Big Data and Security

Jeff Borowitz, PhD

Lecturer

Sam Nunn School of International Affairs

Computer Architecture for Big Data

Big Data Performance

- The other side of the equation is what needs to be computed
- Many computations are of fixed size
  - Calculate something about my genome
  - Calculate the results of the Census
- Some things may grow faster than computational resources
  - Aspects of social network data
  - Log data
- As disk space in particular gets cheaper, new things that were previously not storable will be stored on disk

Georgia Tech

So How Can We Do “Big Data” with These Limits?

- Discussion Questions
  - How would you build a system which can handle more data than fits on a single hard drive?
  - What if you need a system which will serve a website to more customers at once?
  - How would you sort a list which doesn’t fit on one computer?

So How Can We Do “Big Data” with These Limits?

- One Strategy: Better Hardware
  - This is exponentially more expensive, but can be worth it
  - Built by IBM or Cray
  - Up to Petabytes of memory
  - 10,000s of CPU/GPU cores
  - Costs over $100M

- Another Strategy: Distributed Architectures
  - Use more than one cheap computer together!

Georgia Tech

Load Balancing

The best way to distribute work depends a lot on what the work is

What if you just offload computationally intensive functions, which are independent?

- E.g. you have a whole bunch of different problems which look like: choose x to maximize f (x)?

Load Balancing

- The best way to distribute work depends a lot on what the work is
- What if you just offload computationally intensive functions, which are independent?
  - E.g. you have a whole bunch of different problems which look like: choose x to maximize f (x)?
- You could have a network of 100 computers, and give the first problem to the first computer, second to the second, etc.
- This process is called load balancing (round robin)

Databases for Big Data: Master/Worker Architecture

Master/Worker Architecture

- One computer has a list of which data is on which computers (master)
- When you ask for all the data, the leader asks the other computers (worker) to each provide their pieces
- Issues
  - What if the master gets overloaded?
  - What if the master fails?

Databases for Big Data: Eventual Consistency

- What happens if the leader node gets overwhelmed?
- We could have multiple leaders, each of which can task different worker nodes
- Eventual Consistency
  - But how do leaders know what's going on with other workers?
  - They send messages to each other about what they're up to
  - But these messages don't arrive instantly
  - So the first leader can get asked about data which the second knows has changed, but the first hasn't heard about it yet
- This is OK in many contexts, but not in e.g. banking

Georgia Tech

Serving Multiple Customers

- Each computer receives requests, but can only process so many per unit time
- Multiple machines can handle more requests
- But what if you're a website which needs to look up information about a customer?
  - If many servers deal with the same database, that database will be overloaded
  - If you're just reading data, you can have many copies of the database
  - If you sometimes change data, you need to make sure there aren't multiple changes at the same time

Sorting a Big List

- A loose algorithm for sorting a list on N computers
  - On computer 1, sort the data and take each 1/N partition and assign it to a computer
  - Tell each computer to send its values less than 1/N to the first computer, those between 1/N and 2/N to the second, etc.
  - Now we know each computer has data which is either all greater or all less than the data on each other computer
  - So we just sort the data on each computer, and we're done!

Doing More Stuff Like This Sort!

- This was great - we have a way of sorting large amounts of data!
- What probably slows this down?
  - Sending data from one computer to another
  - (Reading data from disk was always going to happen)

Summary

- Supercomputers vs architecture
- We introduced some big data architectural elements
    - Load balancing
    - Master/Worker
    - Eventual consistency
    - Custom algorithms