

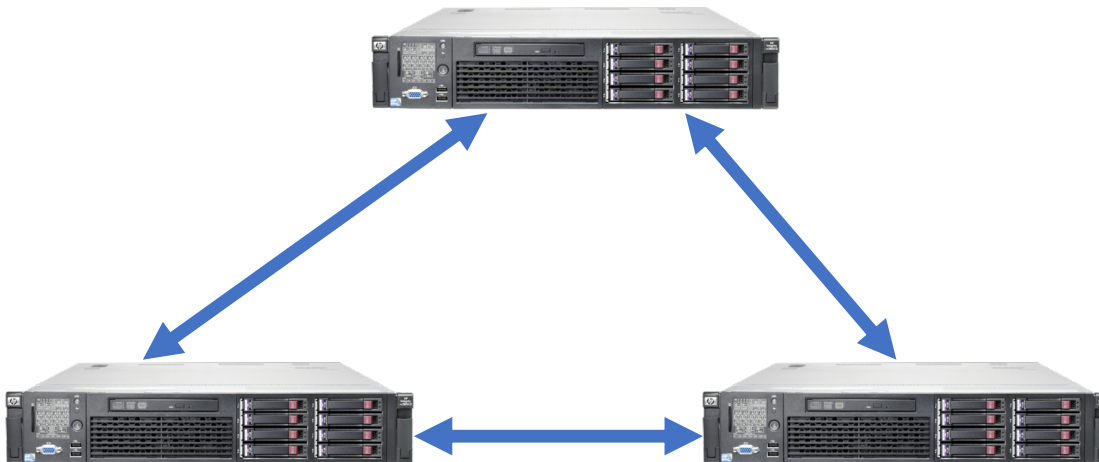
Distributed Systems Intro



COS 418/518: Distributed Systems
Lecture 1

Wyatt Lloyd

Distributed Systems, What?



- 1) Multiple computers
- 2) Connected by a network
- 3) Doing something together

Distributed Systems, Why?

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

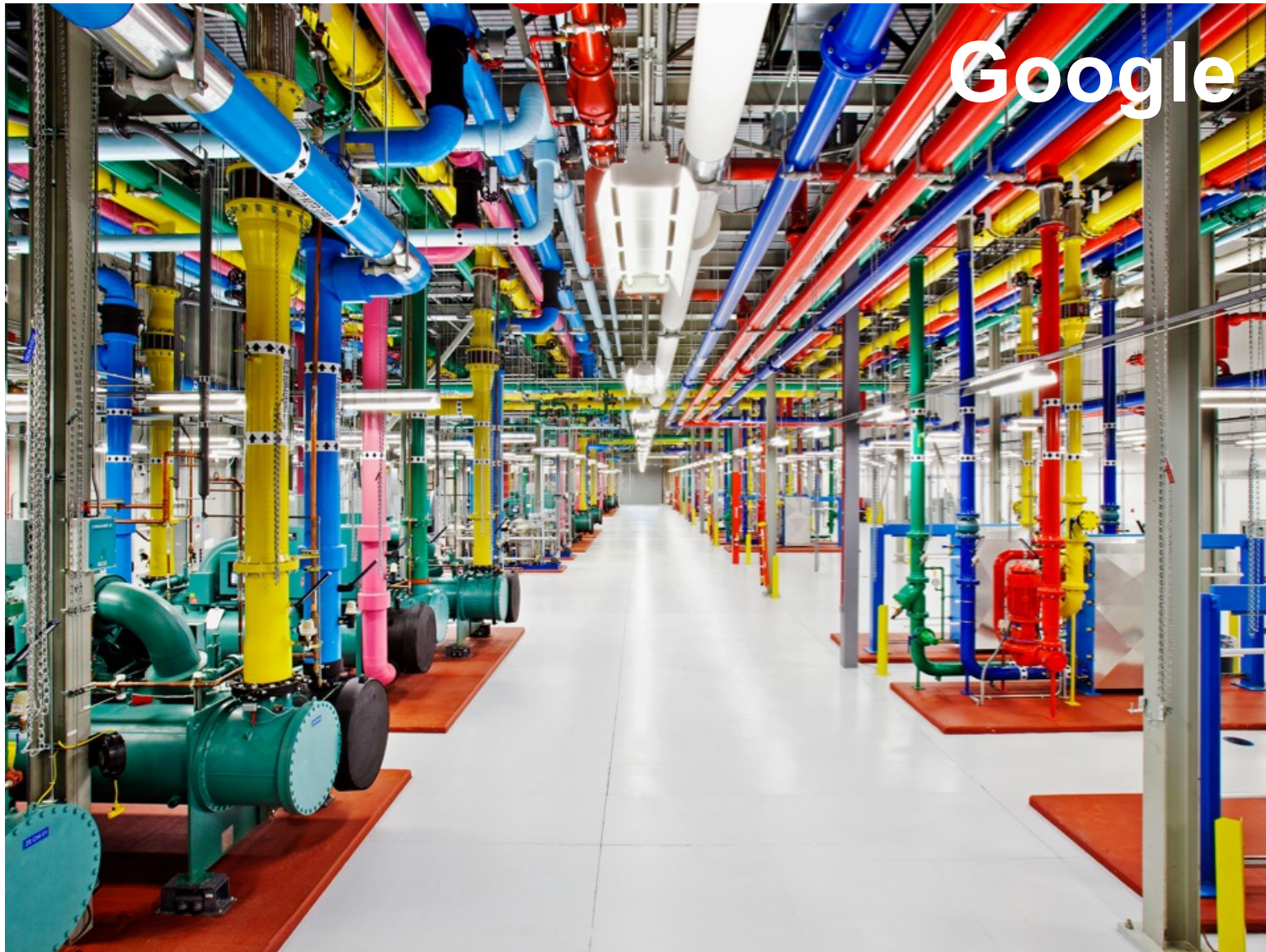
Distributed Systems, Where?

- Web Search (e.g., Google, Bing)
- Shopping (e.g., Amazon, Walmart)
- File Sync (e.g., Dropbox, iCloud)
- Social Networks (e.g., Facebook, Twitter)
- Music (e.g., Spotify, Apple Music)
- Ride Sharing (e.g., Uber, Lyft)
- Video (e.g., Youtube, Netflix)
- Online gaming (e.g., Fortnite, DOTA2)
- ...

**“The Cloud” is not
amorphous**



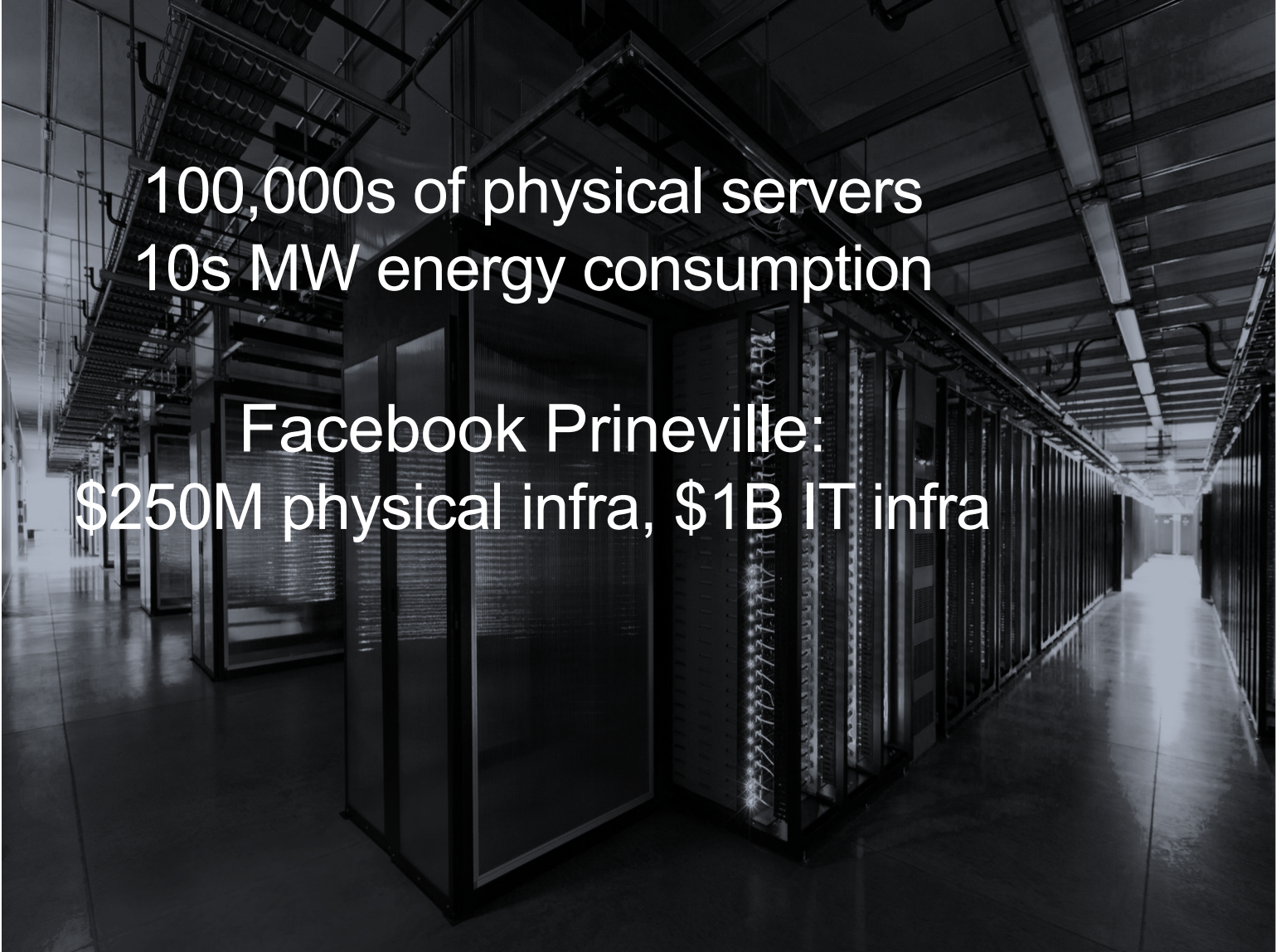
Microsoft





Facebook





100,000s of physical servers
10s MW energy consumption

Facebook Prineville:
\$250M physical infra, \$1B IT infra

Distributed Systems Goal

- Service with higher-level abstractions/interface
 - e.g., file system, database, key-value store, programming model, ...
- Hide complexity
 - Scalable (scale-out)
 - Reliable (fault-tolerant)
 - Well-defined semantics (consistent)
- Do “heavy lifting” so app developer doesn’t need to

Scalable Systems in this Class

- **Scale computation across many machines**
 - MapReduce, Streaming Video Engine
- **Scale storage across many machines**
 - Dynamo, COPS, Spanner

Fault Tolerant Systems in this Class

- Retry on another machine
 - MapReduce, Streaming Video Engine
- Maintain replicas on multiple machines
 - Primary-backup replication
 - Paxos
 - RAFT
 - Bayou
 - Dynamo, COPS, Spanner

Range of Abstractions and Guarantees

- Eventual Consistency
 - Dynamo
- Causal Consistency
 - Bayou, COPS
- Linearizability
 - Paxos, RAFT, Primary-backup replication
- Strict Serializability
 - 2PL, Spanner

Learning Objectives

- Reasoning about concurrency
 - Reasoning about failure
 - Reasoning about performance
-
- Building systems that correctly handle concurrency and failure
 - Knowing specific system designs and design components

Conclusion

- **Distributed Systems**
 - Multiple machines doing something together
 - Pretty much everywhere and everything computing now
- **“Systems”**
 - Hide complexity and do the heavy lifting (i.e., **interesting!**)
 - Scalability, fault tolerance, guarantees

