Scaling Up is Unlikely to Help

- Physical limits
- Economical constraints
- What about fault-tolerance?
- How would you upgrade?
- Anything else?

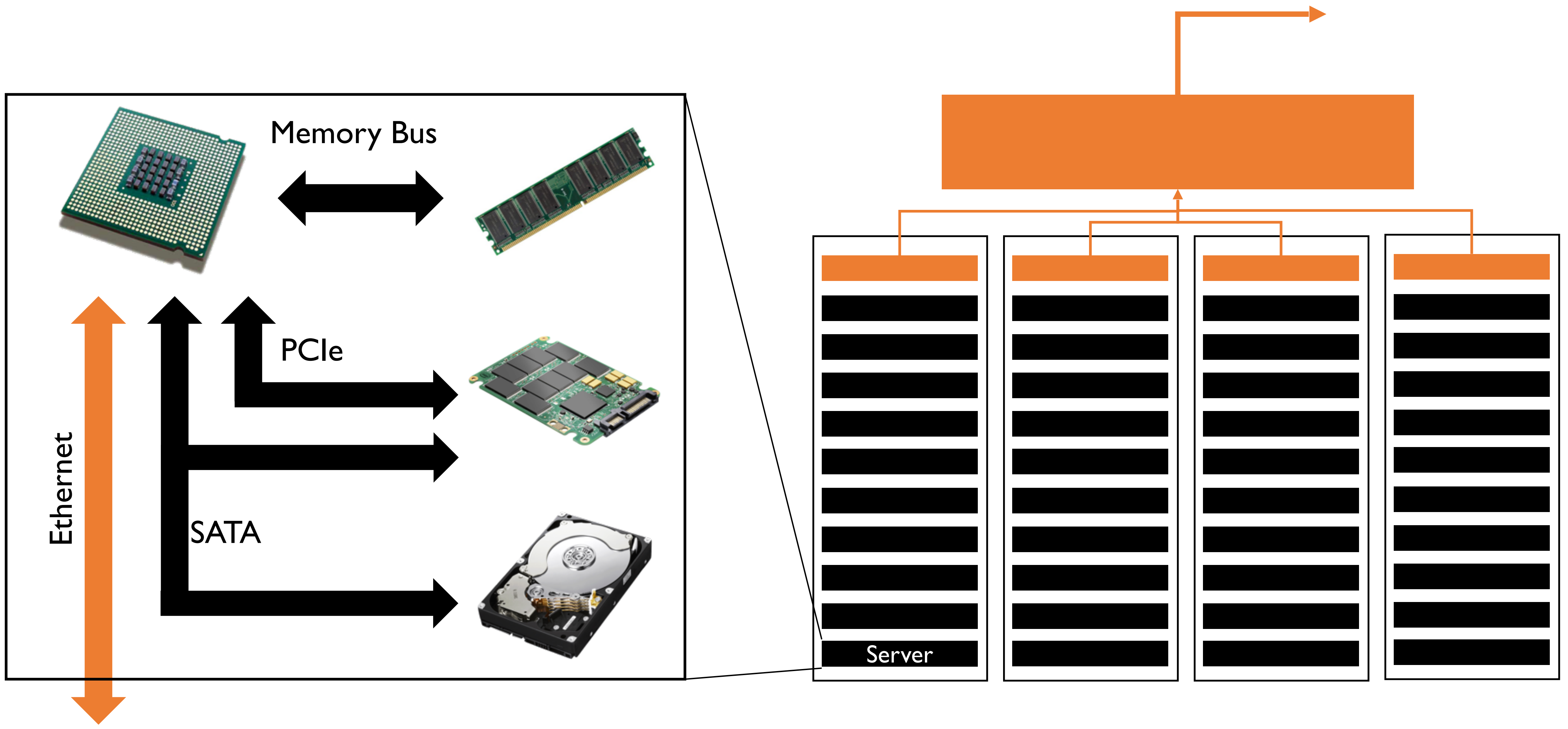
Scale Out: Warehouse-Scale Computers

- Single organization
- Cost efficiency at scale
  - Multiplexing across applications and services
  - Rent it out!
- Homogeneity (to some extent)

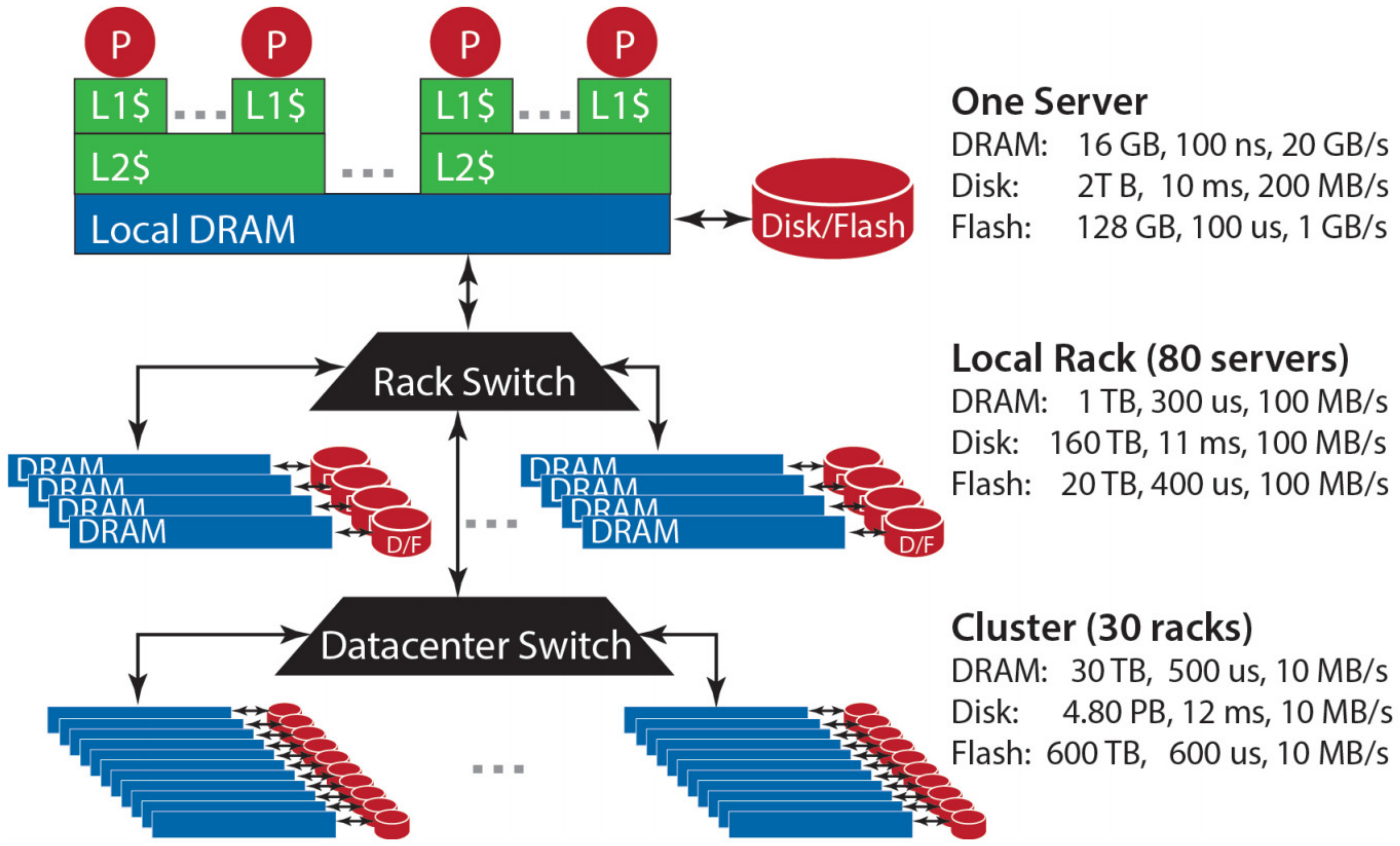
Scale Out: Warehouse-Scale Computers

- Many concerns
  - Infrastructure
  - Networking
  - Storage
  - Software
  - Power/Energy
  - Failure/Recovery
  - …

Architectural Overview



Memory Hierarchy Latency (2013)

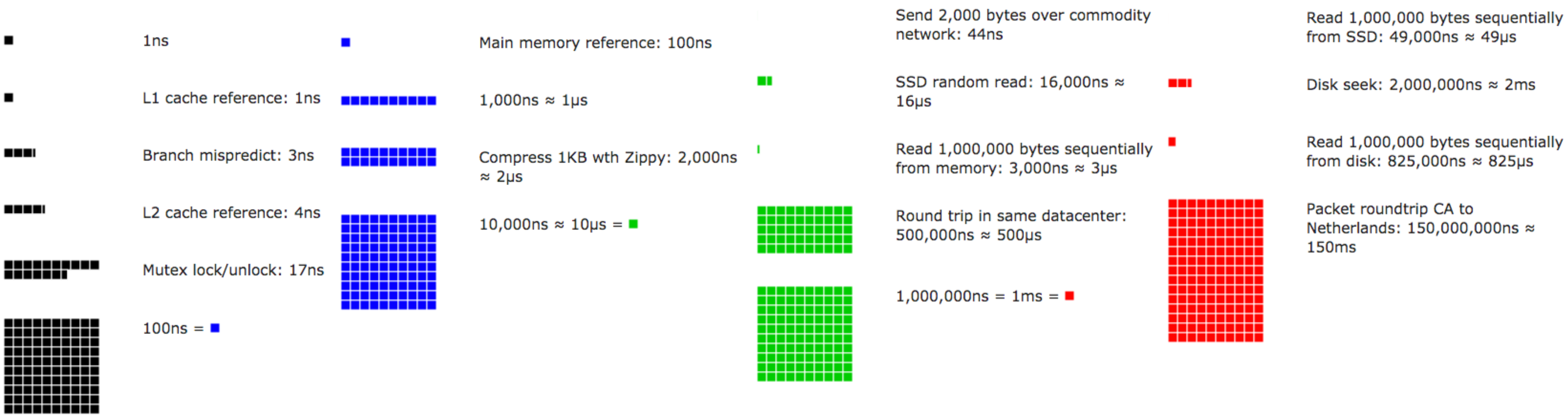


One Server
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

Local Rack (80 servers)
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

Cluster (30 racks)
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Memory Hierarchy Latency Visualized (2002)



Colin Scott: https://people.eecs.berkeley.edu/~rcs/research/interactive_latency.html

Power, Energy, Modeling, Building,…

- Many challenges
- We’ll focus primarily on software infrastructure in this class

Datacenter Needs an Operating System

- Datacenter is a collection of
  - CPU cores
  - Memory modules
  - SSDs and HDDs
- All connected by an interconnect

- A computer is a collection of
  - CPU cores
  - Memory modules
  - SSDs and HDDs
- All connected by an interconnect

Some Differences

1. High-level of parallelism
2. Diversity of workload
3. Resource heterogeneity
4. Failure is the norm
5. Communication dictates performance

Three Categories of Software

1. Platform-level
   - Software firmware that are present in every machine
2. Cluster-level
   - Distributed systems to enable everything
3. Application-level
   - User-facing applications built on top

Datacenter Programming Models

- Fault-tolerance, scalable, and easy access to all the distributed datacenter resources
  - Users submit jobs to these models w/o having to worry about low-level details

- MapReduce
  - Grandfather of big data as we know today
  - Two-stage, disk-based, network-avoiding

- Spark
  - Common substrate for diverse programming requirements
  - Many-stage, memory-first

Resource Management

- Fair and efficient distribution of resources among many competing programming models and jobs
  - Does the dirty work so that users won't have to
- Mesos / YARN
  - Started with a simple question – how to run different versions of Hadoop?
  - Fairness-first allocator
- Borg
  - Google's cluster manager
  - Utilization-first allocator
  - Grand father of Kubernetes

Resource Allocation and Scheduling

- How do we divide the resources anyway?
- DRF
  - Multi-resource max-min fairness
  - Two-level; implemented in Mesos and YARN
  - HUG: DRF + High utilization
  - Carbyne: DRF + Altruism over complex DAGs

File Systems

- Fault-tolerant, efficient access to data
- GFS
  - Data resides with compute resources
  - Compute goes to data; hence, data locality