riak

Control your data, don't let it control you.

NoSQL Atlanta 2009

Justin Sheehy <justin@basho.com>

What is riak?

- a document-oriented database
- a decentralized key-value store
- a fault-tolerant storage solution
- nosql, http, scalable, distributed, reliable...

What is riak?

Influences:	Amazon’s Dynamo
	CAP Theorem
	The Web
	Ops Experience

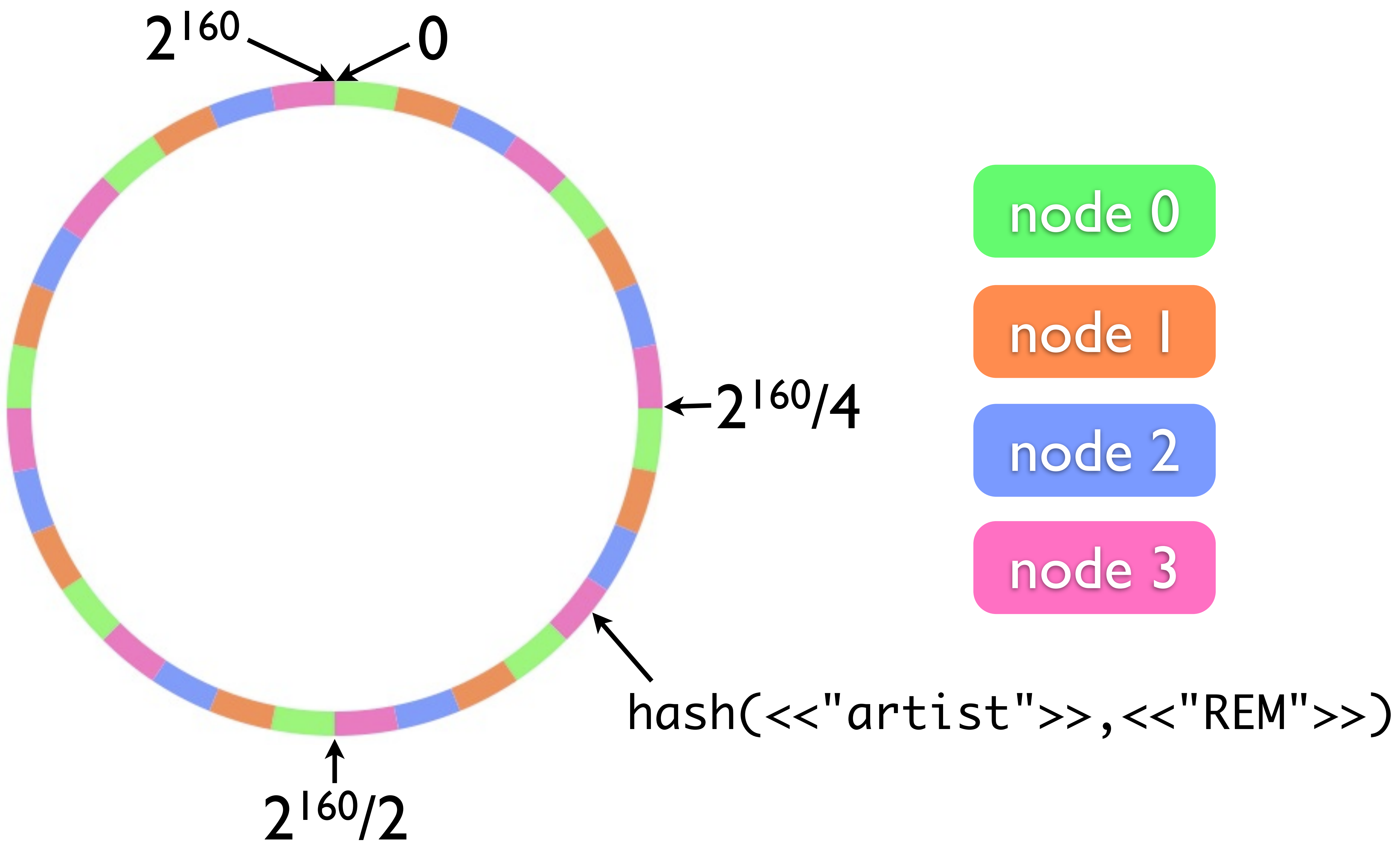
Basic N/R/W

N = number of replicas to store

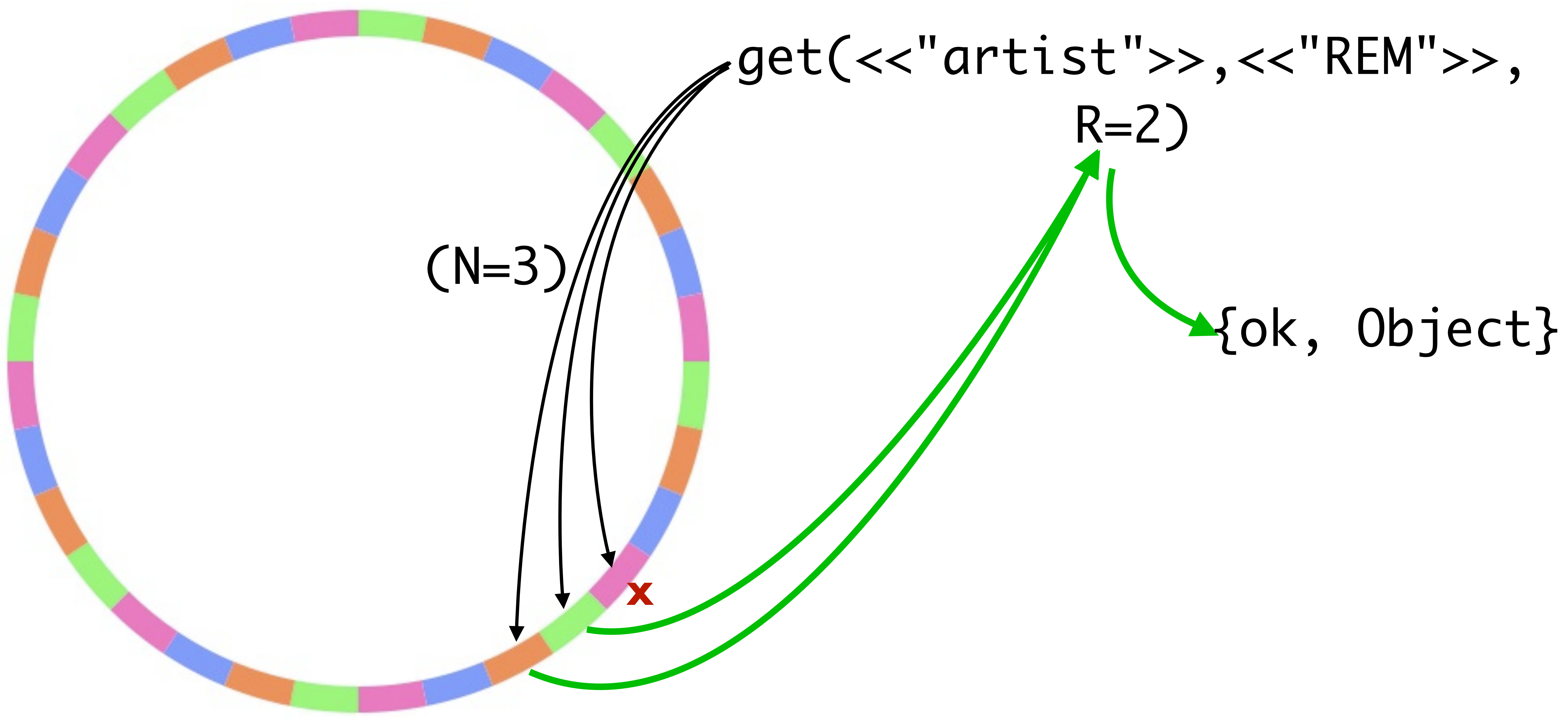
R = number of replicas needed for a read

W = number of replicas needed for a write

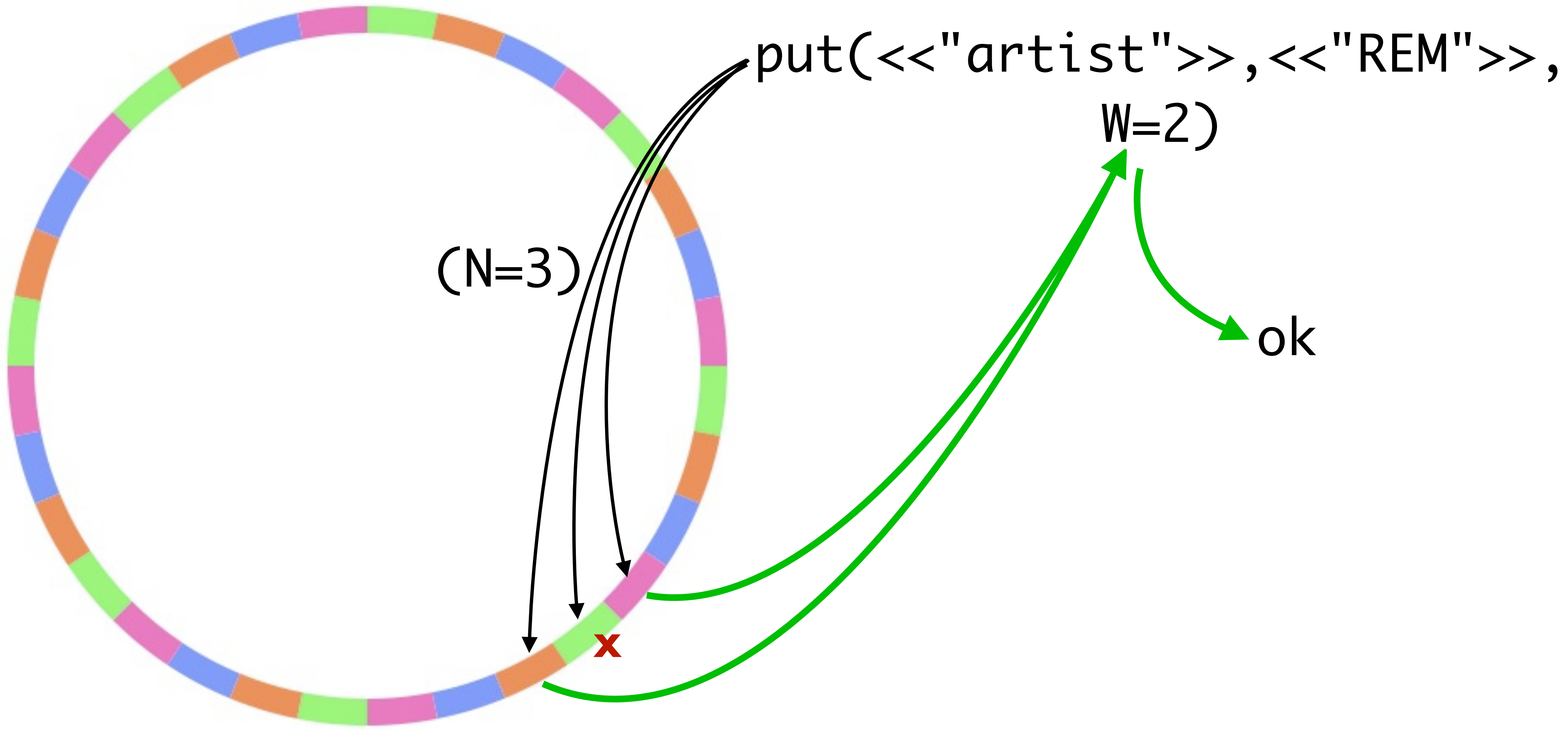
N value



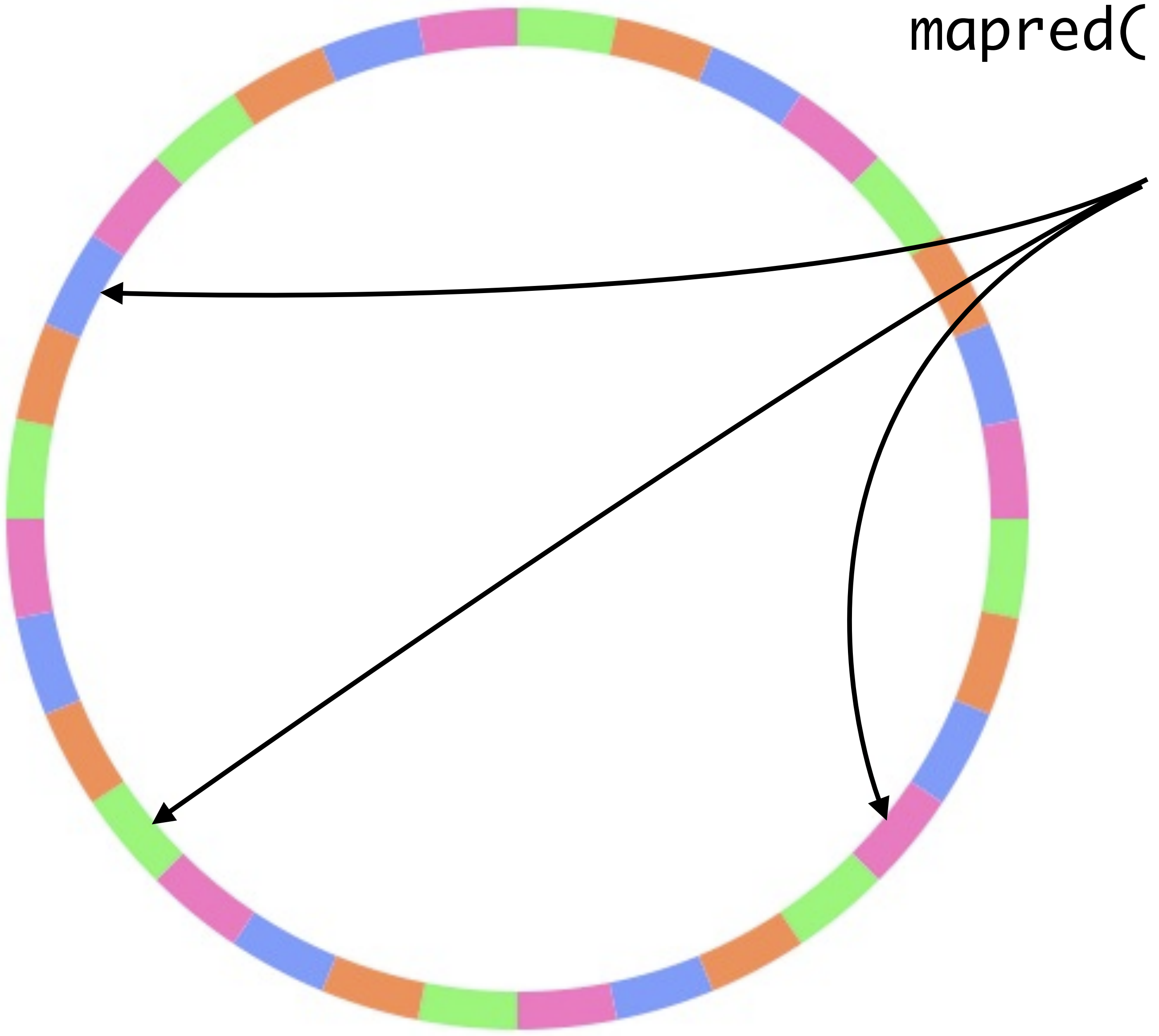
R value



W value

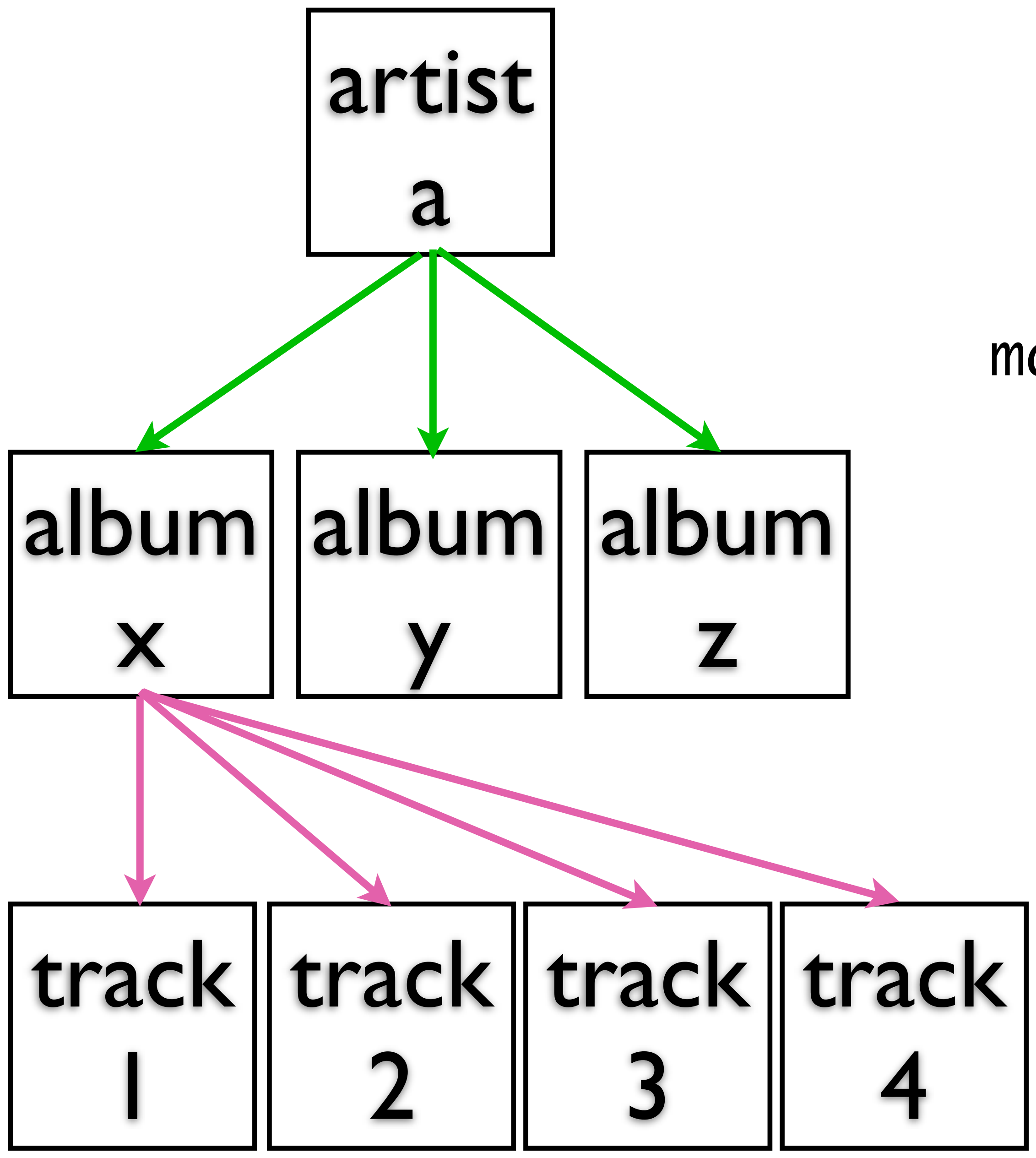


Map/Reduce



```
mapred([{<<"artist">>,<<"REM">>},
        {<<"artist">>,...},...],
       [{map,
         {modfun,artist,member_count},
         none,false},
        {reduce,
         {qfun,fun(L,_,_) ->
                   lists:unique(L)
               end},
         none, true}]).
```

Links



```
mapred([{<<"artist">>,<<"REM">>}],
        [{link,<<"album">>,'_',false},
         {link,<<"track">>,'_',false},
         {map,{modfun,track,name},
          none, true}]).
```

http://host/jiak/artist/REM/album,_,_/track,_,_

Basic CAP

Consistency

Availability

Partition tolerance

Basic CAP

Consistency

Availability

Partition tolerance

“Pick two.”

Basic CAP

Consistency

Availability

Partition tolerance

~~“Pick two.”~~

Basic CAP

Consistency

Availability

Partition tolerance

~~“Pick two.”~~

“Choose your own levels.”

HTTP API

- **`GET /jiak/<bucket>`**
- **`POST /jiak/<bucket>`**
- **`GET /jiak/<bucket>/<key>`**
- **`PUT /jiak/<bucket>/<key>`**
- **`DELETE /jiak/<bucket>/<key>`**

Scalable

"I can add twice as much X to get twice as much Y."

Scalable

$$\begin{tikzcd}\text{computers} \\ \text{"I can add twice as much X to get twice as much Y."} \arrow[u]\end{tikzcd}$$

Scalable

computers

"I can add twice as much X to get twice as much Y."

write-throughput!

storage capacity!

map/red power!

Linearly Scalable

computers

"I can add twice as much X to get twice as much Y."

write-throughput!

storage capacity!

map/red power!

Distributed

Composed of multiple systems, working separately but in harmony.

Distributed, Decentralized, Homogenous

Composed of multiple systems, working separately but in harmony.

No bottlenecks, no SPOFs, no "special" nodes.

Reliable

Doesn't fail?

"Reliable" is tricky.

Everything fails. Ask your sysadmin.

What do we really mean?

Resilient

Assume that failures will happen.

Designing whole systems and components
with individual failures in mind
is a plan for predictable success.

Eventually Consistent

Brief sacrifices of consistency in failure conditions.

CAP tells us we need this when we want availability and partition-tolerance.

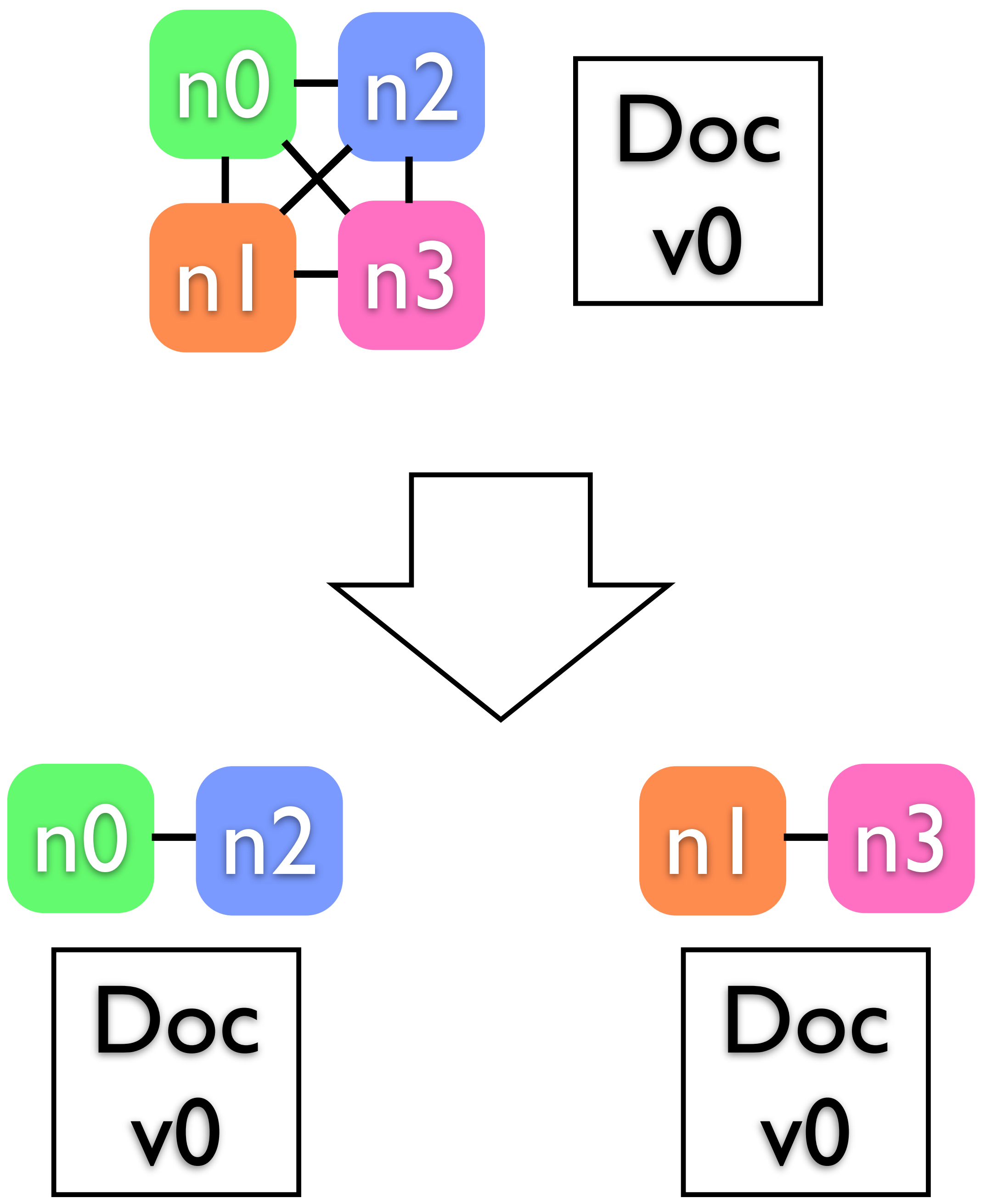
Eventually Consistent

Brief sacrifices of consistency in failure conditions.

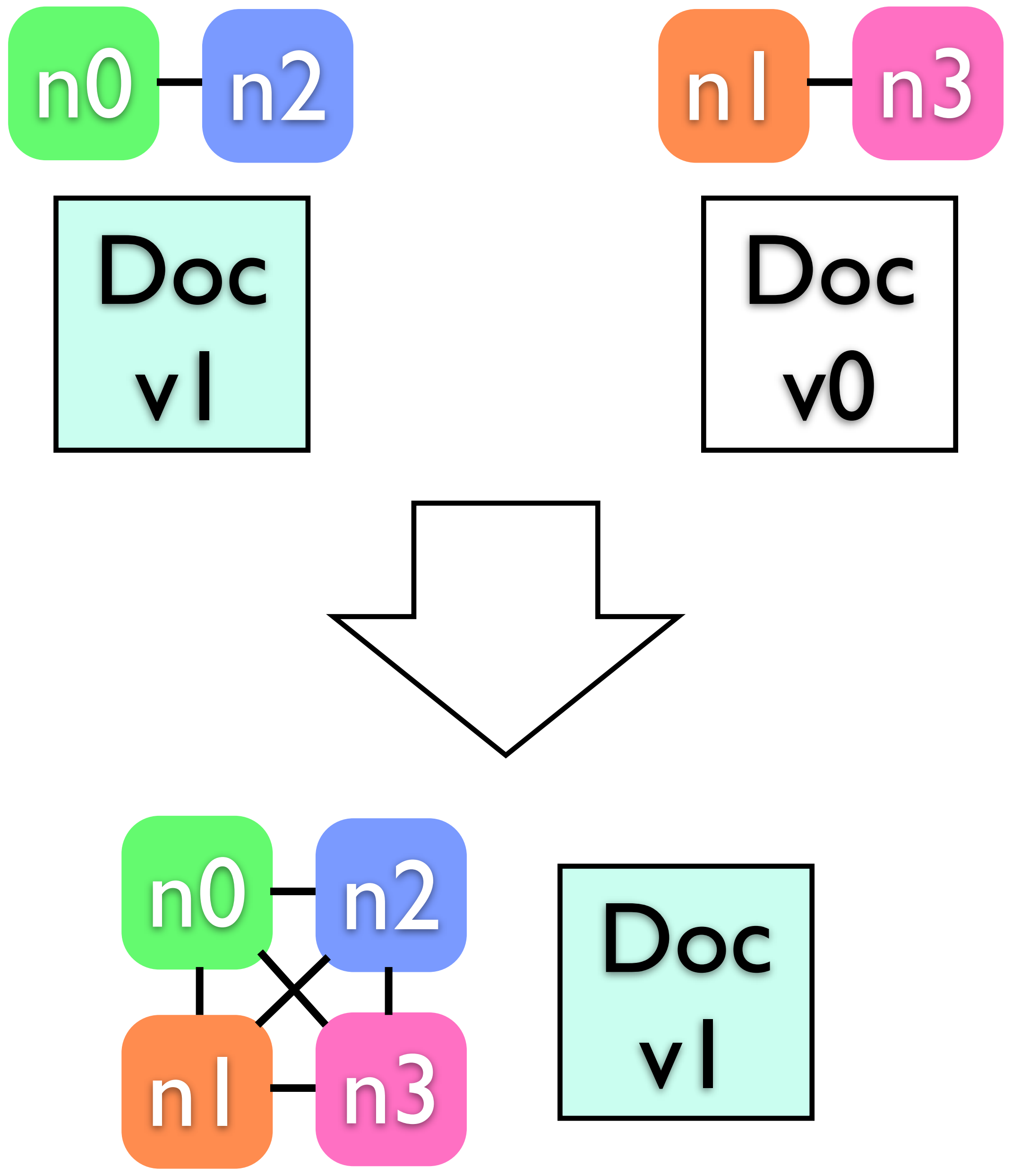
CAP tells us we need this when we want availability and partition-tolerance.

Loss of user data is never acceptable!

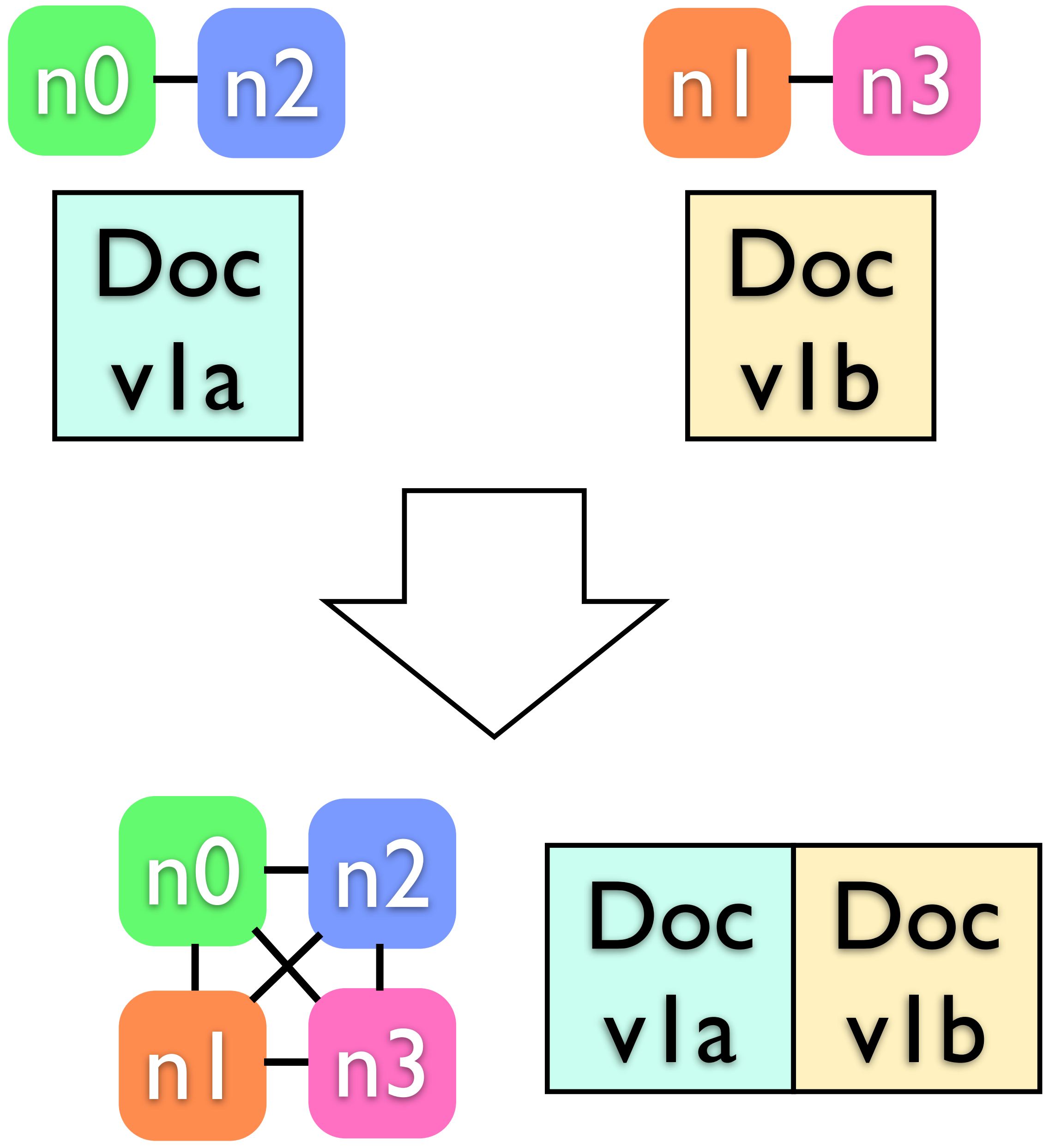
Partition-Tolerance



Partition-Tolerance



Partition-Tolerance



Easy to Use and Operate

devs:

- restful http and json, schemaless
- simple put/get/delete
- scales down easy to your laptop
- ...

ops:

- adding and removing nodes
- managing alerts
- changing configuration over time
- ...

Control your data, don't let it control you.

http://riak.basho.com/

riak-users@lists.basho.com