RAMCloud Design Review

Recovery

Ryan Stutsman

April 1, 2010

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**

- **Leverage scale to get low-latency recovery**
  - Lots of disk heads, NICs, CPUs
  - **Our goal**: recover in **1-2 seconds**
    - Is this good enough?

Overview

- **Master Recovery**
  - 2-Phase
  - Sharding
- **Failures**
  - Backups
  - Rack/Switch
  - Datacenter
  - Power

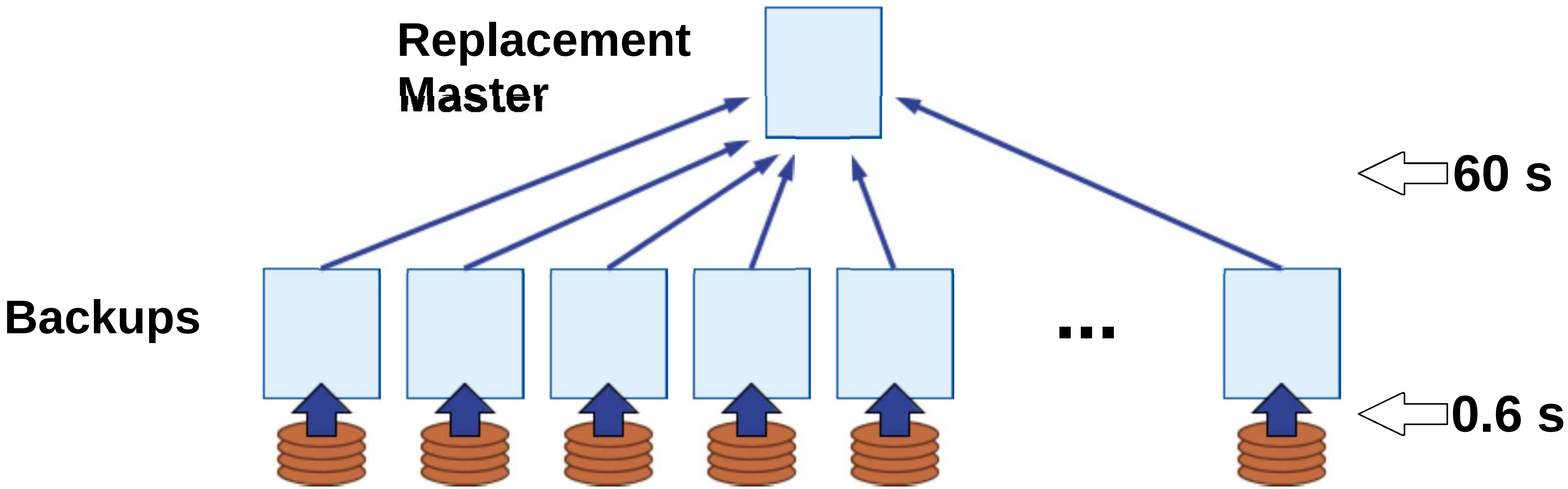
Fast Recovery

- **Idea: Leverage many spindles to recover quickly**
  - Log segments broadly scattered throughout backups
    - Not just great write throughput
    - Take advantage of read throughput
- **Reincarnate masters exactly**
  - Tables
  - Indexes
  - Preserves locality

Fast Recovery: The Problem

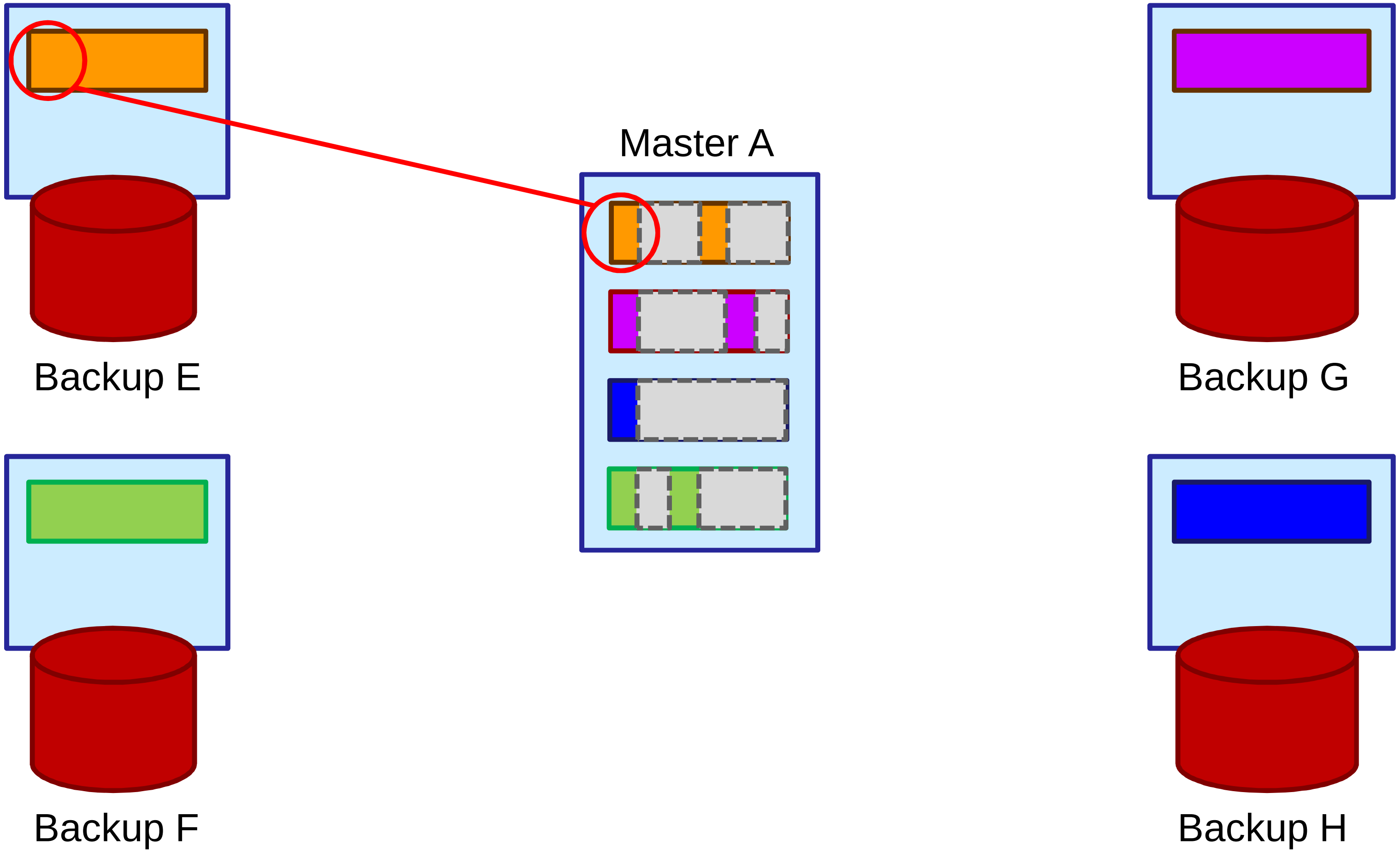
- **After crash, all backups read disks in parallel**
  (64 GB/1000 backups @ 100 MB/sec = **0.6 sec, great!**)

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



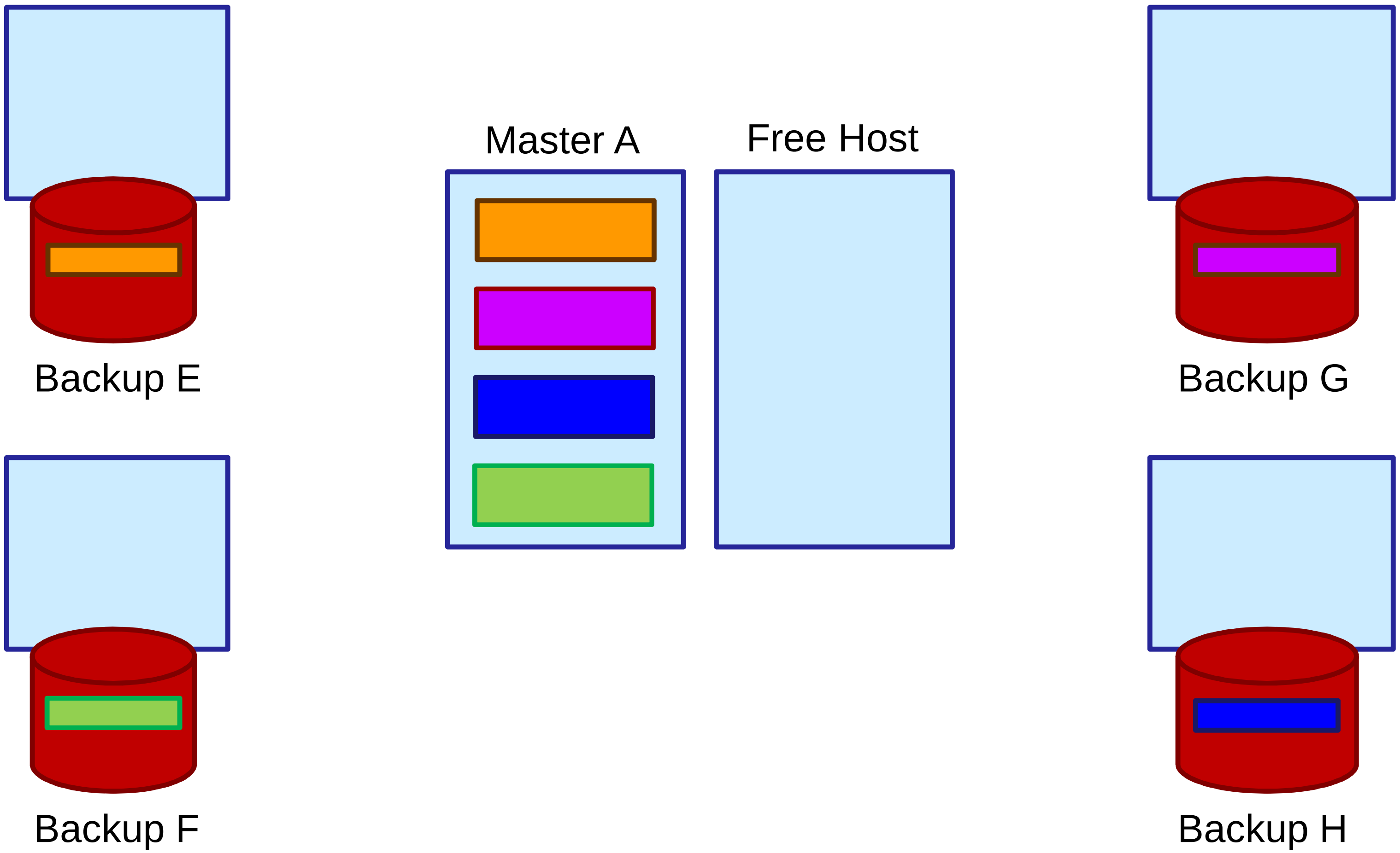
2-Phase Recovery

- **Idea: Is all the data really needed to function?**
  - No
  - Just the hashtable
  - Data already in memory on backups, just need to know where



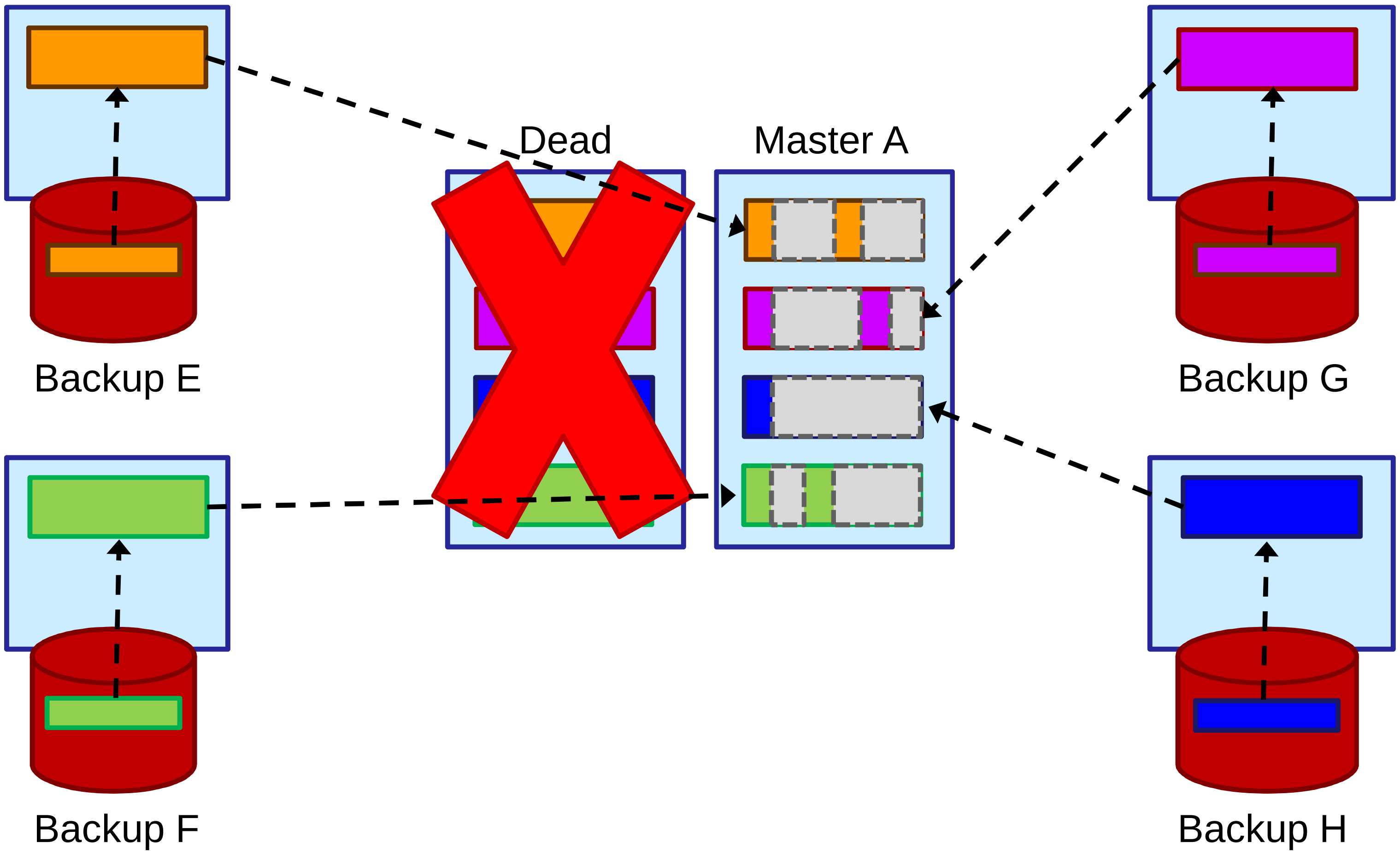
2-Phase Recovery

- **Phase #1: Recover Metadata (< 1s)**
  - Read all segments into memories of backups
  - Send only location info to replacement master
  - Elapsed time depends on # objects



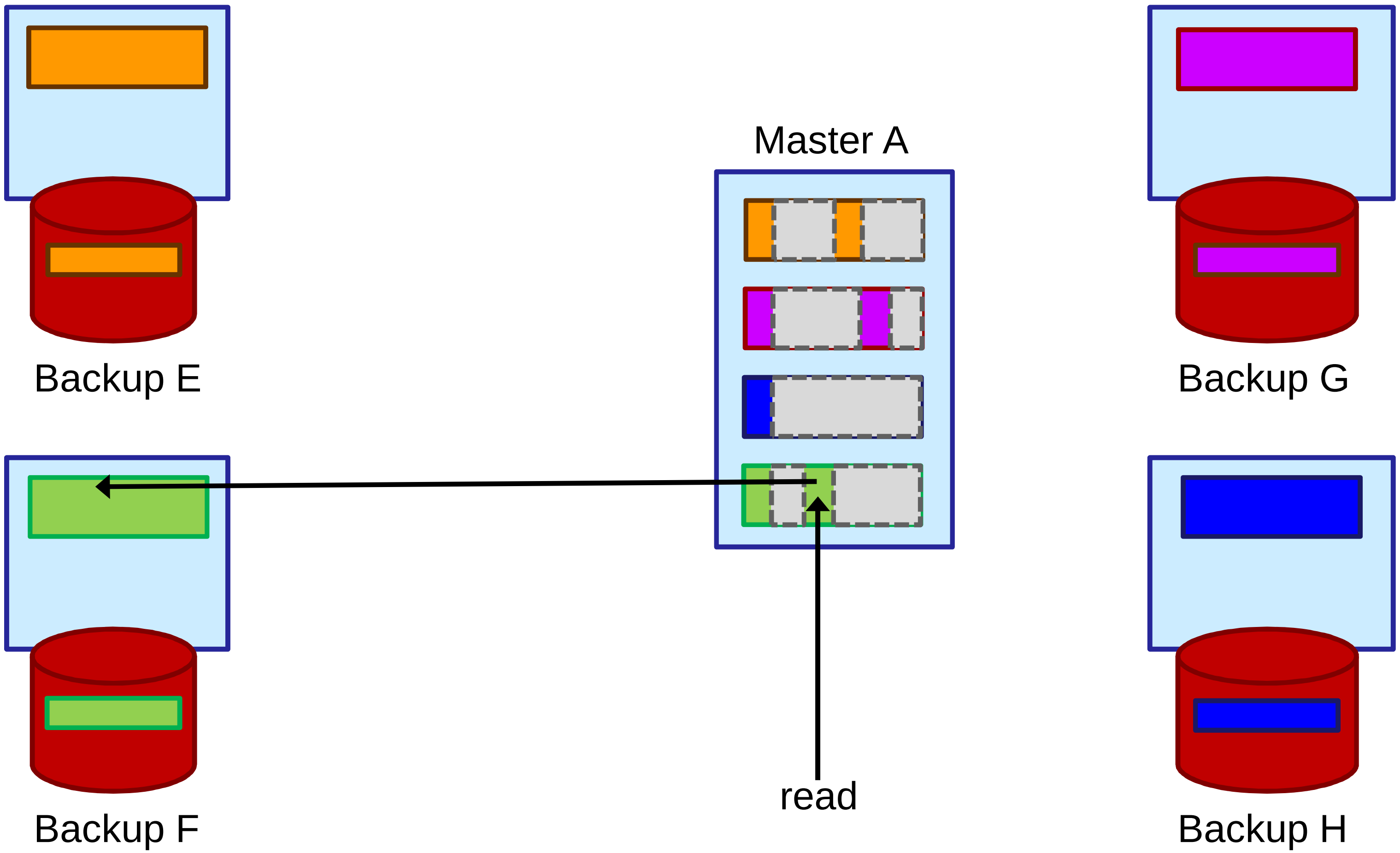
2-Phase Recovery

- **Phase #1: Recover Metadata (< 1s)**
  - Read all segments into memories of backups
  - Send only location info to replacement master
  - Elapsed time depends on # objects



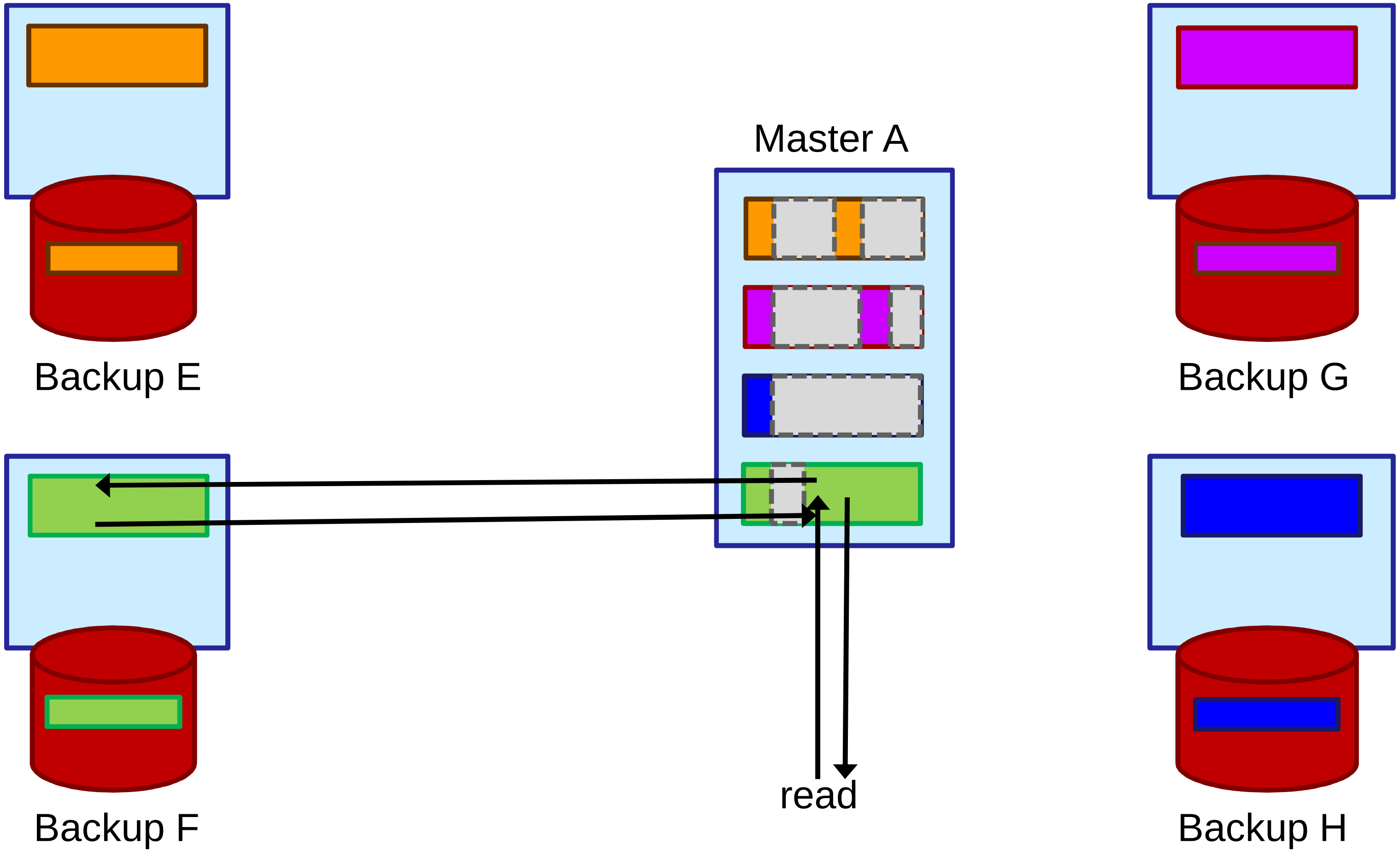
2-Phase Recovery

- **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



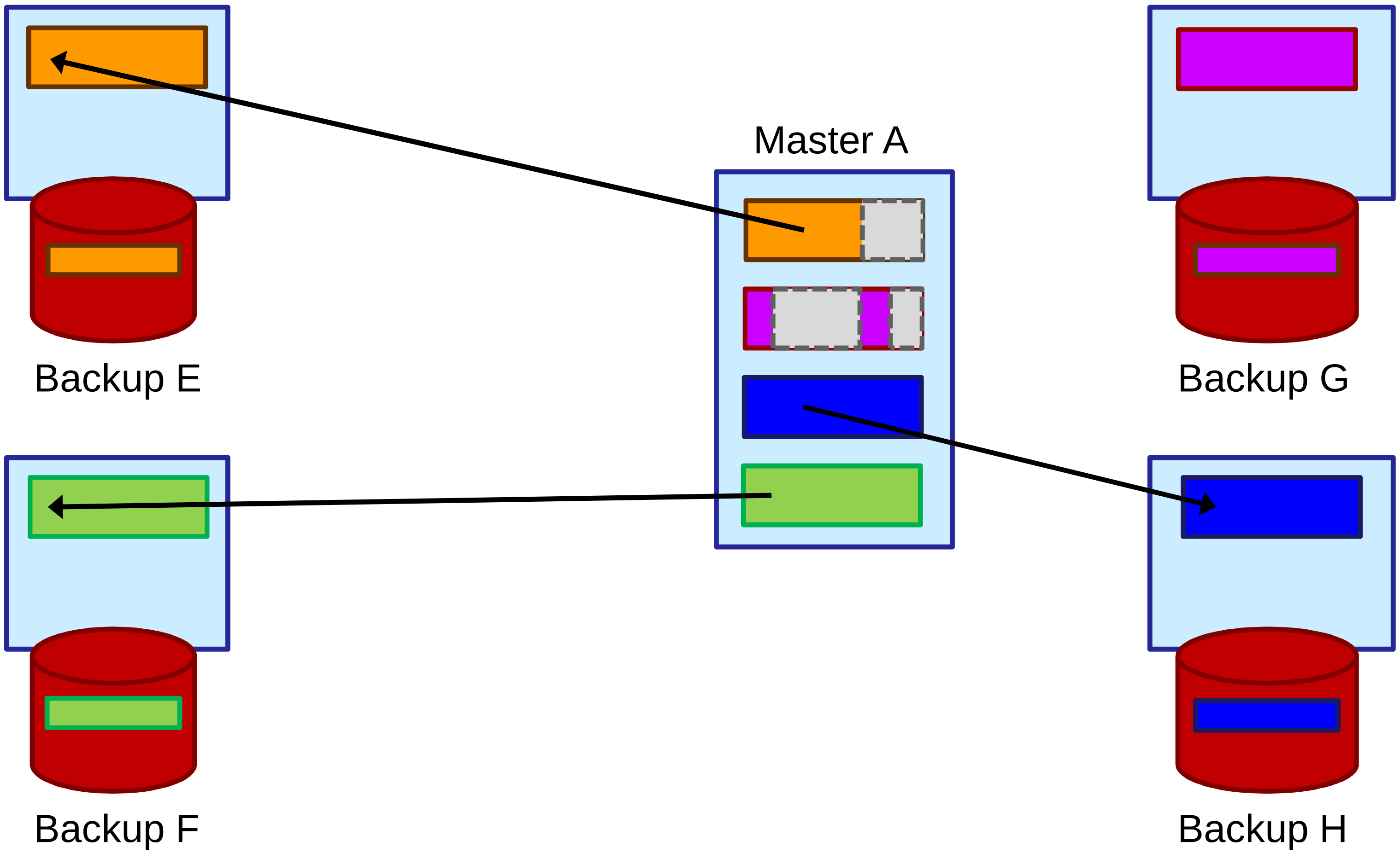
2-Phase Recovery

- **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



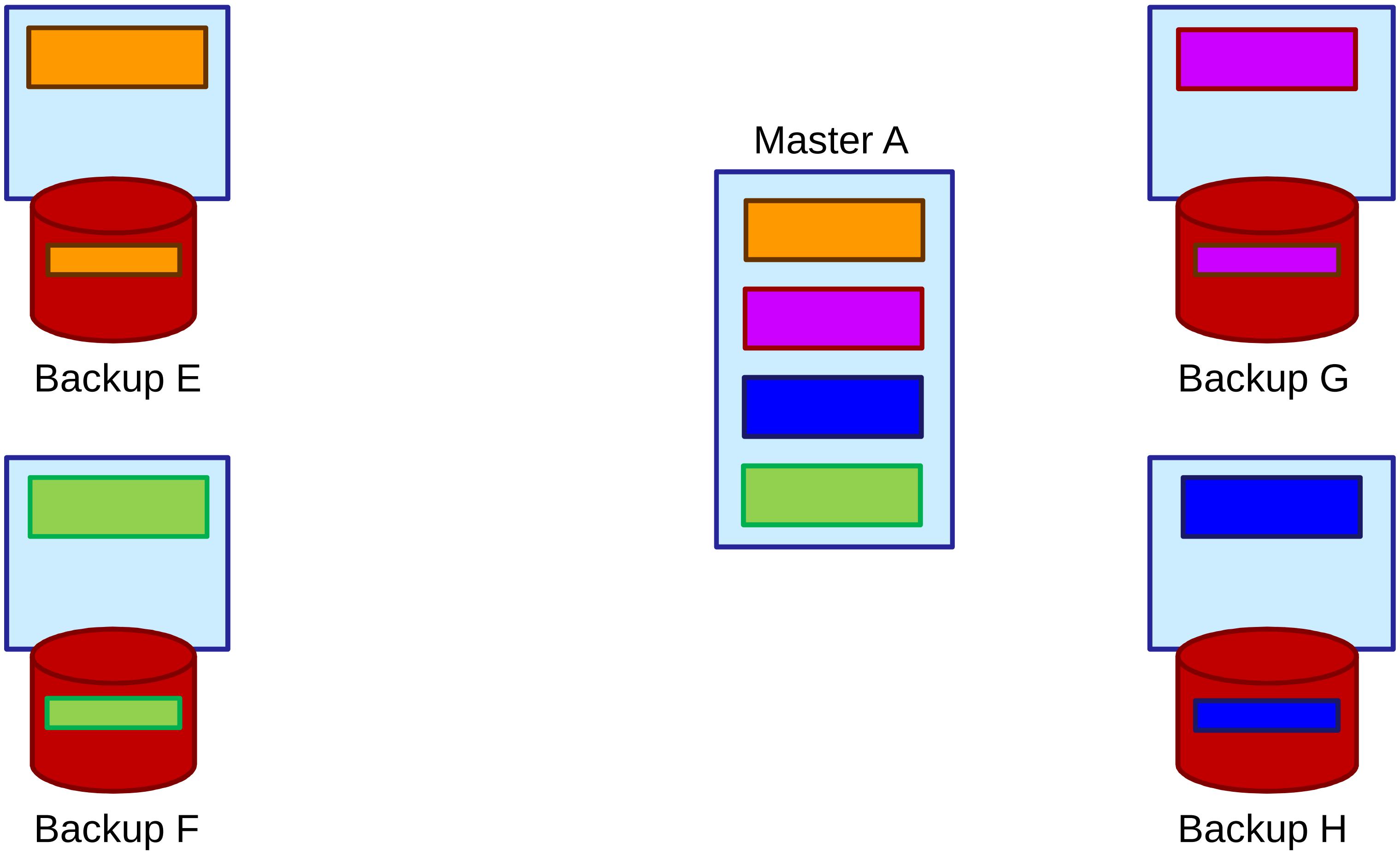
2-Phase Recovery

- **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?
  - Alternatives?
- ✗ **Doesn't deal with heterogeneity**
  - Machine is the unit of recovery
  - Can only recover a machine to one with more capacity
- ✗ **Doesn't solve index recovery**
  - Large indexes need large amount of data to recover
  - 64 GB master containing a 64 GB index
- **Bi-modal Utilization**
  - Must retain pool of empty hosts

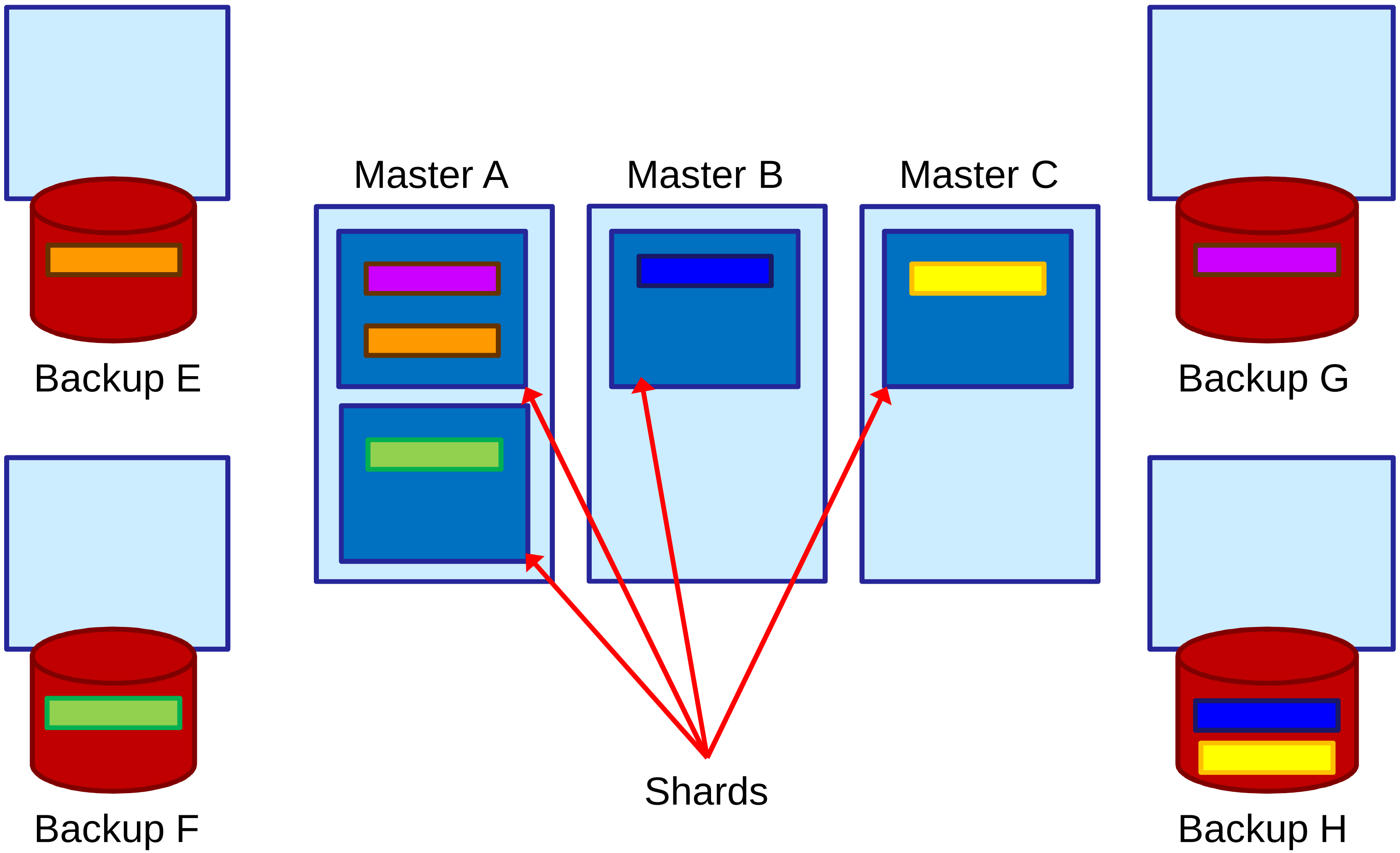
2-Phase Recovery: Problem

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

- **Unsure of a way to recover the master in time**
  - Constrained by both CPU and NIC
  - Recovery to single master is a bottleneck

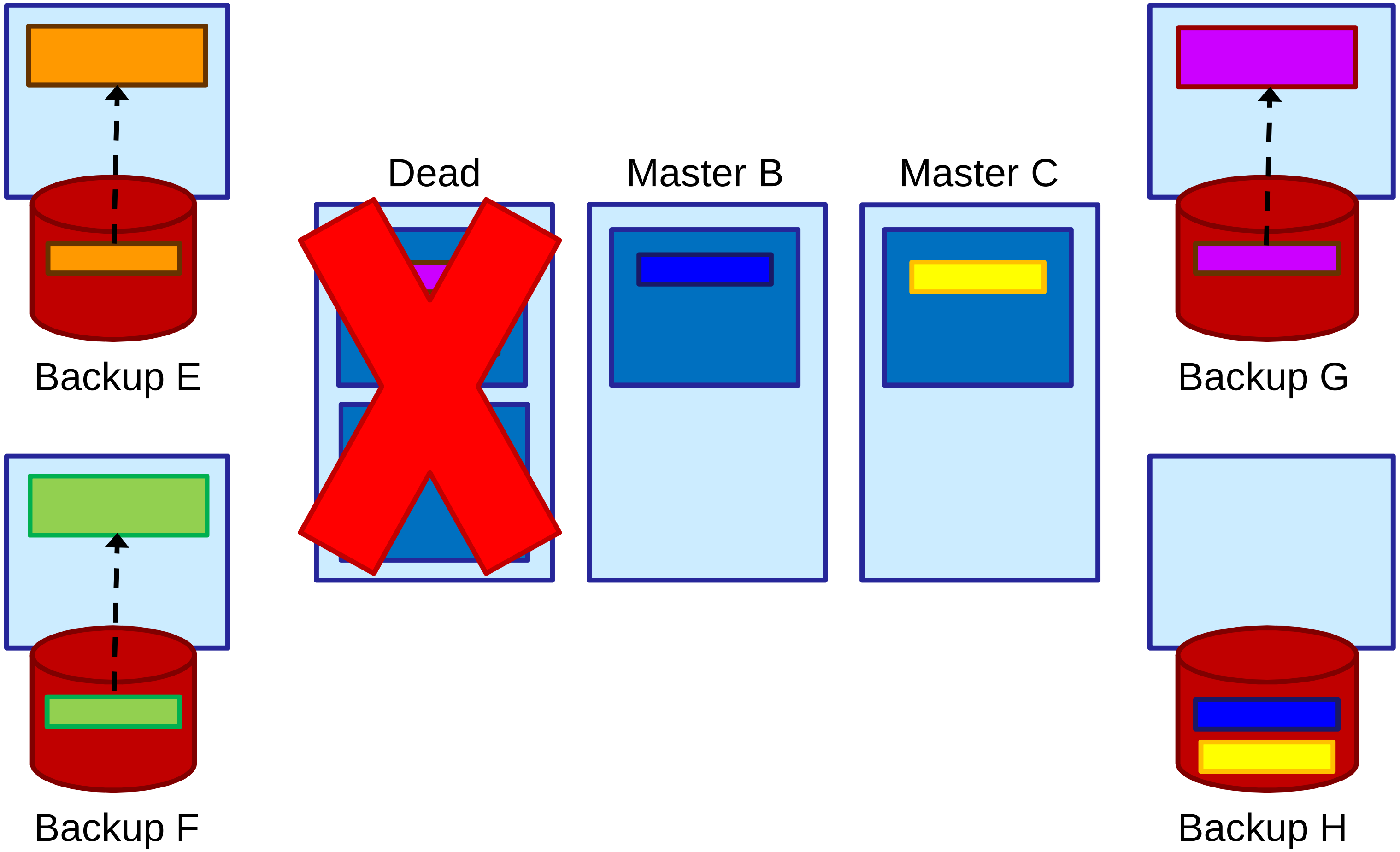
Sharded Recovery

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



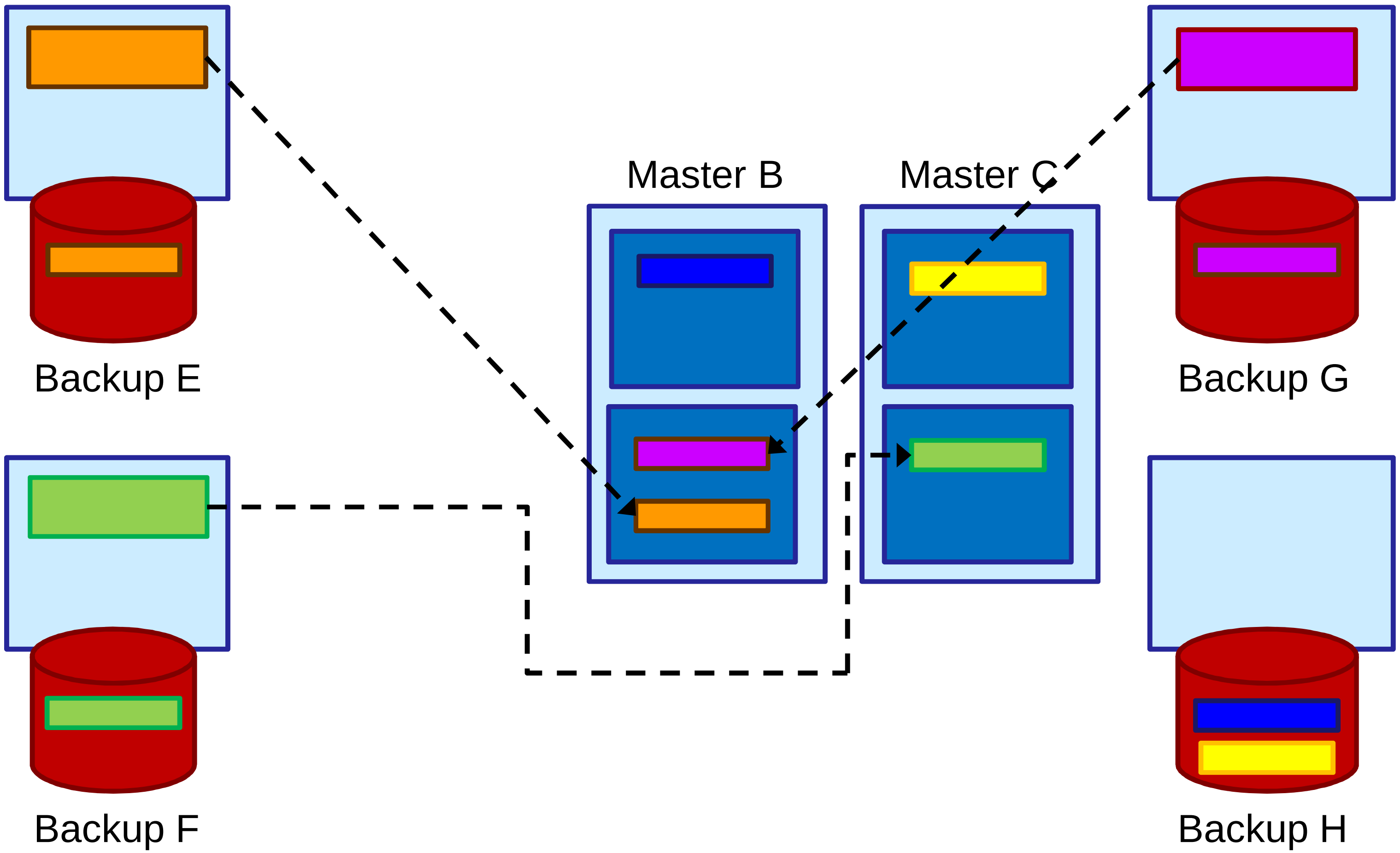
Sharded Recovery

- **Load data from disks**



Sharded Recovery

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

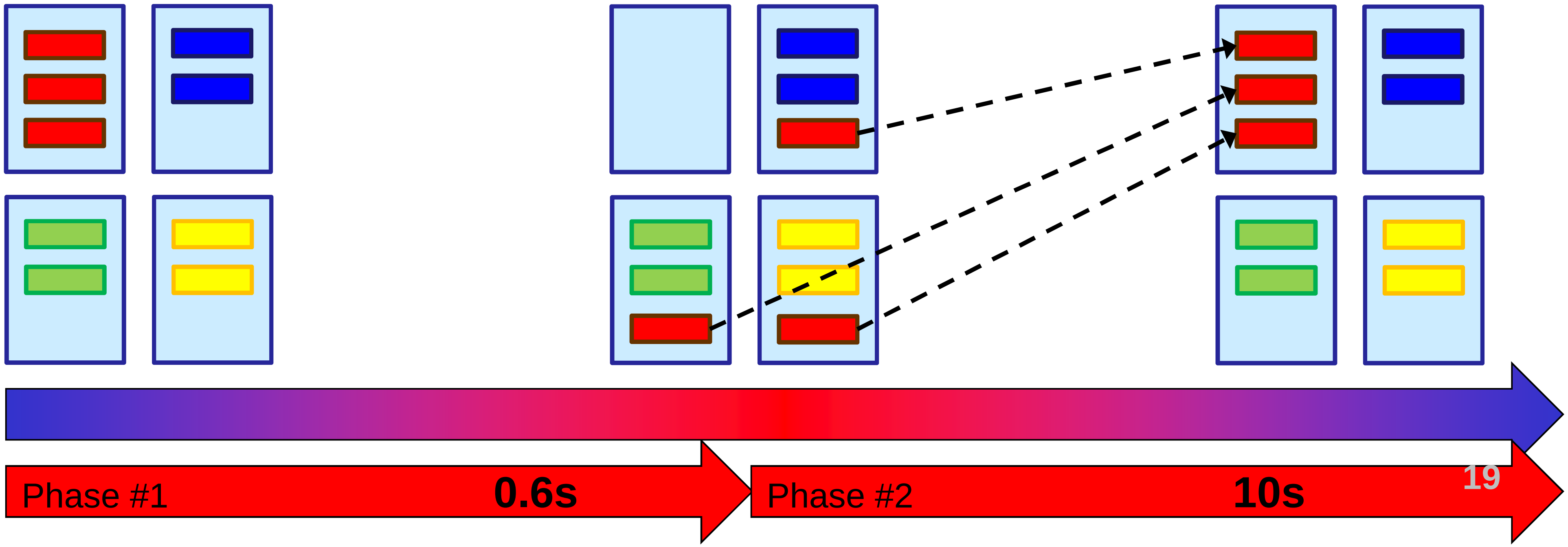


Sharded Recovery: Thoughts

- ✓ **It works: meets availability goals**
  - Can tune time by adjusting shard size
- ✓ **Helps with heterogeneity**
  - Unit of recovery is no longer a machine
- **Increases host/shard related metadata**
  - Coordinator maintains mapping of object ID ranges to masters
    - Clients cache this information
  - Sharding each master 100 ways does not increase metadata 100x
  - Many tables fit within 640 MB
    - These introduce no new mappings
  - Only 100x increase if all tables are on all shards
  - Shards are still large enough to provide locality
- ✗ **Need to talk to all hosts**

Sharded Recovery: Thoughts

- **Recover to least utilized hosts**
  - Using all the machines all the time
  - Based on RAM, NIC, CPU, or something sophisticated
  - Evens out host utilization (unlike 2-Phase approach)
- **Does not recover locality**
  - But, no worse than 2-Phase
  - Shared approach can recover as fast as Phase #1
  - And can restore locality as fast as Phase #2



Master Recovery: Summary

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

Failures: Backups

- On backup failure the coordinator broadcasts
- All masters check their live segments
- If any were backed up on that host
- Rewrite those segments (from RAM) elsewhere

Failures: Racks/Switches

- **Rack failures handled the same as machine failures**
  - Consider all the machines in the rack dead
- **Careful selection of segment backup locations**
  - Write backups for segments to other racks
    - As each other
    - As the master
  - Changes as masters recover
    - Can move between racks
  - Masters fix this on recovery
    - Rewrite segments elsewhere, if needed

- **Question: Minimum RAMCloud that can sustain an entire rack failure and meet recovery goal?**
  - 100 shards to recover a single machine in 0.6s
  - 50 dead * 50 shards, need 2500 machines to make 1.2s
  - Don’t pack storage servers in racks, mix with app servers

Failures: Power

- **Problem: Segments 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 "open" segments on fluctuation**
  - Any battery should be easily sufficient for this
  - About *r* open segments per shard per backup
    - *r* = 3 with 100 shards/master
    - Must flush 300 * 8MB = 24s
- **No battery?**
  - Deal with lower consistency
  - Synchronous writes

- **Question: Is there some cost effective way to get 10-20s of power?**

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 will do much better**
  - Application can batch writes
  - Application understands consistency needs
- **Is this something we need to support?**

Summary

- **Distribute Backup Data**
  - Scatter reads during recovery to overcome disk bottleneck
- **Sharded Recovery**
  - Scatter rebuilding to overcome CPU and network bottlenecks

- **Use scale in two ways to achieve availability**

- **Scale driving lower-latency**

Discussion