

CSE 486/586 Distributed Systems Mid-Semester Overview

Steve Ko
Computer Sciences and Engineering
University at Buffalo

CSE 486/586, Spring 2013

We're at a Mid-Point: What We've Discussed So Far

- Main communication infrastructure: the Internet
- Communication between two processes
 - Socket API
 - RPC
- Communication between multiple processes
 - Multicast algorithms
- Concept of time in distributed systems
- Organization of distributed systems
 - Server-client
 - Peer-to-peer, DHTs
- Impossibility of consensus
- Distributed algorithms
 - Failure detection, global snapshots, mutual exclusion, leader election

CSE 486/586, Spring 2013

2

The Other Half of the Semester

- Distributed storage
- Consensus algorithm: Paxos
- BFT (Byzantine Fault Tolerance)
- Security

CSE 486/586, Spring 2013

3

Data Centers

CSE 486/586, Spring 2013

4

Data Centers

- Hundreds of Locations in the US



CSE 486/586, Spring 2013

5

Inside

- Servers in racks
 - Usually ~40 blades per rack
 - ToR (Top-of-Rack) switch
- Incredible amounts of engineering efforts
 - Power, cooling, etc.

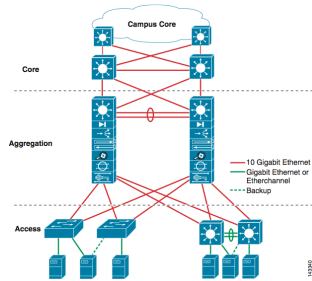


CSE 486/586, Spring 2013

6

Inside

- Network

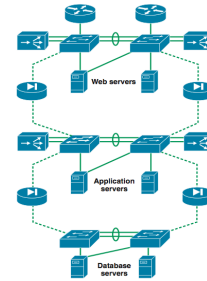


CSE 486/586, Spring 2013

7

Inside

- 3-tier for Web services



CSE 486/586, Spring 2013

8

Web Services

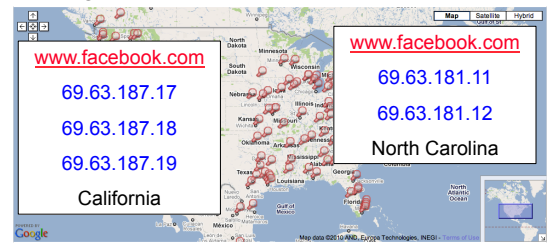
- Amazon, Facebook, Google, Twitter, etc.
- World-wide distribution of data centers
 - Load balance, fault tolerance, performance, etc.
- Replicated service & data
 - Each data center might be a complete stand-alone web service. (It depends though.)
- At the bare minimum, you're doing read/write.
- What needs to be done when you issue a read req?
 - Server selection
- What needs to be done when you issue a write req?
 - Server selection
 - Replicated data store management

CSE 486/586, Spring 2013

9

Server Selection Primer

- Can happen at multiple places
- Server resolution process: DNS -> External IP -> Internal IP
- DNS

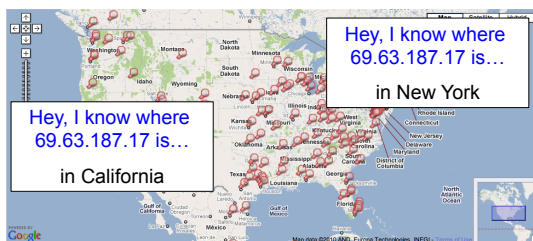


CSE 486/586, Spring 2013

10

IP Anycast

- BGP (Border Gateway Protocol) level

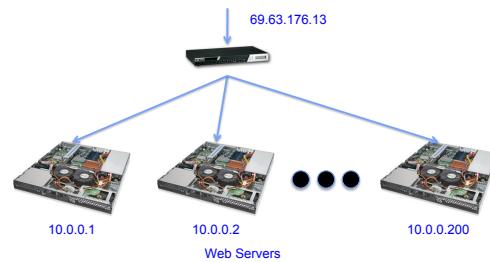


CSE 486/586, Spring 2013

11

Inside

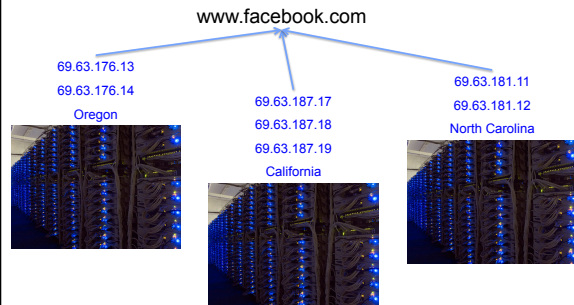
- Load balancers



CSE 486/586, Spring 2013

12

Example: Facebook



CSE 486/586, Spring 2013

13

Example: Facebook Geo-Replication

- (At least in 2008) Lazy primary-backup replication
- All writes go to California, then get propagated.
- Reads can go anywhere (probably to the closest one).
- Ensure (probably sequential) consistency through timestamps
 - Set a browser cookie when there's a write
 - If within the last 20 seconds, reads go to California.
- http://www.facebook.com/note.php?note_id=23844338919

CSE 486/586, Spring 2013

14

Core Issue: Handling Replication

- Replication is (almost) inevitable.
 - Failures, performance, load balance, etc.
- We will spend most of our time looking at this in the second half of the semester.
- Data replication
 - Read/write can go to any server.
 - How to provide a consistent view? (i.e., what consistency guarantee?) linearizability, sequential consistency, causal consistency, etc.
 - What happens when things go wrong?
- State machine replication
 - How to agree on the instructions to execute?
 - How to handle failures and malicious servers?

CSE 486/586, Spring 2013

15

CSE 486/586 Administrivia

- Midterm: 3/12 (Wednesday) in class
 - Everything up to leader election
 - 1-page cheat sheet is allowed.
- Best way to prepare
 - Read the textbook & go over the slides.
 - Go over the problems in the textbook.

CSE 486/586, Spring 2013

16

Banking Example (Once Again)

- Banking transaction for a customer (e.g., at ATM or browser)
 - Transfer \$100 from saving to checking account
 - Transfer \$200 from money-market to checking account
 - Withdraw \$400 from checking account
- Transaction
 1. `savings.deduct(100)`
 2. `checking.add(100)`
 3. `mnymkt.deduct(200)`
 4. `checking.add(200)`
 5. `checking.deduct(400)`
 6. `dispense(400)`

CSE 486/586, Spring 2013

17

Wait...We've Seen This Before...

- What are some things that can go wrong?
 - Multiple clients
 - Multiple servers
- How do you solve this?
 - Group everything as if it's a single step
- Where have we seen this?
 - Mutual exclusion lecture
- So, we're done?
 - No, we're not satisfied.

CSE 486/586, Spring 2013

18

Why Not Satisfied?

• Process 1

```
lock(mutex);
savings.deduct(100);
checking.add(100);
mnymkt.deduct(200);
checking.add(200);
checking.deduct(400);
dispense(400);
unlock(mutex);
```

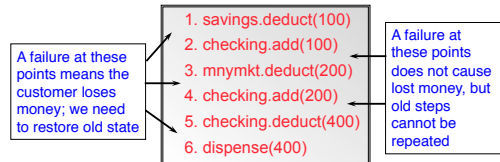
• Process 2

```
lock(mutex);
savings.deduct(100);
checking.add(100);
mnymkt.deduct(200);
checking.add(200);
checking.deduct(400);
dispense(400);
unlock(mutex);
```

CSE 486/586, Spring 2013

19

Why Not Satisfied?



CSE 486/586, Spring 2013

20

Why Not Satisfied?

- What we discussed in mutual exclusion is one big lock.
 - Everyone else has to wait.
 - It does not necessarily deal with failures.
- Performance
 - Observation: we can interleave some operations from different processes.
- Failure
 - If a process crashes while holding a lock
- Let's go beyond simple locking!

CSE 486/586, Spring 2013

21

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).

CSE 486/586, Spring 2013

22