Context
Since we are at the end
Announcements

• This is the last class of the semester -- no more class meetings.

• Please respond to the Doodle poll to set up a 15 minute slot to meet.
  
  • First slot early tomorrow, last slot is at 8:45pm next Wednesday.

  • Must prepare a few slides. Also come prepared to demo if you are done.

• Meeting is required to get a grade for the final project.
Announcements

• Final project report is due next Wednesday at 23:59. No extensions.

• The final project is 40% of the grade, so don't miss this deadline.

• Lab 2 grades will be out by Monday (probably sooner).
From Lecture 1
Three Main Reasons

- Fault tolerance
  - Survive some forms of failures or bug.

- Scalability
  - Use more resources than a single computer can provide.

- Geographic Reach
  - Work even when information is spread across large distances.
But Where?
Look at Three Places

Datacenters  Sensors/Internet of Things  The Internet
Look at Three Places

Datacenters

Sensors/Internet of Things

The Internet

Not in Chronological Order
Same Goals Different Requirements

- Datacenter: Single administrative domain, lots of compute capacity.
  - Control and knowledge of what is running where, etc.
- IoT: Single administrative domain, limited resources.
  - Higher failure rates (potentially). Need to limit resources.
- Internet: Different administrative domains.
  - Figure out incentives for systems to work together, align with economics, etc.
Datacenters
What Datacenters

• The Internet and the web meant lots of clients might be connecting to one service

• Question: How to scale compute to serve all of these clients?

• In the early-1990s a few answers: mainframes, custom computers, etc.

• One research idea: a **network of workstations** that can compute together.
What Datacenters

• What workstation? Over time differences between smaller computers blurred.

• Now: just a building with a lot of servers in racks connected by a fast network.

• How many servers? Do not know for sure, but 50,000 to 100,000 are common.
Challenges

- Where to build?
- How to build?
- How to maintain?
- How to manage infrastructure?
- How to effectively utilize capacity?
Challenges

- Where to build?
- How to build?
- How to maintain?
- How to manage infrastructure?
- How to effectively utilize capacity?
Programming Models for Datacenter
How to Scale Programs for Datacenters

• The answer of course depends on application.

• Going to look at a few examples of how people have used datacenters
  • To serve web requests.
  • To gather and run computations on large amounts of data.
  • Combining the two.
Serving Web Requests
Defining the Problem

Clients  Web Server  Database
Defining the Problem

Categorization:

- Clients
- Web Server
- Database

Diagram:

1. Request
2. Cloud
3. Web Server
4. Query
5. Response
6. Database
7. Result
Defining the Problem

How to handle an increase in the number of clients?
Solving the Problem

Can replicate web servers and put them behind a load balancer

Any problems with this strategy?
Solving the Problem

Assuming most queries are reads can cache data.

Any problems with this strategy?
Solving the Problem

Can shard data (need to be aware of transactions).

Any problems with this strategy?
Sharding is Hard
Solving the Problem

What about fault tolerance?
Gathering and Running Computation
PageRank

- Need to discover and rank pages on the web.
- Was done manually for a while.
- Metric: Pages which are linked to a lot are authoritative.
- Task: Find number of links to each page.
- Challenge: 30 trillion (and growing) pages today.
Web Crawlers

Output
a.com/i -> a.com/j
a.com/i -> a.com/k
a.com/i -> d.com/a
Scaling Web Crawling

- Why independent outputs?
- Is starting from independent pages sufficient?
  - For correctness?
  - For scalability?
- How to address any issues?
Computing PageRank

Map | Shuffle | Reduce
---|---|---
Output 1
a->b a->b y->a
c->b d->c x->j

Output 2
...

Output 3
...

Output 4
...

Count # of unique links
Count # of unique links
Count # of unique links
Count # of unique links
Map Reduce as a Computational Paradigm

- Generalized into a programming framework used to implement
  - Running aggregation queries (e.g., on large amounts of data).
  - Machine learning jobs of some kind.
  - Various other things...
Map Reduce Challenges

- Fault tolerance: need to replicate data and remember locations.
- Scheduling: minimize time and resources used.
- Sharing the cluster across jobs.
- Minimizing compute and network transfer time.
Sensors or IoT
Many Variants, Main Differences

• Usually consider the case of sensors producing data.
  
• Want to compute on the aggregate data from sensors.
  
• For example, to provide early warning for volcanos, storms, earthquakes.
  
• For example, to provide security against intruders.
  
• ...
Challenges

• Sensors have limited compute and power resources.
  • Might not always be on, might be able to do a limited number of tasks.
• Communicate over wireless networks which might not always work reliably.
  • Interference or change in distance might disconnect individual sensors.
Thoughts on solutions?
The Internet
Many Problems, Focusing on One

• There are many problems here.
  • A wide variety of requirements and tradeoffs.

• Focusing on one specific problem here.
  • Why? Seems like a problem that generalizes.

• Also a problem I like.
What is the Internet

A set of networks, each of which is owned by a different entity.
What is the Internet

Must cooperate to get packets to a particular destination
How to Get Cooperation

• What is great about this model:
  • Grows organically to include new areas, don't need a central authority.

• Concerns:
  • Networks cost money, need to ensure economic incentives for transit.
  • Policy/trust about what data is sent where.
  • ...

How it Works Today

Have path to B: M->B
Have path to S: M->B->S
...

Have path to M: B->M
Have path to S: B->S
How it Works Today

Do not consider paths from S.

Prefer paths from D over paths from B.

Prefer paths from M over paths from S. Do not consider paths from F.

...
How it Works Today

Combine policies and announcements to compute path.

Do not consider paths from S.

Prefer paths from D over paths from B.

Prefer paths from M over paths from S.
Do not consider paths from F.
Problems

• How to ensure paths are stable?

• How to ensure quick response after failure?
Benefits

• Range of policies that can be implemented?
How to do better?
Final Thoughts (of the semester)
Final Thoughts

• Given the end of Denard scaling, and what is popular today.
  • If you write programs, very likely to be targeting distributed systems.
  • Probably hidden behind a few layers of abstraction.
• Given this remember just a few rules as you build systems.
  • Avoid coordination when possible, coordination is often slow.
  • But do not shun coordination in exchange for increased complexity.
The End

Please stay to fill out evaluation forms.