What is big data?

There is no hard and fast rule about exactly what size a database needs to be in order for the data inside of it to be considered "big." Instead, what typically defines big data is the need for new techniques and tools in order to be able to process it. In order to use big data, you need programs which span multiple physical and/or virtual machines working together in concert in order to process all of the data in a reasonable span of time.

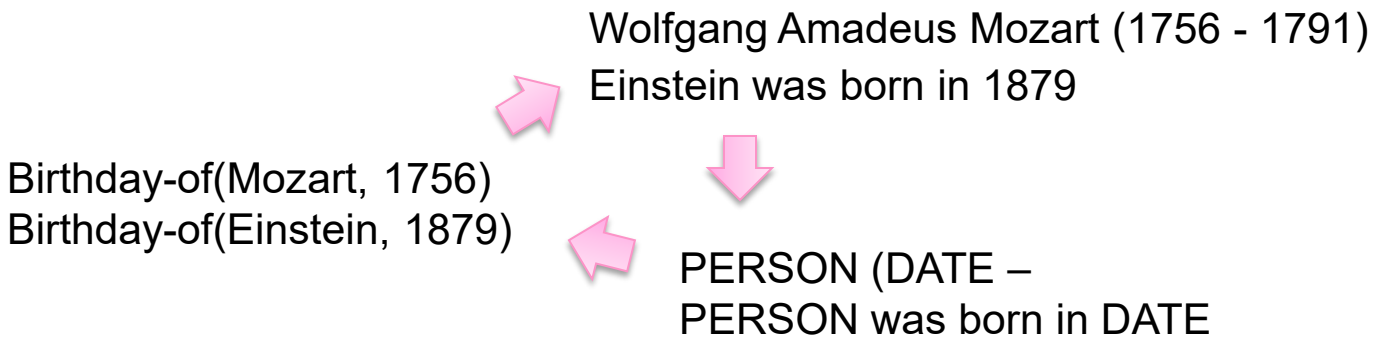
Why large data?

What to do with more data?

- Answering factoid questions
  - Pattern matching on the Web
  - Works amazingly well

Who shot Abraham Lincoln? → X shot Abraham Lincoln

- Learning relations
  - Start with seed instances
  - Search for patterns on the Web
  - Using patterns to find more instances



(Brill et al., TREC 2001; Lin, ACM TOIS 2007)
(Agichtein and Gravano, DL 2000; Ravichandran and Hovy, ACL 2002; …)

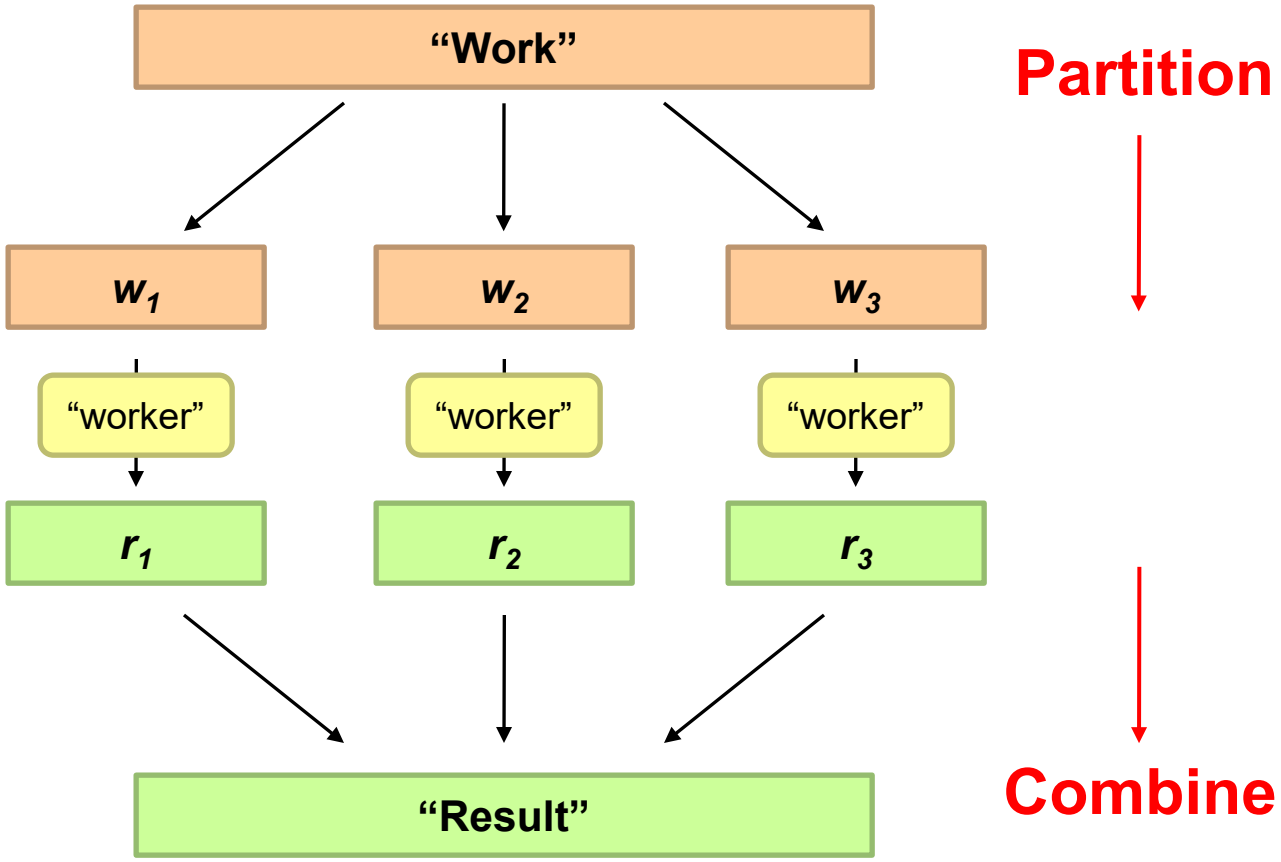
How do we scale ?

Divide and Conquer

In Hardware - Divide and Conquer

Source: Wikipedia (IBM Roadrunner)

In Software - Divide and Conquer



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

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 is the common theme of all of these problems?

Managing Multiple Workers

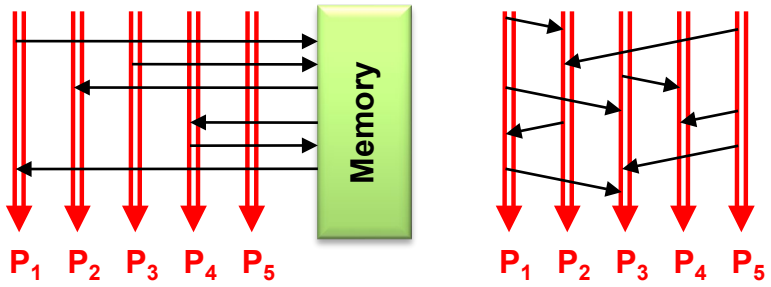
- Difficult because
  - We don't know the order in which workers run
  - We don't know when workers interrupt each other
  - We don't know the order in which workers access shared data
- Thus, we need:
  - Semaphores (lock, unlock)
  - Conditional variables (wait, notify, broadcast)
  - Barriers
- Still,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

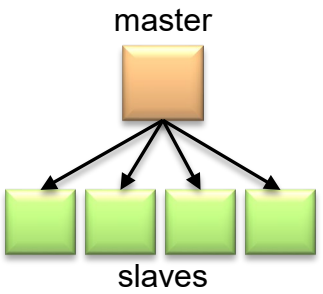
This course...

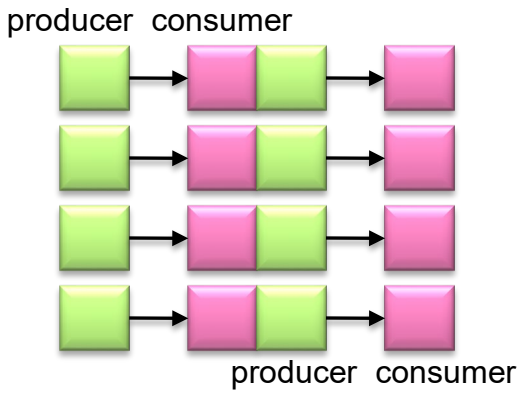
Software Scaling
and
Distributed Data Computing

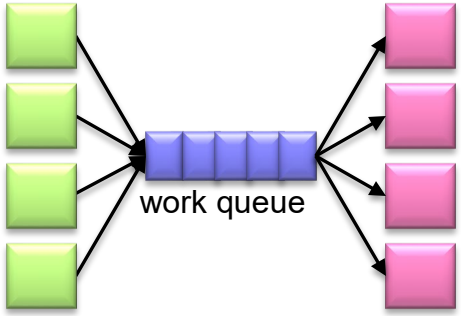
Current Tools

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









Where the rubber meets the road

- Concurrency is difficult to reason about
- Concurrency is even more difficult to reason about
  - At the scale of datacenters (even 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

“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

How is big data analyzed?

One of the best-known methods for turning raw data into useful information is by what is known as MapReduce. MapReduce is a method for taking a large data set and performing computations on it across multiple computers, in parallel. It serves as a model for how to program, and is often used to refer to the actual implementation of this model.