Masterless Distributed Applications With Riak Core

Tim.Tang May 2016

Why Riak Core?

Distributed, Scalable, Failure-tolerant

Why Riak Core?

Distributed, Scalable, Failure-tolerant

No central coordinator.
Easy to setup/operate.

Why Riak Core?

Distributed, Scalable, Failure-tolerant

Horizontally scalable.
Easy add more physical nodes.

Why Riak Core?

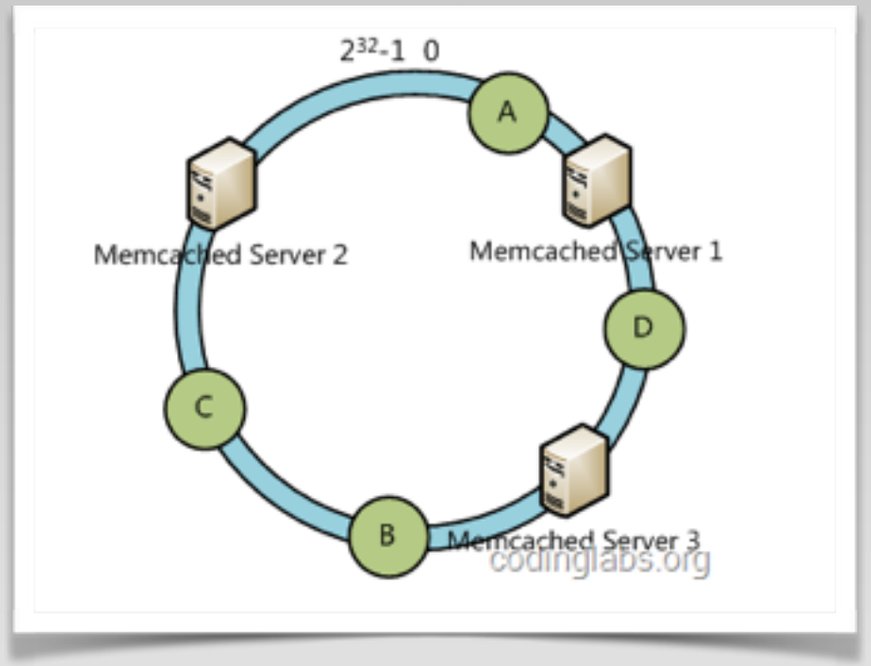
Distributed, Scalable, Failure-tolerant

No single point of failure.
Self-healing.

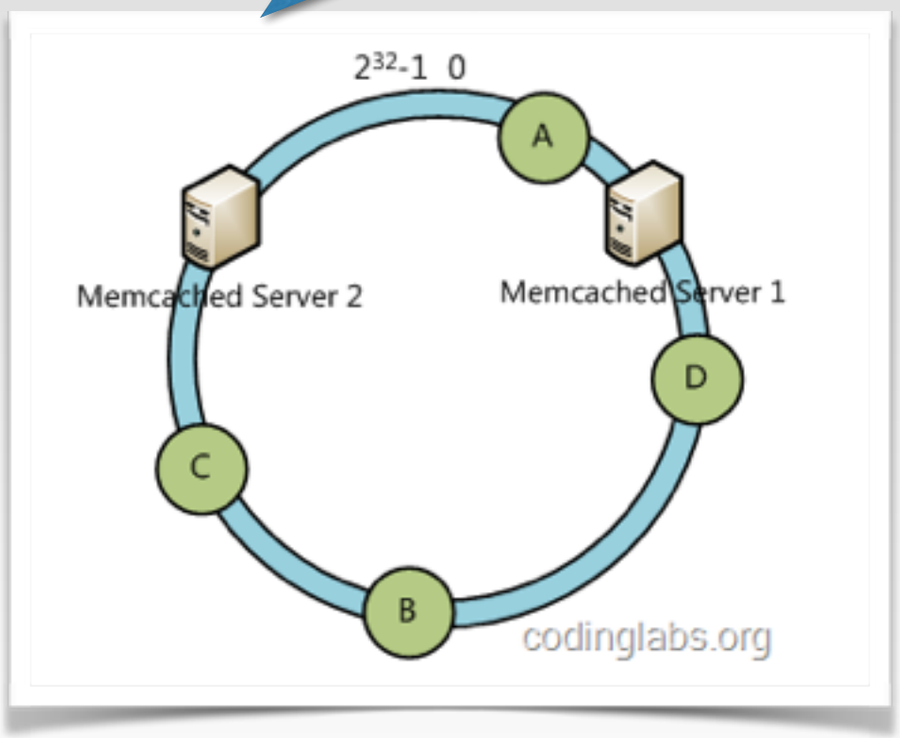
Why Consistent Hash?

- Limits reshuffling of keys when hash table data structure is rebalanced (Add/Remove Nodes).
- Uses consistent hashing to determine where to store data on a primary replica as well as fallbacks if the primary is offline.

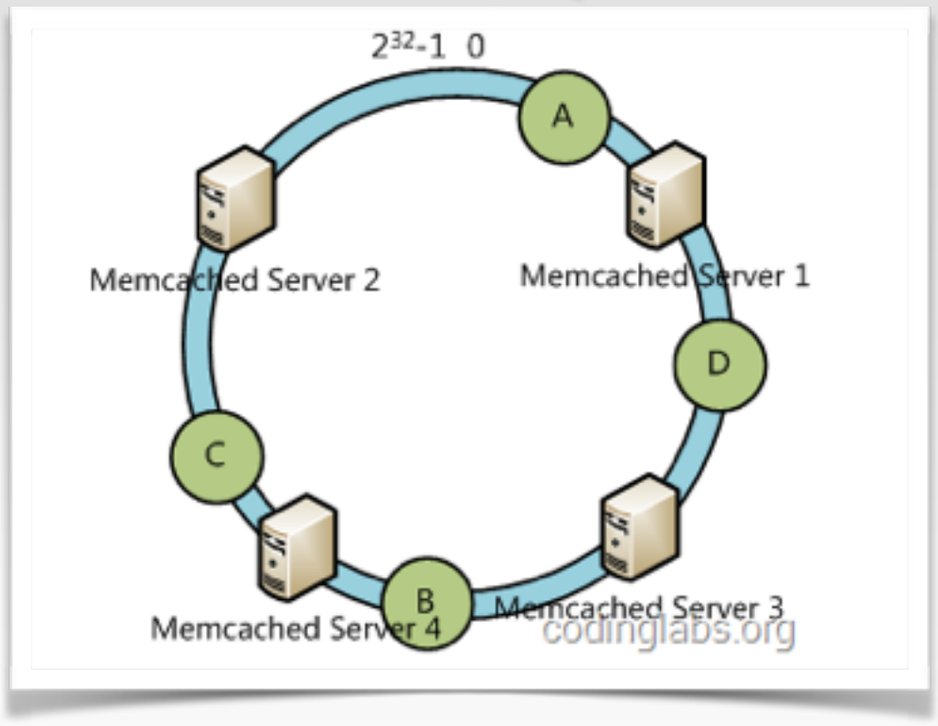
Why Consistent Hash?



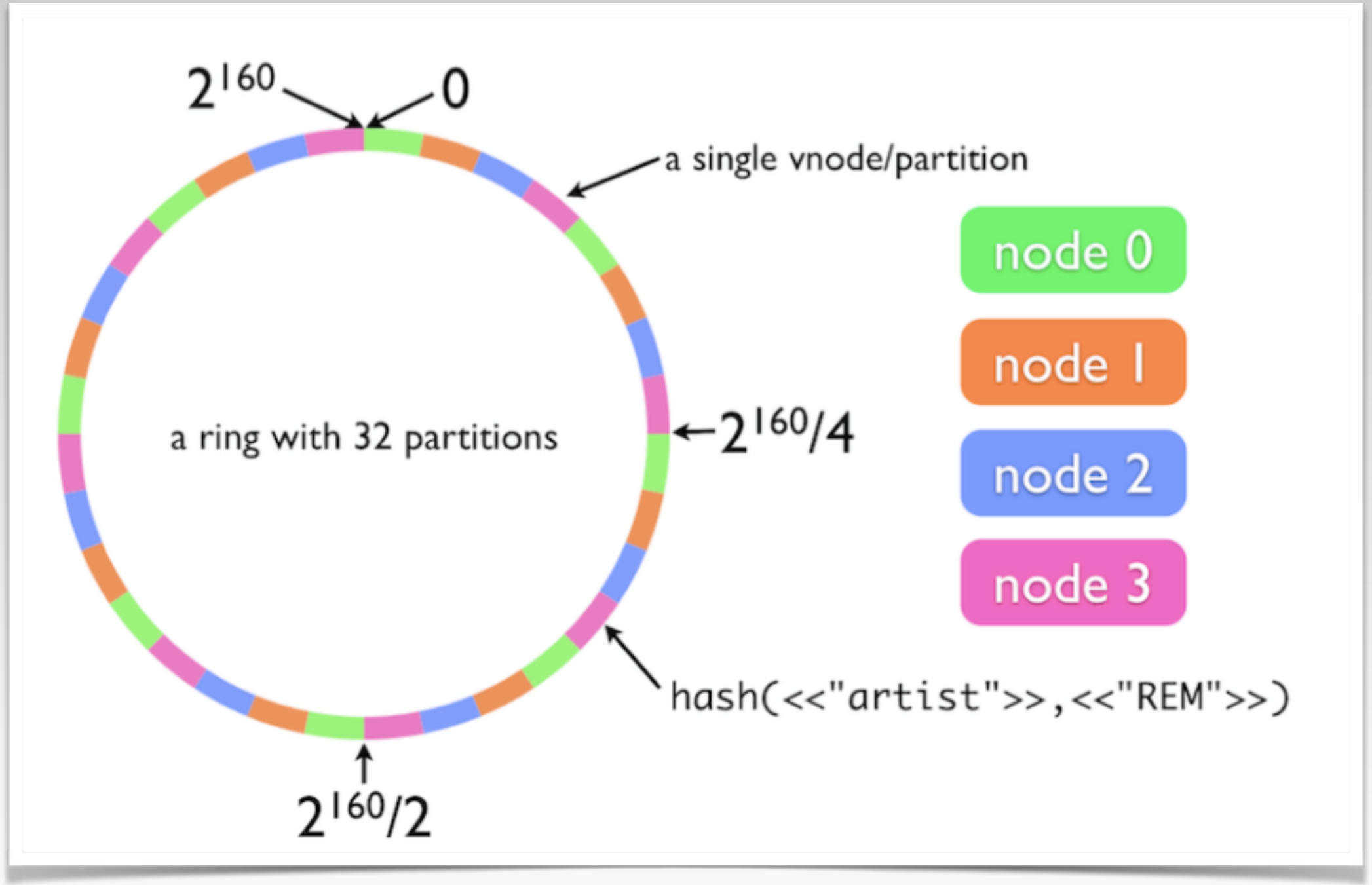
Remove Node 3



Add Node 4



Concepts: The Ring



Concepts: Virtual Node

- One Erlang process per partition in the consistent hashing ring.
- One partition may have multi-vnodes.
- Fundamental unit of replication, fault tolerance, concurrency.
- Receives work for its portion of the hash space.

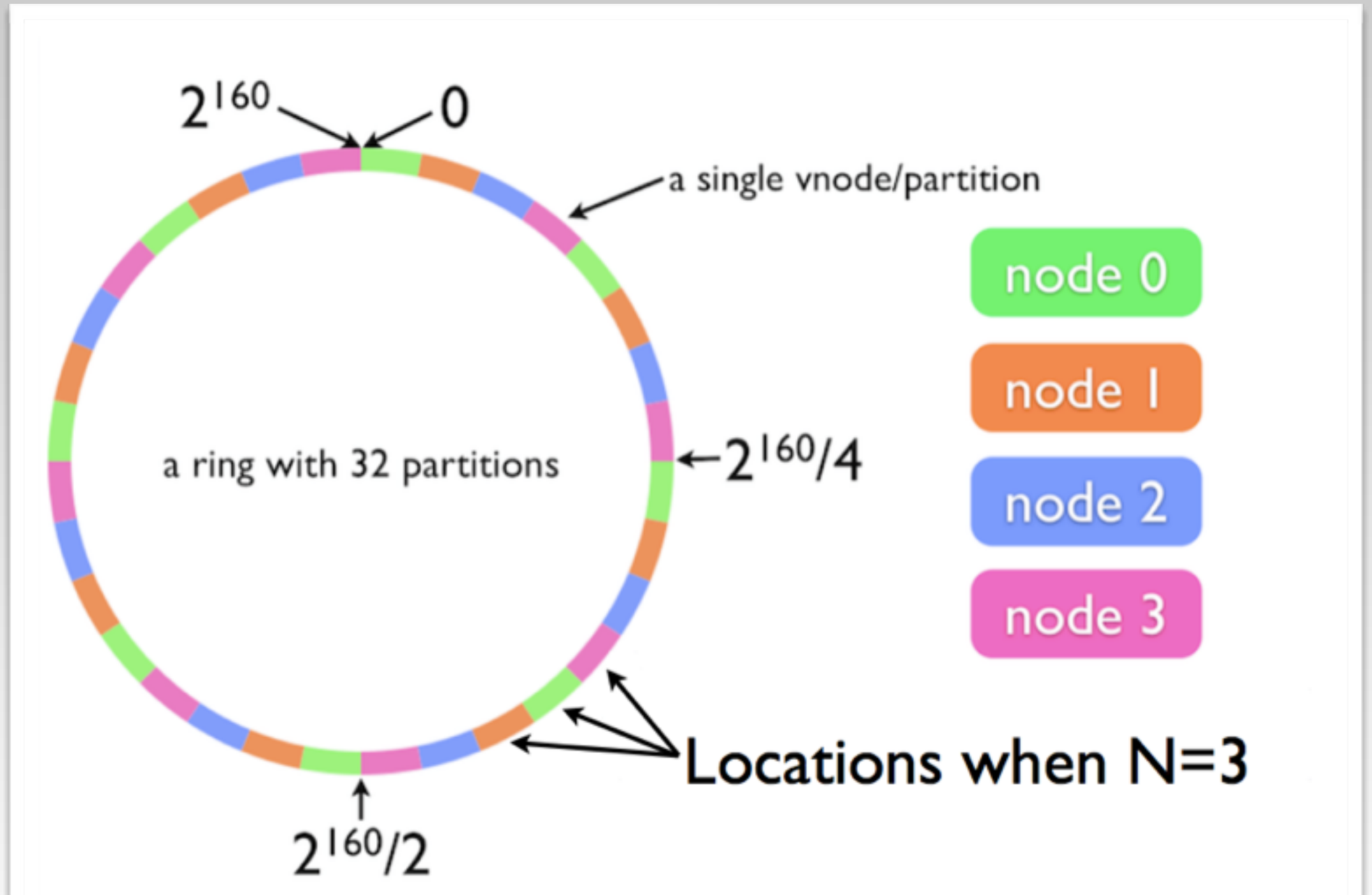
Concepts: Virtual Node Master

- Keep track of all active vnodes on its node receives messages from coordinating FSMs.
- Translates partition numbers to local PIDs and dispatches commands to individual vnodes.
- One vnode_master per Physical Node.

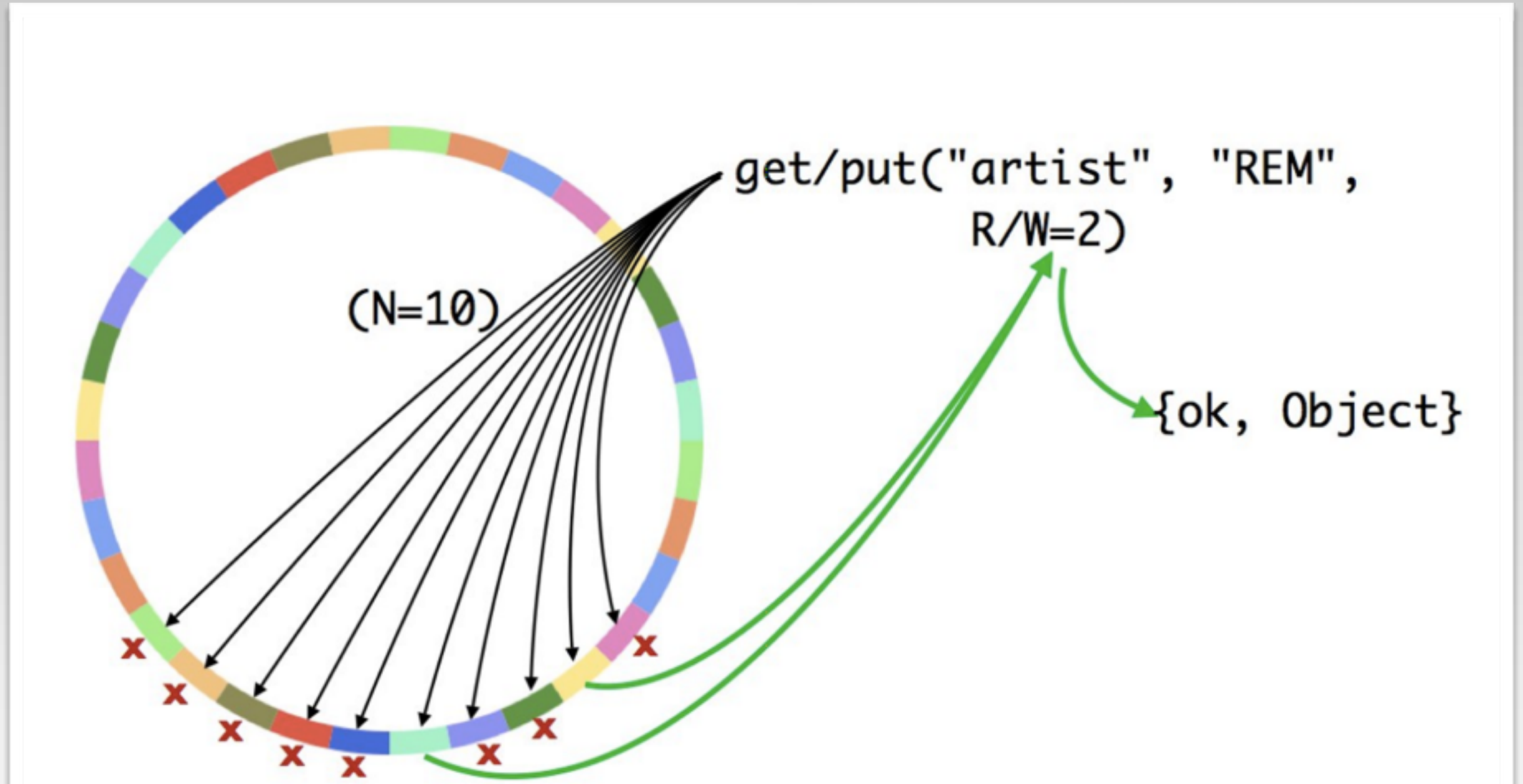
Concepts: N/R/W

- N = number of replicas to store (on distinct nodes)
- R = number of replica responses needed for a successful read per-request
- W = number of replica responses needed for a successful write perrequest

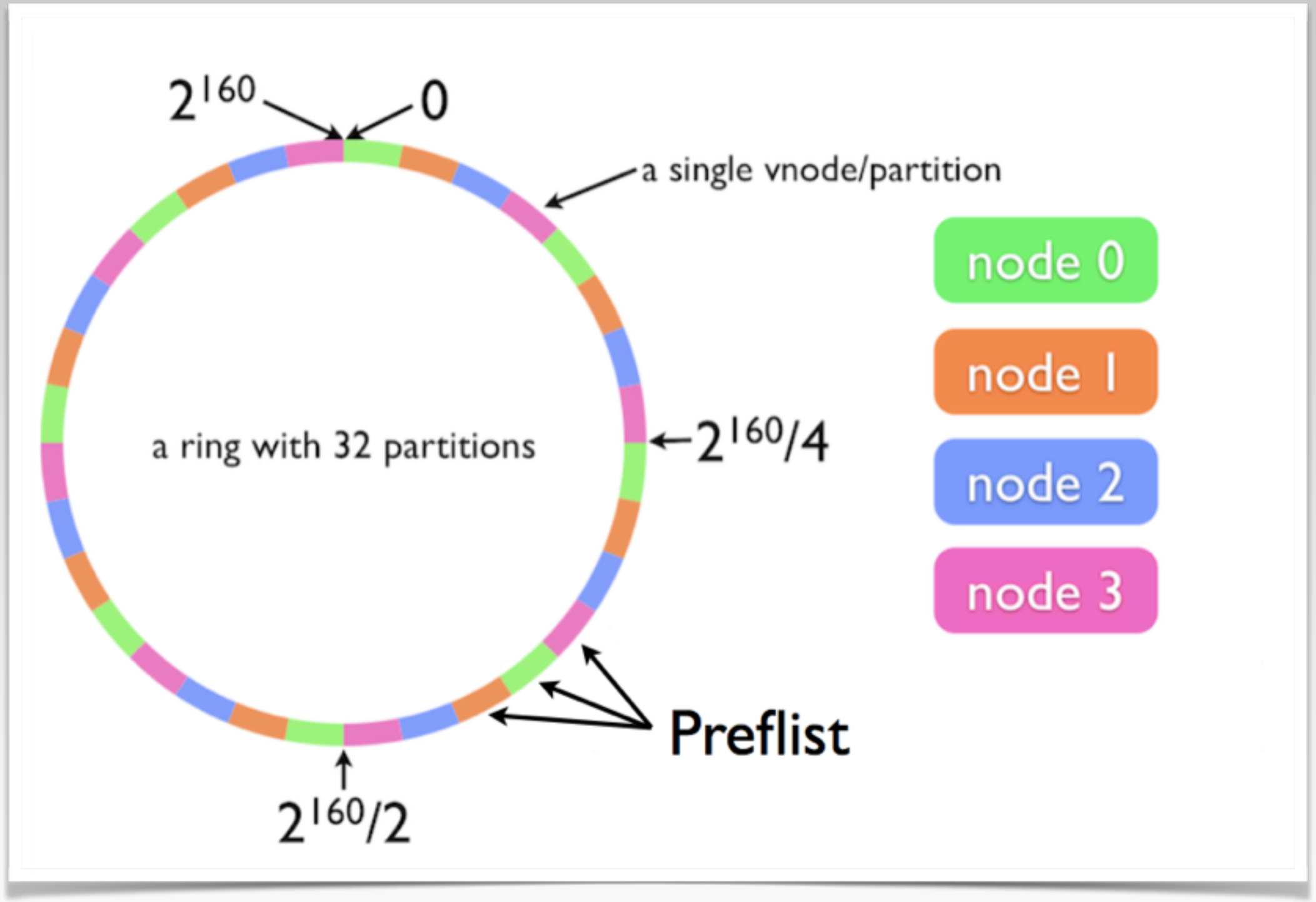
Concepts: N/R/W



Concepts: N/R/W



Concepts:Preference List



Concepts:Read Repair

- If a read detects that a vnode has stale data, it is repaired via asynchronous update.
- Passive Anti-Entropy, helps implement eventual consistency.

Concepts:VClock

Ops	NodeA(midi1@127.0.0.1)	NodeB(midi2@127.0.0.1)	NodeC(midi3@127.0.0.1)
NodeA +500	500 [{A,1}]	500 [{A,1}]	500 [{A,1}]
NodeA +200	700 [{A,2}]	700 [{A,2}]	700 [{A,2}]
NodeC + 300	1050 [{A,2}, {C,1}]	1050 [{A,2}, {C,1}]	1050 [{A,2}, {C,1}]
Network Split -- (A,B), (C) NodeC + 100	1050 [{A,2},{C,1}]	1050 [{A,2},{C,1}]	1150 [{A,2}, {C,2}]
NodeB + 500	1550 [{A,2}, {B,1}, {C,1}]	1550 [{A,2}, {B,1}, {C,1}]	1150 [{A,2}, {C,2}]
Network Repaired -- (A,B,C) NodeA + 50	1600 [{A,3}, {B,1}, {C,1}]	1600 [{A,3}, {B,1}, {C,1}]	1200 [{A,3}, {C,2}]
	Get Request On NodeA, How To Merge Results?		

Concepts:VClock

- Last Write Wins (LWW)
- Allow multiple versions to coexist, caller reconcile the versions with full context.
- Use riak_dt module to handle data conflicting.

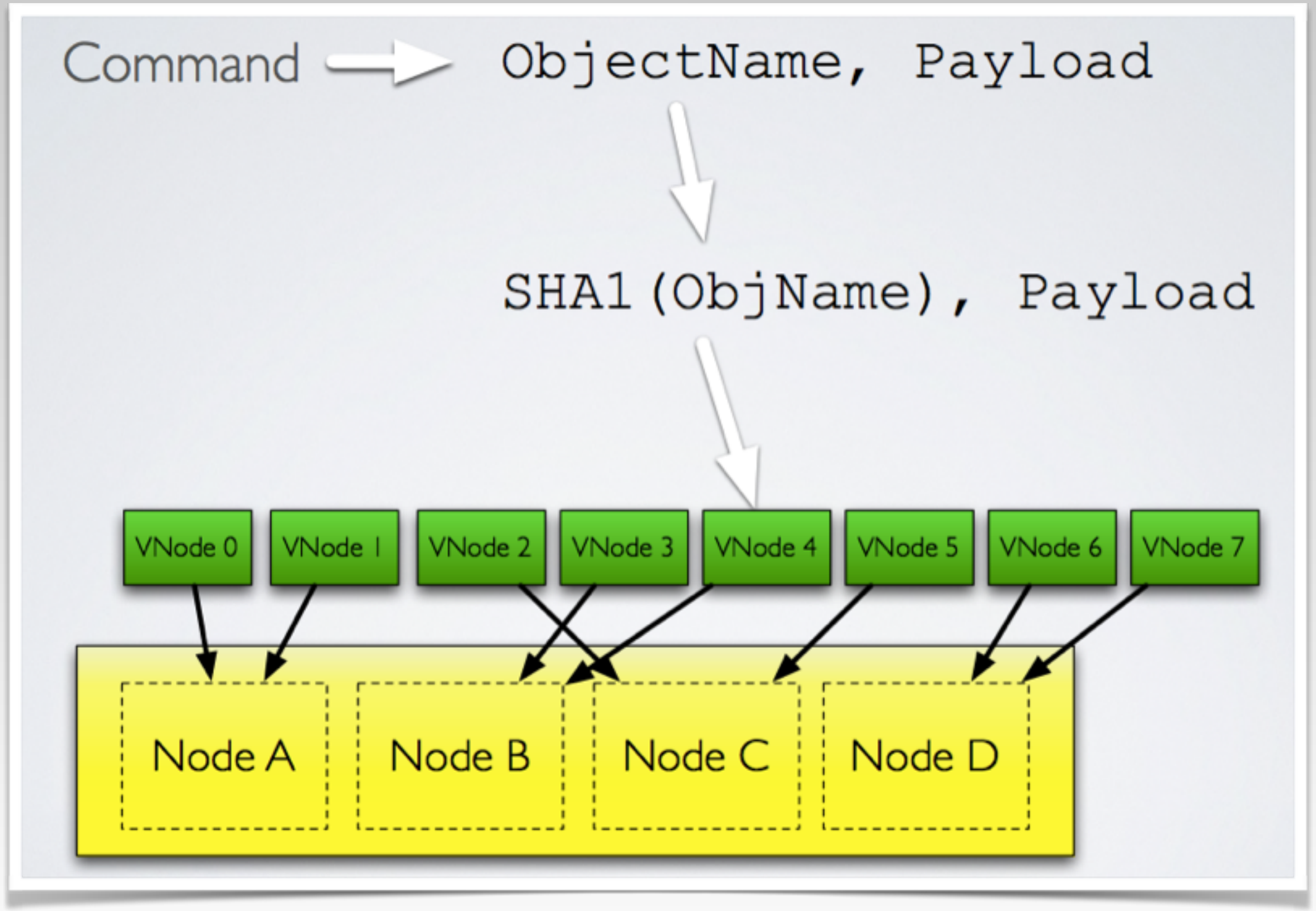
Concepts:Handoff

- Handoff Types:
  - Ownership: occurs when a new node joins the cluster or the vnode needs to be moved.
  - Hinted: occurs when a "fallback" vnode took the responsibility for a "primary" vnode but the primary vnode is reachable again.
  - Repairs: repair handoff happens when your application explicitly calls riak_core_vnode_manager:repair/3.
  - Resize: > Riak core 2.0, riak_core_ring:resize().

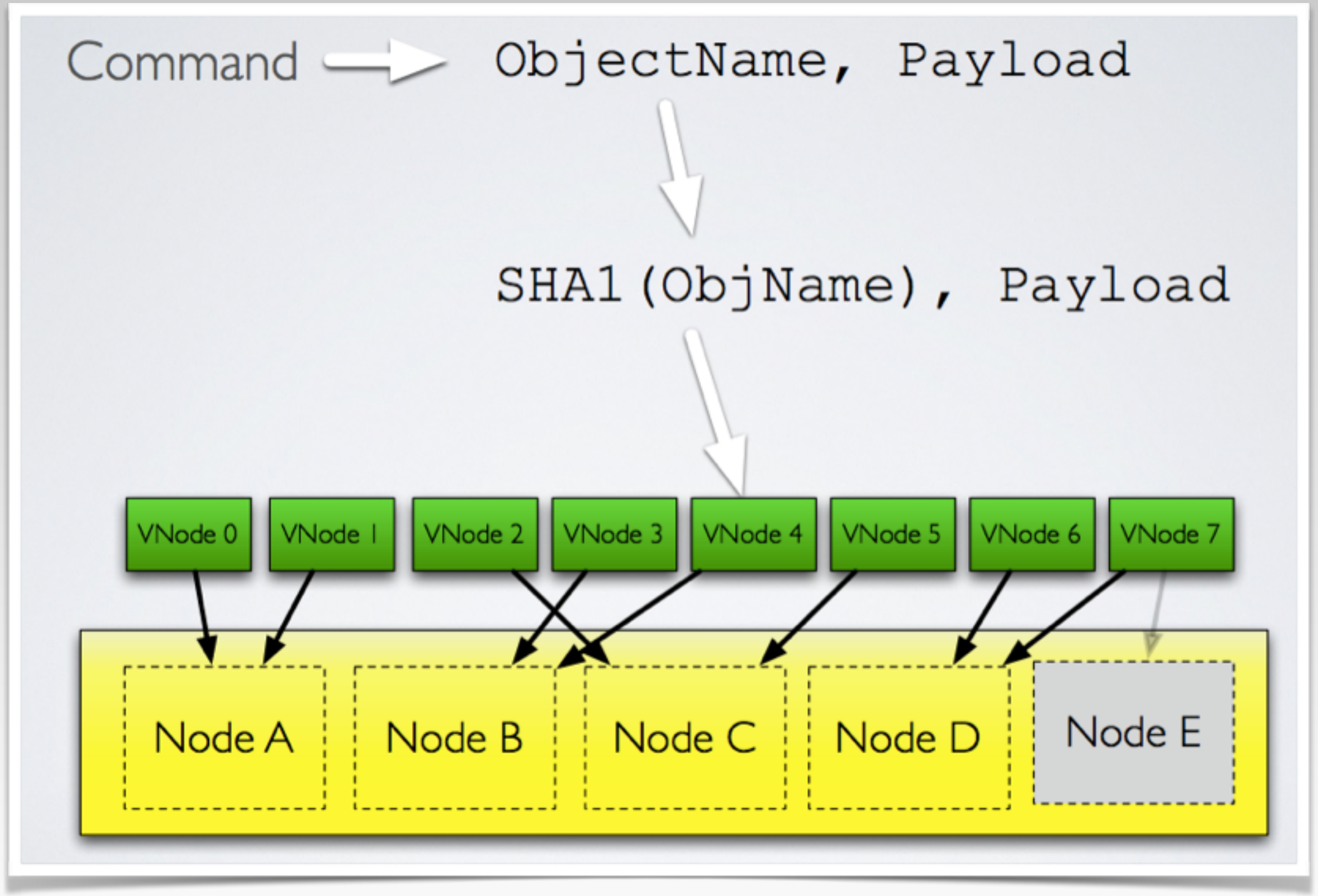
MisConcepts:Fallback

- One Physical Node down, all vnodes(primary) on this physical node status will fallback to another physical node. Switch to type fallback.
- Fallback is never performed by another partition, it goes to another node but keeps the index.
- For some time fallback and primary vnodes coexists.

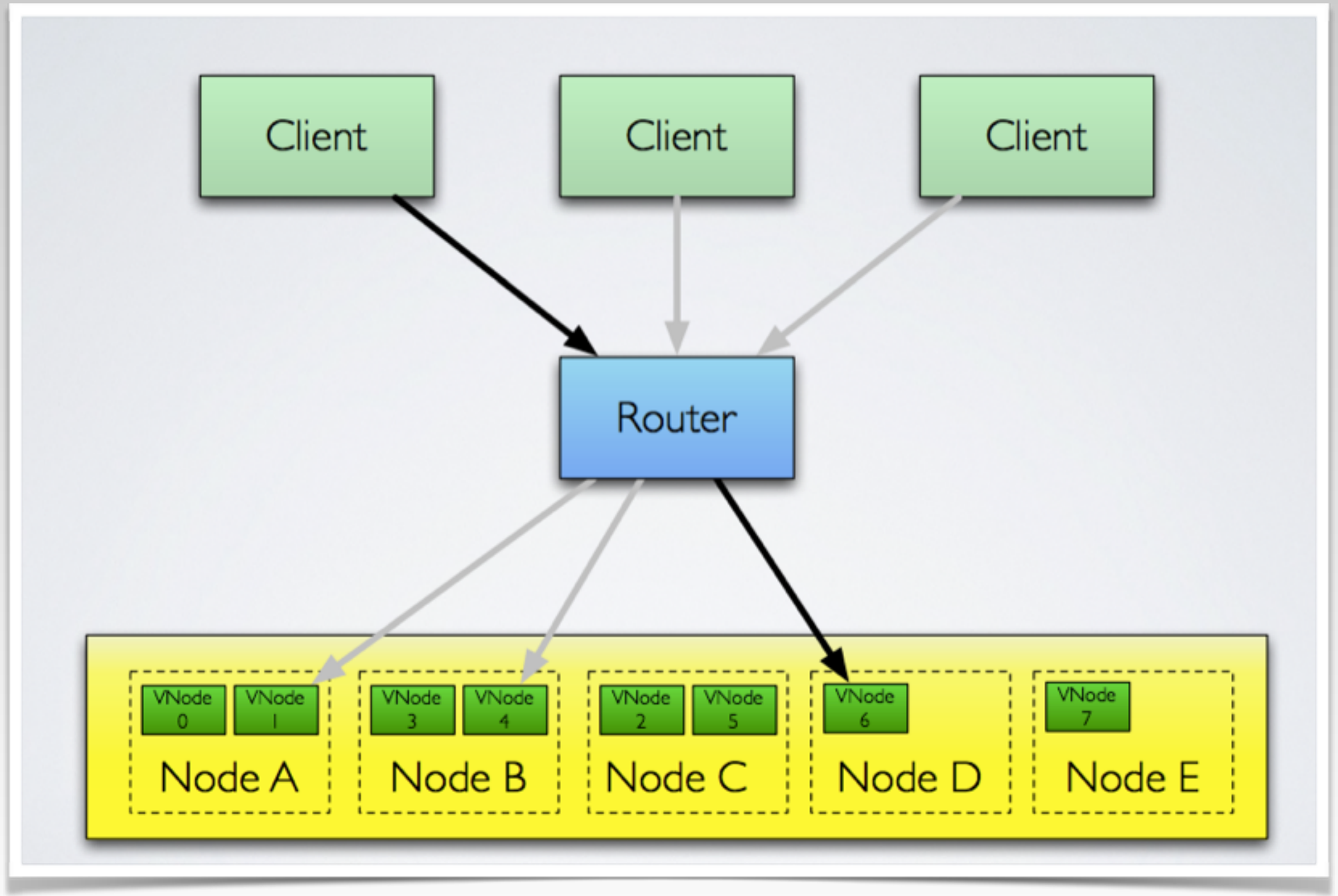
Routing With Consistent Hash



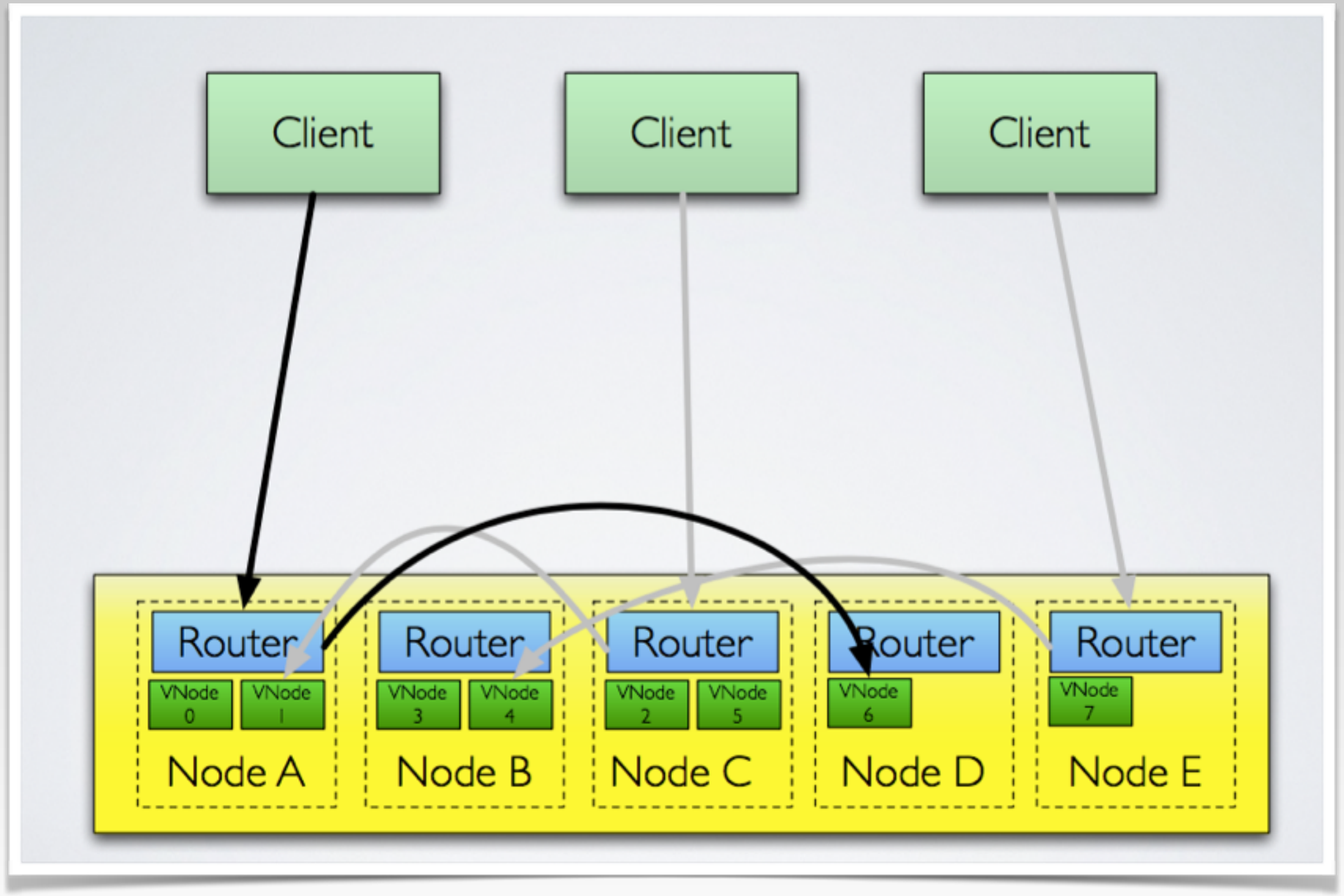
Adding A Node



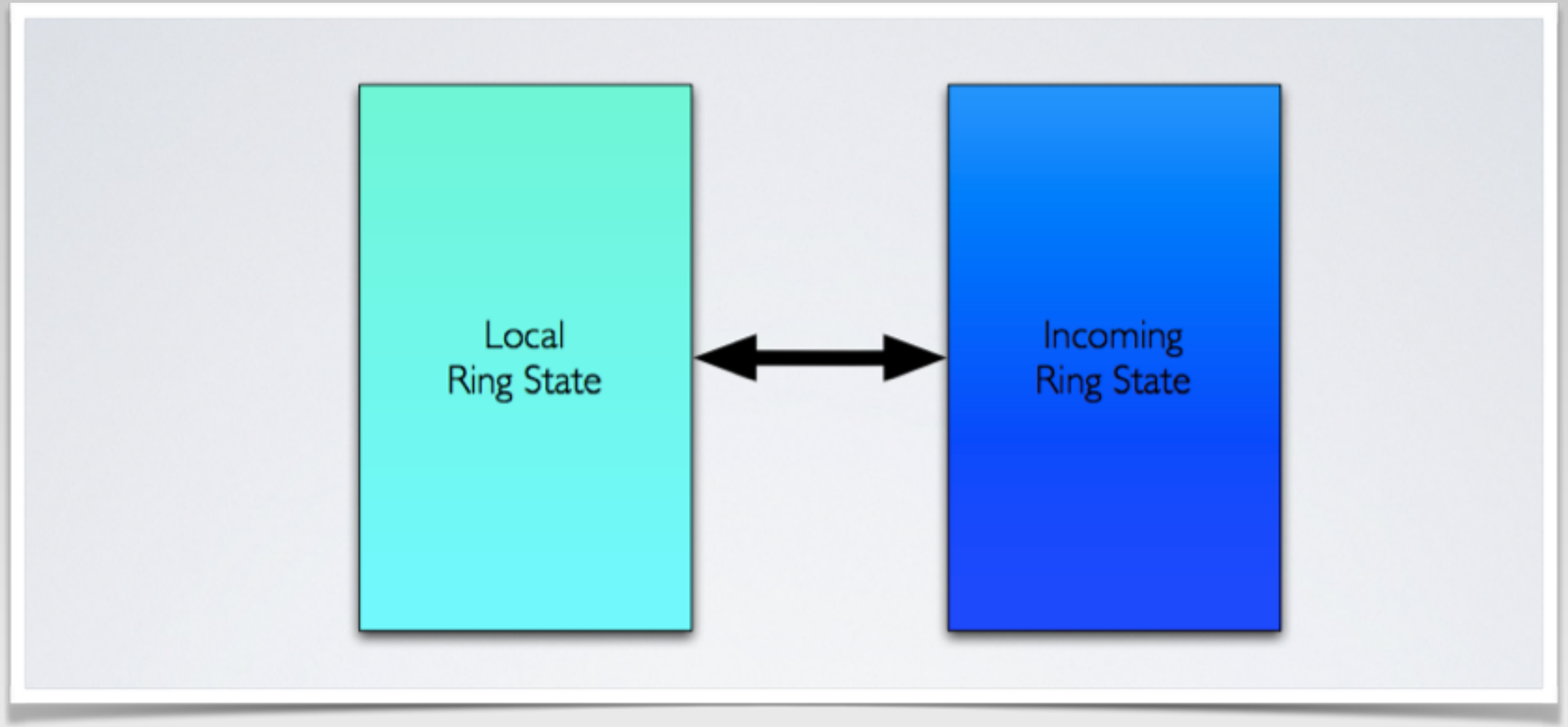
Traditional Router



Riak Core Router



How Do The Routers Reach Agreement?



- Each node has one copy of ring cache.
- Compare ring state strictly by gossip protocol.
- Ring knows each vnode status.

What We Get "Out Of Box"

- Physical Node cluster state management.
- Ring state management.
- Vnode placement and replication.
- Cluster and ring state gossip protocols.
- Consistent hashing utilities.
- Handoff activities, covering set callbacks.
- Rolling upgrade capability.
- Key based request dispatch.
- etc…

Building An Application On Riak Core

- MIDI demo => https://github.com/tim-tang/midi
- Reference:
  - http://marianoguerra.github.io/little-riak-core-book/index.html
  - Rebar3 => https://www.rebar3.org/
  - rebar3_template_riak_core => https://github.com/marianoguerra/rebar3_template_riak_core
  - Riak Core source => https://github.com/basho/riak_core
  - Erlang/OTP 18.

Riak Core Pitfalls

- Cluster membership is controlled by a human, even when a node failure has been (correctly) detected by the cluster manager.
- Vnode distribution around the ring is sometimes suboptimal.

Is it a good fit?

- It expects you to have a "key" that links to a blob of data or service.
- The key (or rather its chash) determines its primary vnode and adjacent replicas.
- The data itself is opaque and has application context.

Not Mentioned

- Merkle Trees (AAE): https://github.com/basho/riak_core/blob/develop/docs/hashtree.md
- Ring Resizing: https://github.com/basho/riak_core/blob/develop/docs/ring-resizing.md

References

- Riak Core Confliction resolution => https://github.com/tim-tang/try-try-try/tree/master/04-riak-core-conflict-resolution
- CRDT LASP => https://github.com/lasp-lang/lasp
- Why Vector Clock are Easy => http://basho.com/posts/technical/why-vector-clocks-are-easy/

Thanks!

Q&A.