Distributed Computation

CSE 550: Systems for All
Autumn 2022

Lequn Chen

Project Proposal

- You decide topics
  - E.g., design and building a system, measurement study, overhead analysis, …
  - Some ideas: here
- Proposal
  - Due: Oct 24 (next Monday)
  - 1-2 page PDF (no formatting requirements)
- Please include the following:
  - The problem
  - Proposed solution
  - Necessary context, tools, libraries and resources you would use in the process.
  - Timeline and Checkpoints

Distributed Systems are everywhere!

- Some of the most powerful services are powered using distributed systems
  - systems that span the world,
  - serve billions of users,
  - and are always up!
- … but also pose some of the hardest CS problems

What is a distributed system?

- Multiple interconnected computers that cooperate to provide some service
- What are some examples of distributed systems?

Why distributed systems?

- Higher capacity and performance
- Geographical distribution
- Build reliable, always-on systems

How is a distributed system different from a single machine?

- “machine”: computation + storage + communication
- Can a multicore machine be called a “distributed system?”

“Fallacies of distributed computing”

(by L Peter Deutsch)

- The network is reliable;
- Latency is zero;
- Bandwidth is infinite;
- The network is secure;
- Topology doesn't change;
- There is one administrator;
- Transport cost is zero;
- The network is homogeneous.

Difference to single machine

- Network
  - Slower latency; Lower bandwidth;
  - Packet might get lost
  - No upper bound on delay
- Storage: message passing vs shared-memory
- Failure: Components can fail
- Clock: Unsynchronized clocks (No upper bound on drift)
- Uncertainties!!!
  - Is the packet lost? Is the network slow or disconnected?
  - Is the primary node down? Should I become the new primary? Do other nodes know that I become the new primary?

What are the challenges in building distributed systems?

- (Partial) List of Challenges
  - Fault tolerance (different failure models, different types of failures)
  - Unsynchronized clocks and ordering events
  - Consistency/correctness of distributed state
  - Performance
  - Scaling
  - Security
  - System design, architecture, testing
- We want to build distributed systems to be more scalable, and more reliable.
- But it’s easy to make a distributed system that’s less performant and less reliable than a centralized one!

Clocks & Events

- Why do we need clocks?
- Why do we need to order events in a distributed system?

Distributed Build System

- Distributed file servers holds source and object files
- Clients specify modification time on uploaded files
- Use timestamps to decide what needs to be rebuilt
  - if output object O depends on source file S, and
  - O.time < S.time, rebuild O
- What can go wrong?

Another example

- On social networking site
  - Remove boss as friend
  - Post: “My boss is the worst, I need a new job!”, visible to friends only
- Social networking site is a distributed system
  - Friendship links, posts, privacy settings stored across a large number of distributed servers
  - lots of copies of data: replicas, caches, cross datacenter replication, etc.
- Don’t want to get a concurrent read to see the wrong order!

Clocks

- Synchronized clocks
  - What are the sources of inaccuracy?
  - Why is it ~~hard~~ impossible to synchronize clocks?
  - How would you synchronize clocks?
    - Broadcast?
    - Ask?
  - What are the certainties that you want to get from synchronized clocks?
  - What about blockchains?
- Logical clocks
  - “Counters”; no real physical clocks are needed.
  - Capture causal relationships: “A happens before B.” “C and D is concurrent.”
  - Lamport clock
  - Vector clock