

riak core.

- *consistent hashing
- *vector clocks
- *sloppy quorums

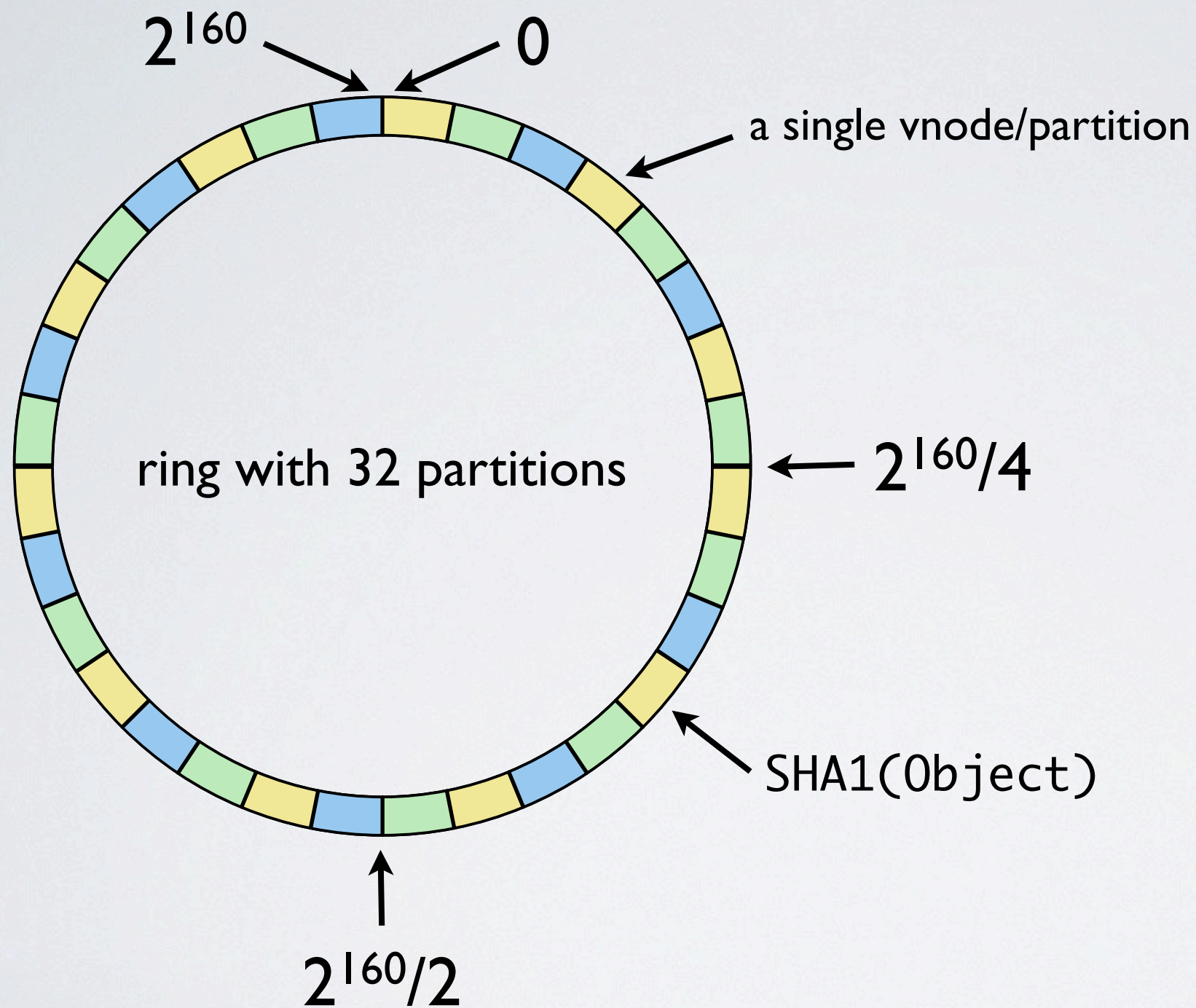
- *gossip protocols
- *virtual nodes
- *hinted handoff

Riak Core is an
Open Source Erlang library
that helps you build
distributed, scalable, failure-tolerant
applications using a
Dynamo-style architecture.

Core Dynamo Principals

- Scalable
 - Consistent hashing
 - Virtual nodes
- Highly Available + Self Healing
 - Always writeable
 - Sloppy quorums
 - Hinted handoff
- All Nodes Equal / Masterless
 - Gossip protocol

Consistent Hashing

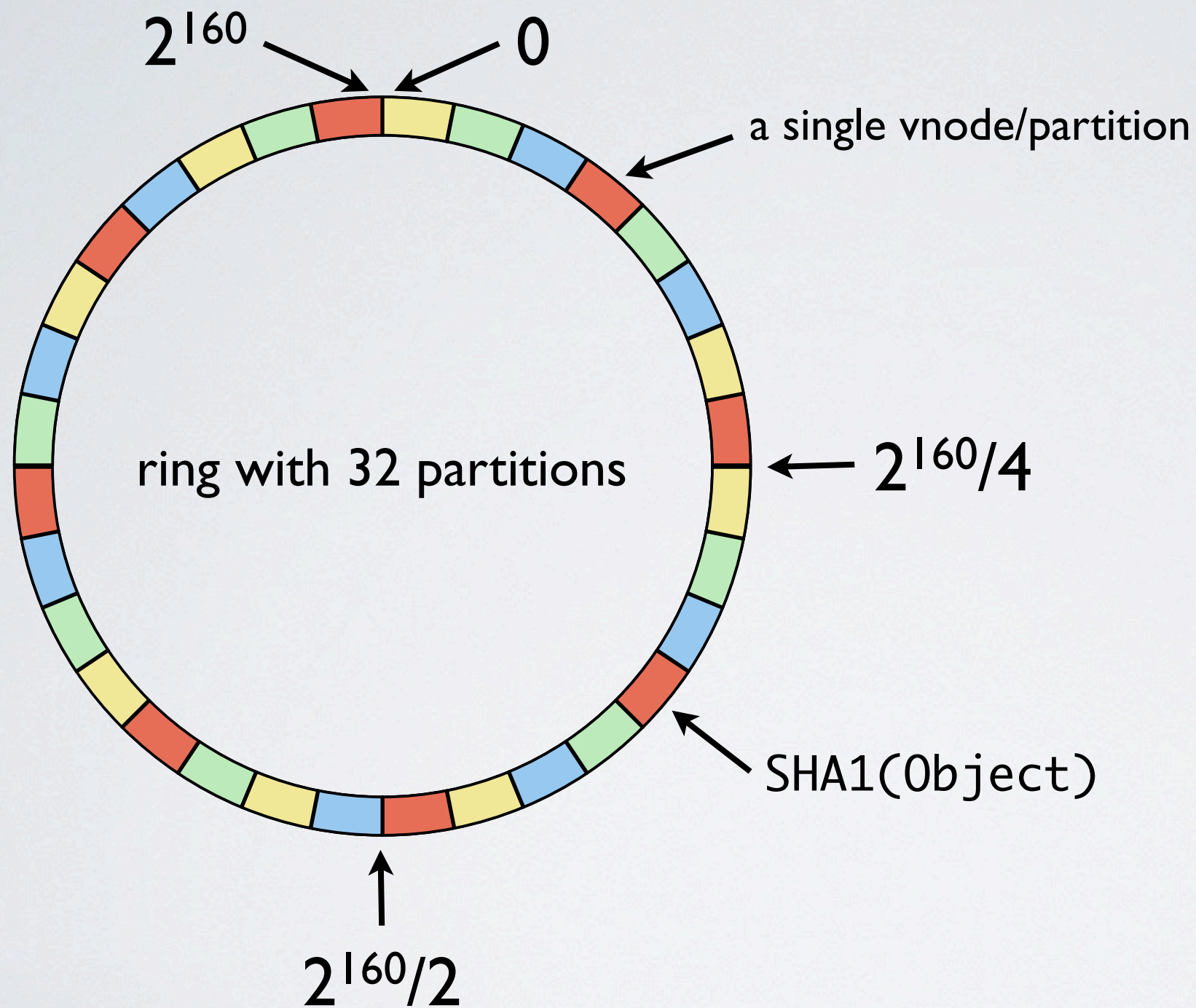


Node 0

Node 1

Node 2

Adding a node



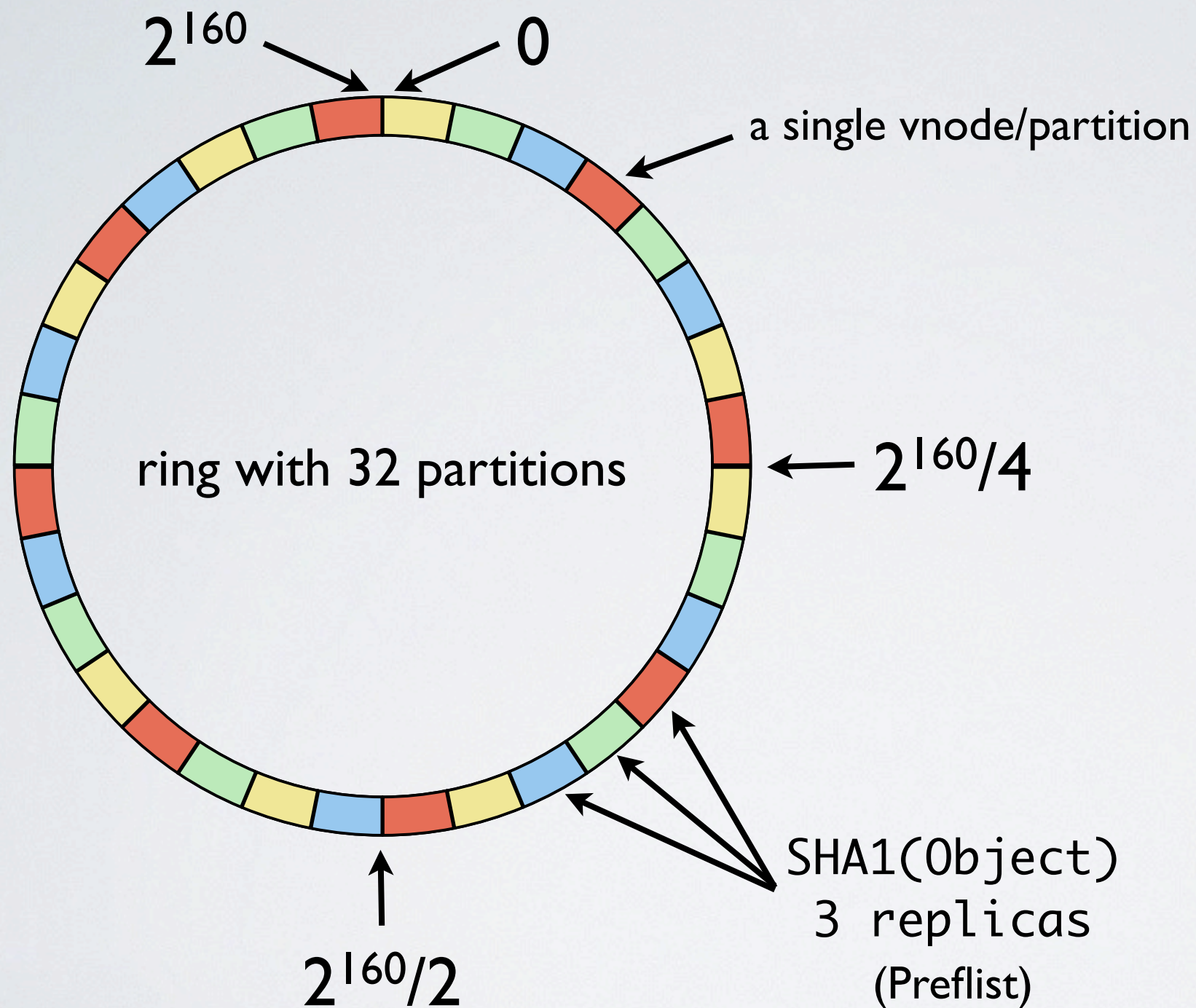
Node 0

Node 1

Node 2

Node 3

Replication



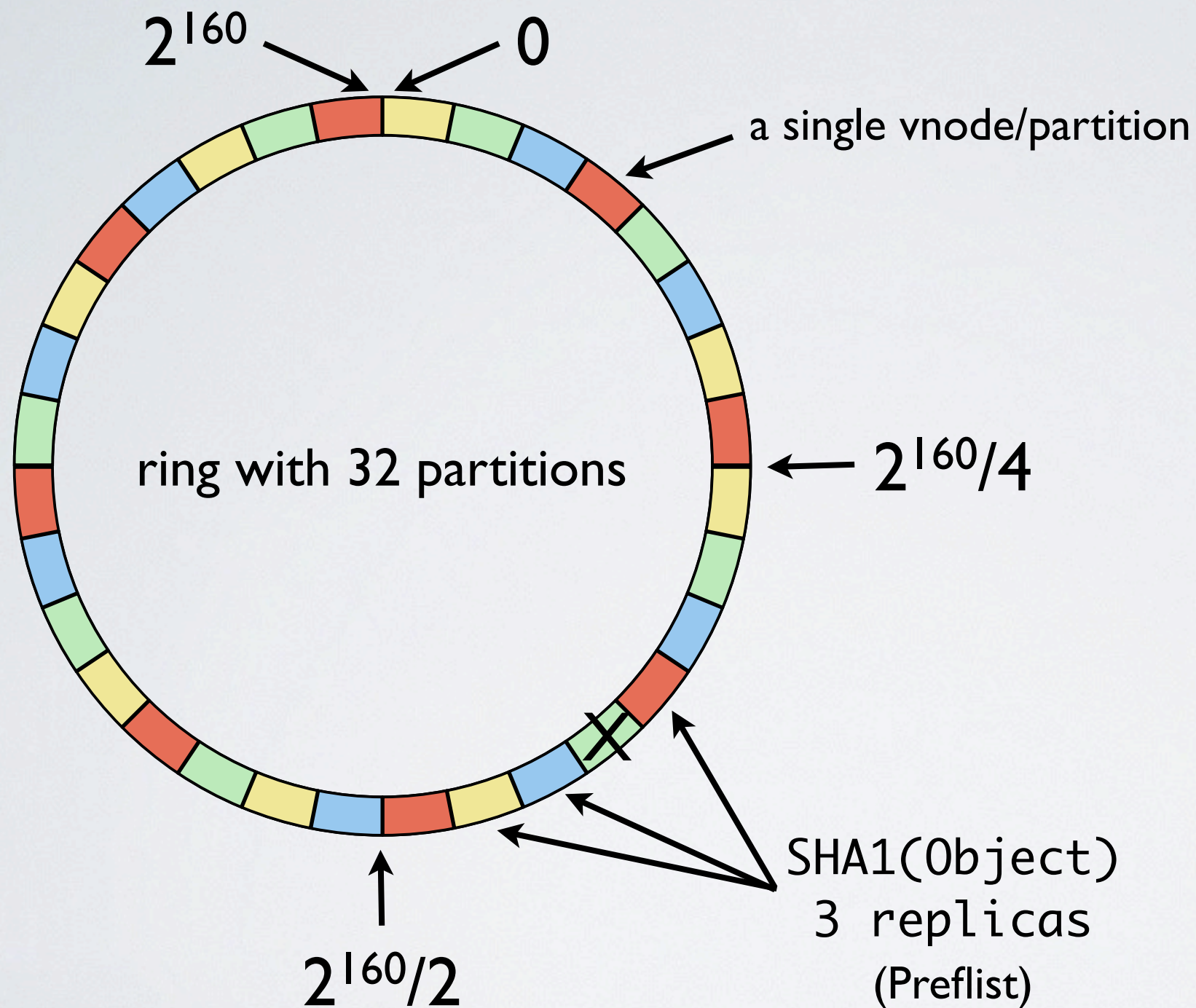
Node 0

Node 1

Node 2

Node 3

Routing around failures



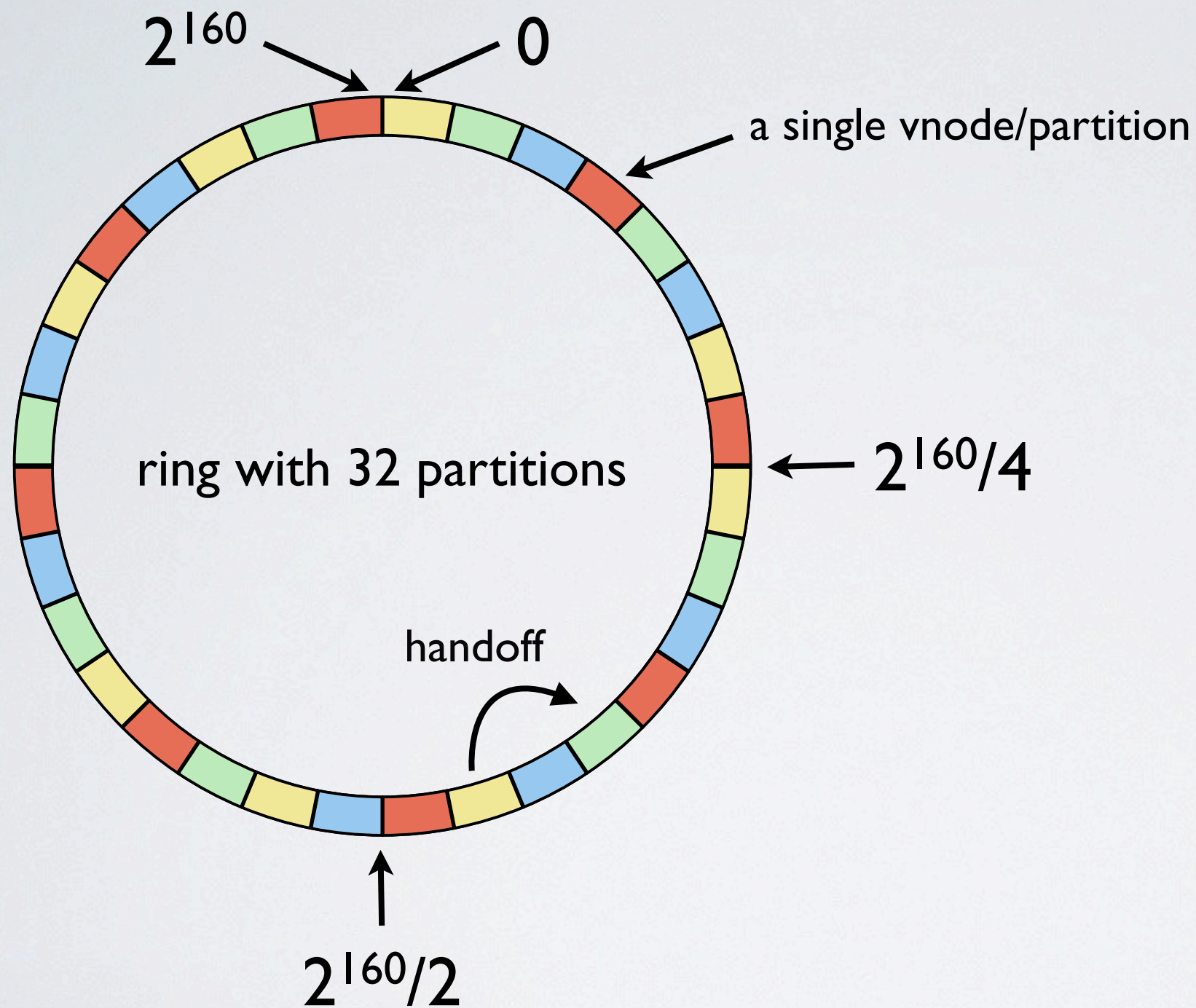
Node 0

Node 1

Node 2

Node 3

Hinted Handoff



Node 0

Node 1

Node 2

Node 3

Riak DT

- Riak Core Application
- Runs alongside Riak KV
- Own Storage

Riak DT

- HTTP API
- `-behaviour(riak_dt).`
- State-based

CRDT Behaviour

- **new/0** *empty CRDT*
- **value/1** *the resolved value*
- **update/3** *mutate CRDT*
- **merge/2** *converge two CRDTs*
- **equal/2** *compare internal value*

CRDTs implemented

■ Counters

- G-Counter
- PN-Counter

■ Sets

- G-Set
- OR-Set

G-Counter

- Simple version vector (28 LoC)
[{ **ActorId**, **Count** }]
- **Update**: increment actor's count
- **Merge**: greatest value per Actor
- **Value**: sum of Counts

G-Counter

```
new() ->  
  [].
```

```
value(GCnt) ->  
  sum([Cnt || {_Act, Cnt} <- GCnt]).
```

```
equal(VA, VB) ->  
  lists:sort(VA) == lists:sort(VB).
```

PN-Counter

```
{  
  P = [{a, 10}, {b, 2}],  
  N = [{a, 1}, {c, 5}]  
}
```

$$\begin{aligned} & (10 + 2) - (1 + 5) \\ &= 12 - 6 \\ &= 6 \end{aligned}$$

- 2 x G-Counter
- $P - N = \text{value}$

Riak DT In Action

- Bitcask storage per vnode
- Value / Update FSM per request
- Webmachine resource(s)
e.g. GET /counters/key

Update FSM

- Sync call **update** on vnode
 - Read, Local Update, Reply
- Async send **merge** to replicas
- Await **W** responses
- Reply to client

Value FSM (Read)

- Async call **value** on all replicas
- Await **R** replies
- Merge all replies with **merge/2**
- Return merged value to client
- Read Repair

Read Repair

- Compare answers to merged result using `equal/2`
- Send `merge` to stale replicas

Multi-Datacenter

- Behaviour addition
 - `rollup/2` *collapsed local view*
- Counters
 - Roll up all actors in cluster:
[`{ClusterId, Count}`]

Trade-Offs

- Update: **Primary only**
 - Secondary/Fallbacks may **Merge**
- Read-before-Write in the request path
- **PW=DW=1** by default

Garbage

- Counters

- Dead actors

- Sets

- Tombstones

Elegance = Punt

- Is GC non-monotonic?
- Needs consensus to collect
- Following research community

And then?

- Stats/Metrics & Polish
- Multi-Datacenter Replication
- Active Anti-Entropy

And then?

- KV as storage
- GC / low garbage datatypes
- Op based / hybrid

Open Source

PUBLIC basho / riak_dt

Pull Request

Watch

Star 6

Fork 0

Code

Network

Pull Requests 0

Issues 12

Wiki

Graphs

Admin

Edit

Riak core app for storing convergent data types — [Read more](#)

Clone in Mac

ZIP

HTTP

SSH

Git Read-Only

git@github.com:basho/riak_dt.git

Read+Write access

branch: master

Files

Commits

Branches 13

Tags

Downloads

Latest commit to the master branch

Start timeout timer from fsm init ...

 russelldb authored 5 days ago

commit 69d8b3060f

riak_dt /

name	age	message	history
 include	6 days ago	Test handoff behavior. [seancribbs]	
 src	2 days ago	Start timeout timer from fsm init [russelldb]	
 test	6 days ago	Test handoff behavior. [seancribbs]	
 .gitignore	a month ago	Ignore the .app file. [seancribbs]	
 .travis.yml	a month ago	Add Travis config. [seancribbs]	
 LICENSE	a month ago	Address review comments with LICENSE and type fix [russelldb]	