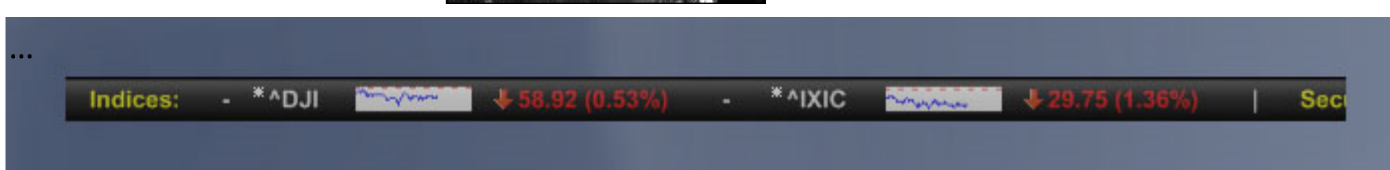
- Internet images, Hollywood movies, MP3 files, …

Medicine

- Drug discovery, Diagnosis, MRI & CT scans, patient records, …

Security

- Unfolding terror groups, …



Science

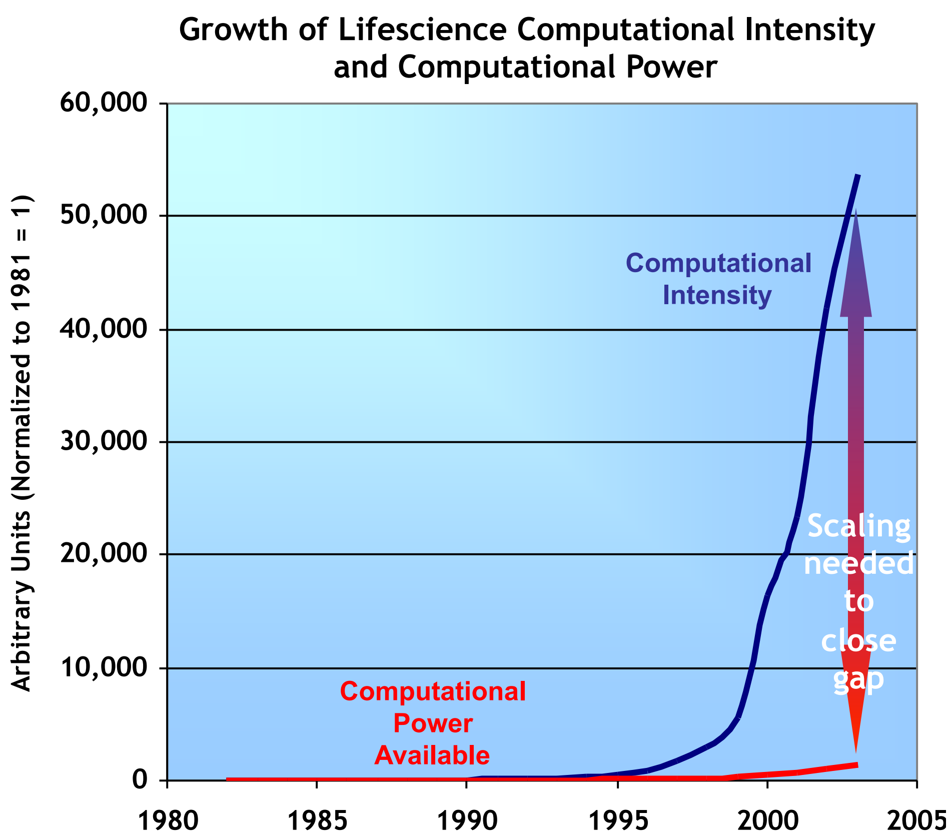
Emergence of the 4^{th} Paradigm

Data-intensive e-Science

Maximilien Brice, © CERN

Need for Change: Lifescience Problems are Computationally Intensive

- Computational Intensity of Life-science processes is increasing rapidly with the growth in:
  - Volume of data
  - Level of automation
  - Complexity of analysis
  - Need for improved decision making
- Users want :
  - Better performance (acceleration)
  - Better price performance (cost)
  - Ease of use
- Linear scaling of performance requires superlinear scaling of infrastructure cost



Administrative Matters

CSE487/587: Data Intensive Computing

Instructor: Vipin Chaudhary
Office: Davis Hall 305, 645-4740
Email: vipin@buffalo.edu
Office Hours : Mon: 1:30-2:30pm

Teaching Assistant: Ruhan Sa, Yi Wei

Guest Lectures:

Text: None

Midterm: #1: March 11, 2015; #2: April 13, 2015

Final: May 11, 2015; 7:15-10:15pm, Knox 110

Reference: Several papers will be posted and discussed in class. Many online tutorials will be used. Some assignments will not be graded (you will be notified) but will help you with your midterms and Programming Assignments.

Programming Assignments Due Dates: Feb 23, March 11, March 30, April 15, May 4.

TA Officer Hours: Tu 2-4pm and Fr 3-4pm in Davis 302;
Fr 4-5pm (Bell 138).

Course Philosophy

- Lecture style
  - Perhaps one break depending on contents of class
  - Have some hands-on work in class (get your laptops)
- 400/500 level course
  - Encourage class participation
  - Encourage self study
  - Reading research papers
- Reduce the pressure of taking exams
  - Smaller fraction of grades contributed by exams
  - Goal: test your knowledge

Course Content

- Trends in Computing - BigData
- Parallel and Distributed File Systems – GFS, HDFS, Ceph
- Accessing data from Social networks
- Resource Management
- Warehouse Scale and Cloud Computing
- Mapreduce
- Pig
- Hive
- R Programming, Statistical Methods
- Hbase
- Accumulo
- Cloud storage – Scale Up, Erasure Coding, Object Store, …
- New Technologies for DIC

Course Web Page

- Entire course is available on the web
  ublearns.buffalo.edu
  - powerpoint files – Lecture notes
  - Papers
  - Software documentation, etc.
  - Most of my communication is via blackboard – make sure your email there works.

Grading Policy

- Composition
  - Programming Assignments are 60%
    - Usually have bonus points for harder extensions of assignments
  - Midterms are 10% each
  - Final is 20%

- Grade assignment
  - 100to 90 is A+; 89 to 85 is A; 84 to 80 is A-
  - 79 to 75 is B+; 74 to 70 is B; 69 to 65 is B-
  - 64 to 60 is C+; 59 to 55 is C; 54 to 50 is C-

Homework Assignments

- Most homework assignments will be posted on the course web page
  - You will be informed of their posting by email, news, and/or in class
- Assignments may not be weighted equally; Grading Criteria will be posted
- Approximate weights of each assignment will be specified when the assignment is posted
- No late homework will be accepted with full credit
  - except under extreme non-academic circumstances discussed with the instructor at least one week before the due date
  - 25% per day penalty for late submission
- Penalty for cheating (fail the course with report to University)
  - Software to be used to detect cheating in programming

Course Problems

- Lack of pre-requisites
  - Familiarize yourself with Java and SQL (useful)
  - knowledge of Java (required)
  - Starting on programs very late
    - There will be long queues in CCR close to deadlines
- Late homework penalty – 25% reduction per day
- What is cheating?
  - Studying together in groups is encouraged
  - Work must be your own
  - Common examples of cheating: running out of time on a assignment and then copy, person asks to borrow solution “just to take a look”, copying an exam question, copying code from internet or others, using help of previous students, ...

Try out your Java/SQL

- For Java
  - http://www.indiabix.com/online-test/java-programming-test/random
- For SQL
  - http://www.javatpoint.com/sql-quiz
  - http://www.w3schools.com/quiztest/quiztest.asp?qtest=SQL
- Revise Java and SQL if your score is low

Background & Research Areas

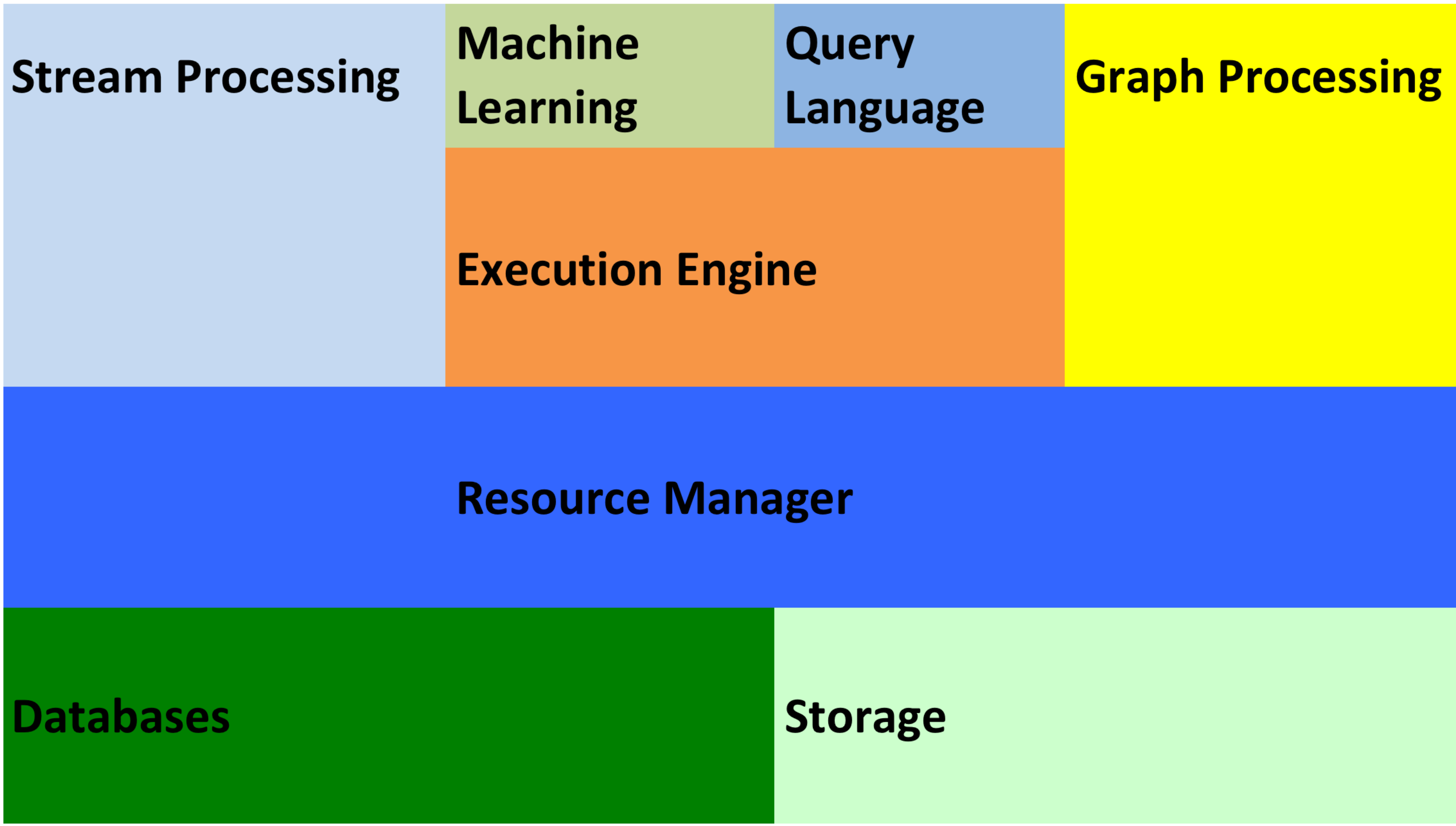
- Industry experience
  - Tata CRL, CEO [sold to TCS – 2012] – HPC solutions
  - Scalable Informatics, President
  - Corio Inc., Chief Architect [IPO 2000; sold to IBM – 2005]
    - Application Service Provider, Hosting infrastructure
  - Cradle Technologies, Inc., Senior Director of Advanced Development [private since 1999]
    - MDSP multicore DSP processor
  - Diagnaid Inc [Founder, 2009]
  - Advisory board of several startups; consultant to VCs.

- Parallel and Distributed Systems (HPC, Grid,…)
- Big Data Computing
- Computer Aided Diagnosis and Interventions

Fun Problem of the Week #1

- Compute the total cost of building a data center that houses 10 ExaBytes of HDD
  - HDDs, networking, racks, building (real-estate), power, cooling, …
  - You can get the cost of various parts online
  - Assume the datacenter is built in Buffalo
  - State all your assumptions
  - Due 1/28 in class

How are Topics Related



Hadoop Resources

- Hadoop on your local machine
- Hadoop in a virtual machine on your local machine
- Hadoop in CCR (where all your submission and grading will happen)

Cloud Computing

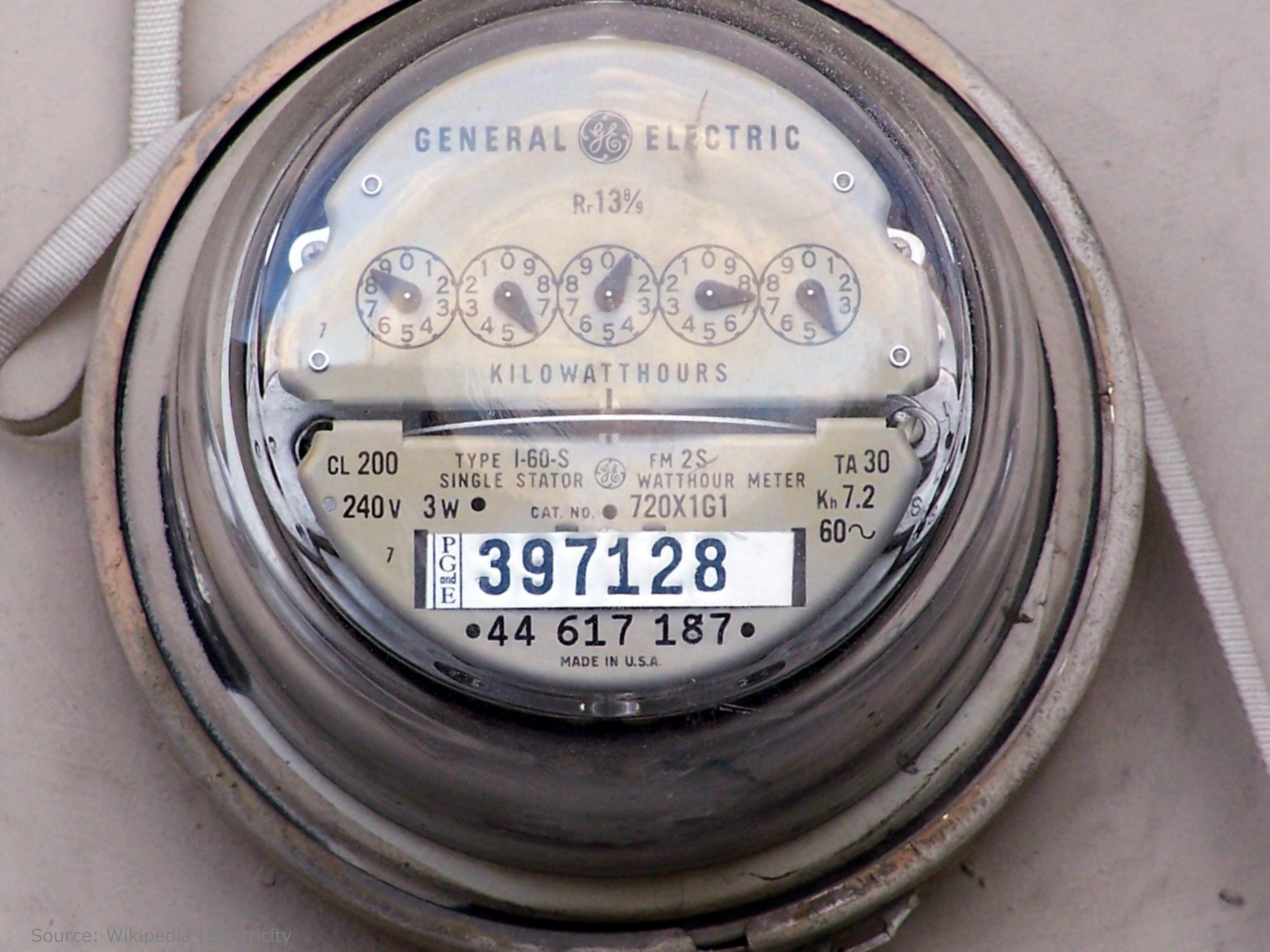
Source: Wikipedia (Clouds)

The best thing since sliced bread?

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

Rebranding of web 2.0

- Rich, interactive web applications
  - Clouds refer to the servers that run them
  - AJAX as the de facto standard
  - Examples: Facebook, YouTube, Gmail, …
- “The network is the computer”: take two
  - User data is stored “in the clouds”
  - Rise of the netbook, smartphones, etc.
  - Browser *is* the OS

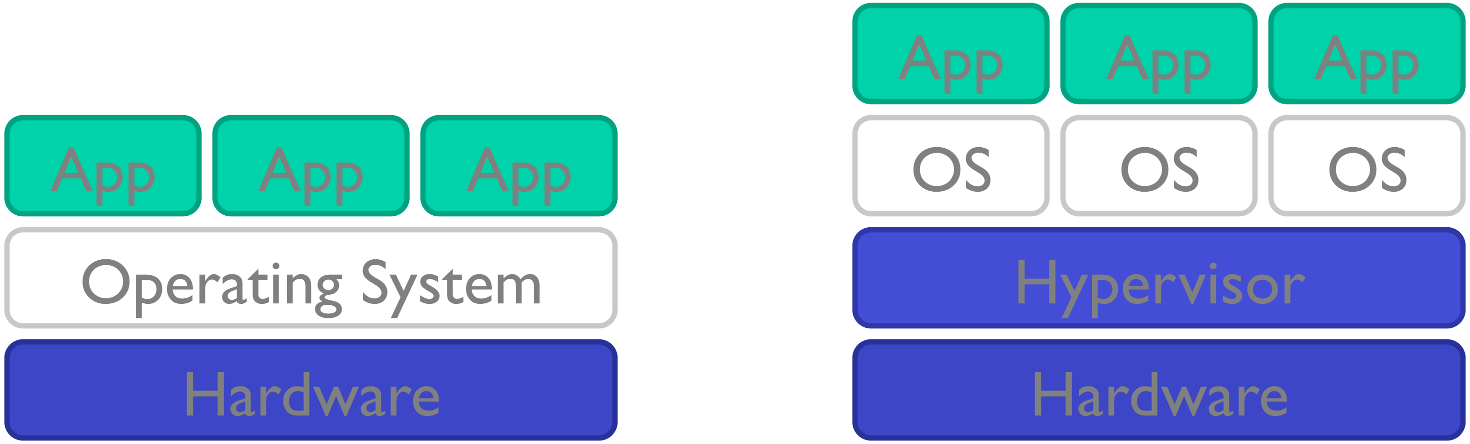


Source: Wikipedia (Electricity

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

Enabling Technology: Virtualization



Everything as a Service

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon's EC2, Rackspace, Lucerahq
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades, …
  - Example: Google App Engine, Amazon
- Software as a Service (SaaS)
  - Runs out of the box
  - Example: Gmail, Salesforce

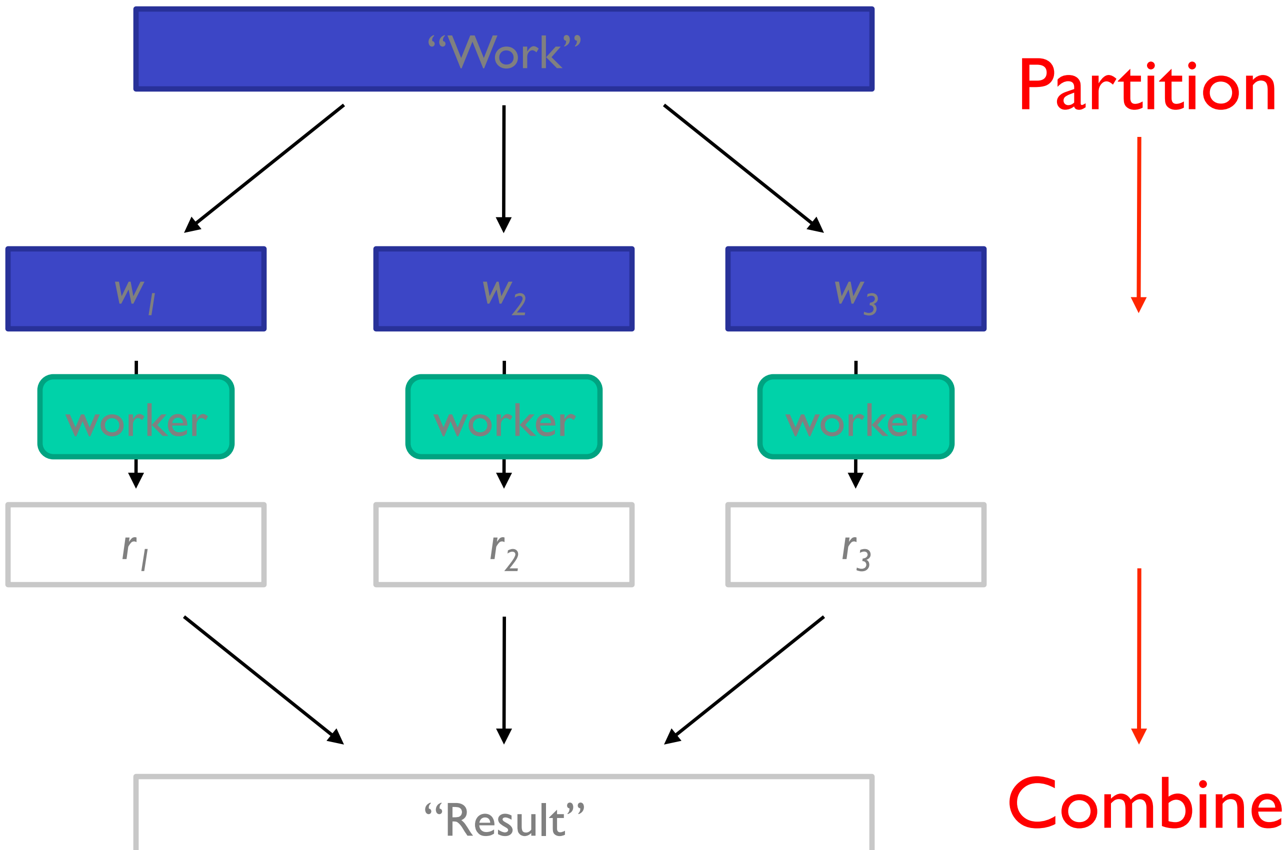
Who cares?

- Ready-made big data problems
  - Social media, user-generated content = big data
  - Examples: Facebook friend suggestions, Google ad placement
  - Business intelligence: gather everything in a data warehouse and run analytics to generate insight
- Utility computing provides:
  - Ability to provision Hadoop clusters on-demand in the cloud
  - Lower barrier to entry for tackling big data problems
  - Commoditization and democratization of big data capabilities

Tackling Big Data

Source: Google

Divide and Conquer



Parallelization Challenges

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

What's the common theme of all of these problems?

Common Theme?

- Parallelization problems arise from:
  - Communication between workers (e.g., to exchange state)
  - Access to shared resources (e.g., data)
- Thus, we need a synchronization mechanism

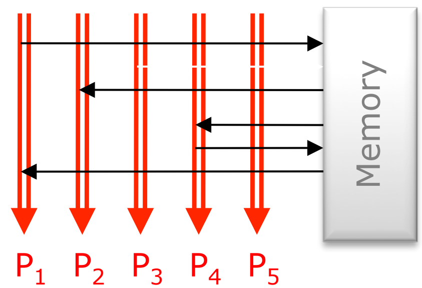
Source: HindustanTimes/Sonu Mehta

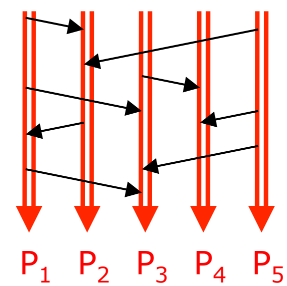
Managing Multiple Workers

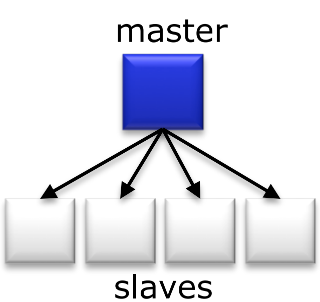
- Difficult because
  - We don’t know the order in which workers run
  - We don’t know when workers interrupt each other
  - We don’t know when workers need to communicate partial results
  - We don’t know the order in which workers access shared data
- Thus, we need:
  - Semaphores (lock, unlock)
  - Conditional variables (wait, notify, broadcast)
  - Barriers
- Still, lots of problems:
  - Deadlock, livelock, race conditions...
  - Dining philosophers, sleeping barbers, …
- Moral of the story: be careful!

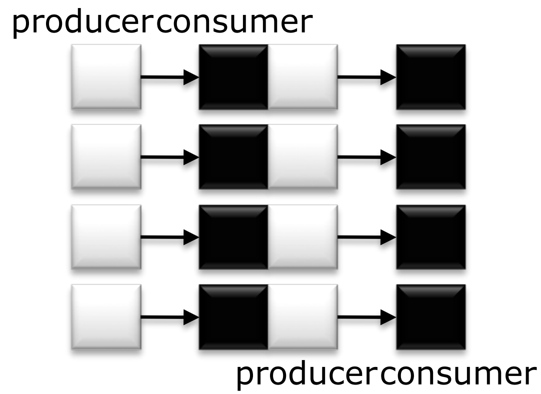
Current Tools

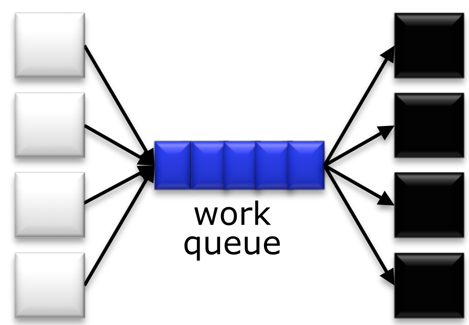
- Programming models
  - Shared memory (pthreads)
  - Message passing (MPI)
- Design Patterns
  - Master-slaves
  - Producer-consumer flows
  - Shared work queues











Practically?

- Concurrency is difficult to reason about
- Concurrency is even more difficult to reason about
    - At the scale of datacenters and across datacenters
    - In the presence of failures
    - In terms of multiple interacting services
- Not to mention debugging…
- The reality:
    - Lots of one-off solutions, custom code
    - Write you own dedicated library, then program with it
    - Burden on the programmer to explicitly manage everything


The datacenter *is* the computer!
Source: Google

What’s the point?

- It’s all about the right level of abstraction
  - Moving beyond the von Neumann architecture
  - We need better programming models
- Hide system-level details from the developers
  - No more race conditions, lock contention, etc.
- Separating the *what* from *how*
  - Developer specifies the computation that needs to be performed
  - Execution framework (“runtime”) handles actual execution

“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

Thank you!