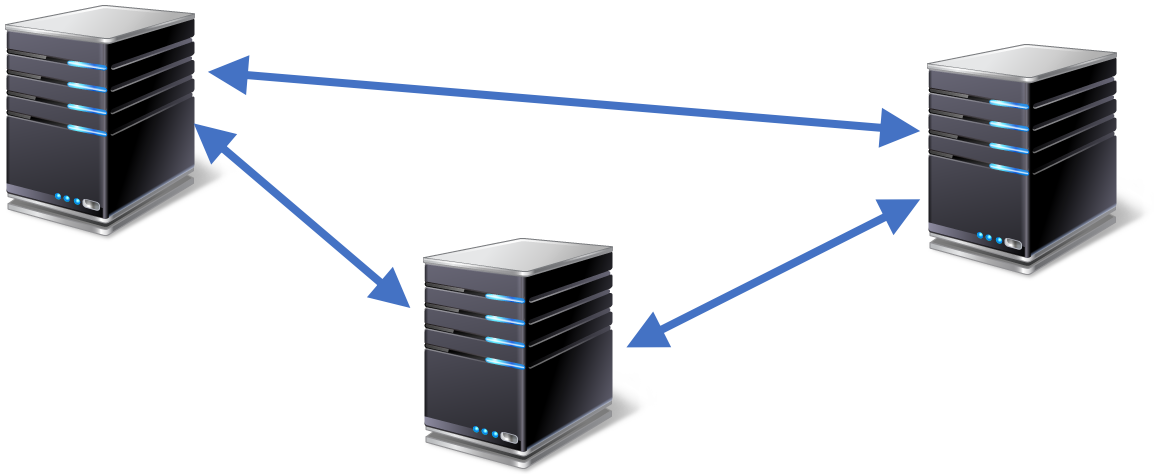
Putting it all together – Final Review

CS 475: Concurrent & Distributed Systems (Fall 2021)

Yue Cheng

Back in Lec-1…

Distributed systems: What?



- Multiple cooperating computers
  - Connected by a network
  - Doing something together

- Lots of critical infrastructure are distributed

Distributed systems: Why?

- Or, why not 1 computer to rule them all?
- Failure
- Limited computation/storage
- Physical location

Distributed systems: Why?

- Or, why not 1 computer to rule them all?
- Failure ➢Fault tolerance
- Limited computation/storage ➢Scalability
- Physical location ➢Availability, low latency

Goals of "distributed systems"

- Service with higher-level abstractions/interface
  - E.g., key-value store, programming model, …
- High complexity
  - Scalable (scale-out)
  - Reliable (fault-tolerant)
  - Well-defined semantics (consistent)
- Do "**heavy lifting**" so app developers don't need to

Theme

- Fundamental building blocks
- Abstractions and programming models
- Production system designs

Theme

- Fundamental building blocks
- Abstractions and programming models
- Production system designs

Fundamental building blocks

- Remote procedure calls (RPCs)

Fundamental building blocks

- Remote procedure calls (RPCs)

- Time & clocks

Fundamental building blocks

- Remote procedure calls (RPCs)
- Time & clocks
- Consensus algorithms

Fundamental building blocks

- Remote procedure calls (RPCs)
- Time & clocks
- Consensus algorithms
- Replication, sharding, transactions

Fundamental building blocks

- Remote procedure calls (RPCs)
- Time & clocks
  - **Vector clocks**
- Consensus algorithms
  - **Raft**
- Replication, sharding, transactions
  - **Serializability**

Theme

- Fundamental building blocks

- Abstractions and programming models

- Production system designs

Programming models

- MapReduce

- Spark

Programming models

- MapReduce

- **Spark**

Resilient Distributed Datasets & Spark

- Transformations and actions
- persist()
  - Not an action nor a transformation – tell which RDDs should materialize
- PageRank example
  - How iterative PR algorithm works
  - Where to place persist() in iterative PR

Theme

- Fundamental building blocks
- Abstractions and programming models
- Production system designs

Production system designs

- Amazon Dynamo

- Facebook memcache

Production system designs

- Amazon Dynamo

- **Facebook memcache**

Facebook memcache

- Memcache as a demand-filled, look-aside cache
  - `Read()` and `write()`

- Interesting problems solved in FB's production-scale memcache deployments

  1. Stale set: **a single region** vs. **geographically distributed**
  2. Thundering herds
  3. Incast congestion
  4. Incorporating McSqueal for what?

Final exam

- Thursday, Dec 09th, 7:30 – 10:00am
  - 150 minutes
  - Open-book, open-notes (you may use class notes, papers, and lab materials; you may read them on your laptop, **but you are not allowed to use any network**)
  - Let me know if you need testing center accommodation ASAP (**no guarantee if you send me the form one day before the final exam**)

- Covering topics from lec-1 to lec-17
  - High-level design questions
  - **30%** before midterm **70%** after midterm

Topics

1. Vector clocks
2. Raft
3. Transactions
4. Spark
5. Facebook memcache

Don’t forget to fill out the course evaluation form

Good luck! ☺