RAMCloud: Low-latency DRAM-based storage

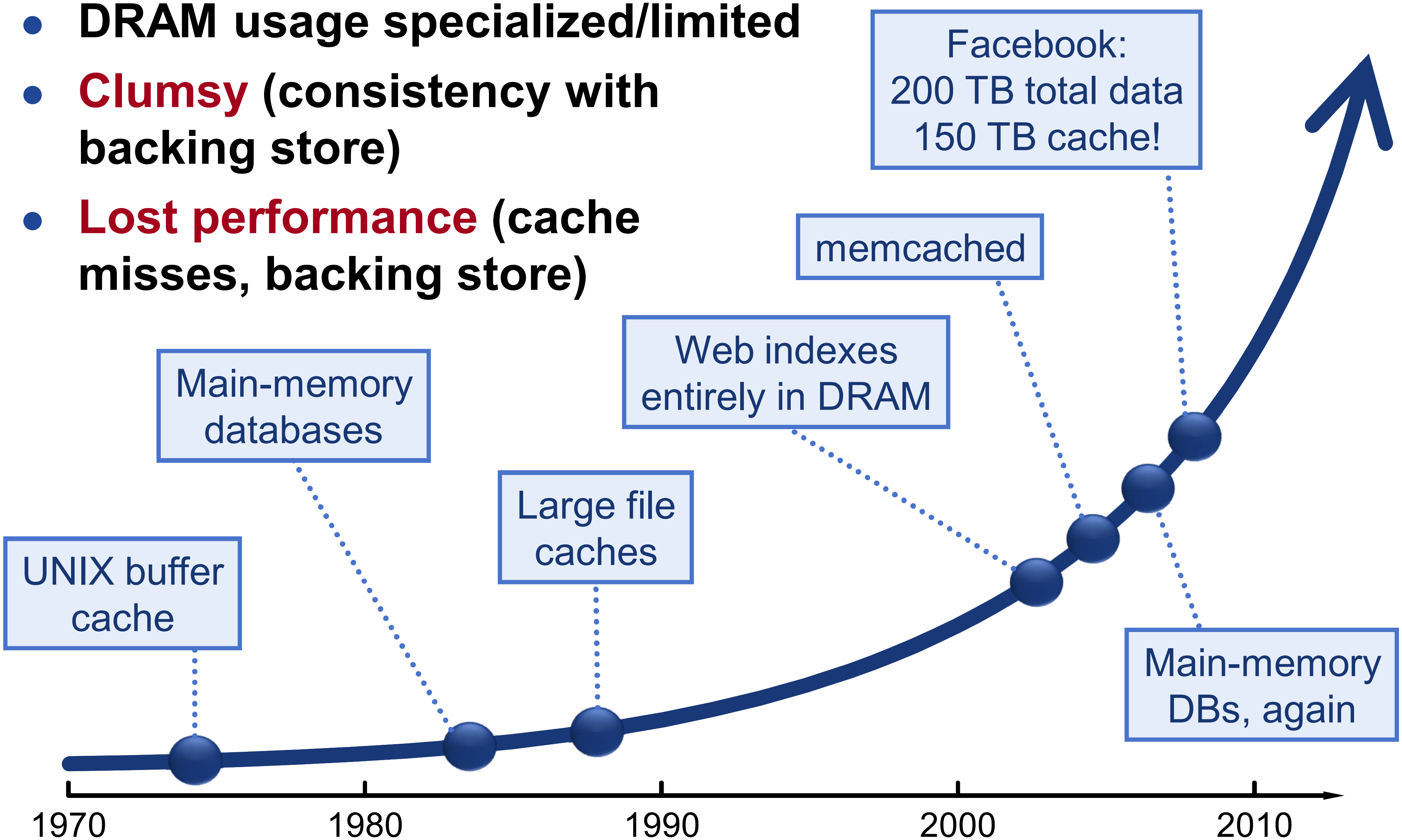
Jonathan Ellithorpe, Arjun Gopalan, Ashish Gupta, Ankita Kejriwal, Collin Lee, Behnam Montazeri, Diego Ongaro, John Ousterhout, Seo Jin Park, Henry Qin, Mendel Rosenblum, Stephen Rumble, Ryan Stutsman

Stanford University

http://ramcloud.stanford.edu

DRAM in Storage Systems

- **DRAM usage specialized/limited**
- **Clumsy (consistency with backing store)**
- **Lost performance (cache misses, backing store)**



RAMCloud Overview

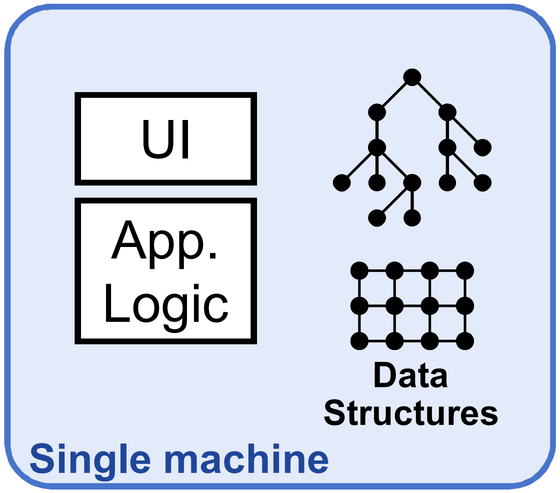
Harness full performance potential of large-scale DRAM storage:

- **General-purpose storage system**
- **All data always in DRAM (no cache misses)**
- **Durable and available**
- **Scale: 1000+ servers, 100+ TB**
- **Low latency: 5-10µs remote access**

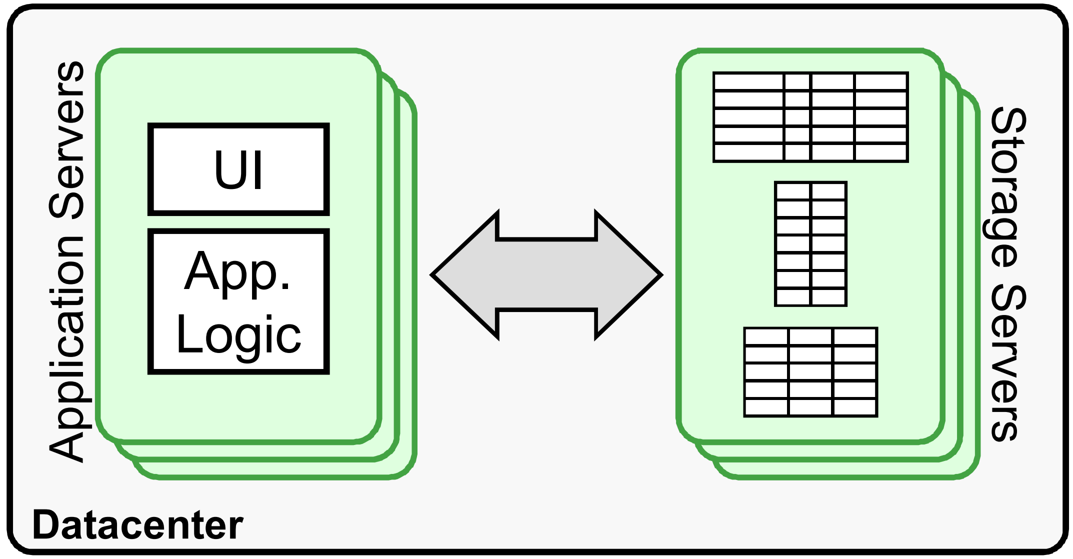
Potential impact: enable new class of applications

Why Does Latency Matter?

Traditional Application



Web Application



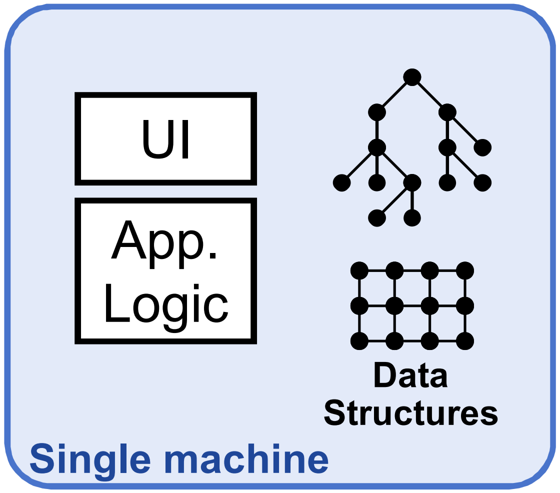
<< 1µs latency

0.5-10ms latency

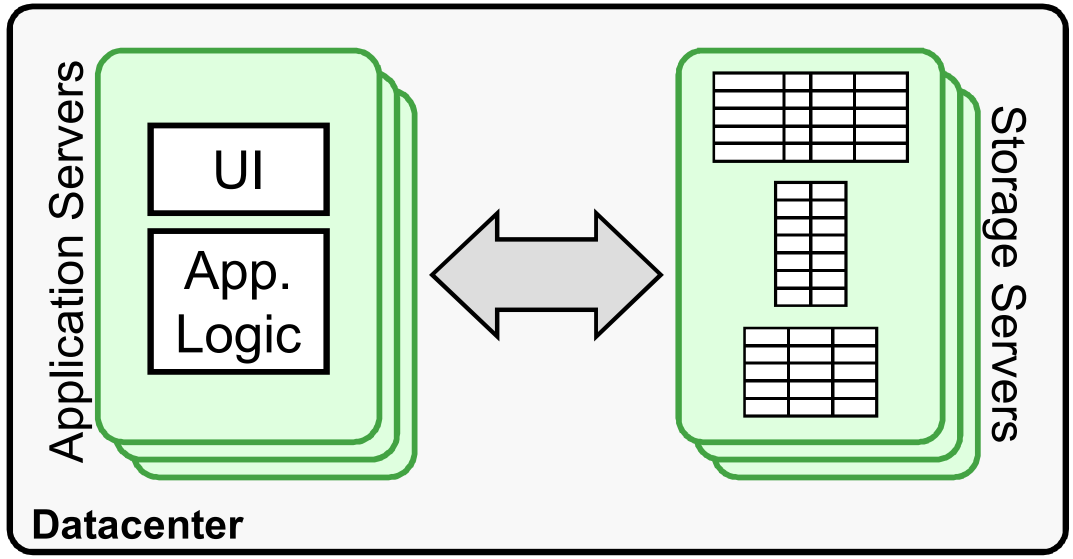
- **Large-scale apps struggle with high latency**
  - Random access data rate has not scaled!
  - Facebook: can only make 100-150 internal requests per page

Goal: Scale and Latency

Traditional Application



Web Application



<< 1µs latency

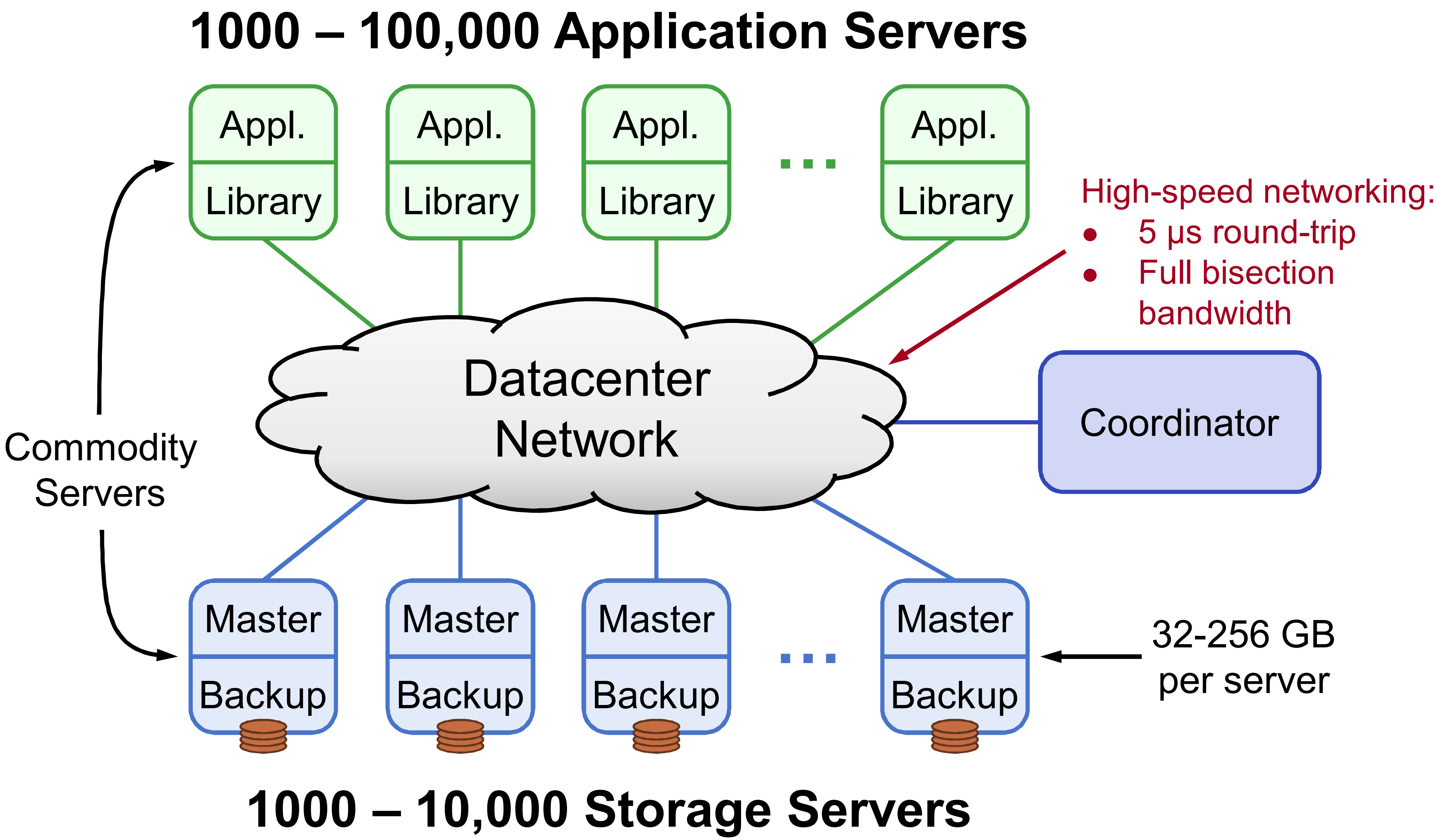
~~0.5-10ms~~ latency
5-10µs

- **Enable new class of applications:**
  - Crowd-level collaboration
  - Large-scale graph algorithms
  - Real-time information-intensive applications

Project Status

- **Started in 2009**
- **Building production-quality code**
  - About 125,000 lines of C++, permissive license
- **RAMCloud 1.0 released**
- **Looking for more users!**

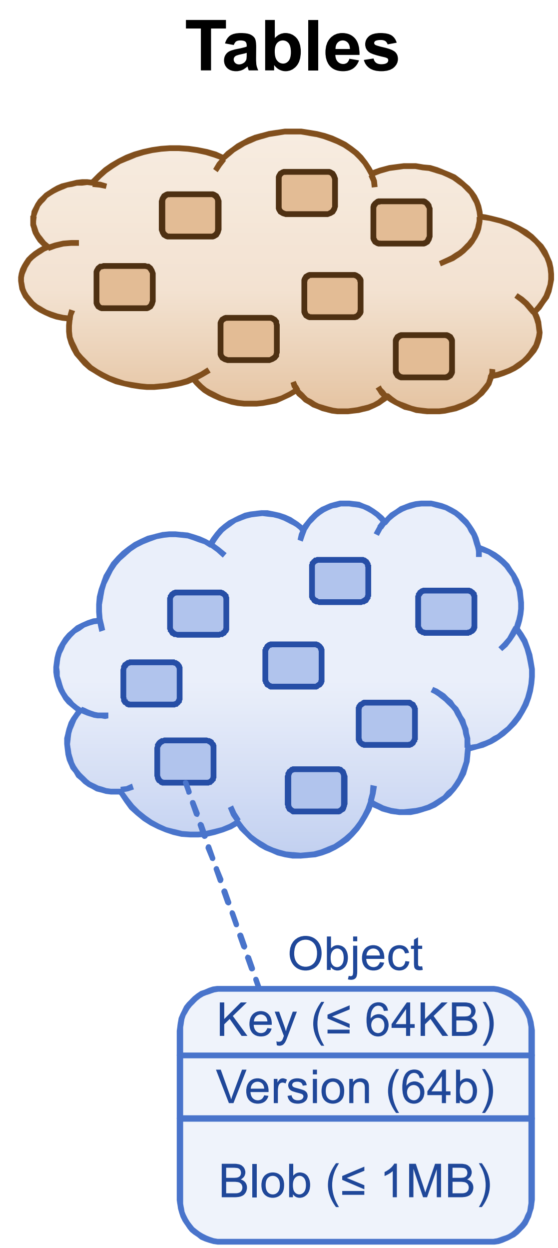
RAMCloud Architecture



Data Model: Key-Value Store

- **Basic operations:**
  - `read(tableId, key)`
    `=> blob, version`
  - `write(tableId, key, blob)`
    `=> version`
  - `delete(tableId, key)`
- **Other operations:**
  - `cwrite(tableId, key, blob, version)`
    `=> version`

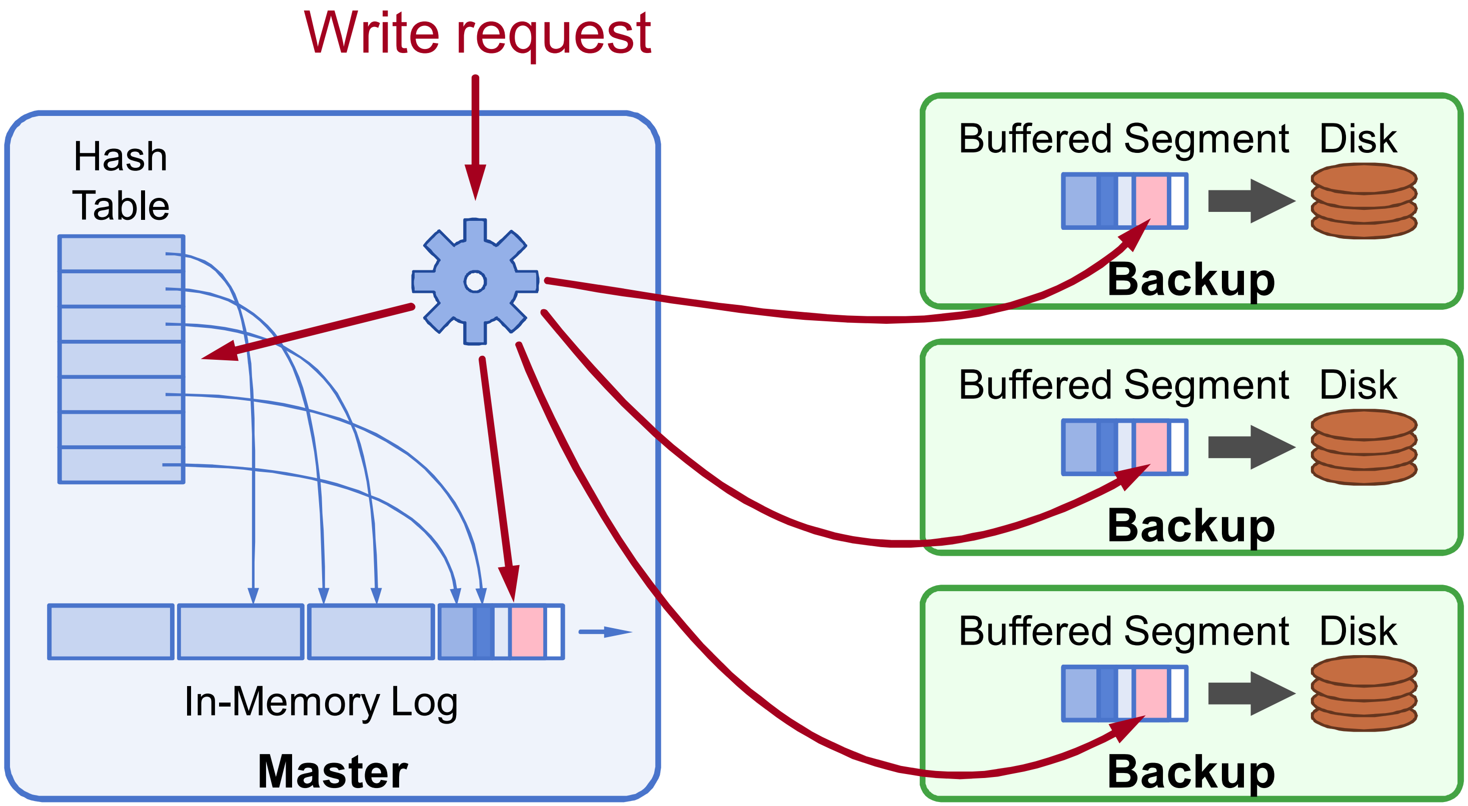
    (Only overwrite if version matches)
  - Enumerate objects in table
  - Efficient multi-read, multi-write
  - Atomic increment
- **Planned / in development:**
  - Atomic updates of multiple objects
  - Secondary indexes



Durability and Availability

- **Challenge: making volatile memory durable**
- **Keep replicas in DRAM of other servers?**
  - 3x system cost, energy
  - Still have to handle power failures
- **RAMCloud approach:**
  - 1 copy in DRAM
  - Backup copies on disk/flash: durability ~ free!

Buffered Logging



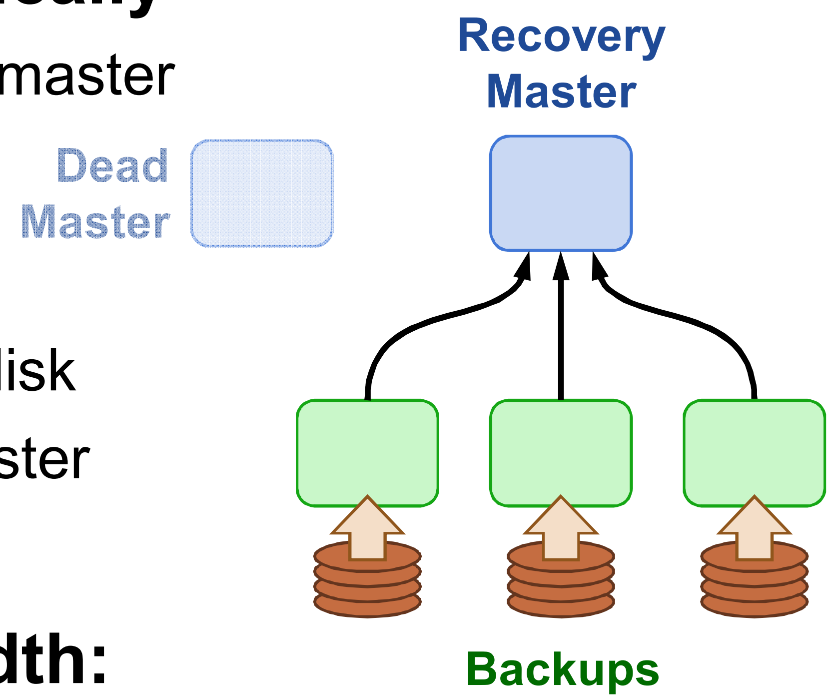
- **Log-structured: backup disk and master's memory**
- **Log cleaning ~ generational garbage collection**
- **No disk I/O during write requests**
  - Backups need ~64 MB NVRAM (or battery) for power failures

Crash Recovery

- **Applications depend on 5µs latency: must recover crashed servers quickly!**
- **Traditional approaches don't work**
  - Can't afford latency of paging in from disk
  - Replication in DRAM too expensive, doesn't solve power loss
- **Crash recovery:**
  - Need data loaded from backup disks back into DRAM
  - Must replay log to reconstruct hash table
  - Meanwhile, data is unavailable
  - Solution: fast crash recovery (1-2 seconds)
  - If fast enough, failures will not be noticed
- **Key to fast recovery: use system scale**

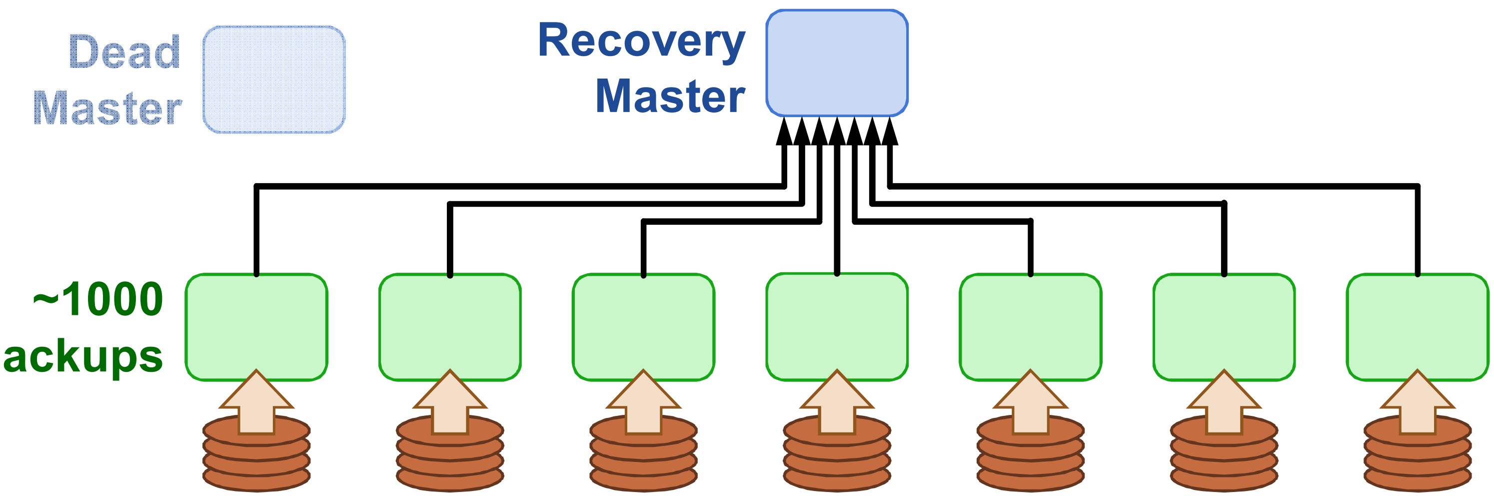
Recovery, First Try

- **Master chooses backups statically**
  - Each backup mirrors entire log for master
- **Crash recovery:**
  - Choose recovery master
  - Backups read log segments from disk
  - Transfer segments to recovery master
  - Recovery master replays log
- **First bottleneck: disk bandwidth:**
  - 64 GB / 3 backups / 100 MB/sec/disk
    ≈ 210 seconds
- **Solution: more disks (and backups)**

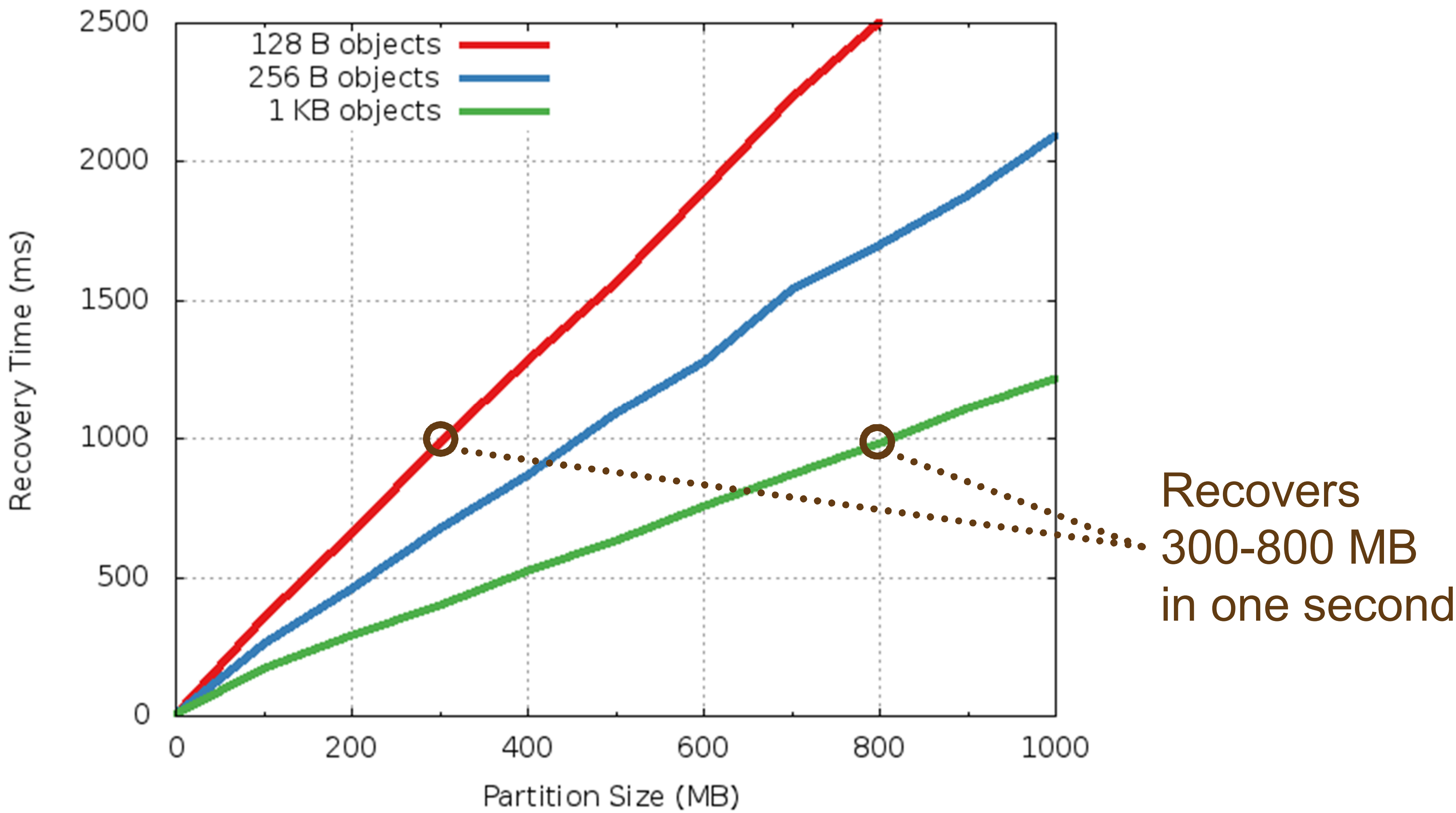


Recovery, Second Try

- **Scatter logs:**
  - Each log divided into 8MB segments
  - Master chooses different backups for each segment (randomly)
  - Segments scattered across all servers in the cluster
- **Crash recovery:**
  - All backups read from disk in parallel (100-500 MB each)
  - Transmit data over network to recovery master



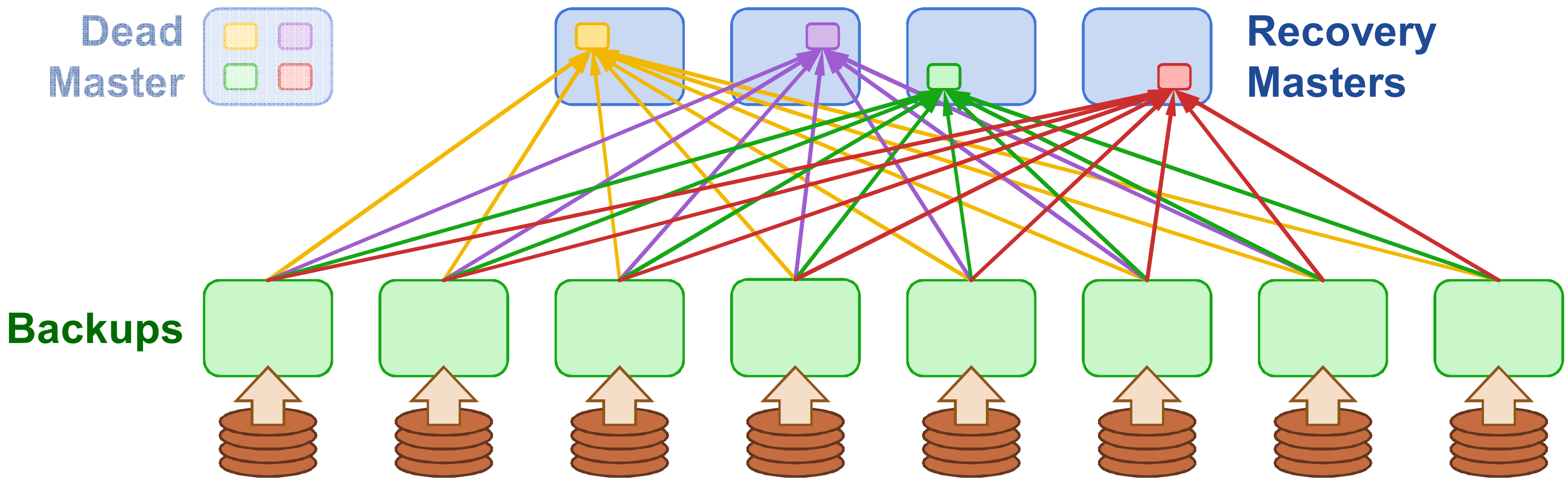
Single Recovery Master



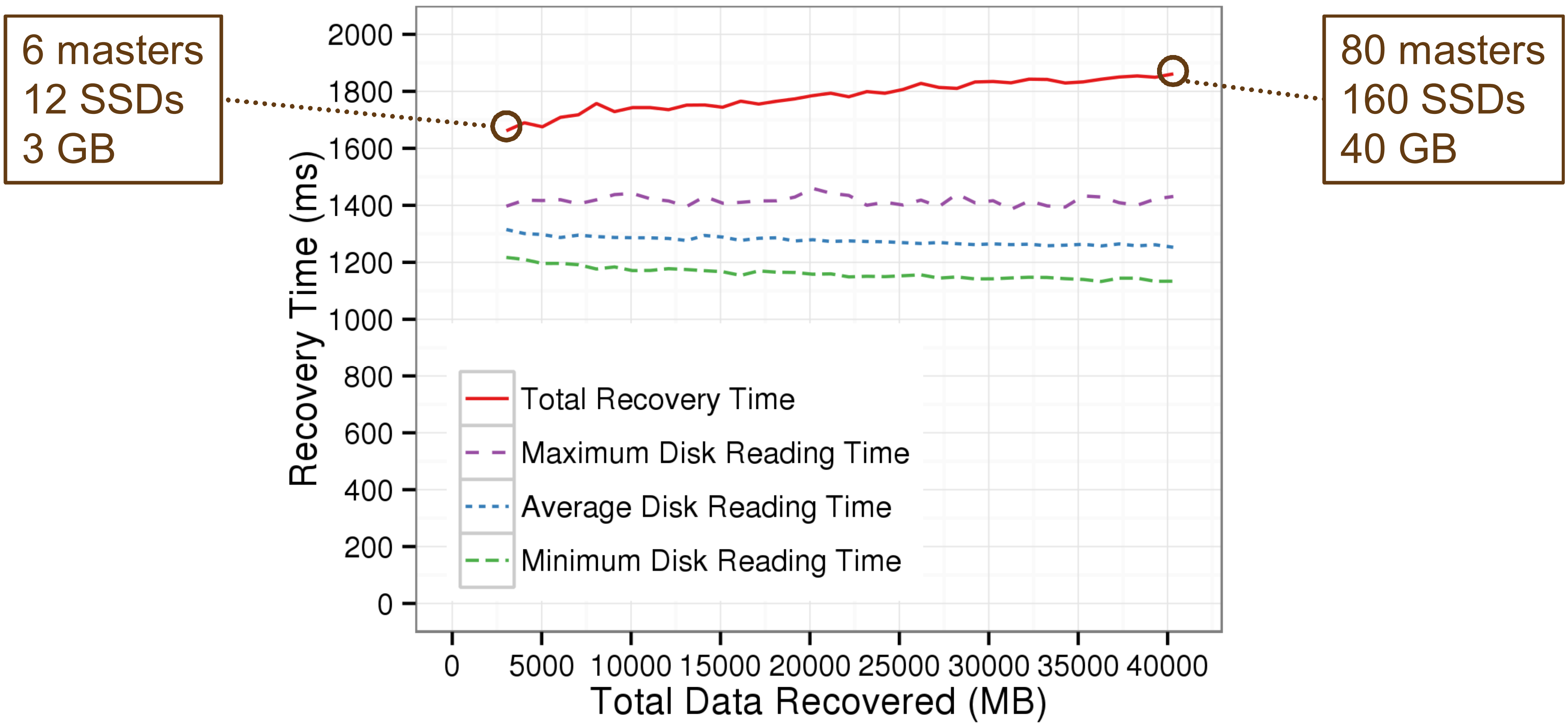
- **New bottlenecks: recovery master's NIC and CPU**

Recovery, Third Try

- Divide dead master's data into partitions
  - Recover each partition on a separate recovery master
  - Partitions based on key ranges, *not log segment*
  - Each backup divides its log data among recovery masters
  - Highly parallel and pipelined



Scalability



- **Nearly flat (needs to reach 120-200 masters)**
- **Expect faster: out of memory bandwidth (old servers)**

RAMCloud Summary

- **RAMCloud goals:**
  - Harness full performance potential of DRAM-based storage
  - Enable new applications: intensive manipulation of big data
- **Achieved low latency (at small scale)**
- **Not yet at large scale (but scalability encouraging)**
- **Fast crash recovery:**
  - Recovered 40GB in < 2 seconds; more with larger clusters
  - Durable + available DRAM storage for the cost of volatile cache
- **Looking for users!**
- **http://ramcloud.stanford.edu for code, papers**

Diego Ongaro, @ongardie