# RAMCloud 1.0

**John Ousterhout**

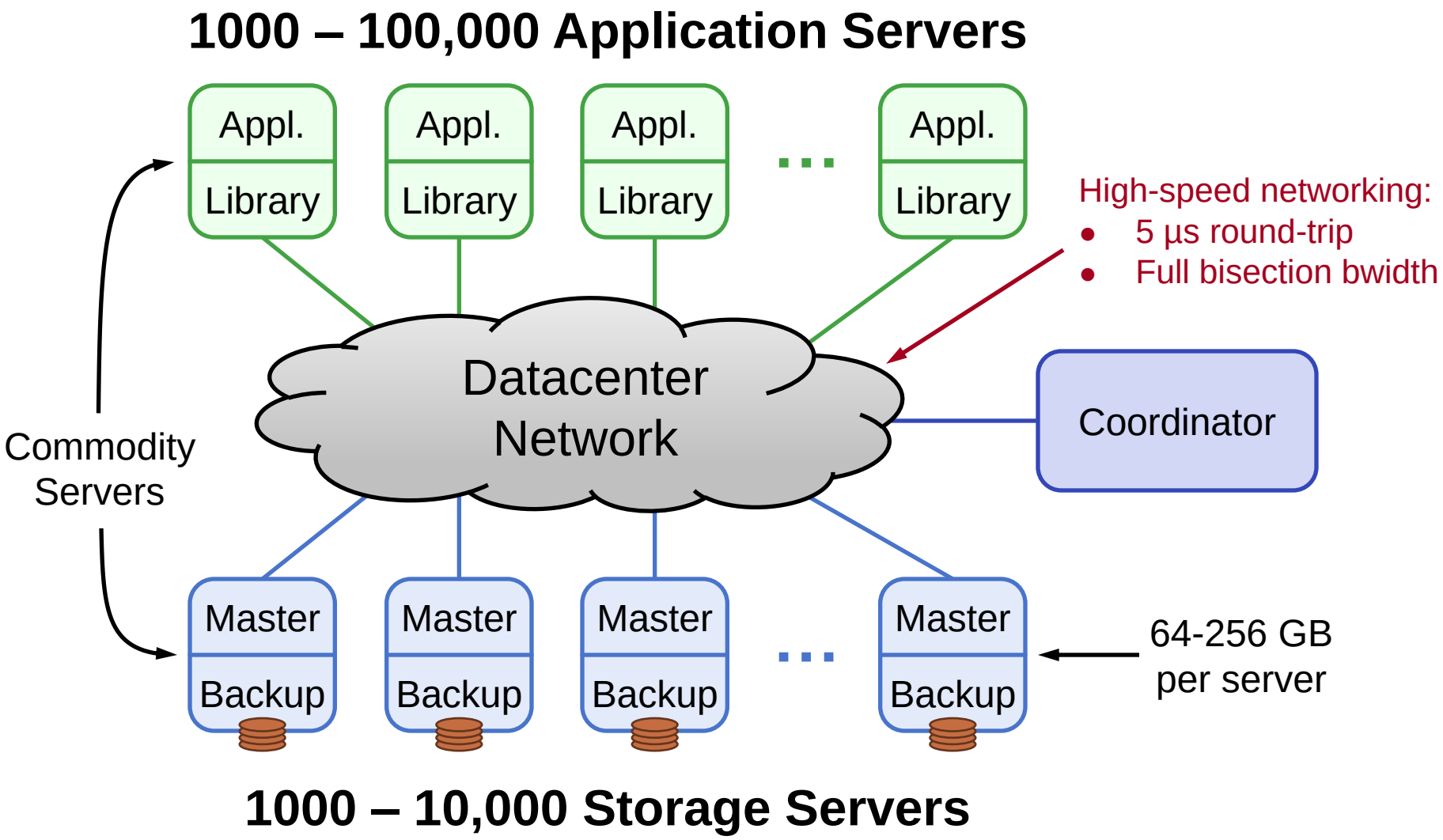
**Stanford University**

**(with Arjun Gopalan, Ashish Gupta, Ankita Kejriwal, Collin Lee, Behnam Montazeri, Diego Ongaro, Seo Jin Park, Henry Qin, Mendel Rosenblum, Stephen Rumble, and Ryan Stutsman)**

# Overview

- **RAMCloud project in transition**
- **Phase 1 complete:**
  - RAMCloud 1.0 released
    - Key-value store
    - Memory management
    - Fast crash recovery
  - First PhD students graduating
- **Phase 2 starting up:**
  - New projects:
    - Higher-level data models
    - Datacenter RPC revisited
  - Many new students

# RAMCloud Architecture

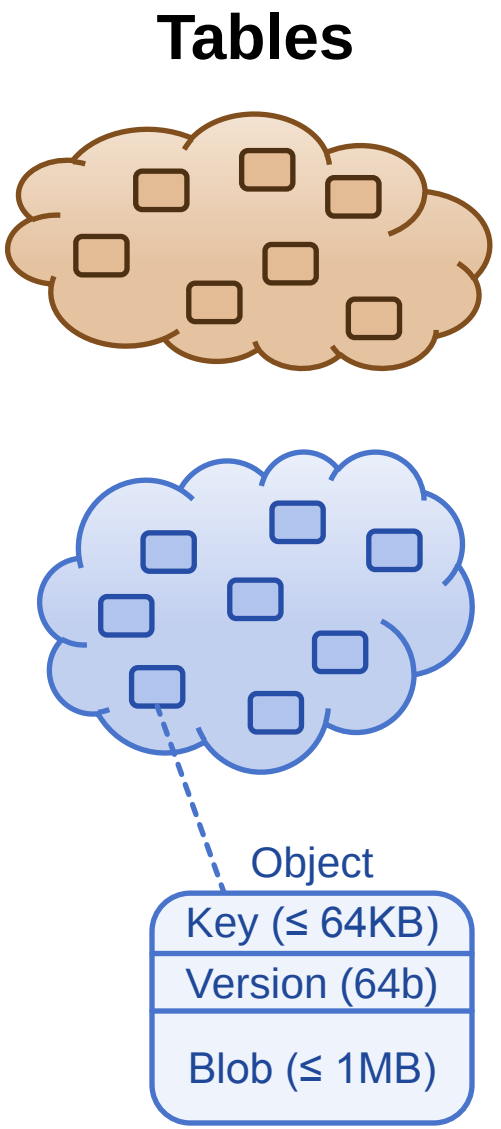


# Data Model: Key-Value Store

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

(Only overwrite if version matches)

- **Other operations:**
  - `cwrite(tableId, key, blob, version)`
    `=> version`
  - Enumerate objects in table
  - Efficient multi-read, multi-write
  - Atomic increment
- **Not currently available:**
  - Atomic updates of multiple objects
  - Secondary indexes



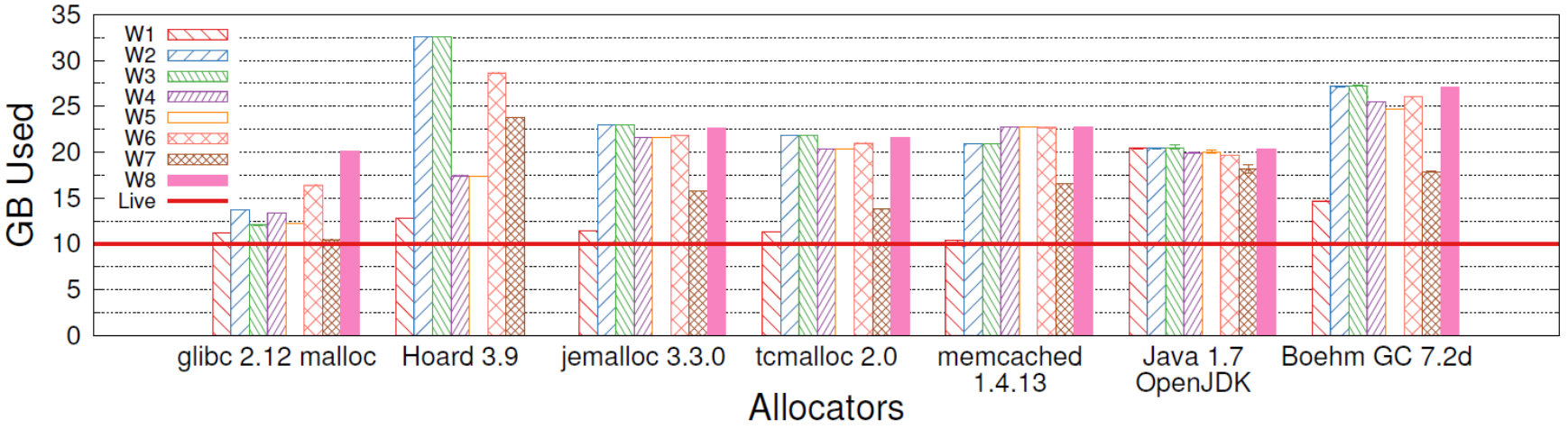
# Performance

- **Using Infiniband networking**
  - 24 Gb/sec effective bandwidth
  - Kernel bypass
  - Can also use other networking, but slower
- **Reads:**
  - 100B objects: 5µs
  - 10KB objects: 10µs
  - Single-server throughput (100B objects): 700 Kops/sec.
  - Small-object multi-reads: 1-2M objects/sec.
- **Writes:**
  - 100B objects: 15µs
  - 10KB objects: 40µs

# Steve Rumble's Dissertation

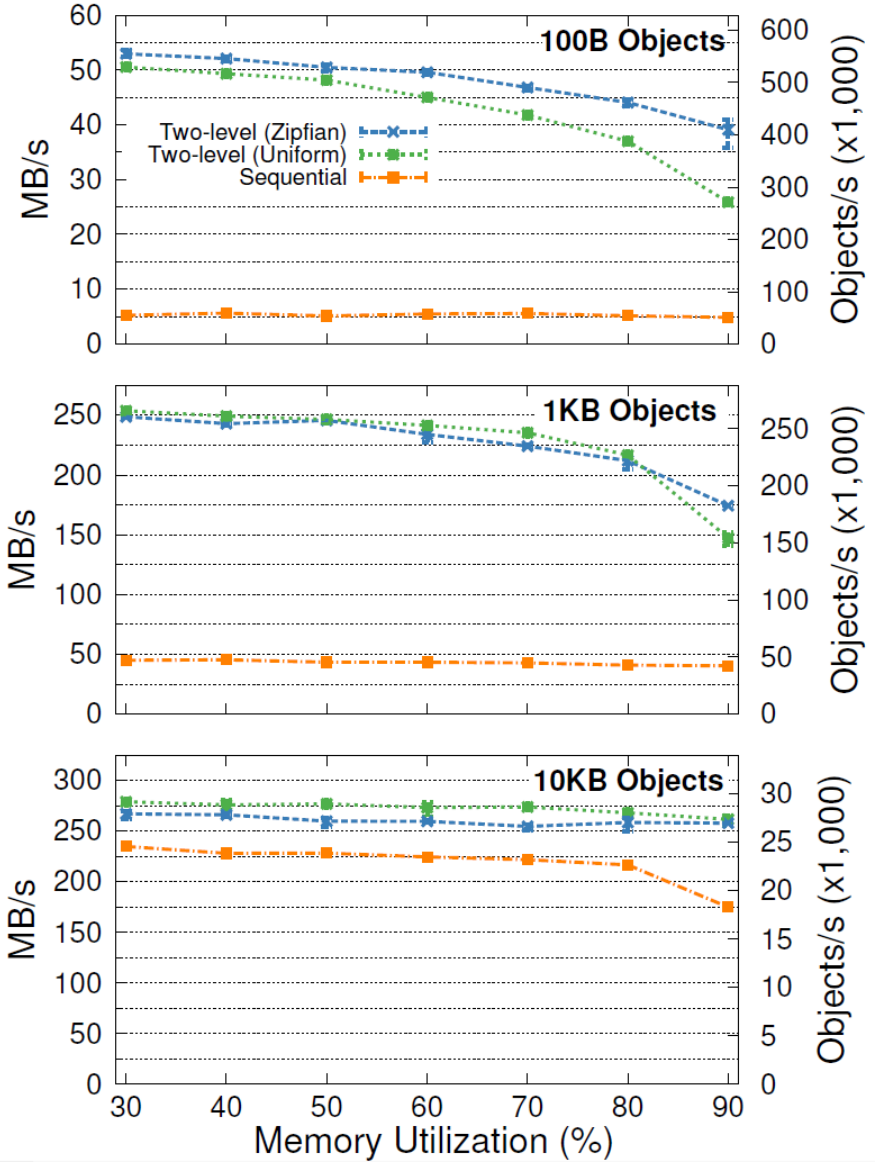
- **How to manage objects in DRAM?**
  - High write performance
  - High memory efficiency (80-90% utilization)
- **Uniform log structure for all info (both DRAM and disk)**
  - Log cleaner => incremental generational garbage collector
- **Innovative aspects:**
  - 2-level cleaning (different policies for DRAM, disk)
  - Parallel cleaning (hides cost of cleaning)
  - Improved LFS segment selection formula
- **Paper in FAST 2014, dissertation nearing completion; Steve is working at Google Zurich**

# Existing Allocators Waste Memory




- **Allocators waste memory if workloads change:**
  - E.g., W2 (simulates schema change):
    - Allocate 100B objects
    - Gradually overwrite with 130B objects
- **All existing allocators waste at least 50% of memory under some conditions**

# Client Write Throughput




| Memory Utilization | Performance Degradation |
|---|---|
| 80% | 17-27% |
| 90% | 26-49% |
| 80% | 14-15% |
| 90% | 30-42% |
| 80% | 3-4% |
| 90% | 3-6% |

# Ryan Stutsman's Dissertation

**Durability and availability for data in DRAM**

- **Fault-tolerant log for each master:**
  - Decentralized log management
  - Finding log after crashes
  - Restoring redundancy after backup crashes
- **Fast crash recovery:**
  - Use thousands of servers concurrently to reload data from a crashed master
  - 1-2 second recovery
- **Handling simultaneous master/backup failures**
- **Dissertation filed December 2013**
  **Ryan is now a post-doc at Microsoft Research**

# Recovery Scalability

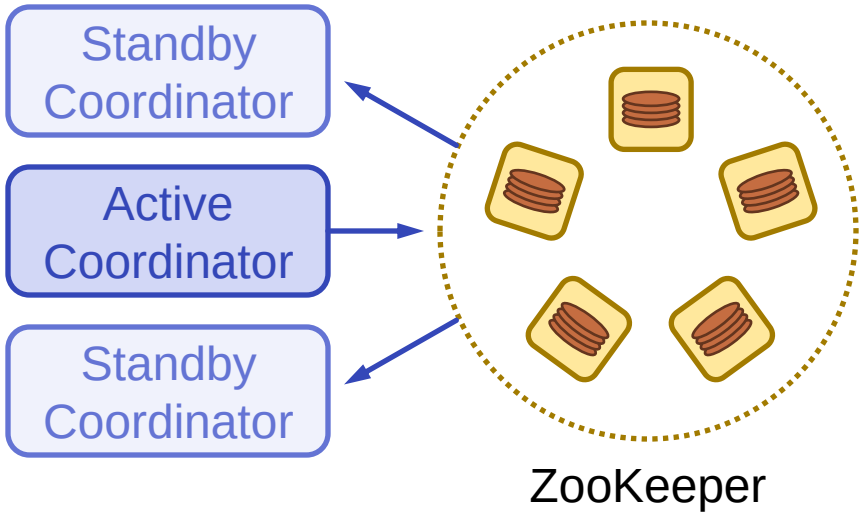
1 master
2 backups
2 SSDs
500 MB


80 masters
160 backups
160 SSDs
40 GB

- **Will improve with newer machines**
  - More cores (our nodes: 4 cores)
  - More memory bandwidth (our nodes: 11 GB/sec)
- **Bottom line: recover 500 MB/sec/server**

# Coordinator Crash Recovery

- **Active/standby model**
- **Use replicated external storage for configuration data:**
  - Cluster membership
  - Table metadata
- **Distributed "commit" mechanism:**
  - Record intent in storage
  - Notify relevant servers
  - (Eventually) mark storage "completed"
  - During restart, find and finish uncompleted ops



# Missing from RAMCloud 1.0

- **Table configuration management:**
  - Tablets not moved after creation
  - RAMCloud has mechanisms for splitting, migrating tablet
  - No policy code yet
- **Crude reconfiguration: crash server!**
  - Tablets split among many other servers

# New Project: Data Model

- **Goal: higher-level data model than just key-value store:**
  - Secondary indexes
  - Transactions spanning multiple objects and servers
  - Graph-processing primitives (sets)
- **Can RAMCloud support these without sacrificing**
  - Latency?
  - Scalability?
- **First project: secondary indexes (SLIK) (Arjun Gopalan, Ashish Gupta, Ankita Kejriwal)**
  - Design complete, implementation underway
  - Ankita will discuss design issues

# New Work: Datacenter RPC

## Complete redesign of RAMCloud RPC

- **General purpose (not just RAMCloud)**
- **Latency:**
  - Analyze, reduce latency
  - Explore alternative threading strategies
  - Optimize for kernel bypass
- **Scale:**
  - Support 1M clients/server (minimal state/connection)
  - Congestion control: reservation based?
- **Initial projects underway (Behnam Montazeri, Henry Qin)**
  - Analyze latency of existing RPCs
  - Support for pFabric, SolarFlare NIC

# Other Projects

- **Raft: new consensus protocol (Diego Ongaro)**
- **Graph processing on RAMCloud (Jonathan Ellithorpe)**
- **Using a rule-based approach for distributed, concurrent, fault-tolerant code (Collin Lee, Ryan Stutsman)**

# Conclusion

- **We now have a usable system**
- **Still many open research problems**
- **Real usage should generate additional information, ideas**