Low-Latency Datacenters

John Ousterhout
Platform Lab Retreat
May 29, 2015
Datacenters: Scale and Latency

- **Scale:**
  - 1M+ cores
  - 1-10 PB memory
  - 200 PB disk storage

- **Latency:**
  - < 0.5 μs speed-of-light delay

- **Most work so far has focused on scale:**
  - One app, many resources
  - Map-Reduce, etc.

- **Latency potential unrealized:**
  - High-latency hardware/software
  - Most apps designed to tolerate latency (communication via large blocks)
• Round-trip times (100K servers):
  ▪ Today: 100-500 µs best case
  ▪ Often much worse because of congestion
  ▪ Hardware limit: ~2 µs

• Storage latency dropping:
  ▪ Disk → Flash → DRAM

• Can we create a new platform that makes the hardware limit accessible to applications?

• If so, will it enable important new applications?
Clean-Slate Low-Latency Datacenter

- New switching architecture (30 ns per switch)
- NIC fused with CPU cores; on-chip routing
- User-level networking, polling instead of interrupts
- New transport protocol
- Storage systems based primarily in DRAM
- New software stack
New class of datacenter storage:
- All data in DRAM at all times (disk/flash for backup only)
- Large scale: aggregate 1000’s of servers
- Low latency: 5-10µs remote access

1000x improvements over disk in
- Performance
- Energy/op

Goal: enable a new class of data-intensive applications
TCP protocol optimized for:
- Throughput, not latency
- Long-haul networks (high latency)
- Congestion throughout
- Modest # connections/server

Future datacenters:
- High performance networking fabric:
  - Low latency
  - Multi-path
- Congestion primarily at edges
  - Little congestion in core
- Many connections/server (1M?)

Need new transport protocol
New Transport Protocol, cont’d

- Greatest obstacle to low latency:
  - Congestion at receiver’s link
  - Large messages delay small ones

- Solution: drive congestion control from receiver
  - Schedule incoming traffic
  - Prioritize small messages

- Behnam Montazeri will present work in progress
Low-Latency Software Stacks?

- Today’s stacks: highly layered
- Good for structuring software
  - Each layer solves one problem
- Bad for performance
  - Each layer adds latency
- Example: Thrift RPC system
  - Handles several problems: marshalling, threading, etc.
  - General-purpose: re-pluggable components
  - Adds 7 µs latency

For low latency, must replace the entire software stack
Reducing Software Stack Latency

1. Optimize layers (specialize?)
2. Eliminate layers
3. Bypass layers
Integrate NIC Into CPU Chip?

- Core
- Core
- Core
- Core
- Core
- Core

Switch

Per-Core Mini-NIC

OS controls routing tables for incoming packets

Network
Low Latency => New Applications?

- Does 2 µs latency matter?
- Use low latency for collecting data?
  - Small chunks of data
  - Random access
  - Dependencies serialize accesses
  - Need a lot of chunks in a small amount of time:
    - 20K chunks in 50 ms?
- Use low latency for new computational models?
  - Independent compute-storage elements
  - Low latency allows high coherency
Discussion Topics

- What are the key elements of a low-latency platform for datacenters?
- What will a new software stack look like?
- What applications could make use of a low-latency datacenter?