The Rise of RAMCloud

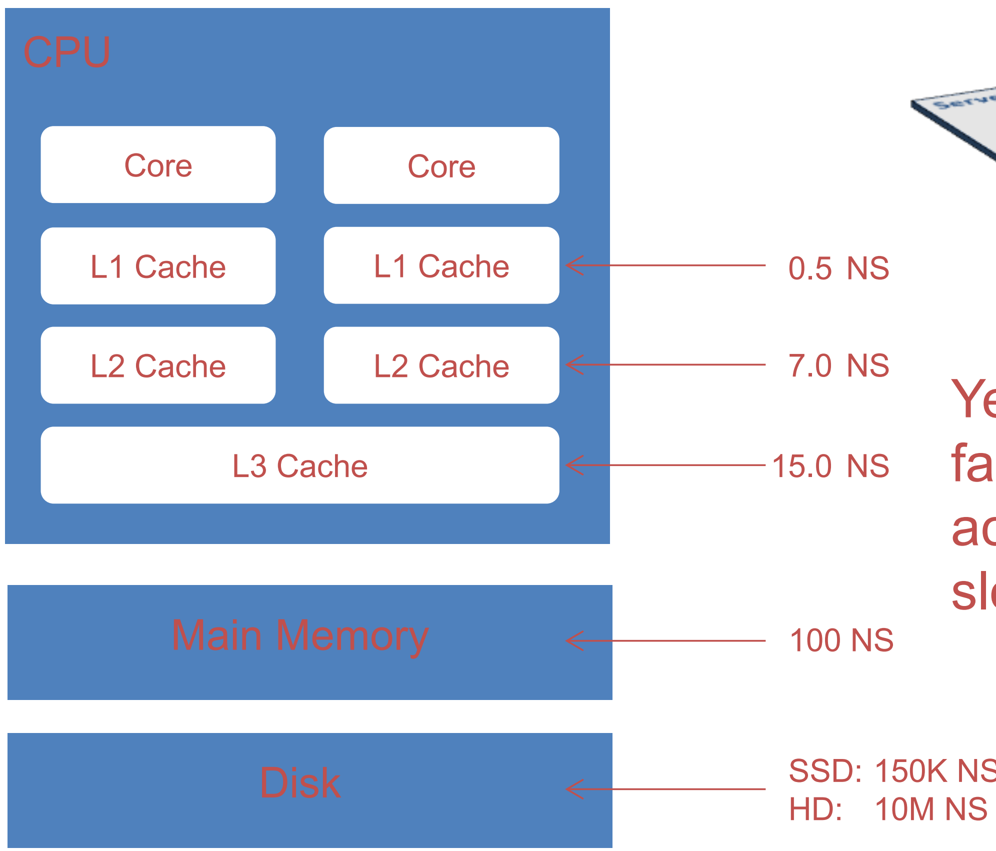
Rong Chen

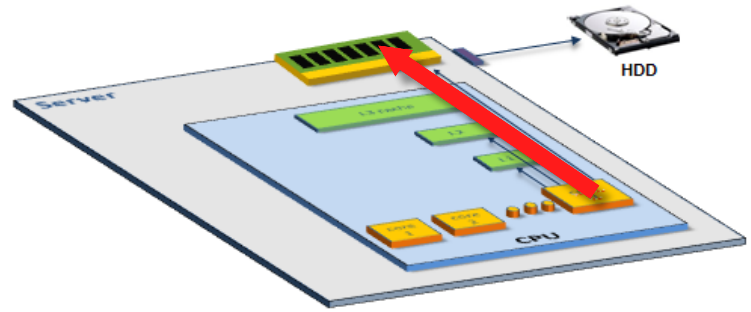
Slides from John Ousterhout@Stanford

Outline

- Overview
- Motivation
- Durability
- Crash Recovery

Review: Latency Numbers





Yes, DRAM is 100,000 times faster than disk, but DRAM access is still 6-200 times slower than on-chip caches

Source: SAP

RAMCloud Overview

Storage for datacenters

1000-10000 commodity servers

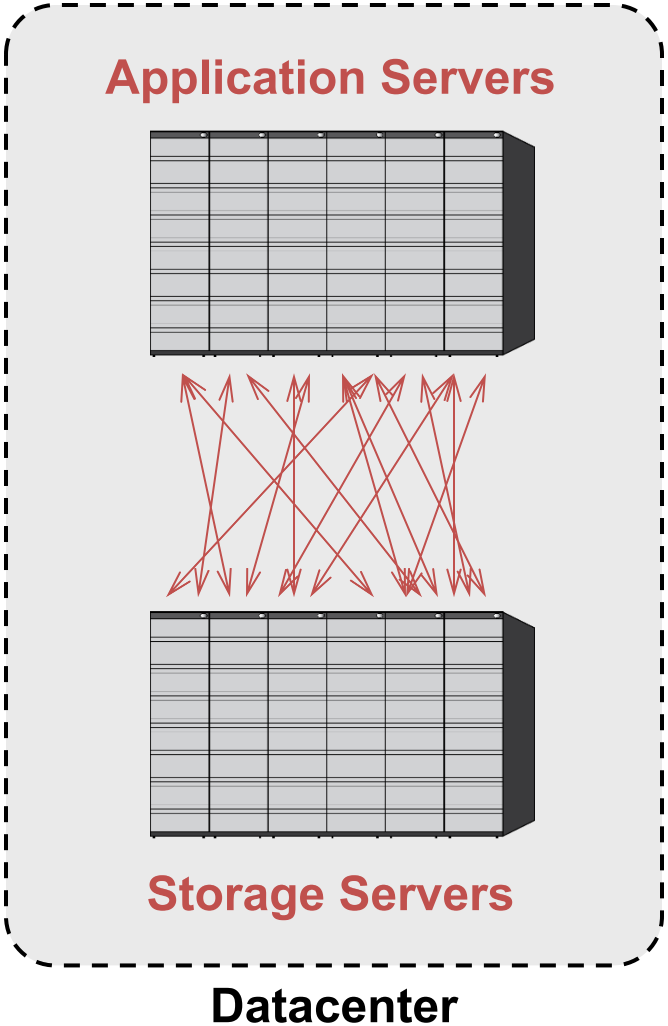
64 GB DRAM/server

All data always in RAM

Durable and available

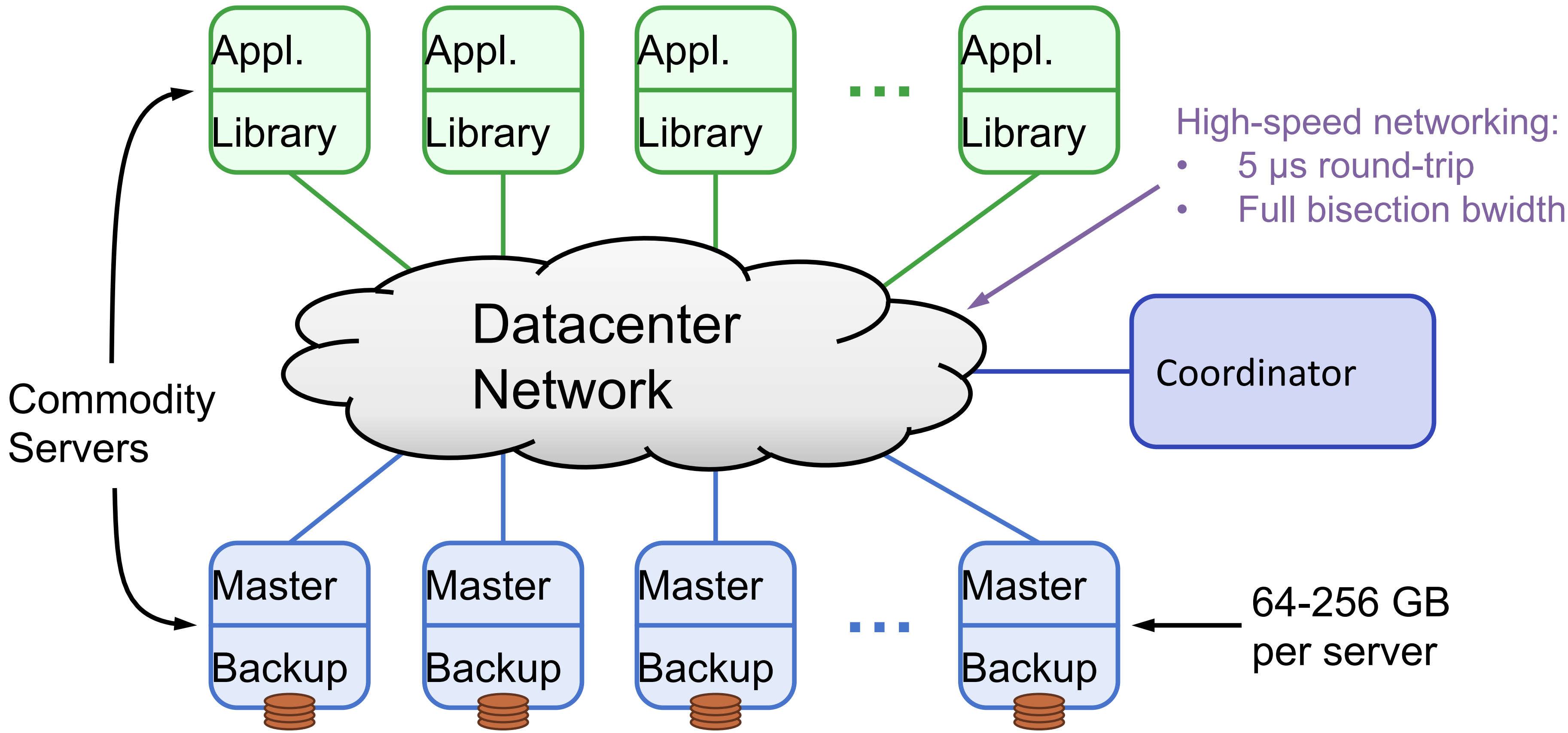
Performance goals:

- High throughput: 1M ops/sec/server
- Low-latency access: 5-10μs RPC



RAMCloud Architecture

1000 – 100,000 Application Servers

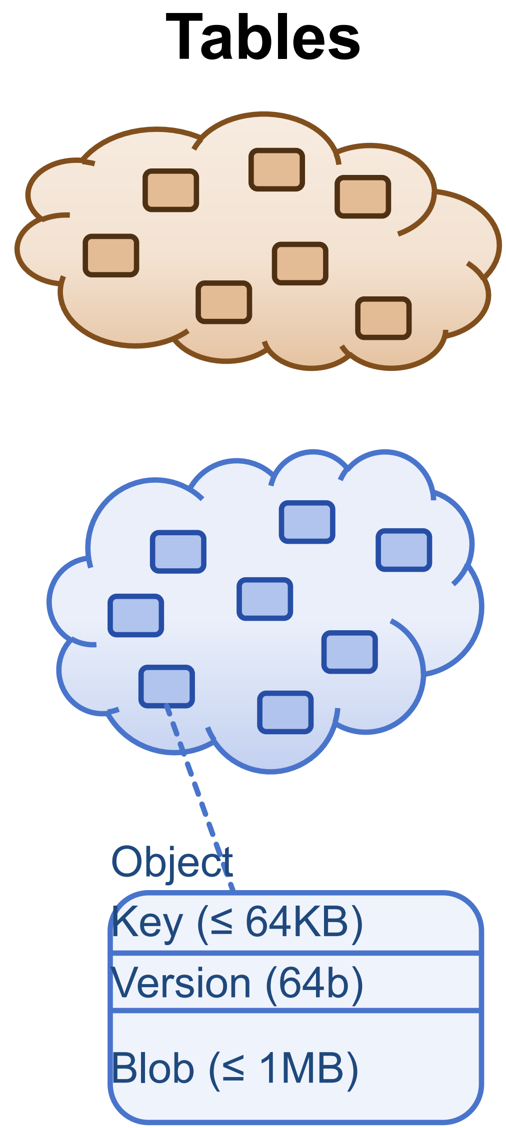


1000 – 10,000 Storage Servers

Data Model: Key-Value Store

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

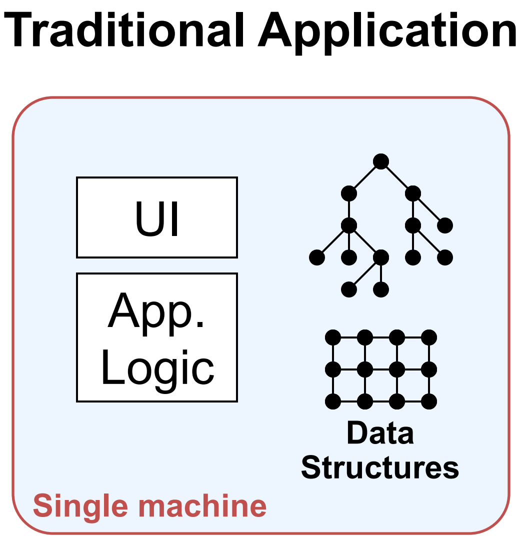
  (Only overwrite if version matches)
  - `cwrite(tableId, key, blob, version)`
    `=> version`
  - Enumerate objects in table
  - Efficient multi-read, multi-write
  - Atomic increment
- Not in RAMCloud 1.0:
  - Atomic updates of multiple objects
  - Secondary indexes

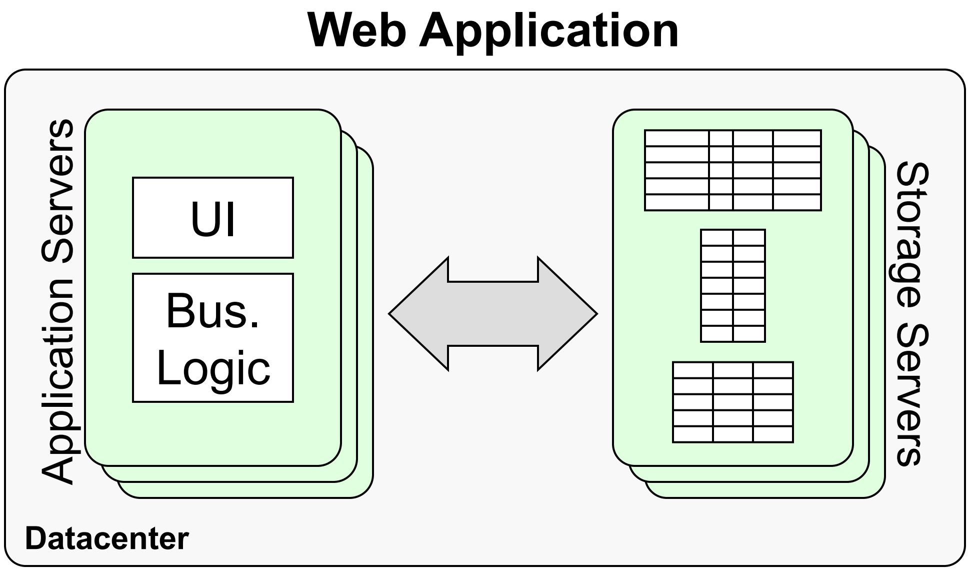


Example Configurations

	2010	5-10 years
# servers	1000	1000
GB/server	64GB	1024GB
Total capacity	64TB	1PB
Total server cost	$4M	$4M
$/GB	$60	$4

Motivation: Latency



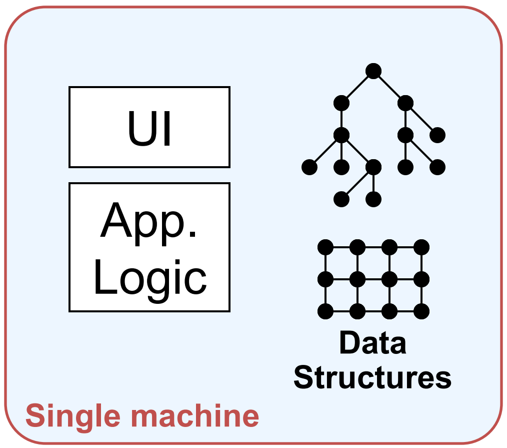


<< 1µs latency **0.5-10ms latency**

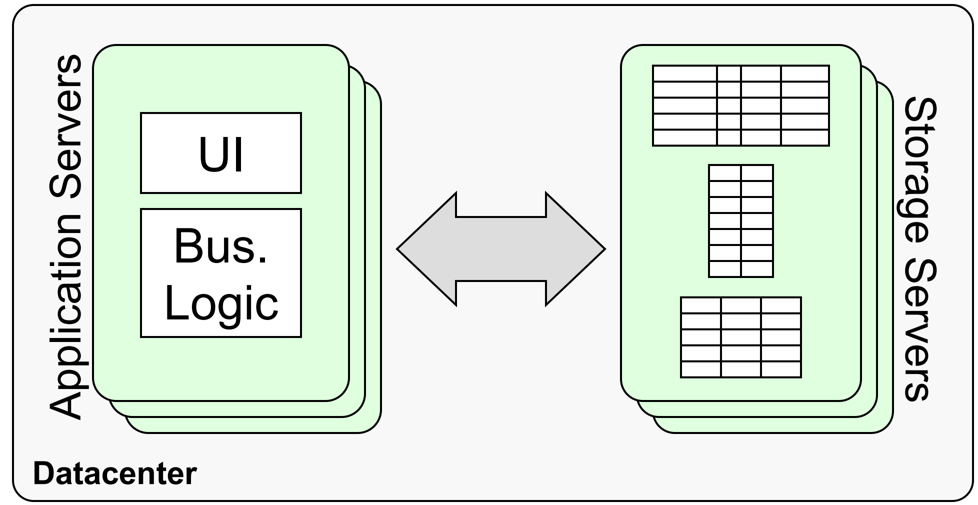
- Large-scale apps struggle with high latency
  - Facebook: can only make 100-150 internal requests per page

Motivation: Latency

Traditional Application



Web Application



<< 1µs latency

~~0.5-10ms~~ latency
5-10µs

- RAMCloud goal: large scale and low latency
- Enable a new breed of information-intensive applications

Motivation: Scalability

Relational databases don't scale

Every large-scale Web application has problems:

- Facebook: 4000 MySQL instances + 2000 memcached servers

New forms of storage starting to appear:

- Bigtable
- Dynamo
- PNUTS
- Sinfonia
- H-store
- memcached

Motivation: Technology

Disk access rate not keeping up with capacity:

	Mid-1980's	2009	Change
Disk capacity	30 MB	500 GB	16667x
Max. transfer rate	2 MB/s	100 MB/s	50x
Latency (seek & rotate)	20 ms	10 ms	2x
Capacity/bandwidth (large blocks)	15 s	5000 s	333x
Capacity/bandwidth (1KB blocks)	600 s	58 days	8333x

Disks must become more archival

More information must move to memory

Why Not a Caching Approach?

Lost performance:

- 1% misses → 10x performance degradation

Won’t save much money:

- Already have to keep information in memory
- Example: Facebook caches ~75% of data size

Changes disk management issues:

- Optimize for reads, vs. writes & recovery

Why not Flash Memory?

Many candidate technologies besides DRAM

- Flash (NAND, NOR)
- PC RAM
- …

DRAM enables lowest latency:

- 5-10x faster than flash

Most RAMCloud techniques will apply to other technologies

Ultimately, choose storage technology based on cost, performance, energy, not volatility

Is RAMCloud Capacity Sufficient?

Facebook: 200 TB of (non-image) data today

Amazon:

Revenues/year:	$16B
Orders/year:	400M? ($40/order?)
Bytes/order:	1000-10000?
Order data/year:	0.4-4.0 TB?
RAMCloud cost:	$24-240K?

United Airlines:

Total flights/day:	4000? (30,000 for all airlines in U.S.)
Passenger flights/year:	200M?
Bytes/passenger-flight:	1000-10000?
Order data/year:	0.2-2.0 TB?
RAMCloud cost:	$13-130K?

Ready today for all online data; media soon

RAMCloud Overview

Data Durability and Logging

Data Recovery

RAMCloud Overview

- Data Durability and Logging

- Data Recovery

Data Durability

Want durability with single copy in RAM

Use disks for replicated backup copies (cheap, non-volatile)

Natural to use a logging approach

Exploit high sequential I/O bandwidth

Avoid high access latencies

Scatter backups across cluster

Fast recovery

High burst write bandwidth

Next part will discuss recovery from durable storage

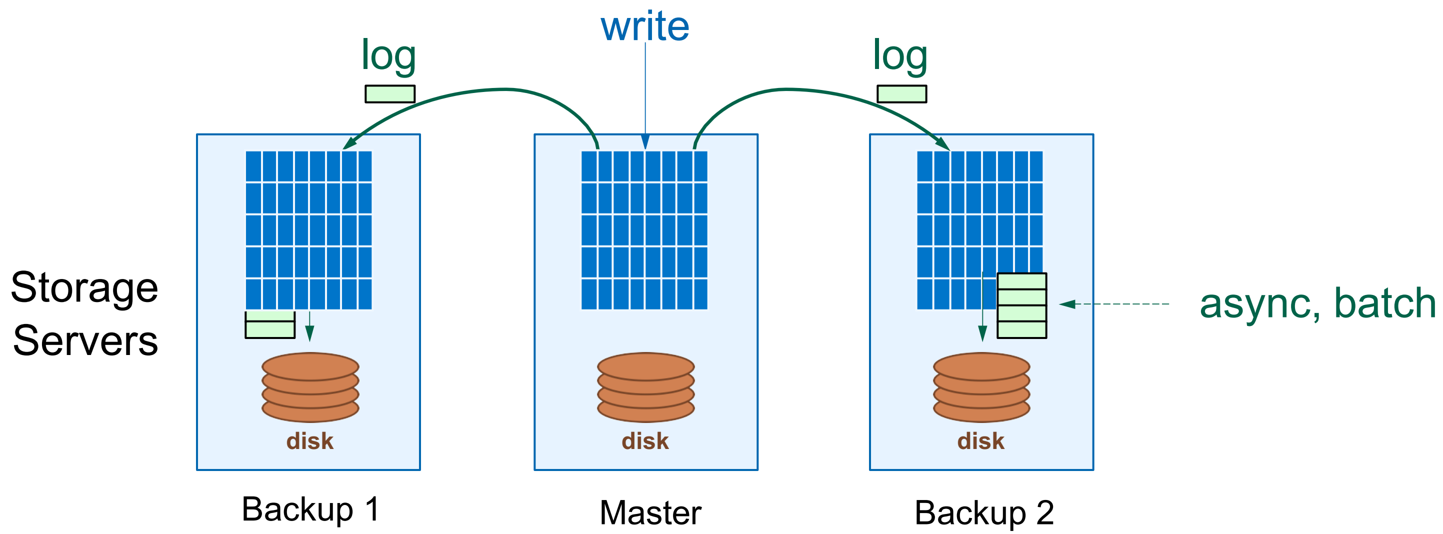
Data Durability

- **We need durability**
  - Servers will fail
  - The power will go out
  - Failures will be frequent
    - System always in recovery?

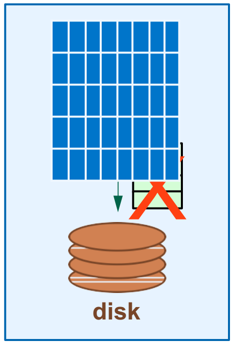
- **We need to replicate main memory contents**
  - RAM is not feasible
    - Highest performance, but:
      - Assumes we can keep all RAM powered at all times
      - Too expensive: increases cost per bit by replication factor
  - Local disks are not feasible
    - Require synchronous writes (latency much too high!)
    - Too slow to recover (single spindle)
    - What if the box dies?

Buffered Logging

- **Each server maintains a log of updates to its objects**
  - We call the owner server a "*master*"
- **Masters' log updates are sent to *R* backups**
  - RPCs return when backups have updates buffered in RAM
  - Backups batch and write to disk asynchronously
  - Assume for now each master always uses same *R* backups
- **Each master is also a backup for other servers**



Backup Buffer Volatility



- **Problem: Backup buffers are in volatile DRAM**
  - Vulnerable until disk flush
- **Solution 1: Synchronous disk writes**
  - 2,000 - 8,000 microsecond latency!
  - No write batching => very low bandwidth (< 1% of sequential I/O)
- **Solution 2: Synchronous flash SSD writes**
  - ~50 - 100 microsecond latency, still a big non-sequential I/O penalty
- **Solution 3: Reduce consistency guarantees**
  - "Sorry, we lost your data. Deal with it."

Solution 4: Battery Backups

Batteries provide enough power to flush buffers

Log-Structured Backups

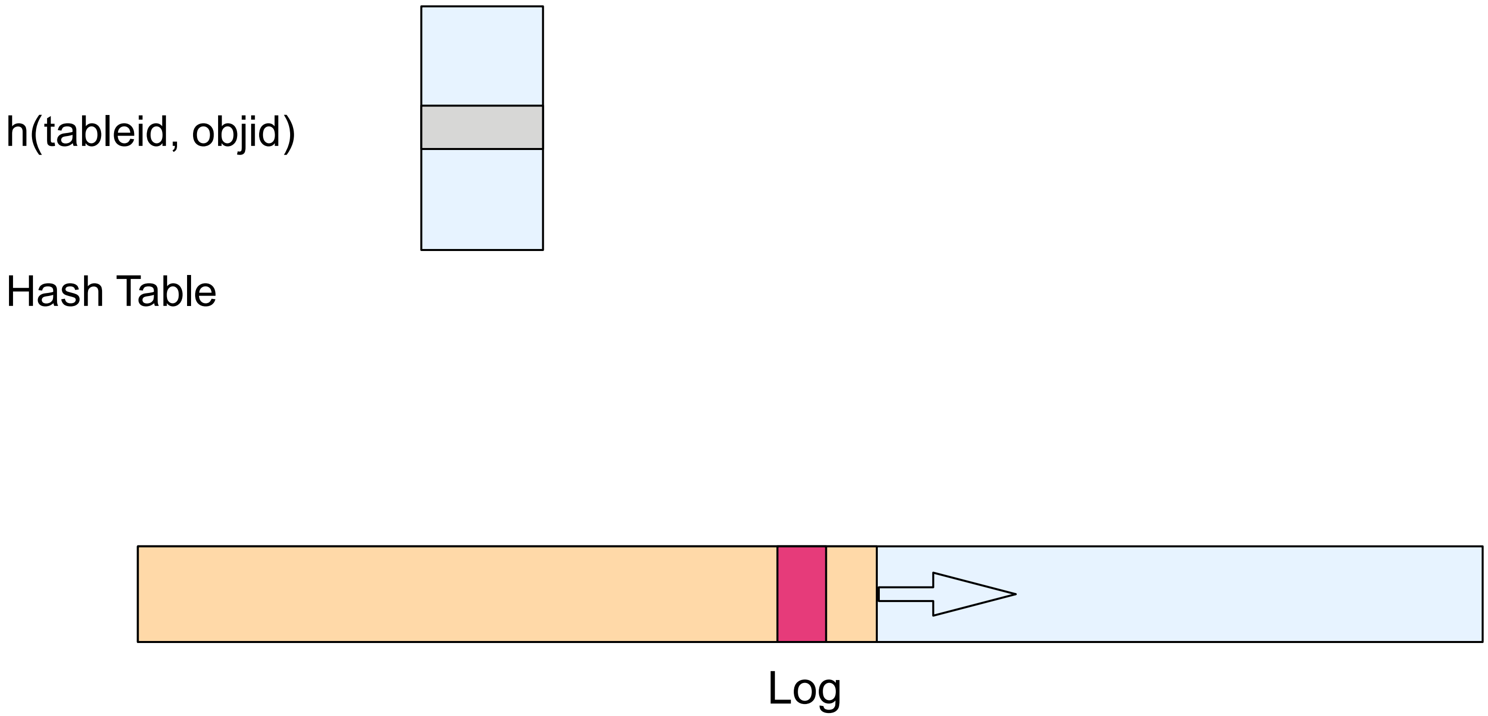
- **Problem: Need fast write rates, but have disks**
  - RAMCloud is about performance, after all

- **Solution: Log-structure on disks**
  - Exploits sequential I/O
  - But we need to do cleaning

- **What about log cleaning overheads?**
  - All data is in RAM, so no need to re-read for cleaning
    - 50% of the overhead immediately out the window
  - Don't need to use disks efficiently
    - Worry about RAM utilization
    - Assume backups have capacity to spare.

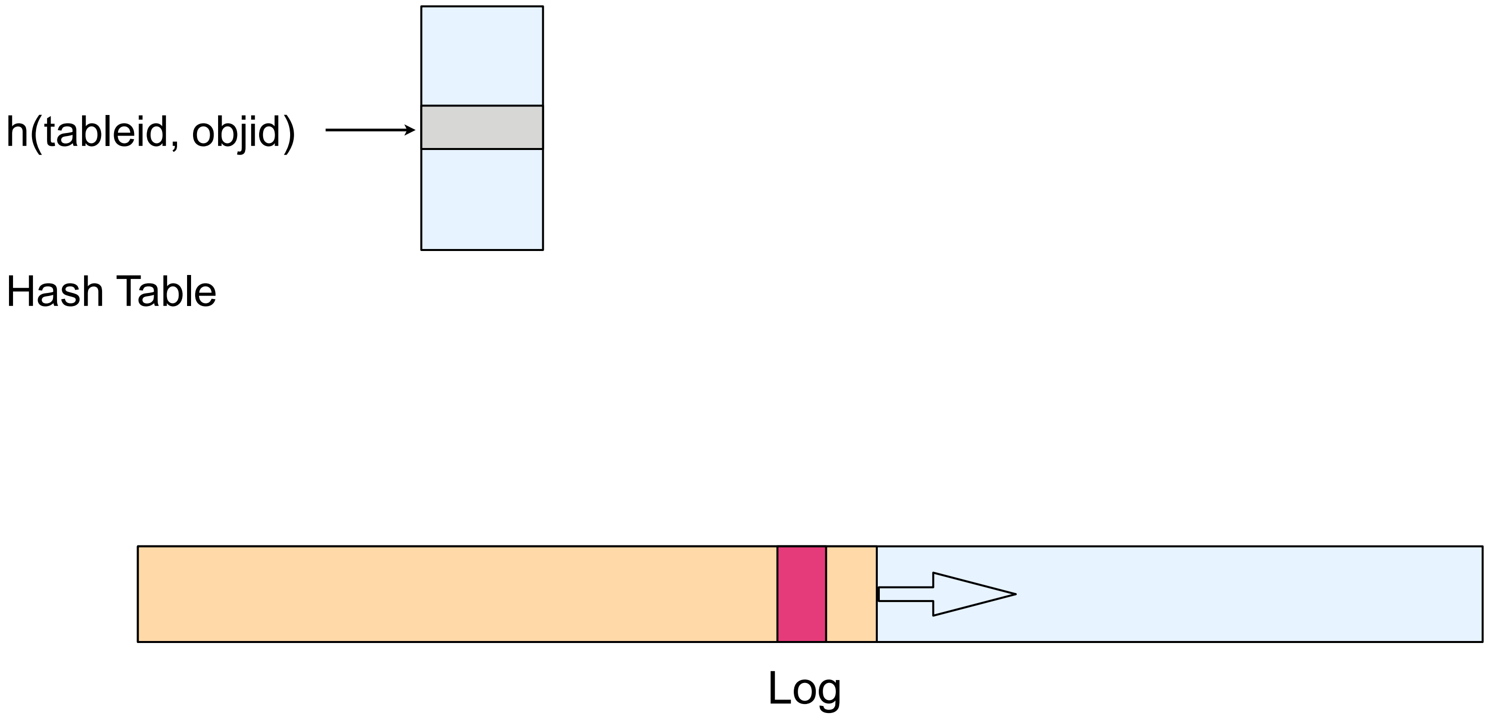
Locating Objects in the Log

- **How do we find objects in the main-memory Log?**
  - Hash table lookup
  - Two cache misses from (tableId, objectId) to object
    - Extremely fast, despite complex memory management scheme



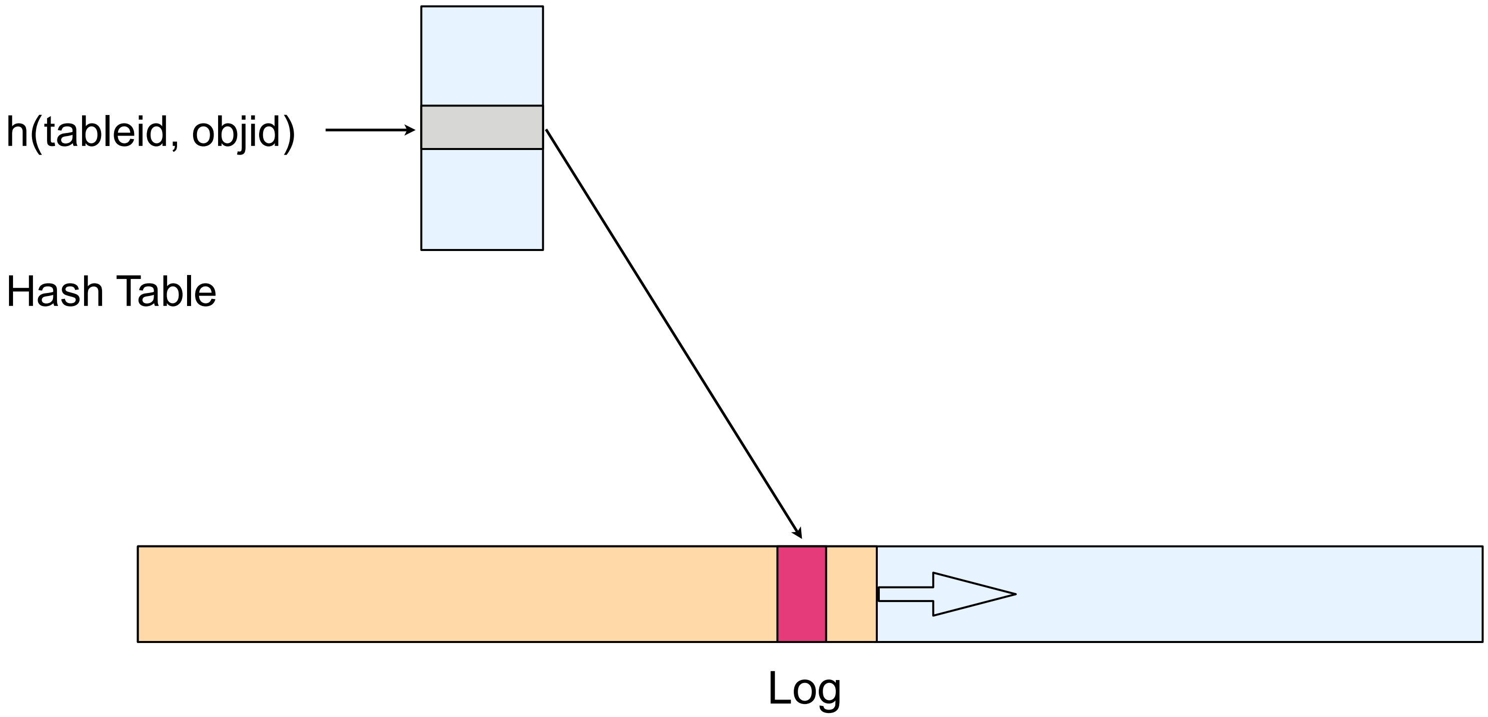
Locating Objects in the Log

- **How do we find objects in the main-memory Log?**
  - Hash table lookup
  - Two cache misses from (tableId, objectId) to object
    - Extremely fast, despite complex memory management scheme

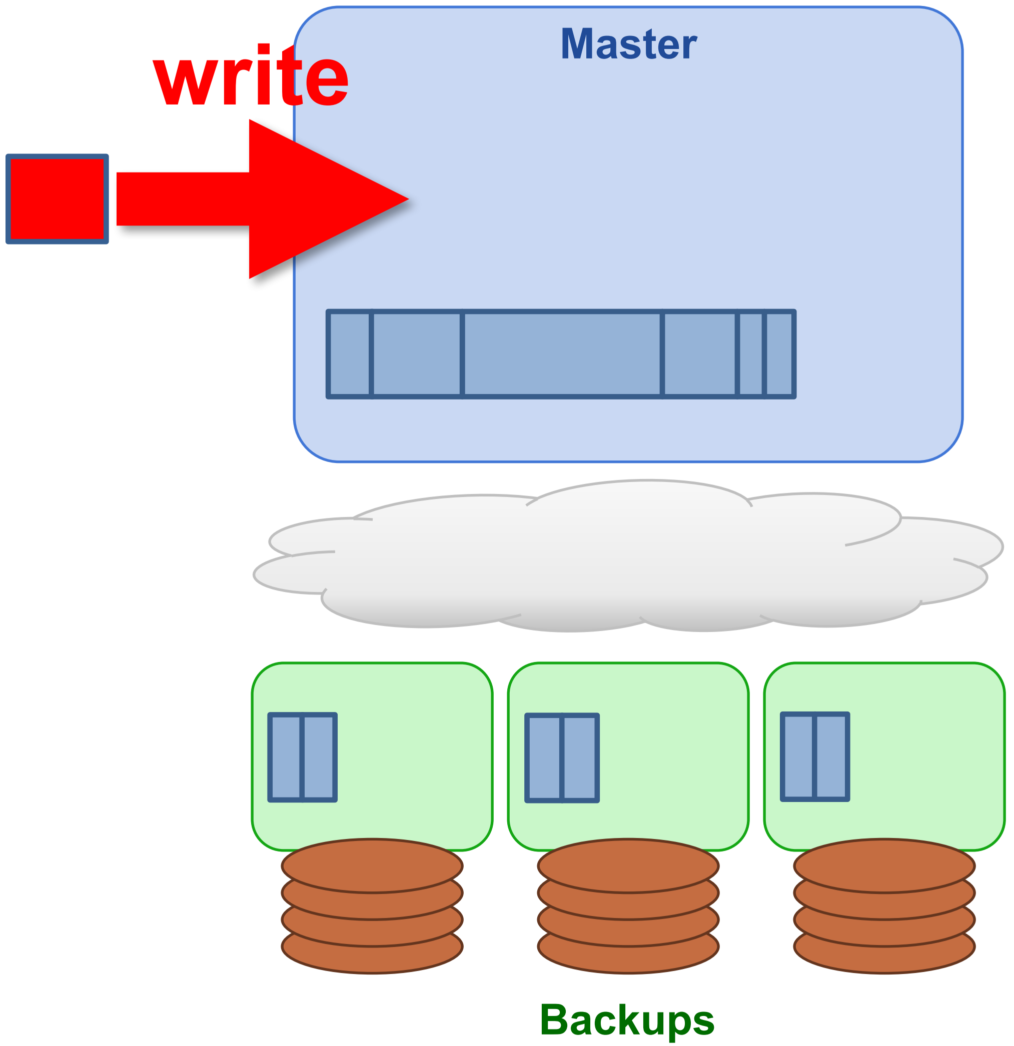


Locating Objects in the Log

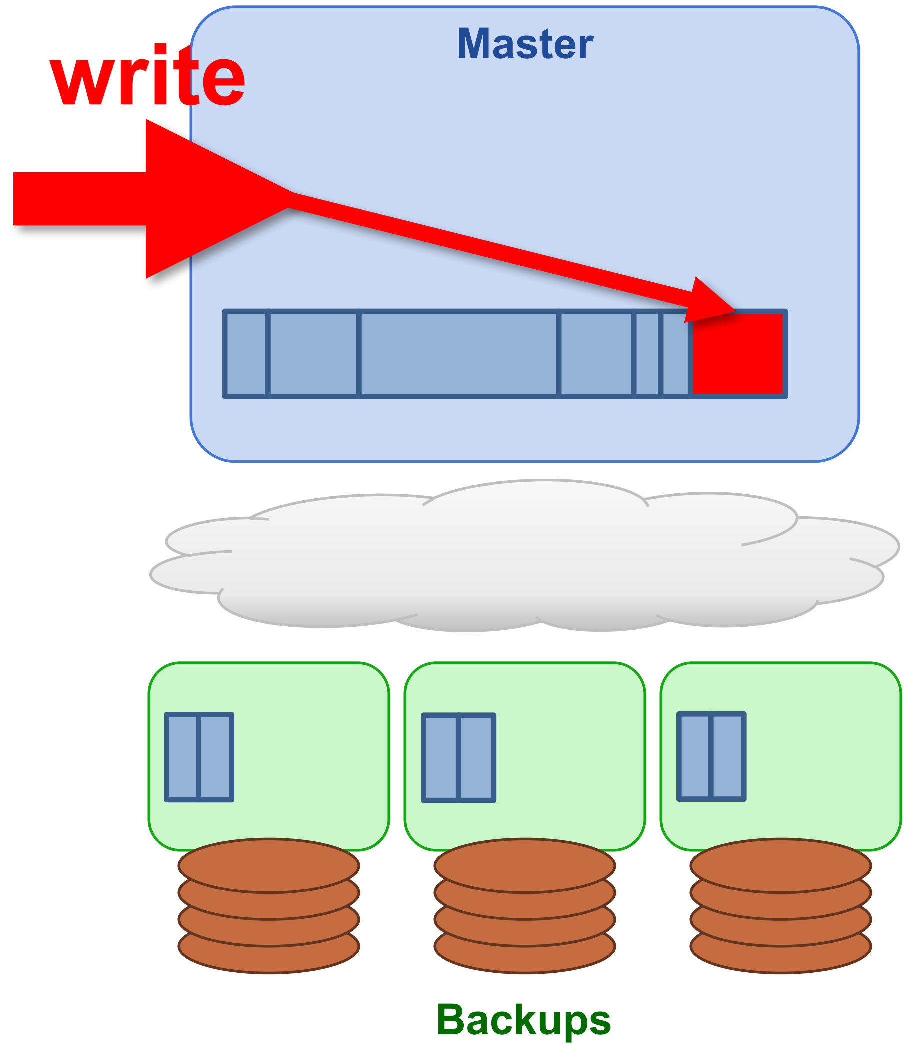
- **How do we find objects in the main-memory Log?**
  - Hash table lookup
  - Two cache misses from (tableId, objectId) to object
    - Extremely fast, despite complex memory management scheme



Durability with RAM performance

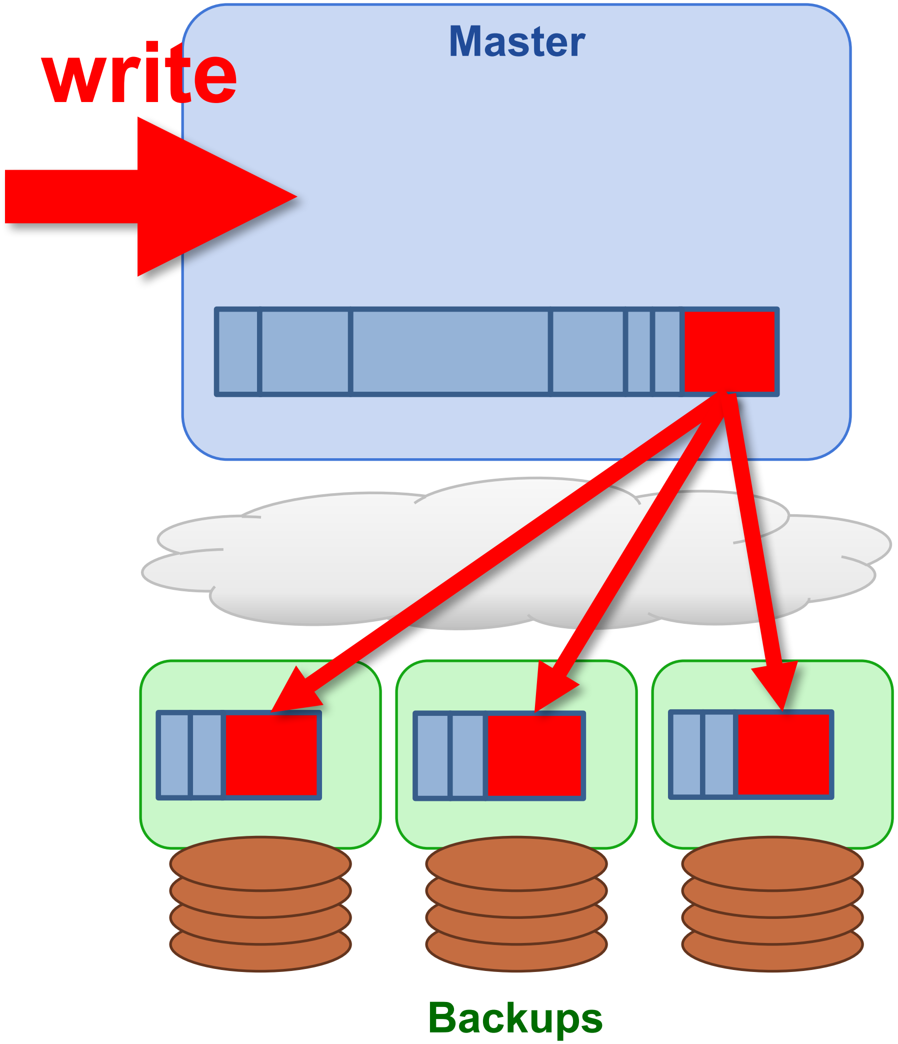


Durability with RAM performance



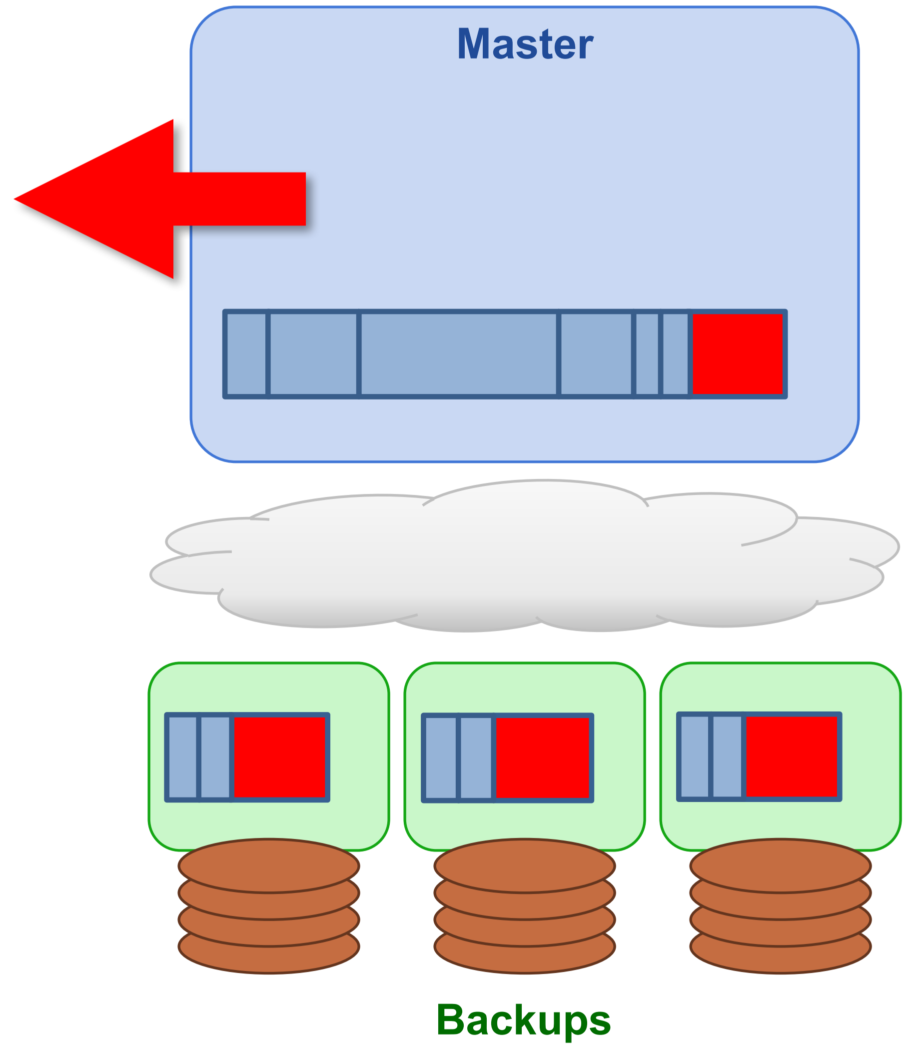
Durability with RAM performance

- Backups buffer update
  - No synchronous disk write



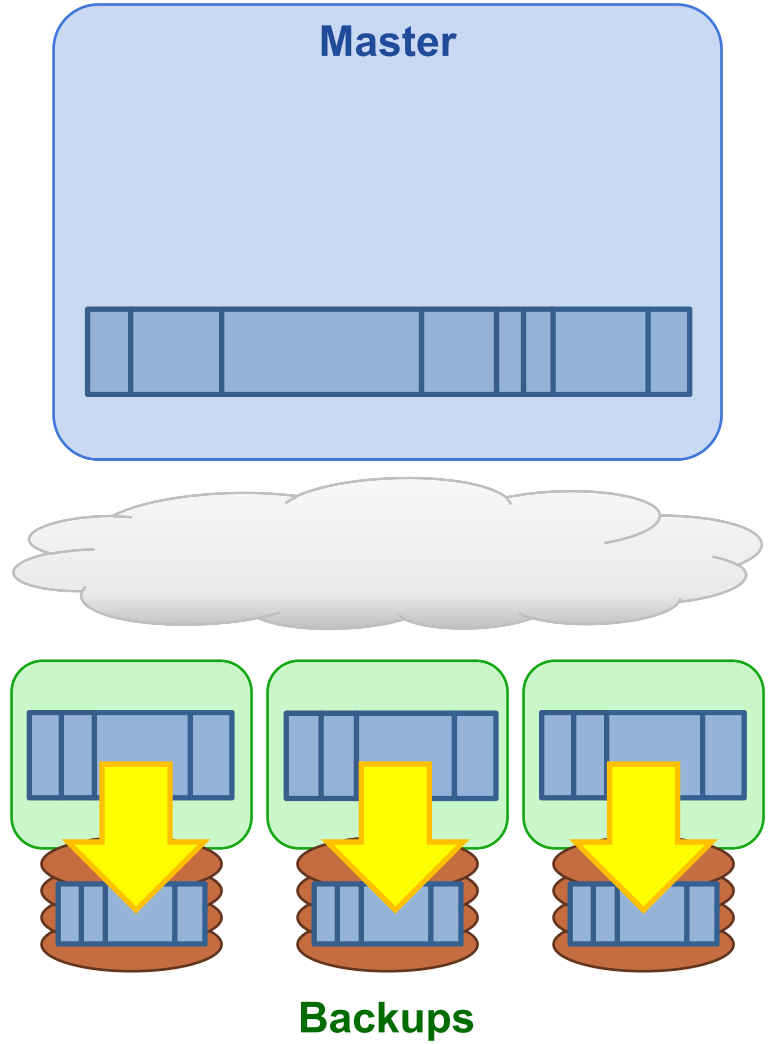
Durability with RAM performance

- Backups buffer update
  - No synchronous disk write



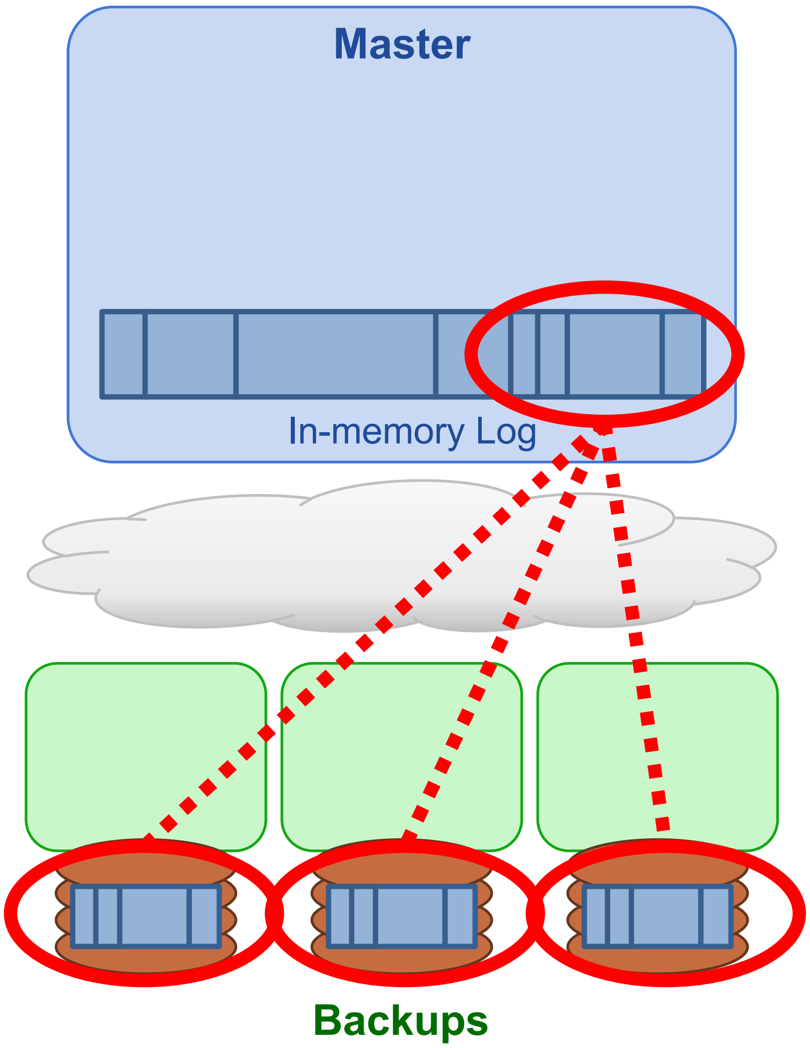
Durability with RAM performance

- Backups buffer update
  - No synchronous disk write
- Bulk writes in background
  - Must flush on power loss



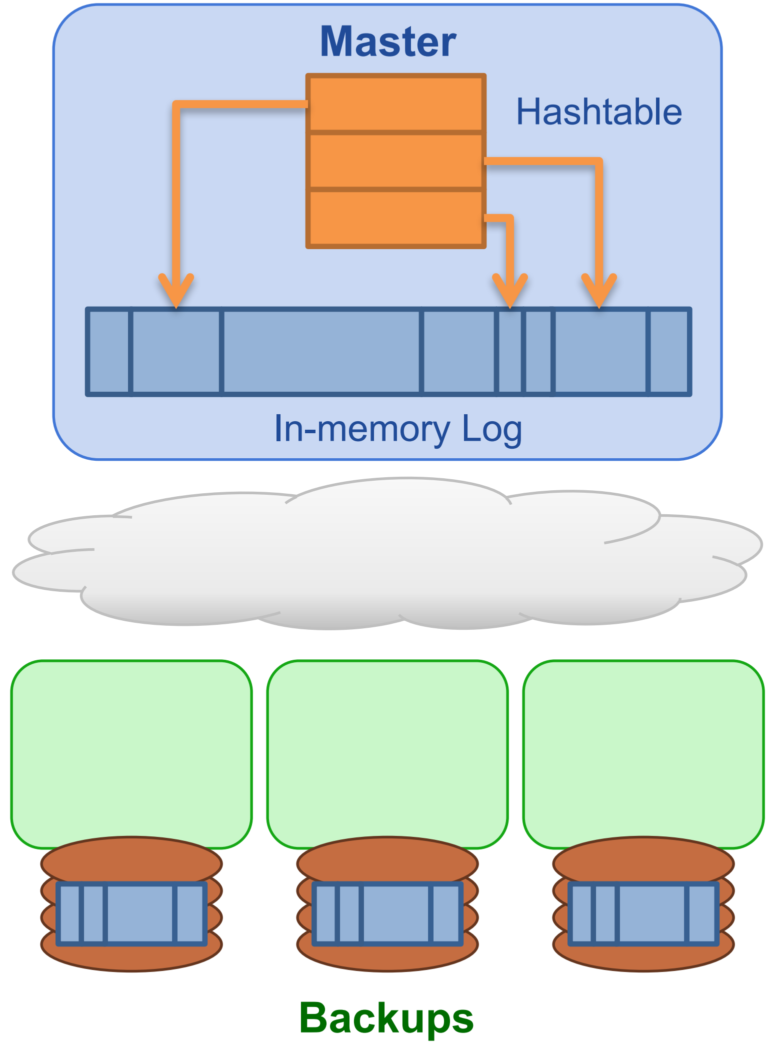
Durability with RAM performance

- Backups buffer update
  - No synchronous disk write
- Bulk writes in background
  - Must flush on power loss
- Pervasive log structure
  - Even RAM is a log
  - Log cleaner



Durability with RAM performance

- Backups buffer update
  - No synchronous disk write
- Bulk writes in background
  - Must flush on power loss
- Pervasive log structure
  - Even RAM is a log
  - Log cleaner
- Hashtable, key → location



Data Durability Summary

- Durability with one copy in RAM
- Logging approach for disk I/O utilization
- Backups scattered across cluster for recovery and bursty load

Outline

- Data Durability and Logging
- Data Recovery

Data Recovery Overview

- **Master Recovery**
  - 2-Phase
  - Partitioned

- **Failures**
  - Backups
  - Rack/Switch
  - Power
  - Datacenter

Implications of Single Copy in Memory

- **Problem: Unavailability**
  - If master crashes unavailable until read from disks on backups
  - Read 64 GB from one disk? **10 minutes**

- **Use scale to get low-latency recovery**
  - Lots of disk heads, NICs, CPUs
  - **Our goal**: recover in **1-2 seconds**
    - Is this good enough?
  - Applications just see “hiccups”

Fast Recovery

- **Problem: Disk bottleneck for recovery**

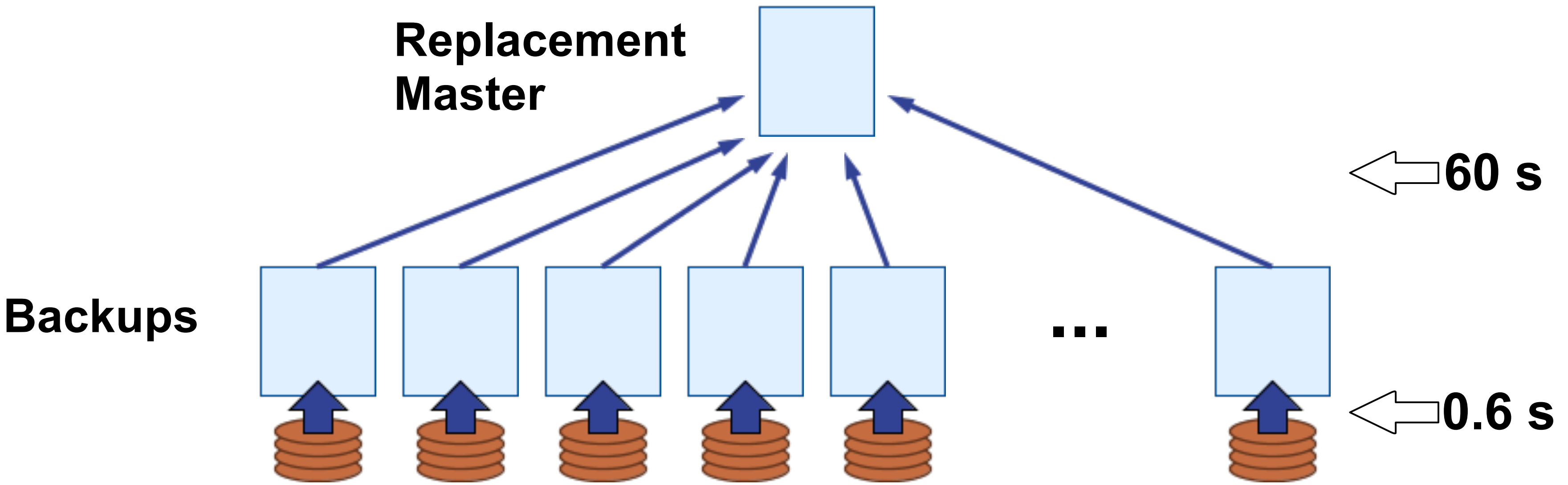
- **Idea: Leverage many spindles to recover quickly**
  - Data broadly scattered throughout backups
    - Not just great write throughput
    - Take advantage of read throughput

- **Reincarnate masters**
  - With same tables
  - With same indexes
  - Preserves locality

Fast Recovery: The Problem

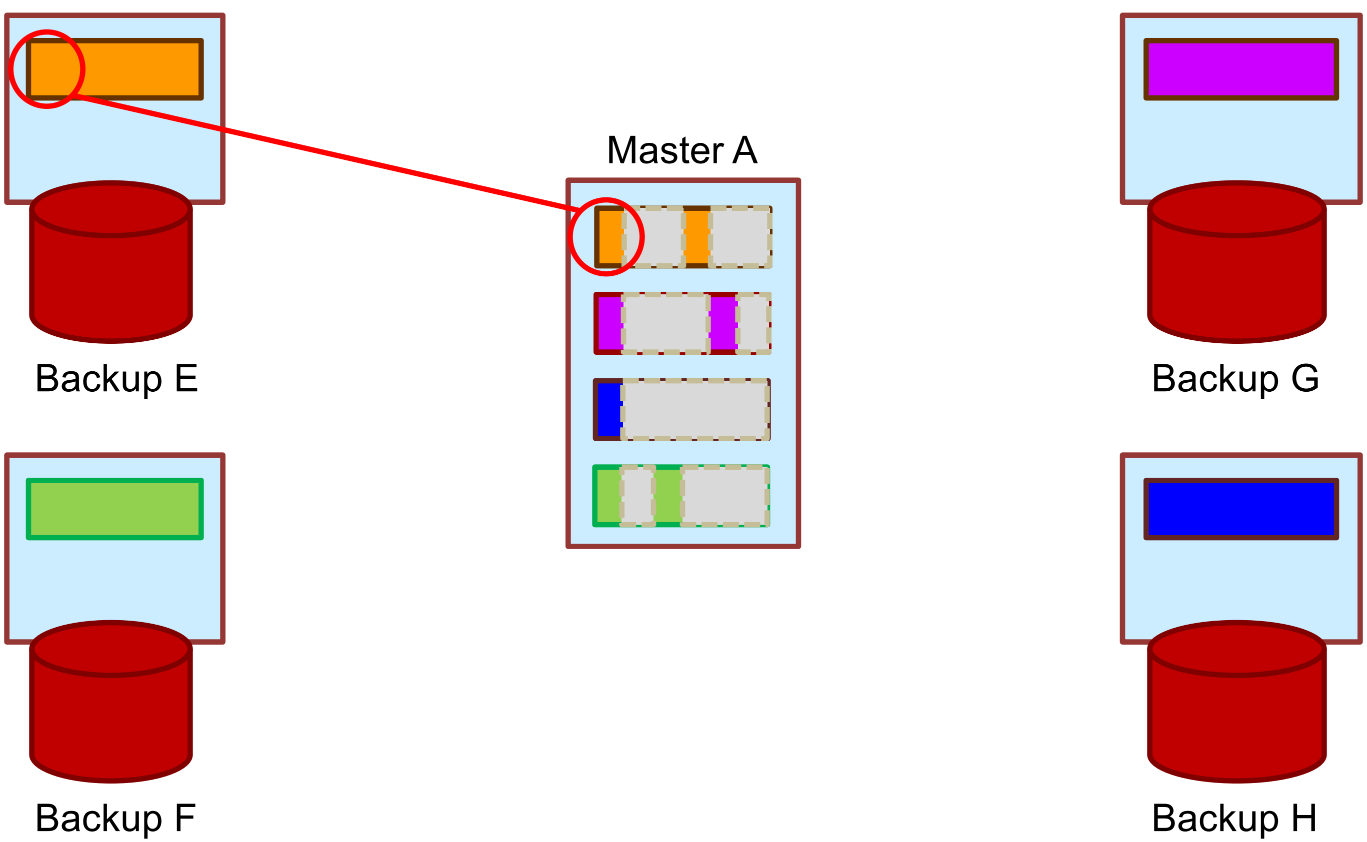
- **After crash, all backups read disks in parallel**
  (64 GB/1000 backups @ 100 MB/sec = **0.6 sec, great!**)

- **Collect all backup data on replacement master**
  (64 GB/10Gbit/sec ~ **60 sec: too slow!**)
  Problem: Network is now the bottleneck!



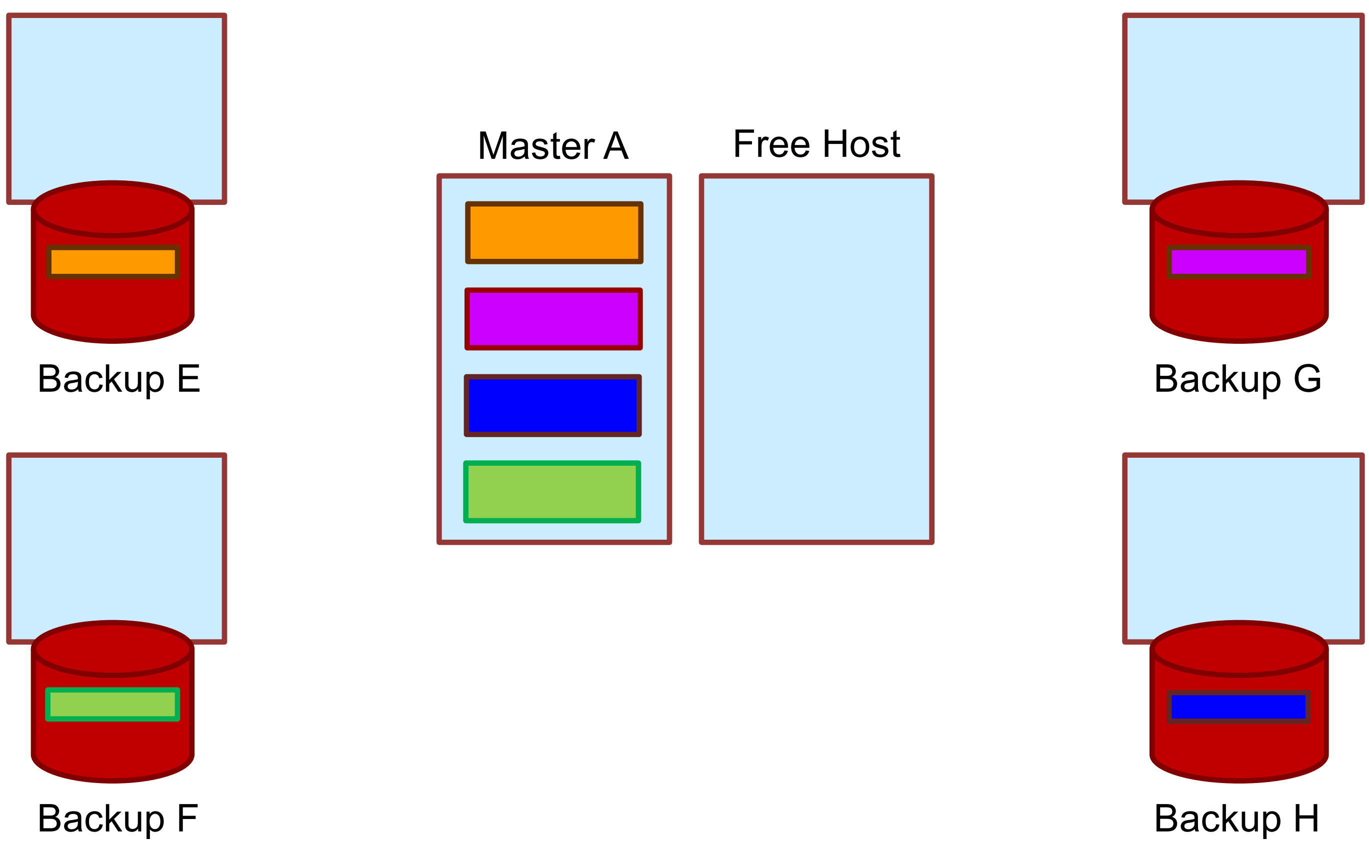
2-Phase Recovery

- **Idea: Data already in memory on backups, just need to know where**



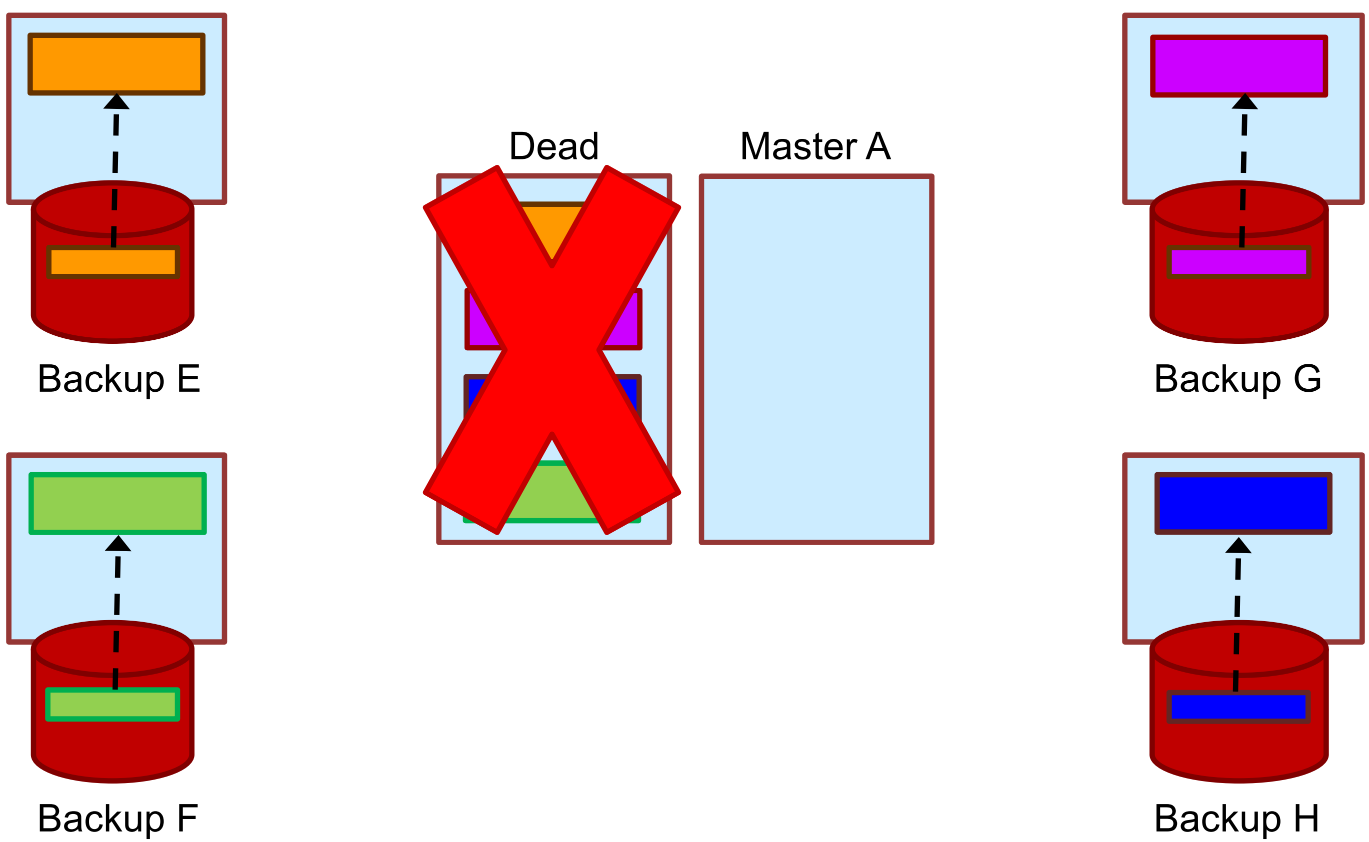
2-Phase Recovery

- **Phase #1: Recover location info (< 1s)**



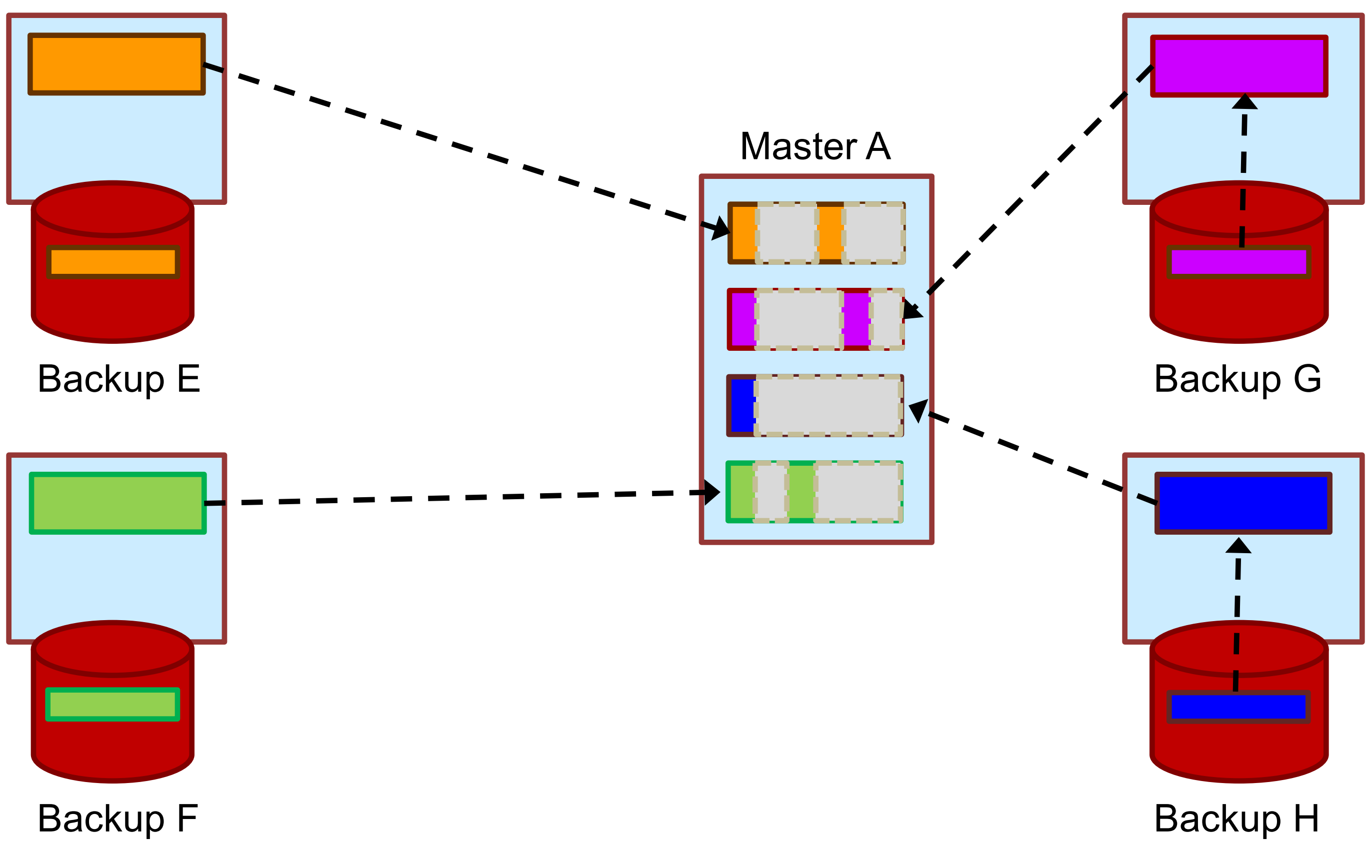
2-Phase Recovery

- **Phase #1: Recover location info (< 1s)**
  - Read all data into memories of backups



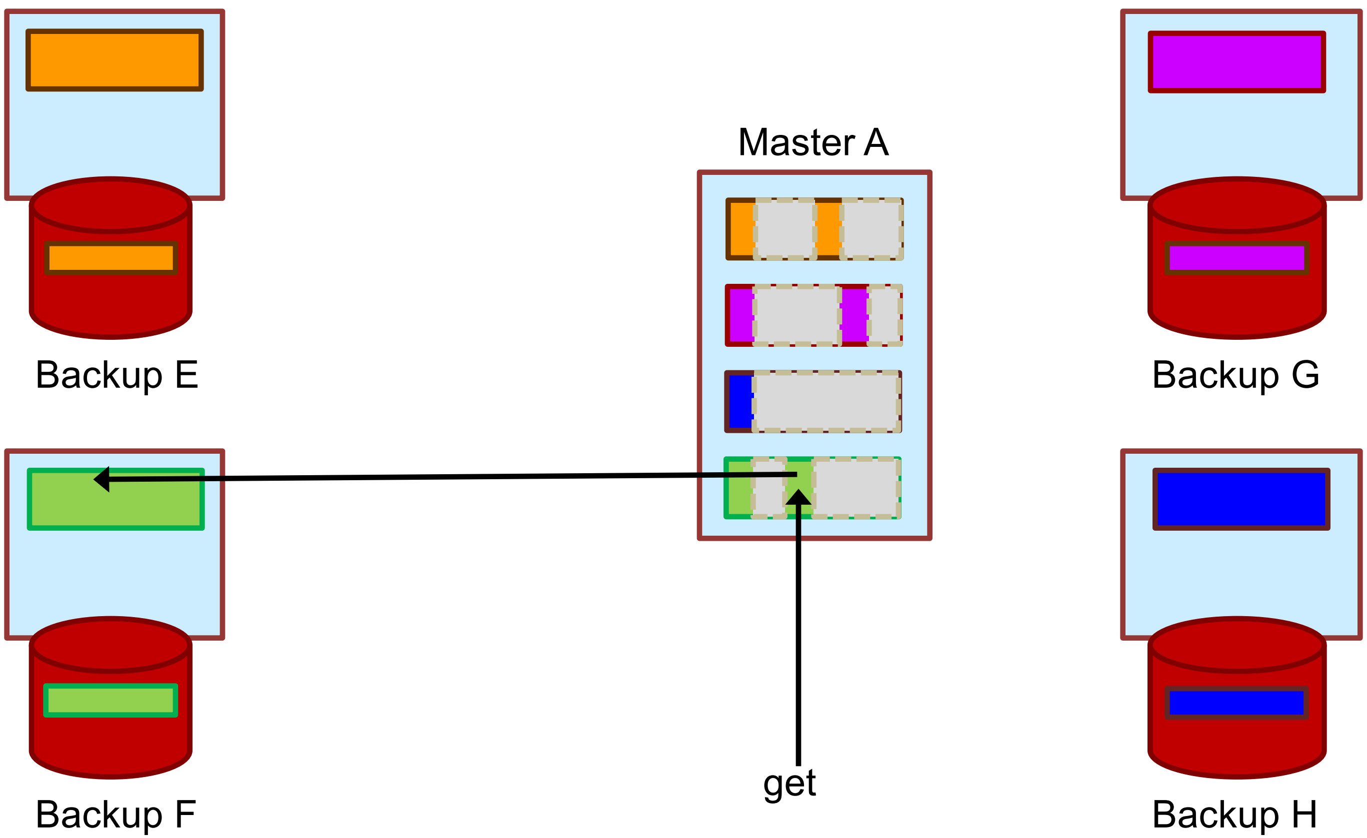
2-Phase Recovery

- **Phase #1: Recover location info (< 1s)**
  - Read all data into memories of backups
  - Send only location info to replacement master



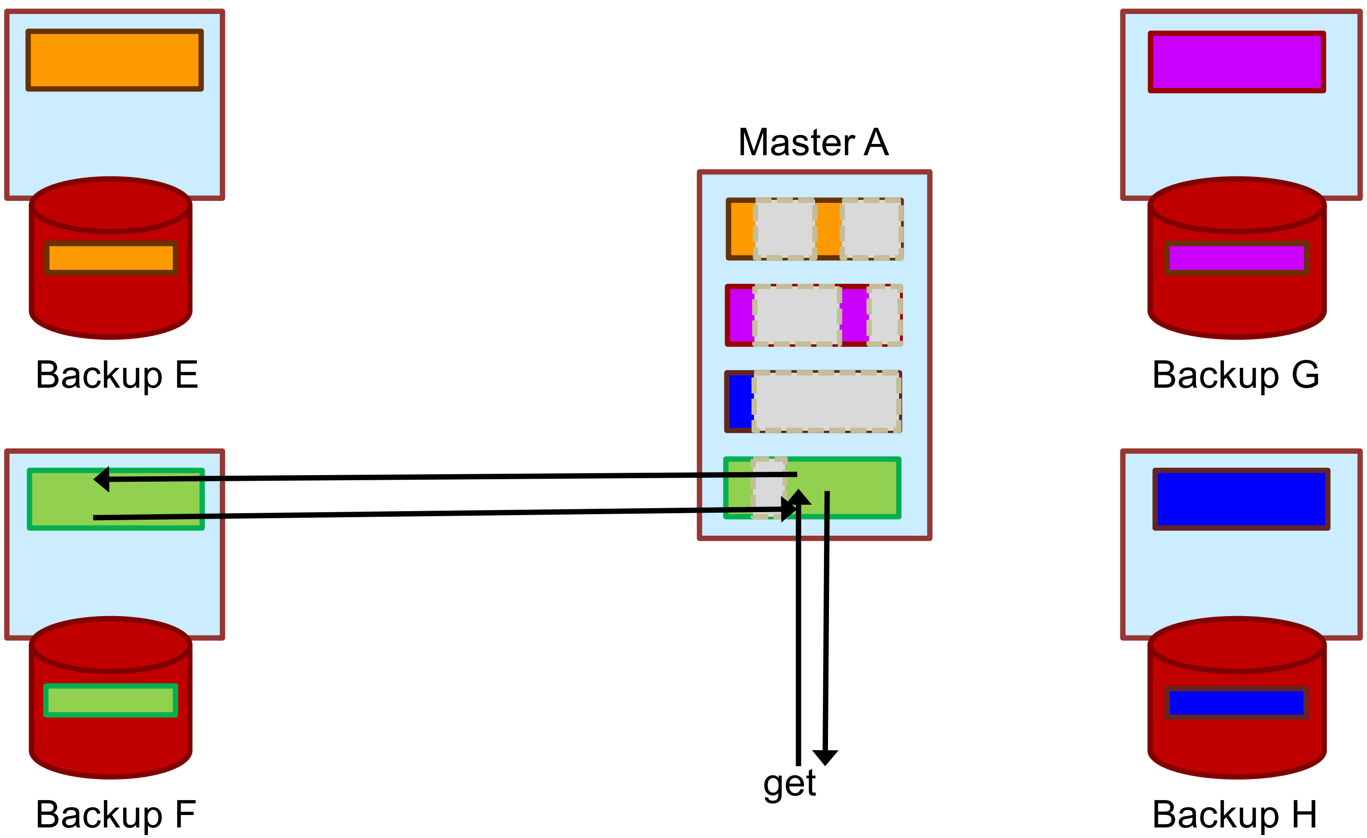
2-Phase Recovery

- **Phase #2: Proxy & Recover Full Data (~60s)**
  - **System resumes operation:**
    - Fetch on demand from backups
    - 1 extra round trip on first read of an object
    - Writes are full speed



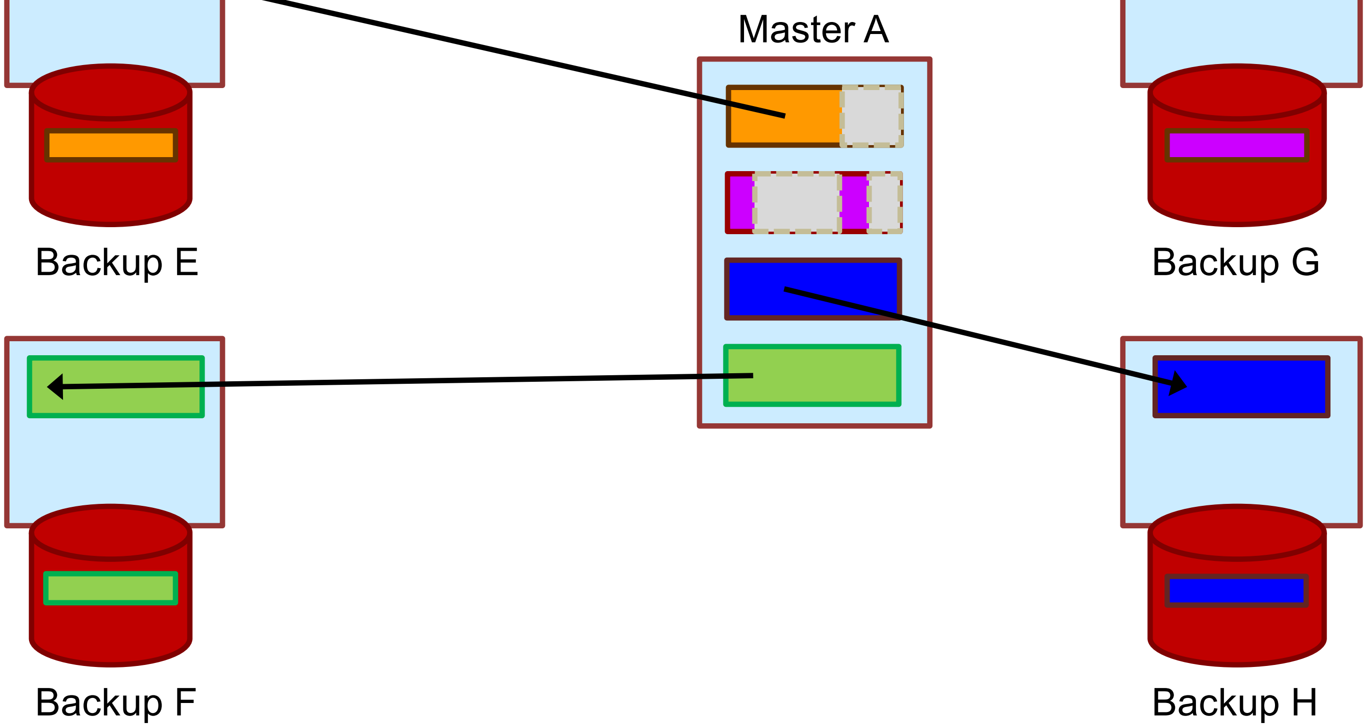
2-Phase Recovery

- **Phase #2: Proxy & Recover Full Data (~60s)**
  - **System resumes operation:**
    - Fetch on demand from backups
    - 1 extra round trip on first read of an object
    - Writes are full speed



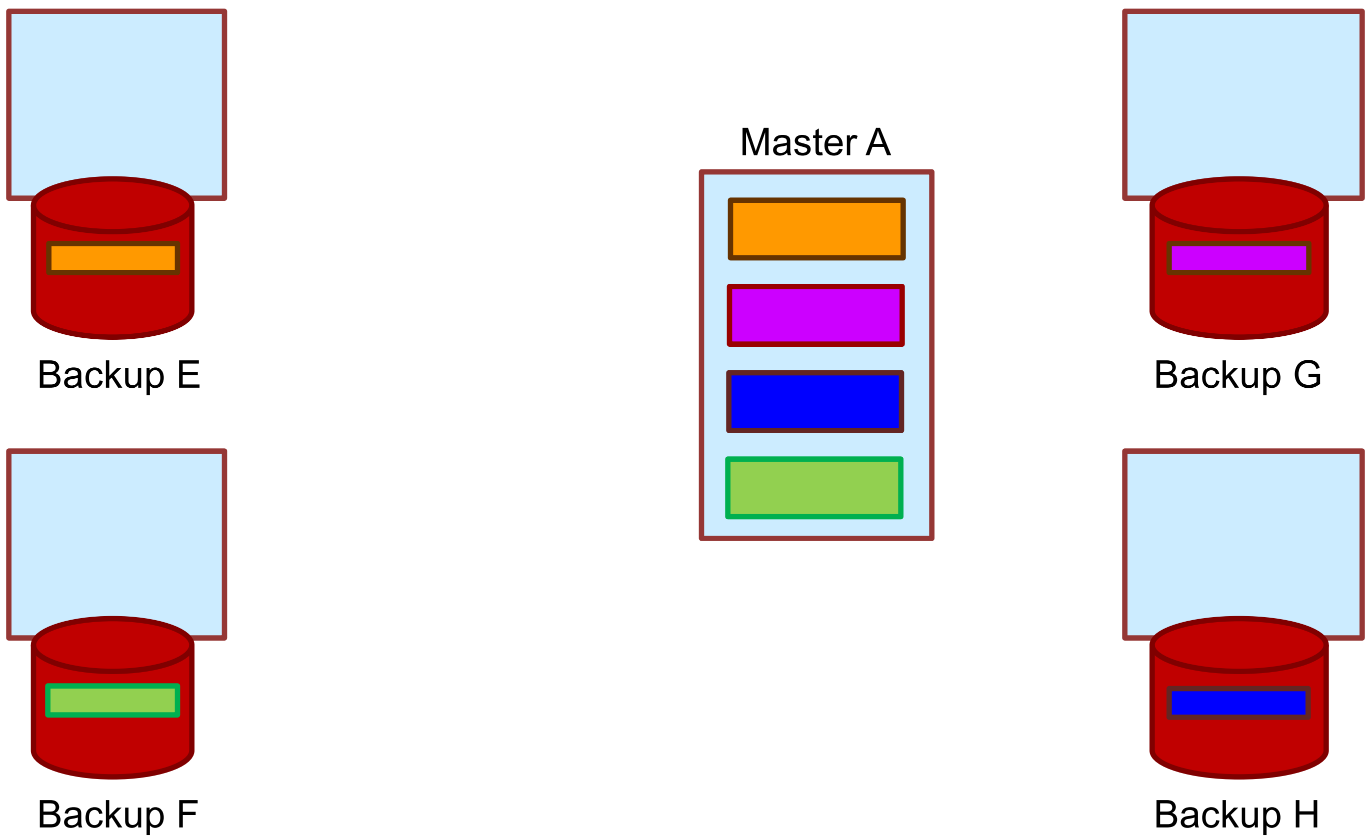
2-Phase Recovery

- **Phase #2: Proxy & Recover Full Data (~60s)**
  - Transfer data from backups in between servicing requests



2-Phase Recovery

- **Performance normal after Phase #2 completes**



2-Phase Recovery: Thoughts

- ✓ **Recovers locality by recovering machines**
- ✗ **Need to talk to all hosts**
  - Because backup data for a single master is on all machines
  - How bad is this?
- ✗ **Doesn't deal with heterogeneity**
  - Machine is the unit of recovery
  - Can only recover a machine to one with more capacity
- **Bi-modal Utilization**
  - Must retain pool of empty hosts

2-Phase Recovery: Problem

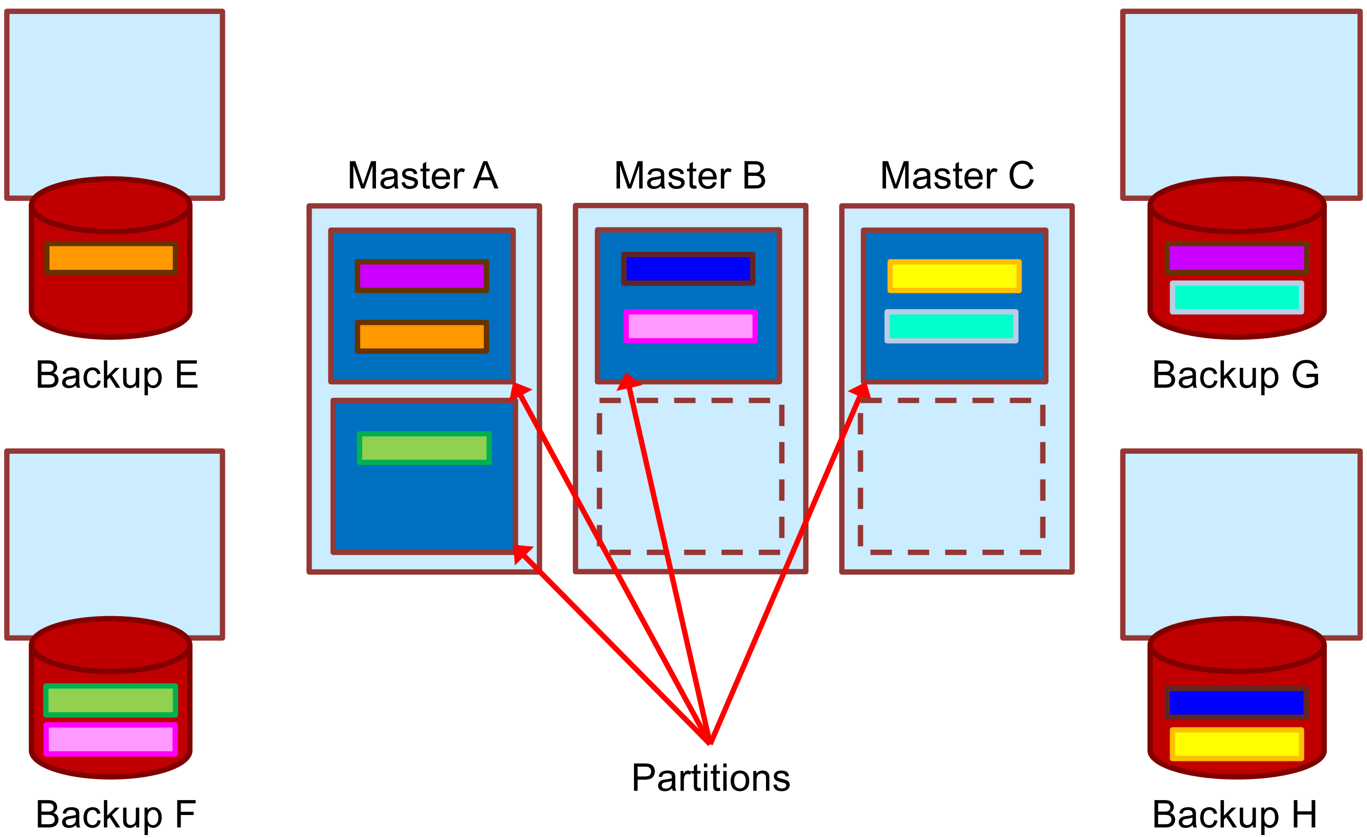
- **Hashtable inserts become the new bottleneck**
  - Phase #1 must place all objects in the hashtable
  - Master can have 64 million 1 KB objects
  - Hashtable can sustain about 10 million inserts/s
  - 6.4s is over our budget
  - Can use additional cores, but objects could be even smaller

- **Unsure of a way to recover exact master quickly**
  - Constrained by both CPU and NIC
  - Recovery to single host is a bottleneck

- **Another way?**

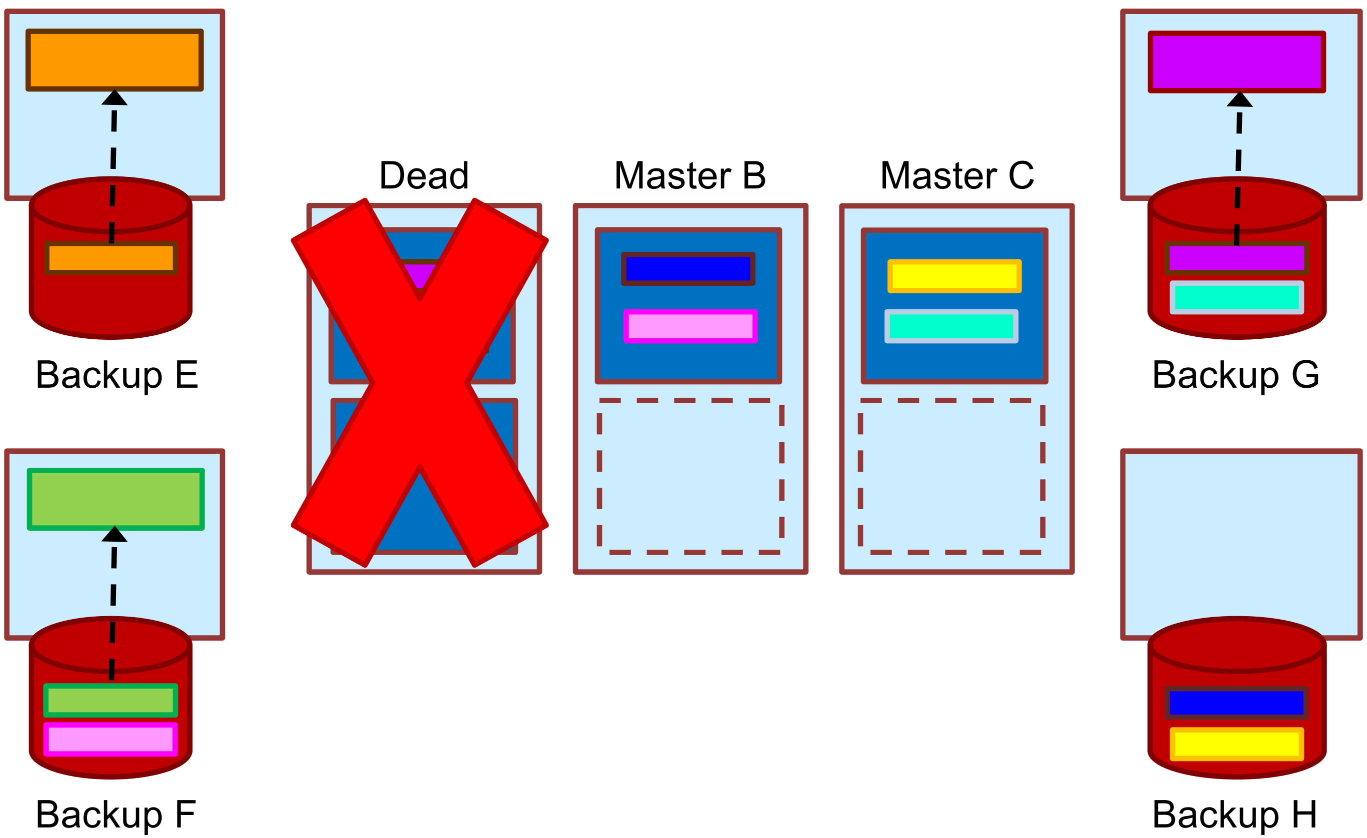
Partitioned Recovery

- **Idea: Leverage many hosts to overcome bottleneck**
  - Problem is machines are large so divide them into partitions
  - Recover each partition to a different master
  - Just like a machine
    - Contains any number of tables, table fragments, indexes, etc.



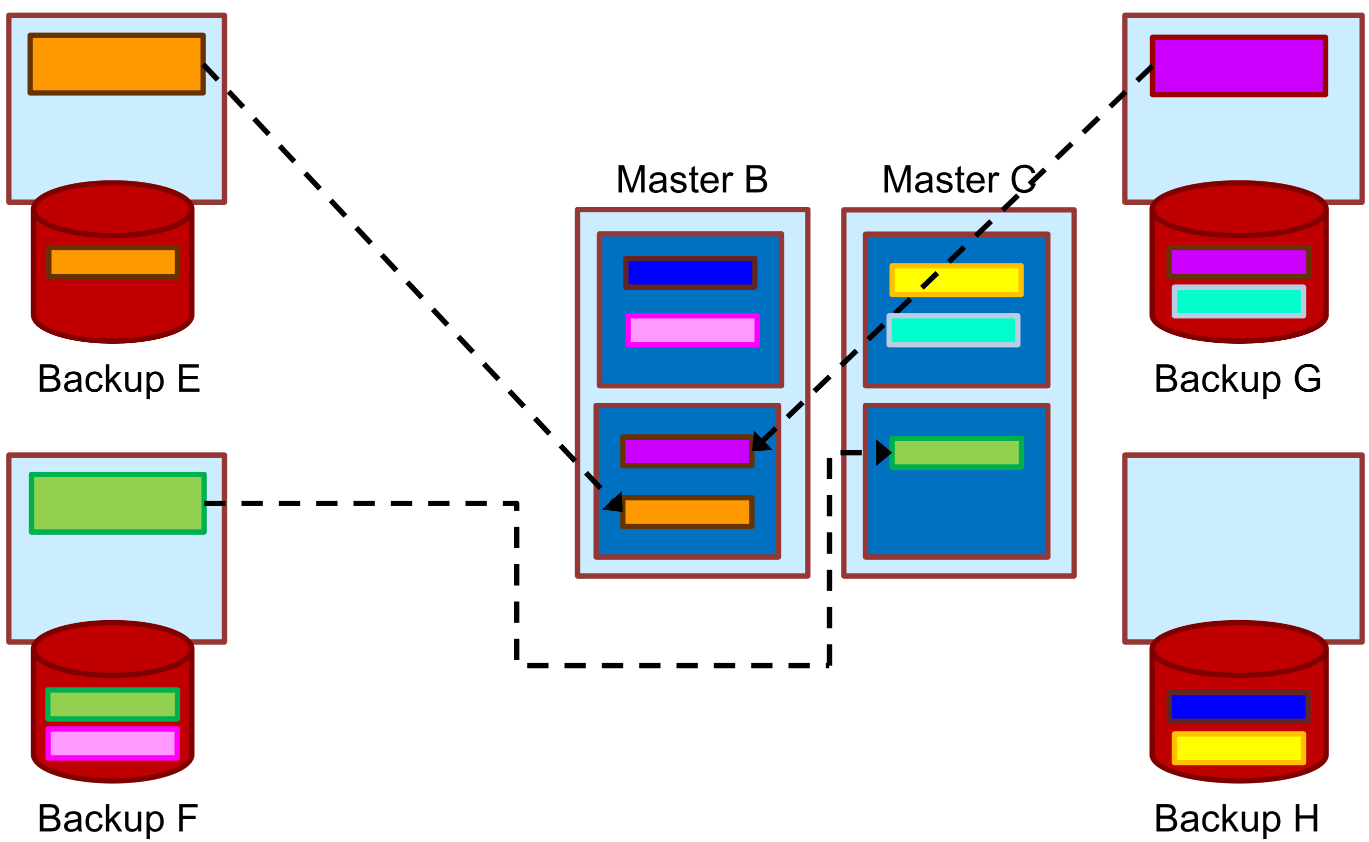
Partitioned Recovery

- **Load data from disks**



Partitioned Recovery

- Reconstitute partitions on many hosts
- 64 GB / 100 partitions = 640 MB
- 640 MB / 10 GBit/s = 0.6s for full recovery

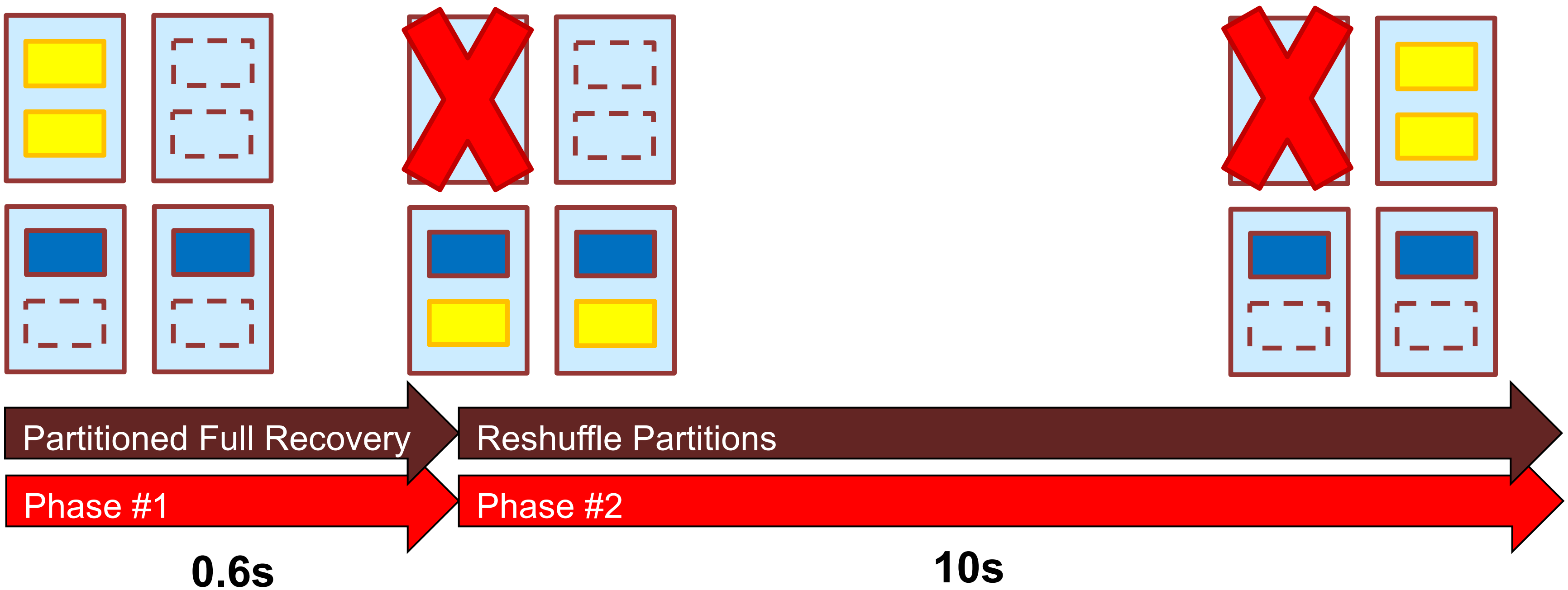


Partitioned Recovery: Thoughts

- ✓ **It works: meets availability goals**
  - Can tune time by adjusting partition size
- ✓ **Helps with heterogeneity**
  - Unit of recovery is no longer a machine
- ✘ **Increases host/partition related metadata**
  - Partitions still large enough to provide app locality
- ✘ **Need to talk to all hosts**
- **No hot spares**

Partitioned Recovery: Thoughts

- **Does not recover locality**
  - But, no worse than 2-Phase
  - Partitioned approach recovers as fast as Phase #1
  - Can restore locality as fast as Phase #2



Failures: Backups

- **On backup failure the coordinator broadcasts**
  - More information during coordinator discussion
- **All masters check their live data**
- **If objects were on that backup rewrite elsewhere (from RAM)**

- **No recovery of backups themselves**

Failures: Racks/Switches

- **Careful selection of backup locations**
  - Write backups for objects to other racks
    - As each other
    - As the master
  - Changes as masters recover
    - Can move between racks
  - Masters fix this on recovery

- **Rack failures handled the same as machine failures**
  - Consider all the machines in the rack dead

- **Question: Minimum RAMCloud that can sustain an entire rack failure and meet recovery goal?**
  - 100 partitions to recover a single machine in 0.6s
  - 50 dead * 100 partitions, need 5000 machines to make 0.6s
  - Don’t pack storage servers in racks, mix with app servers

Failures: Power

- **Problem: Objects are buffered temporarily in RAM**
  - Even after the put has returned as successful to the application
- **Solution: All hosts have on-board battery backup**
- **Flush all dirty objects on fluctuation**
  - Any battery should be easily sufficient for this
  - Each master has *r* active buffers on backups
  - Buffers are 8MB, for 3 disk copies
    - 3 * 8MB takes 0.24s to flush

- **Question: Cost effective way to get 10-20s of power?**

- **No battery?**
  - Deal with lower consistency
  - Synchronous writes

Failures: Datacenter

- **Durability guaranteed by disks, no availability**
  - Modulo nuclear attacks
- **No cross-DC replication in version 1**
  - Latency can't be reconciled with consistency
  - Aggregate write bandwidth of 1000 host RAMCloud
    - 100 MB/s * 1000 = 1 Tbit/s
- **Application level replication will do much better**
  - Application can batch writes
  - Application understands consistency needs
- **Is this something we need to support?**

Recovery Summary

- **Use scale in two ways to achieve availability**
  - Scatter reads to overcome disk bottleneck
  - Scatter rebuilding to overcome CPU and network bottlenecks
- **Scale driving lower-latency**
- **Remaining Issue: How do we get information we need for recovery?**
  - Every master recovery involves all backups

Conclusion

Interesting combination of scale and latency

Enable more powerful uses of information at scale:

- 1000-10000 clients
- 100TB - 1PB
- 5-10 µs latency