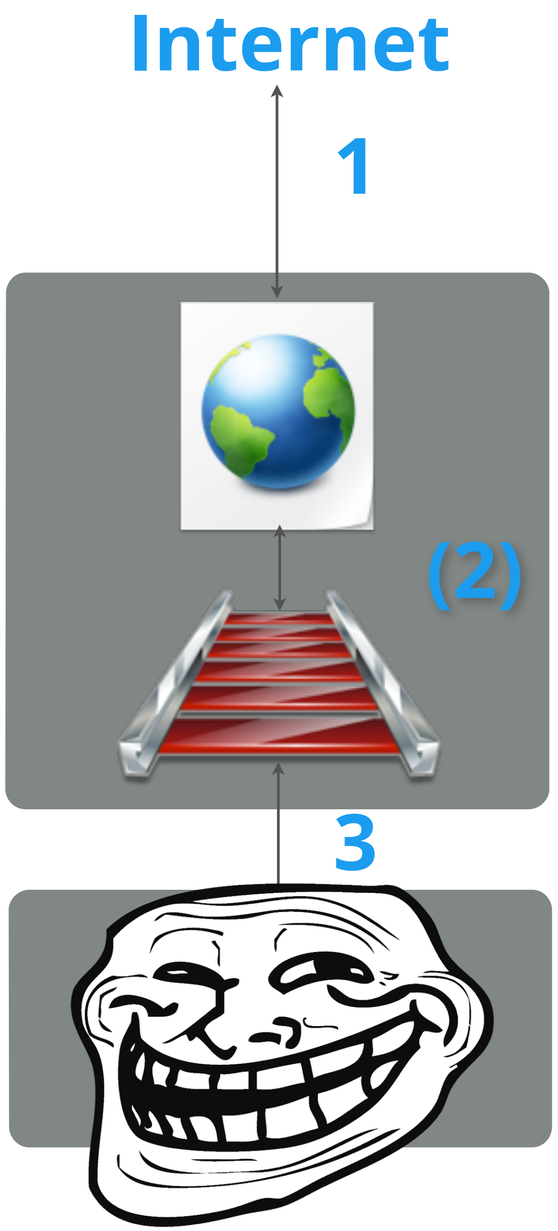
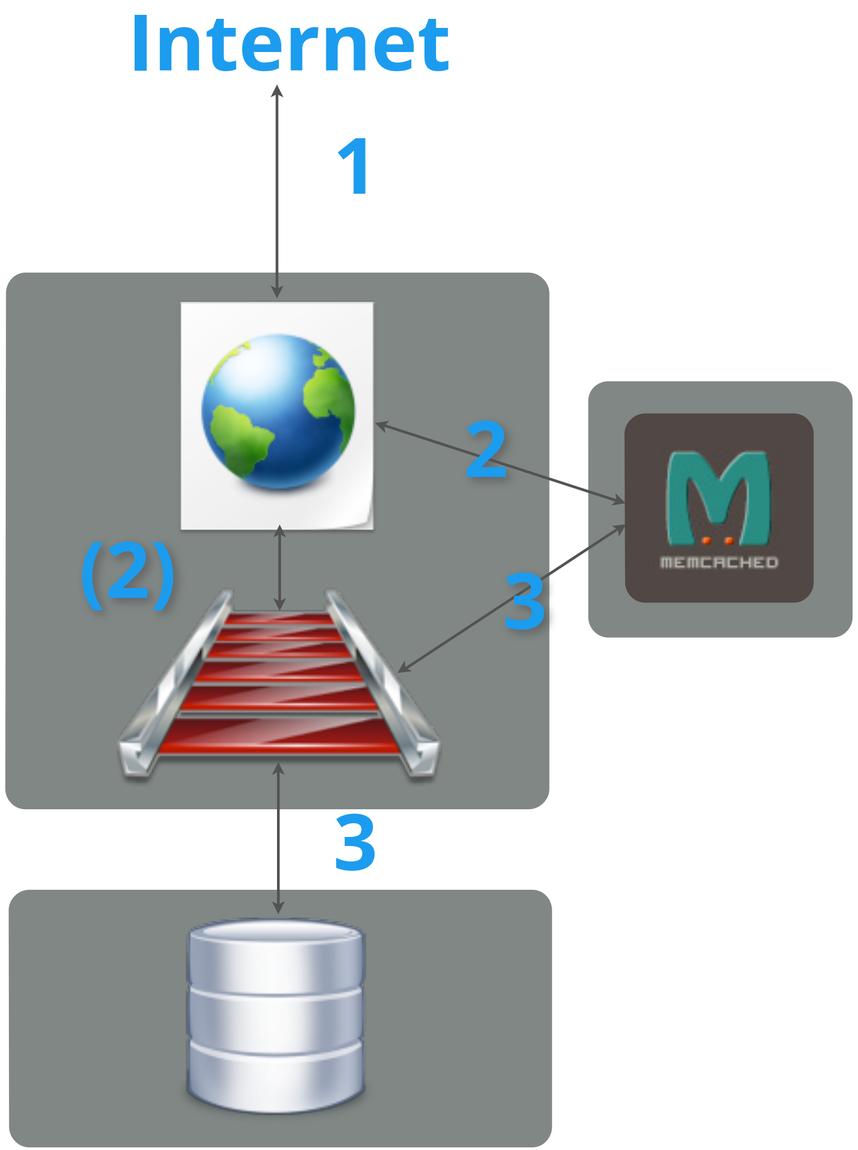
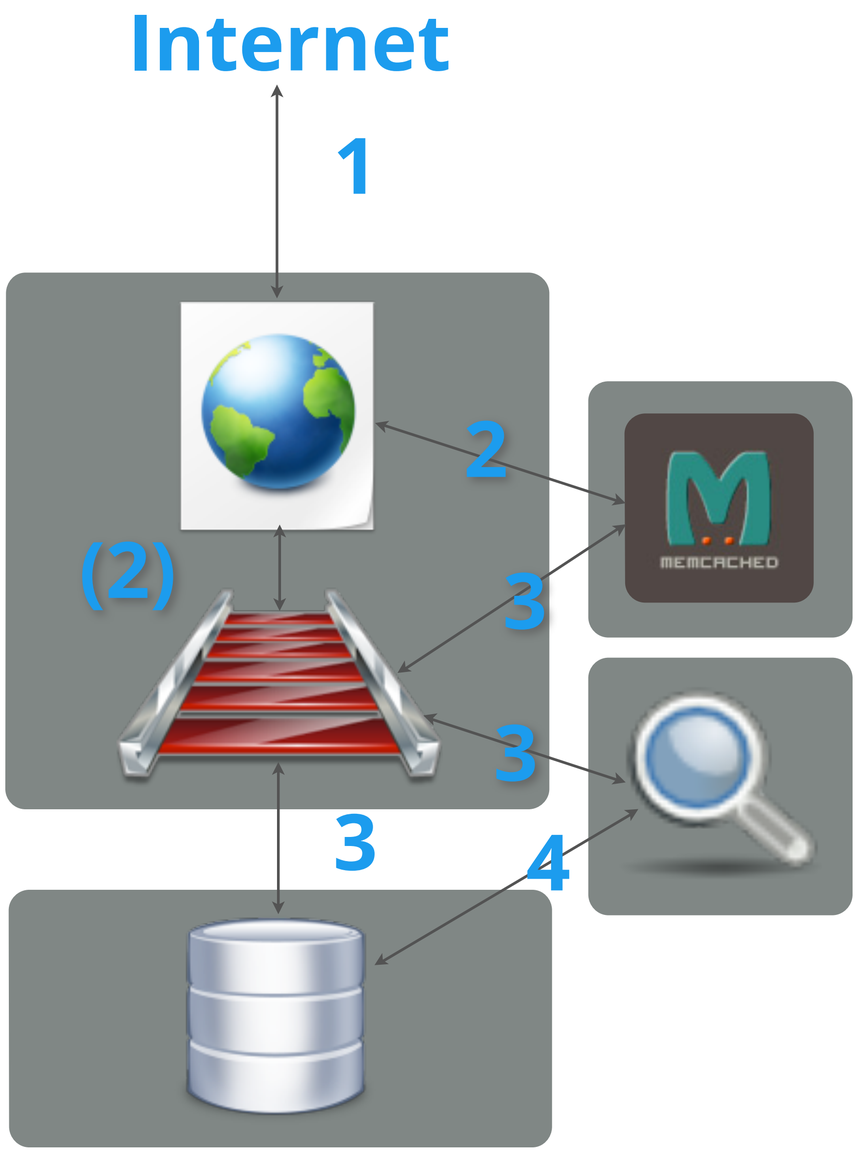
Scaling Data Safely *with* Riak

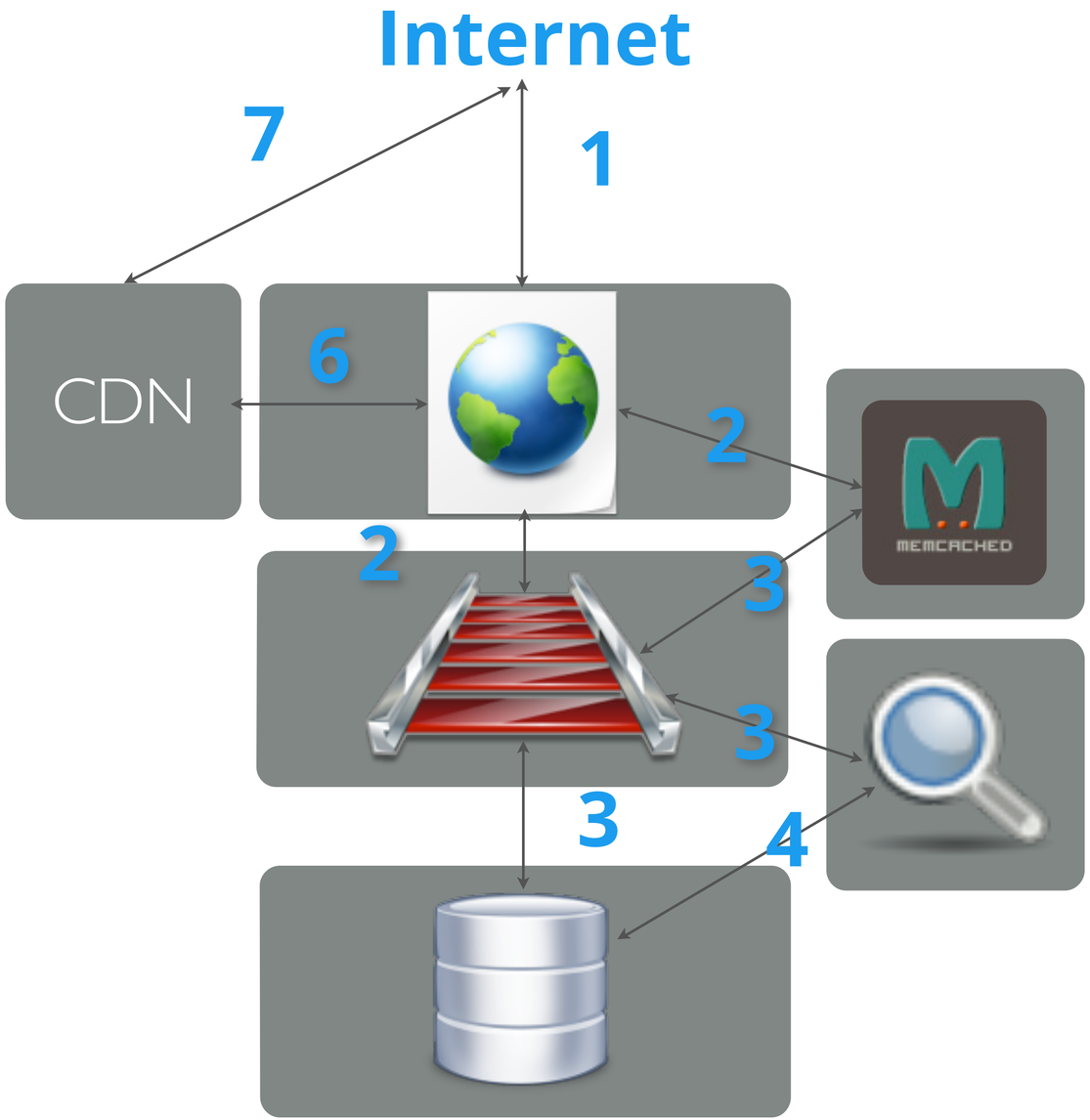
@alexmoore

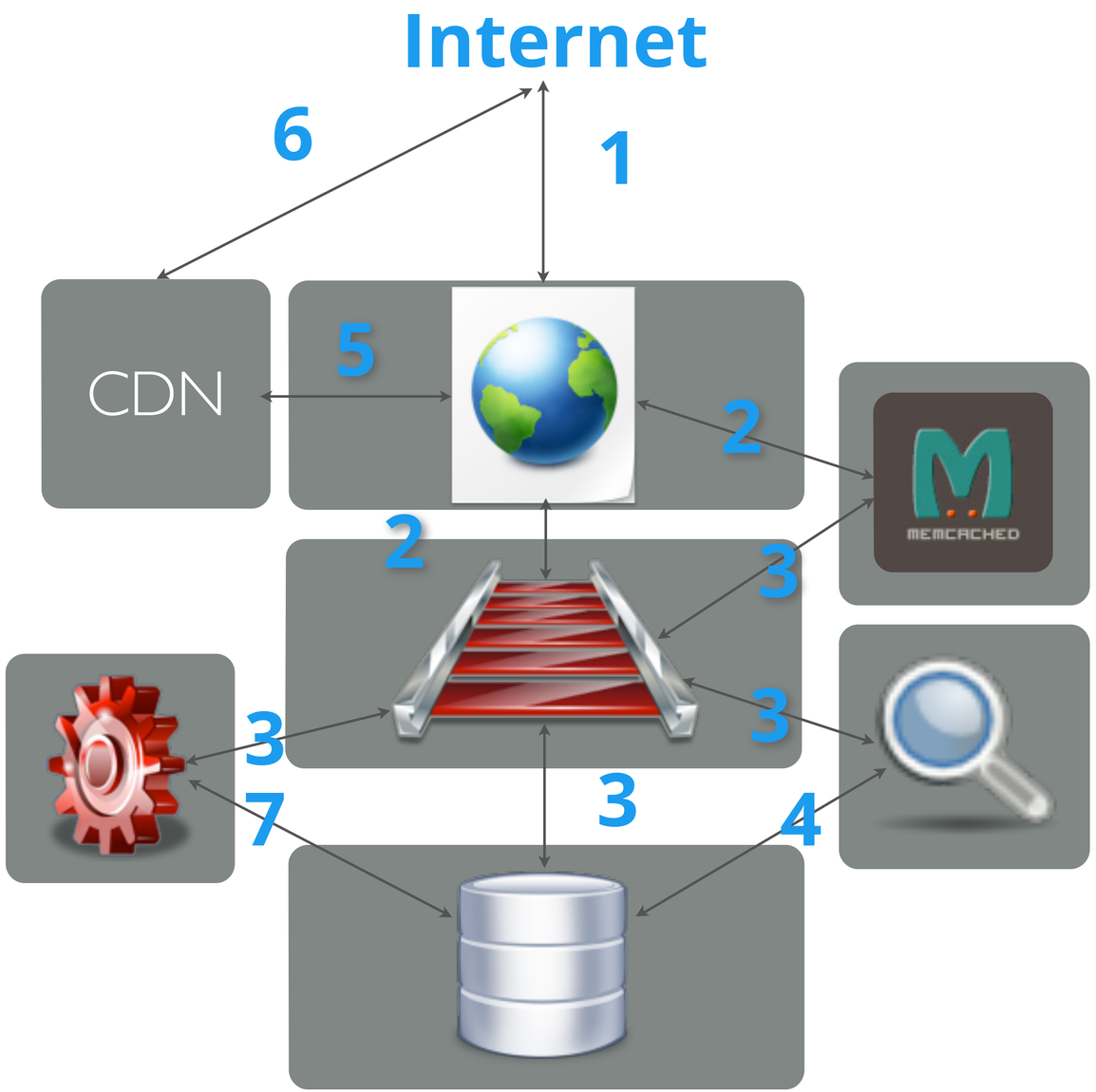
Timeout
1
503

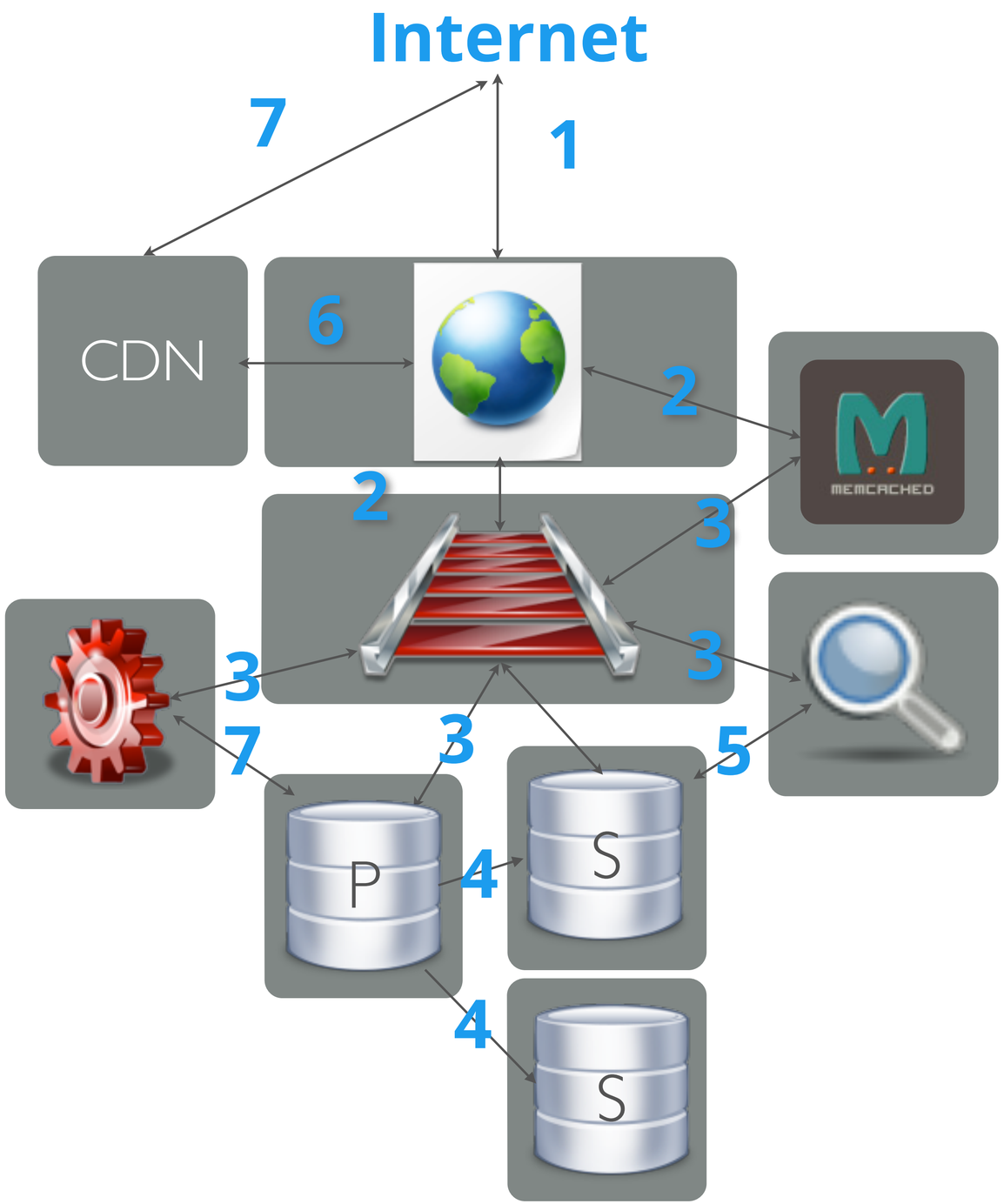

Internet
1
(2)
3


Internet
1
2
(2)
3
3
MEMCACHED


Internet
1
2
(2)
3
3
3
4
MEMCACHED

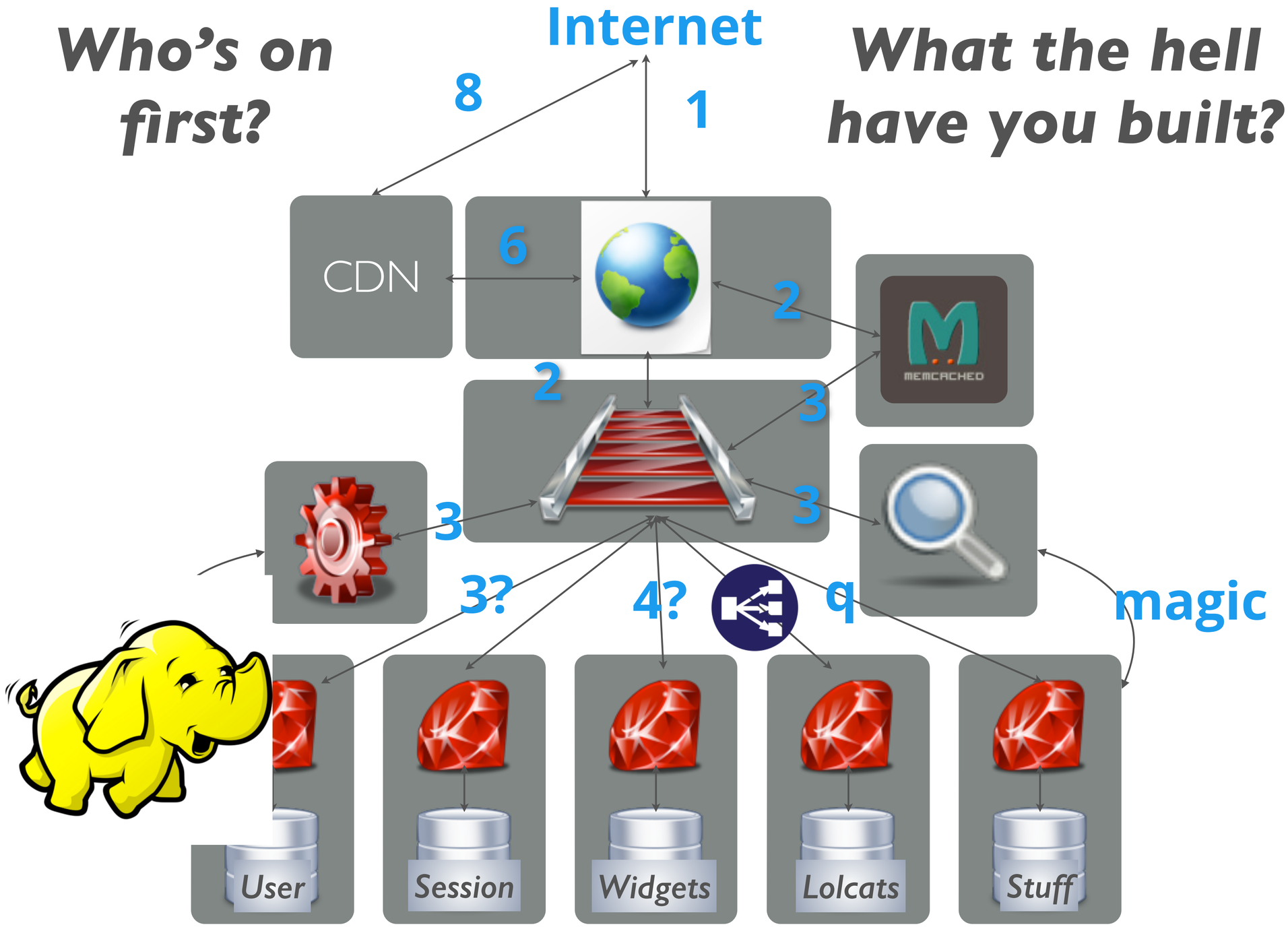

Internet
7
1
CDN
6
2
2
3
3
3
4
MEMCACHED


Internet
6
1
CDN
5
2
2
3
MEMCACHED
3
3
3
7
4


Internet
7
1
CDN
6
2
2
MEMCACHED
3
3
3
7
3
5
P
4
S
4
S

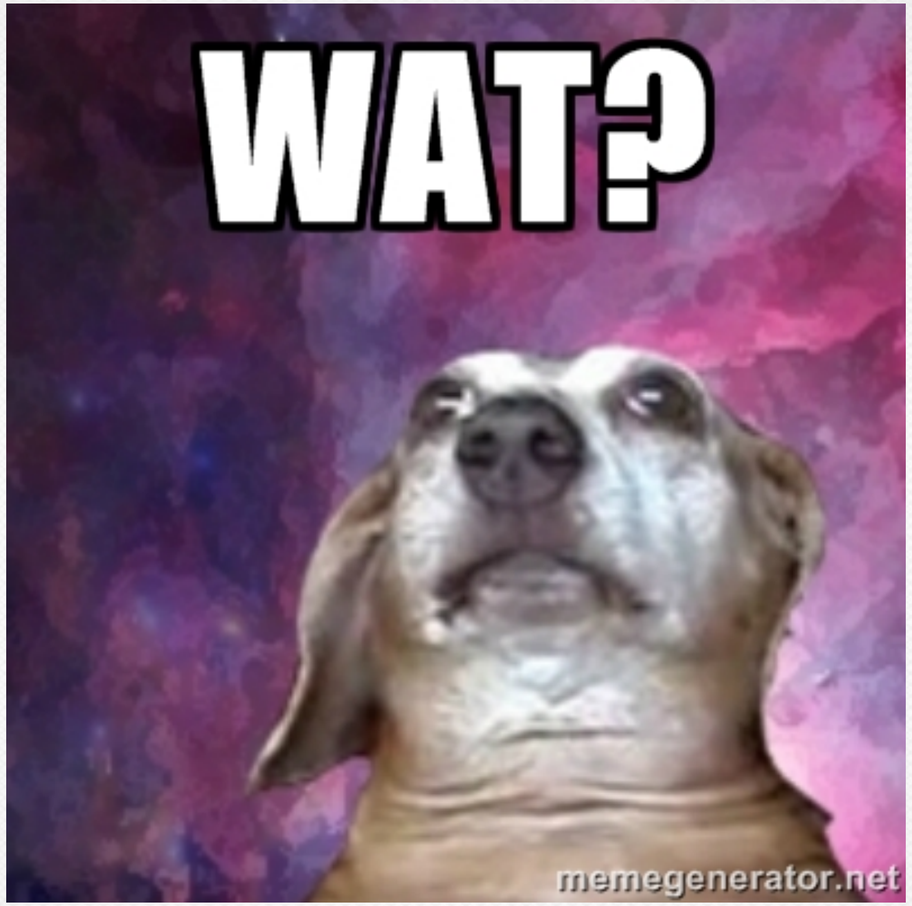
Who's on first?

What the hell have you built?



Enter Riak

Riak is a distributed, fault-tolerant, highly available database


WAT?
memegenerator.net

Distributed

a cluster of **nodes**

Distributed

each node has
an identical role

Fault-tolerant

symmetrical

Fault-Tolerant

no Single Point of Failure

Highly Available

Highly Available

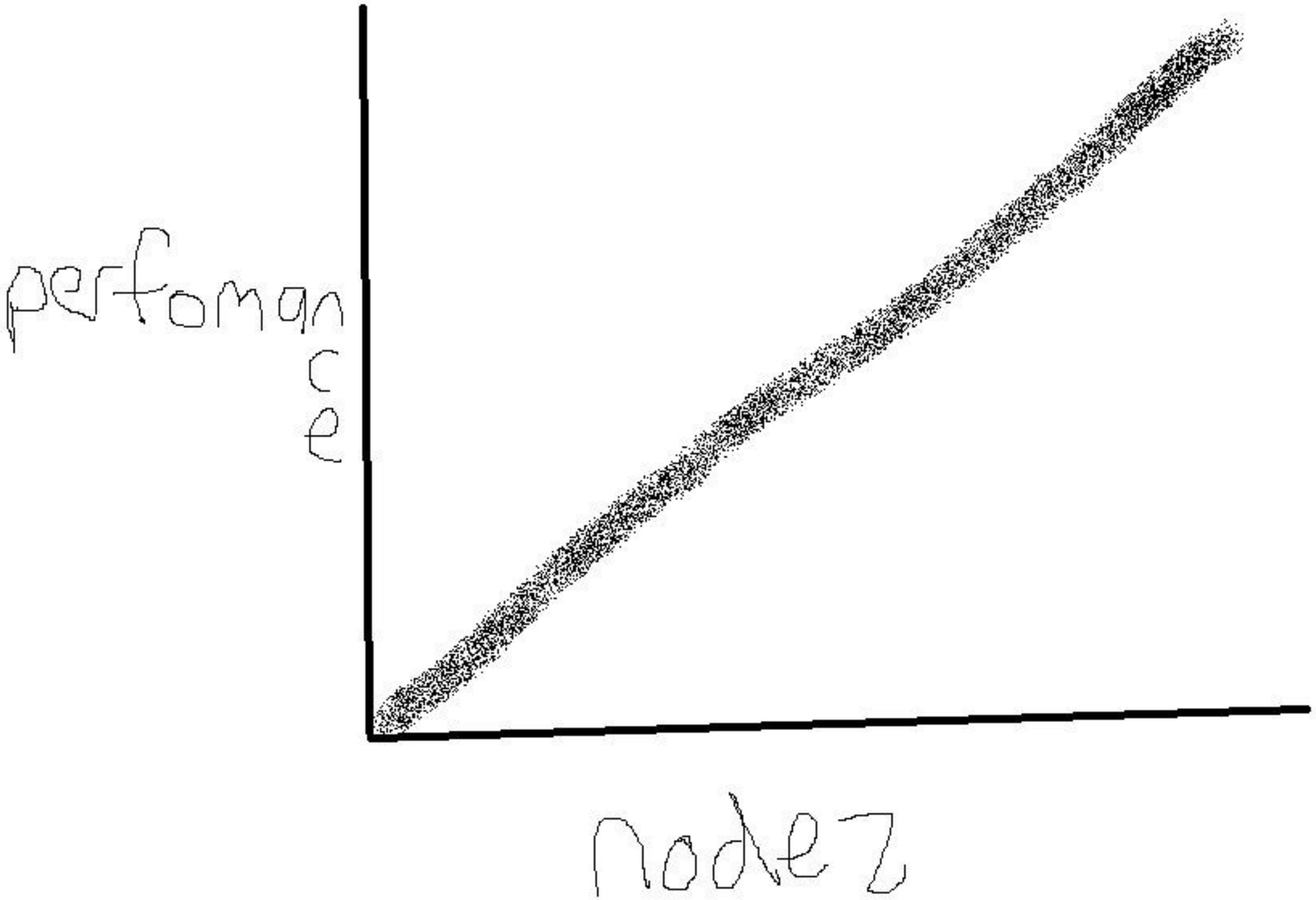
replicas & redundancy

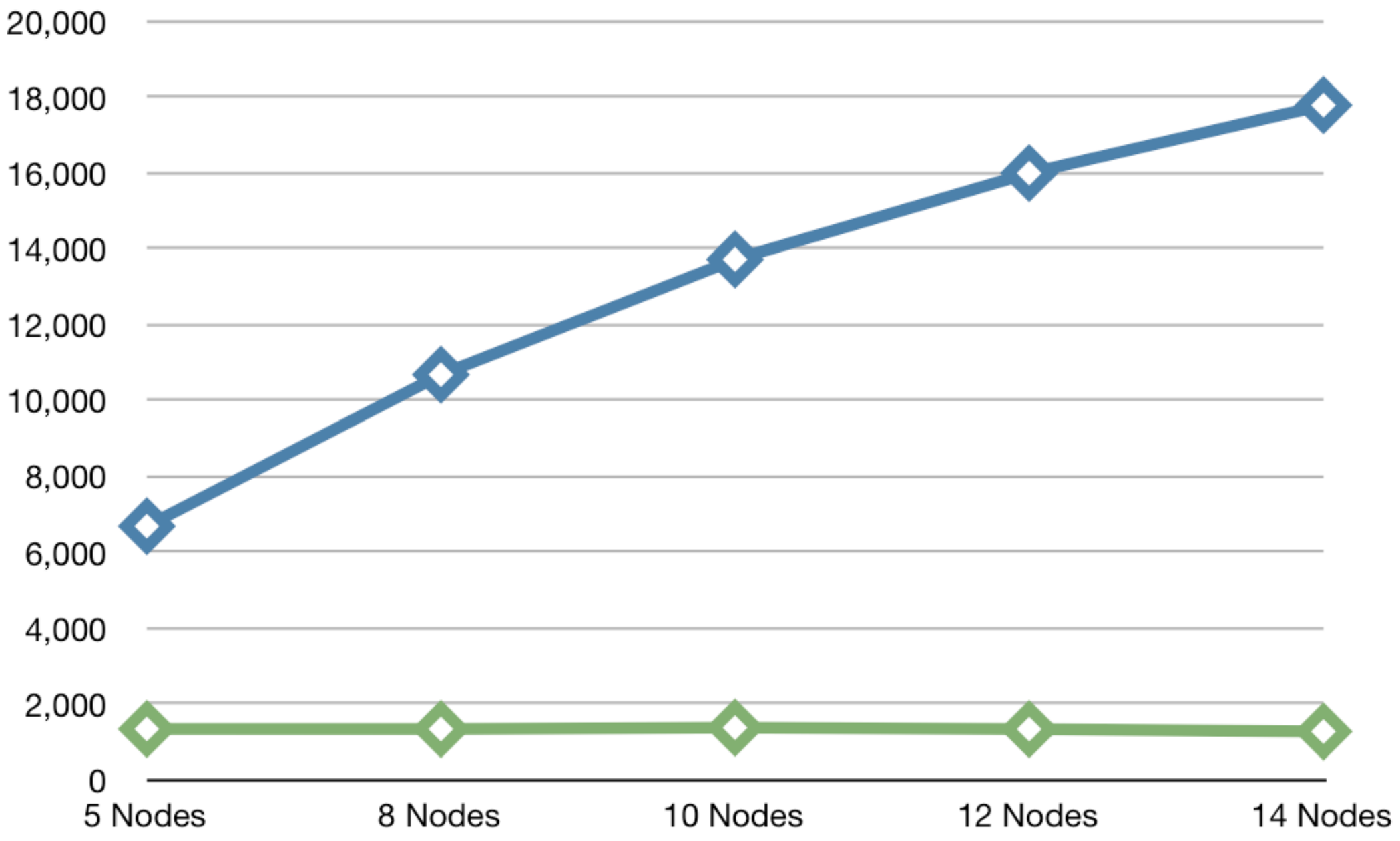
Scalable

add or remove nodes

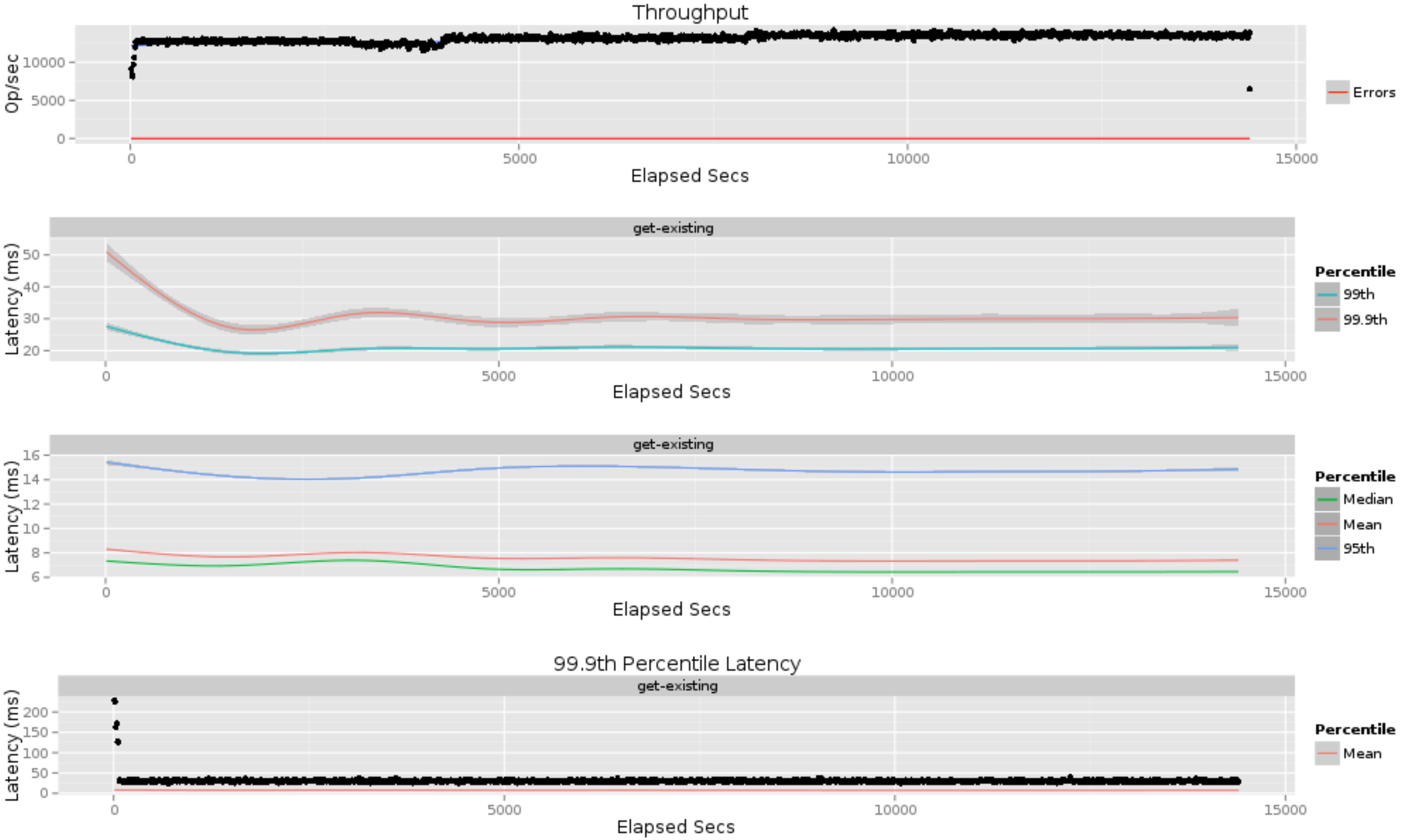
Scalable

linear scalability


performance
nodez

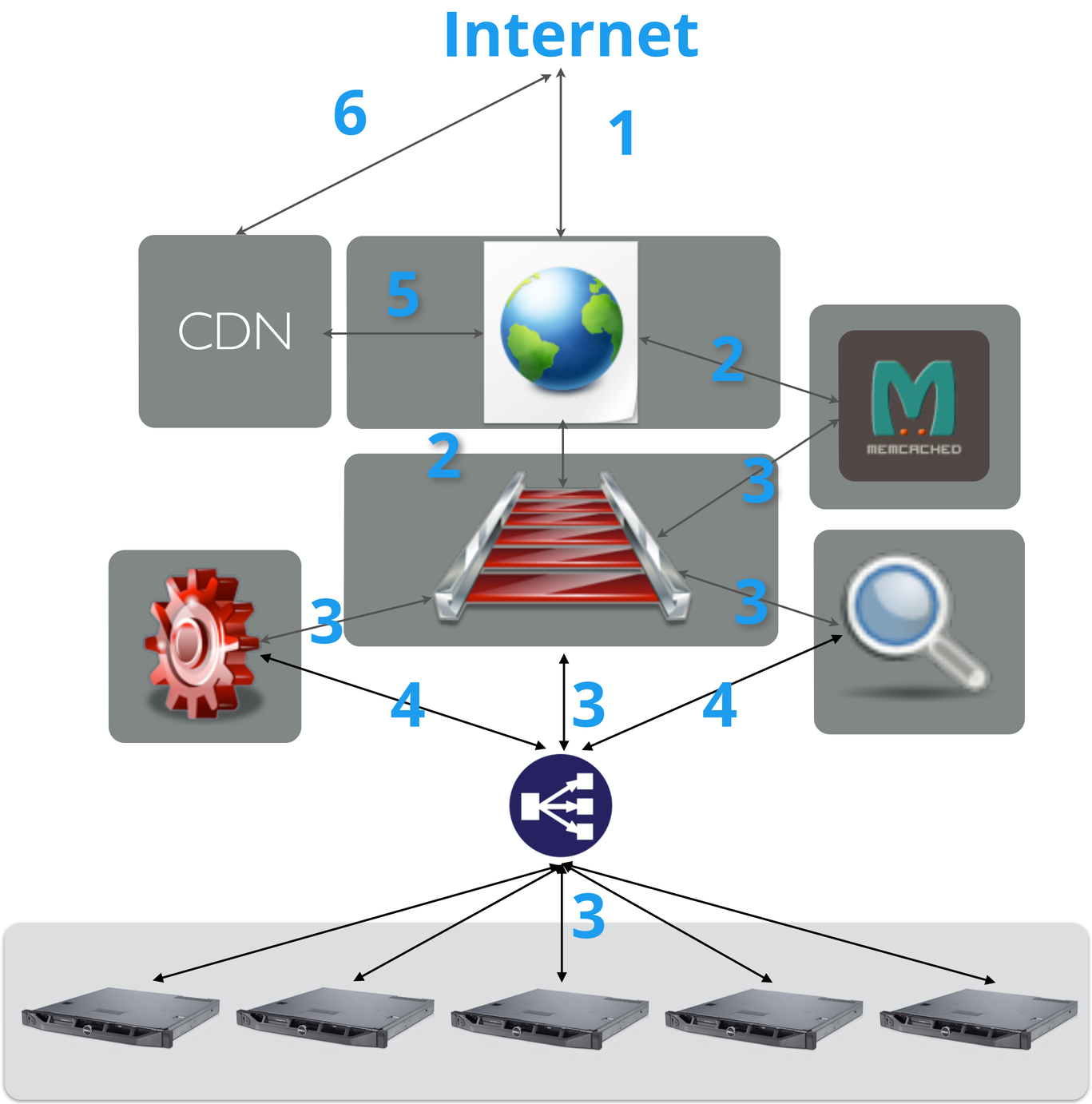

20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
5 Nodes
8 Nodes
10 Nodes
12 Nodes
14 Nodes

Predictable Performance


Throughput
Op/sec
10000
5000
0
0
5000
10000
15000
Elapsed Secs
Errors
get-existing
Latency (ms)
50
40
30
20
0
5000
10000
15000
Elapsed Secs
Percentile
99th
99.9th
get-existing
Latency (ms)
16
14
12
10
8
6
0
5000
10000
15000
Elapsed Secs
Percentile
Median
Mean
95th
99.9th Percentile Latency
get-existing
Latency (ms)
200
150
100
50
0
0
5000
10000
15000
Elapsed Secs
Percentile
Mean

Multiple Platforms

Multiple Platforms


Internet
6
1
CDN
5
2
2
MEMCACHED
3
3
3
4
3
4
3

Building A Cluster

```
# All Nodes
$ riak start

# Nodes 2..n.each
$ riak-admin cluster join riak@node1.company.com

# Any Node
$ riak-admin cluster plan
$ riak-admin cluster commit
```

```
$./riak-admin cluster plan
================================ Staged Changes ==================
Action         Details(s)
------------------------------------------------------------------
join           'dev5@127.0.0.1'
------------------------------------------------------------------


NOTE: Applying these changes will result in 1 cluster transition

##################################################################
                         After cluster transition 1/1
##################################################################

================================= Membership ====================
Status     Ring    Pending    Node
------------------------------------------------------------------
valid      25.0%     18.8%    'dev1@127.0.0.1'
valid      25.0%     18.8%    'dev2@127.0.0.1'
valid      25.0%     18.8%    'dev3@127.0.0.1'
valid      25.0%     25.0%    'dev4@127.0.0.1'
valid       0.0%     18.8%    'dev5@127.0.0.1'
------------------------------------------------------------------
Valid:5 / Leaving:0 / Exiting:0 / Joining:0 / Down:0

Transfers resulting from cluster changes: 12
  4 transfers from 'dev3@127.0.0.1' to 'dev5@127.0.0.1'
  4 transfers from 'dev2@127.0.0.1' to 'dev5@127.0.0.1'
  4 transfers from 'dev1@127.0.0.1' to 'dev5@127.0.0.1'
```

```
$./riak-admin transfers
'dev3@127.0.0.1' waiting to handoff 3 partitions
'dev2@127.0.0.1' waiting to handoff 4 partitions
'dev1@127.0.0.1' waiting to handoff 2 partitions

Active Transfers:


$./riak-admin member-status
================================= Membership ==========================
Status     Ring    Pending    Node
-------------------------------------------------------------------------
valid      18.8%      --      'dev1@127.0.0.1'
valid      18.8%      --      'dev2@127.0.0.1'
valid      18.8%      --      'dev3@127.0.0.1'
valid      25.0%      --      'dev4@127.0.0.1'
valid      18.8%      --      'dev5@127.0.0.1'
-------------------------------------------------------------------------
Valid:5 / Leaving:0 / Exiting:0 / Joining:0 / Down:0
```

Architecture

(safety built in)

Riak is the
ops-friendly
database

Riak is

distributed

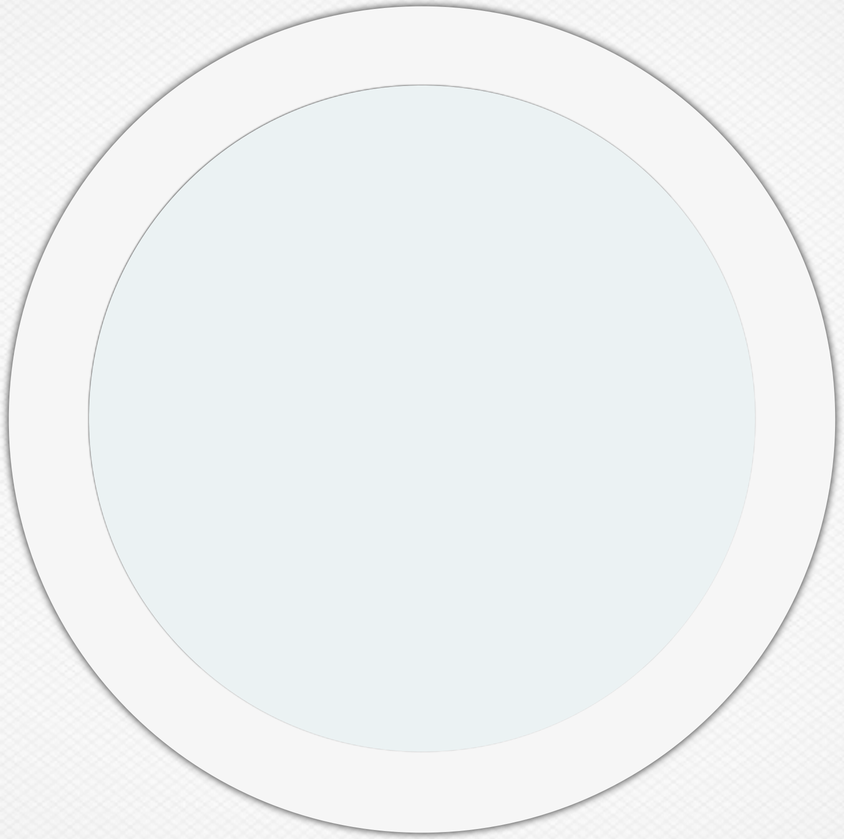
Consistent Hashing

hash(“**bucket**/**key**”)

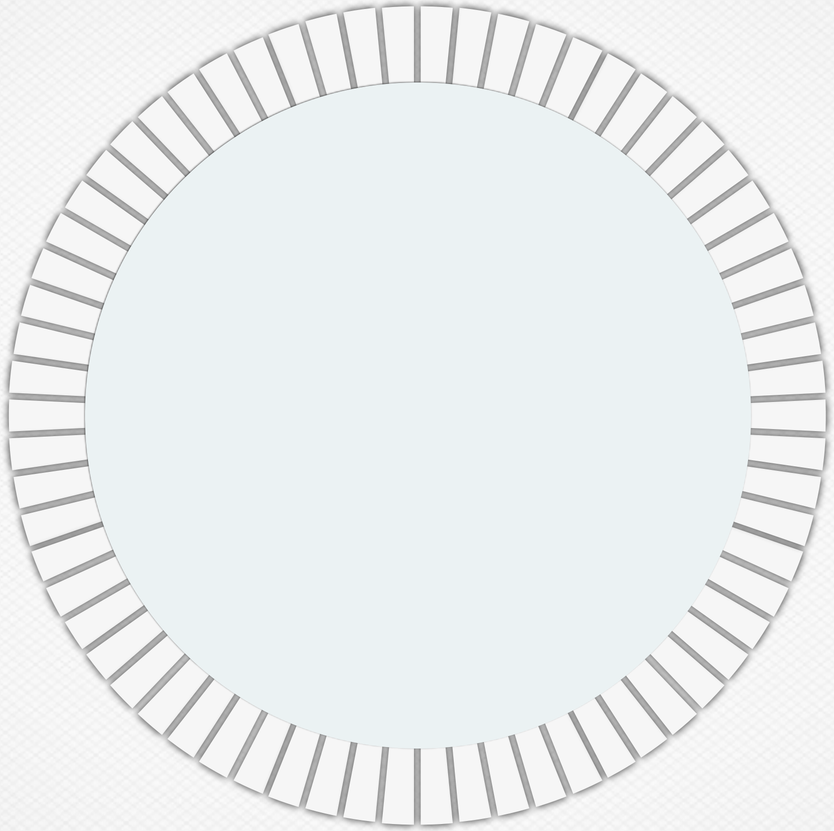
$0 - 2^{160}$

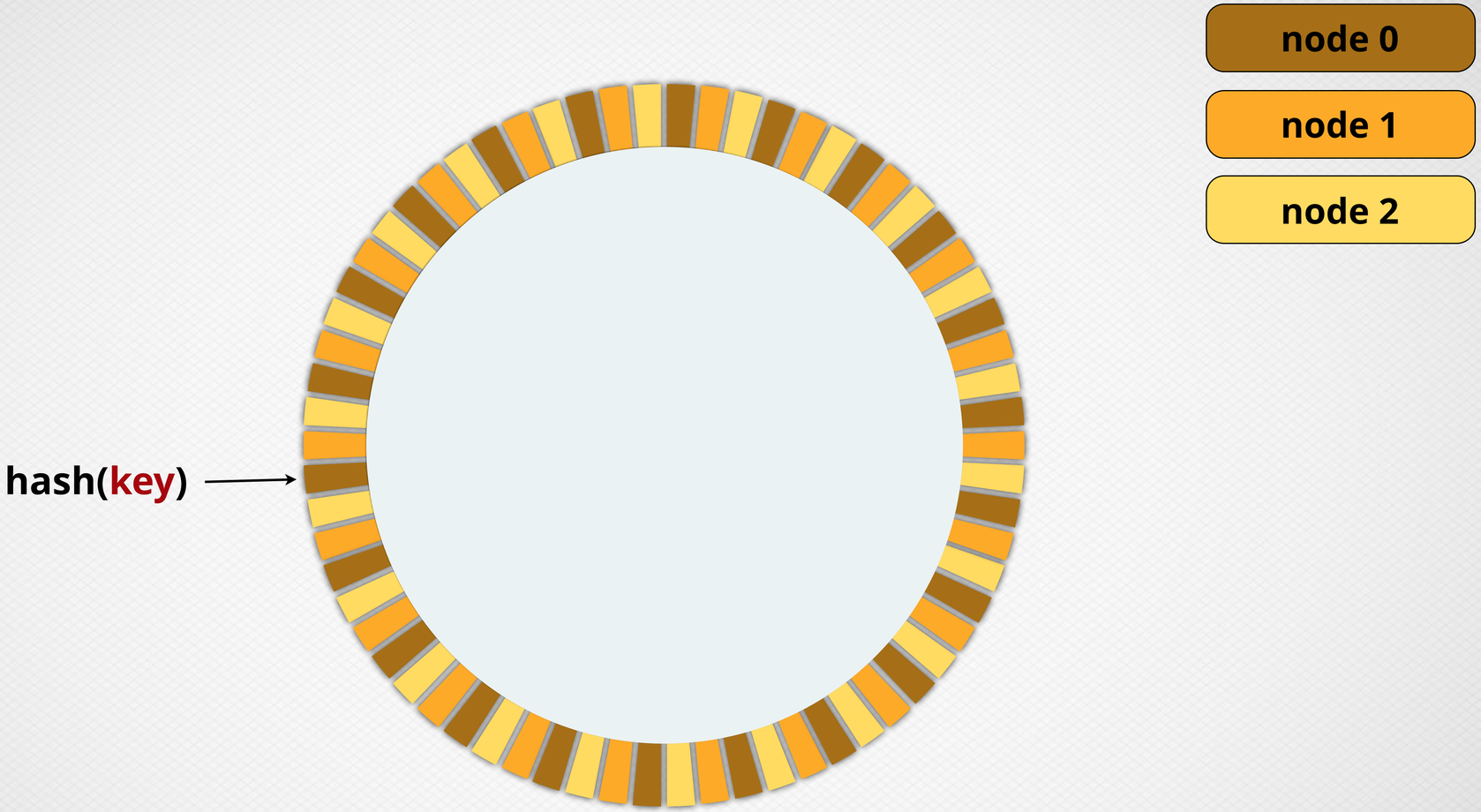
(1461501637330902918203684832716283019655932542976)

hash ring



tokenize it




node 0
node 1
node 2
hash(key)

Scaling out

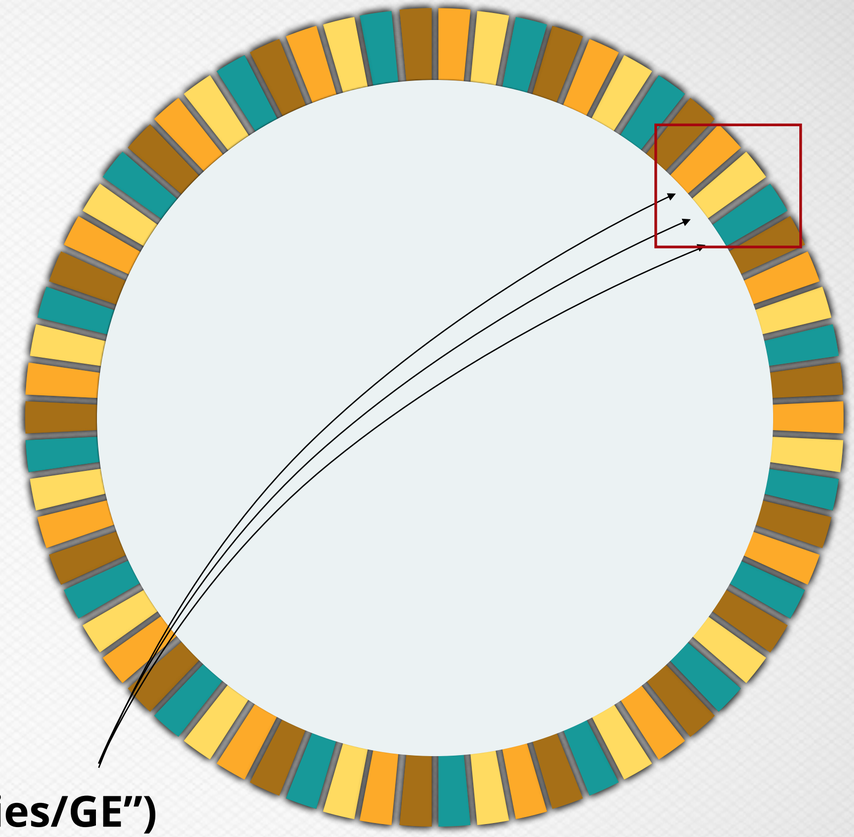
node 0

node 1

node 2

node 3 +

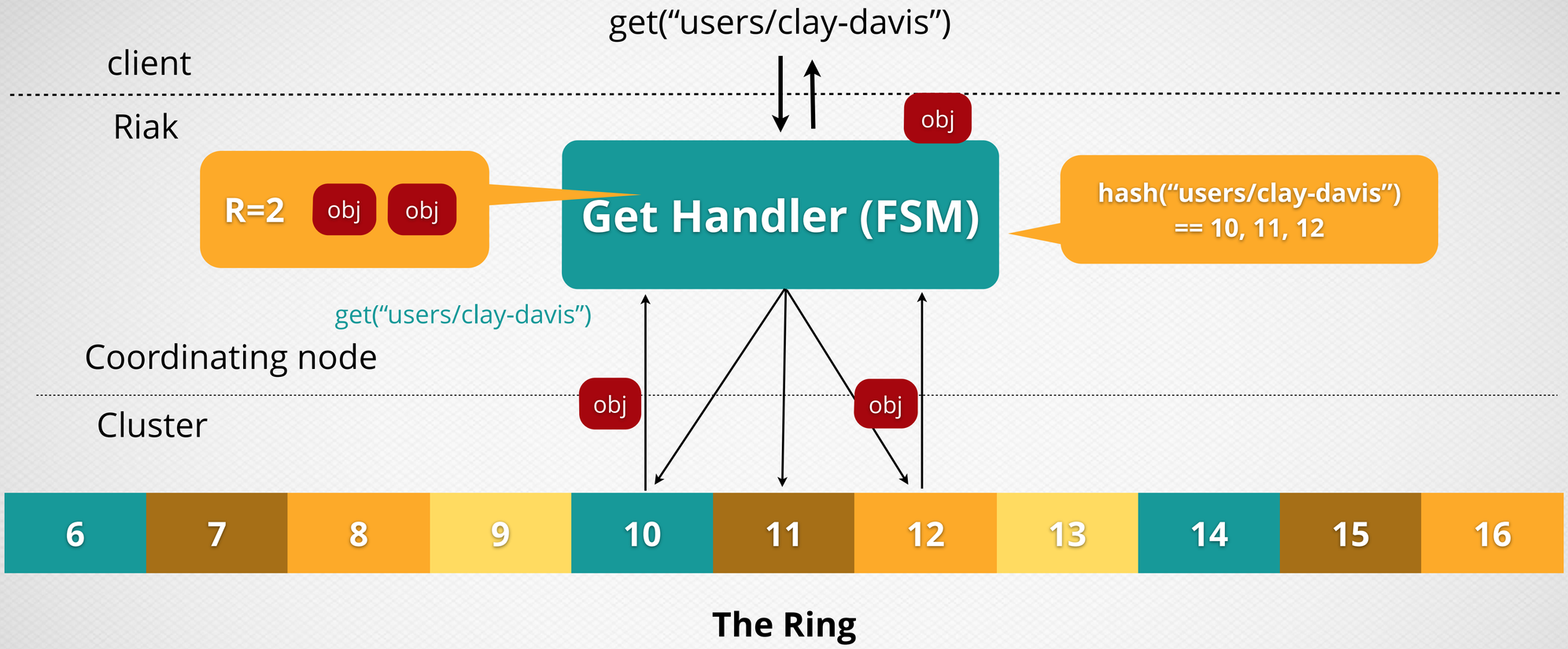
Replicas are stored to the N-1 contiguous partitions



hash(“companies/GE”)

N R W PR/PW DW

Quorum requests


get("users/clay-davis")
client
Riak
obj
R=2
obj
obj
Get Handler (FSM)
hash("users/clay-davis")
== 10, 11, 12
get("users/clay-davis")
Coordinating node
Cluster
obj
obj
6
7
8
9
10
11
12
13
14
15
16
The Ring

Read Repair

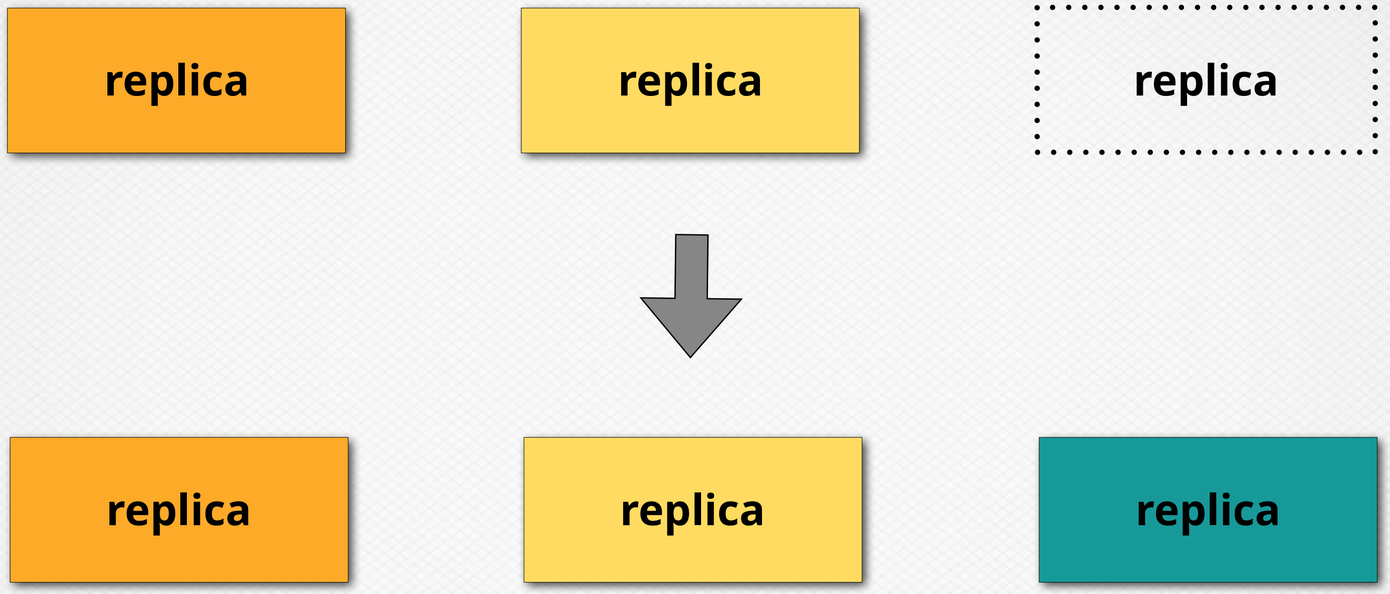
(Anti-Entropy)

replica

replica

replica

replica
replica
replica


replica
replica
replica
replica
replica
replica

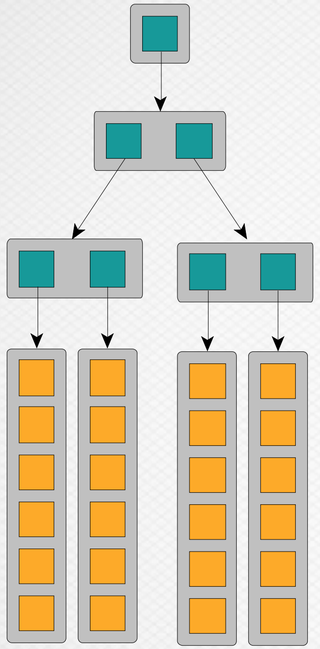
Active Anti-Entropy
(self healing clusters)

disk-based

persistent

real-time updates

non-blocking

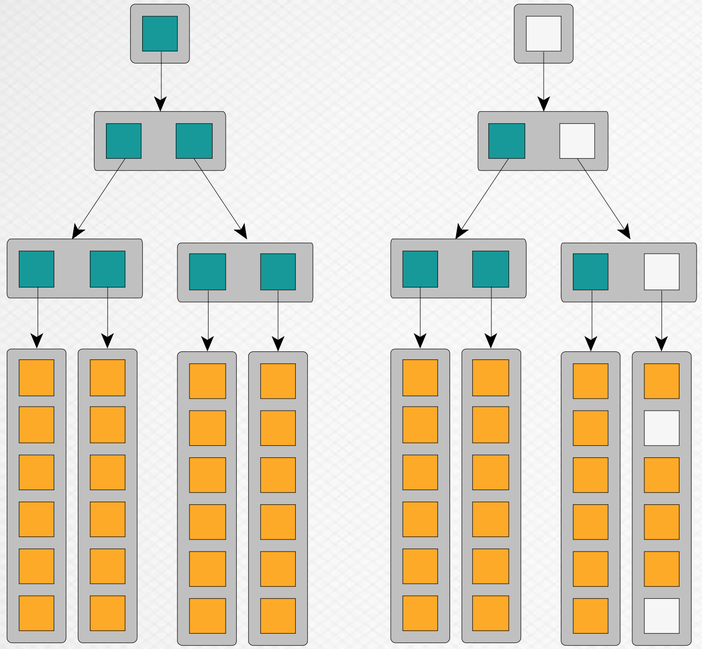


merkle tree to track changes
coordinated at the vnode level

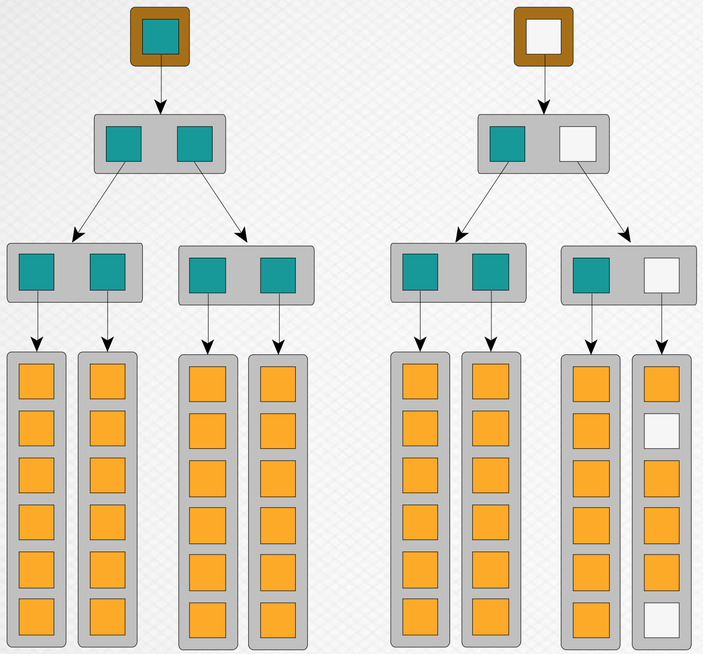
runs as a background process

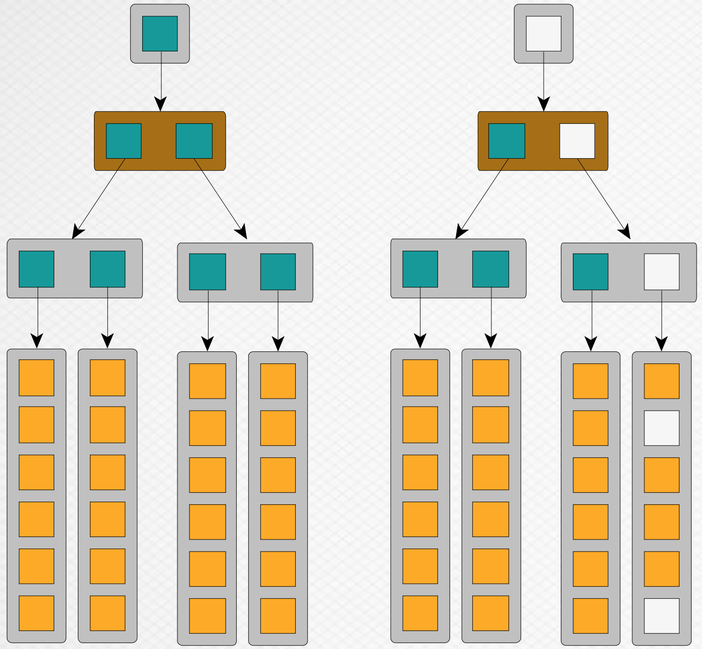
exchange with
neighbor vnodes for inconsistencies

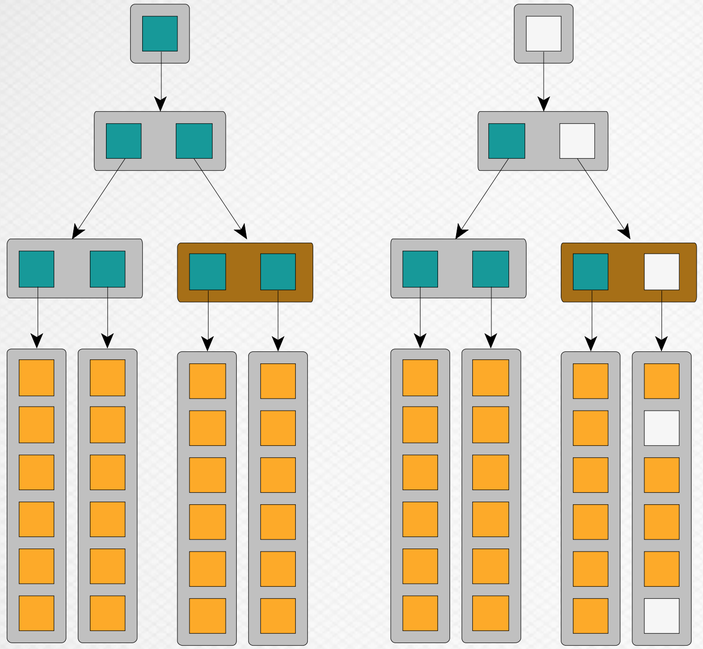
resolution semantics:
trigger read-repair

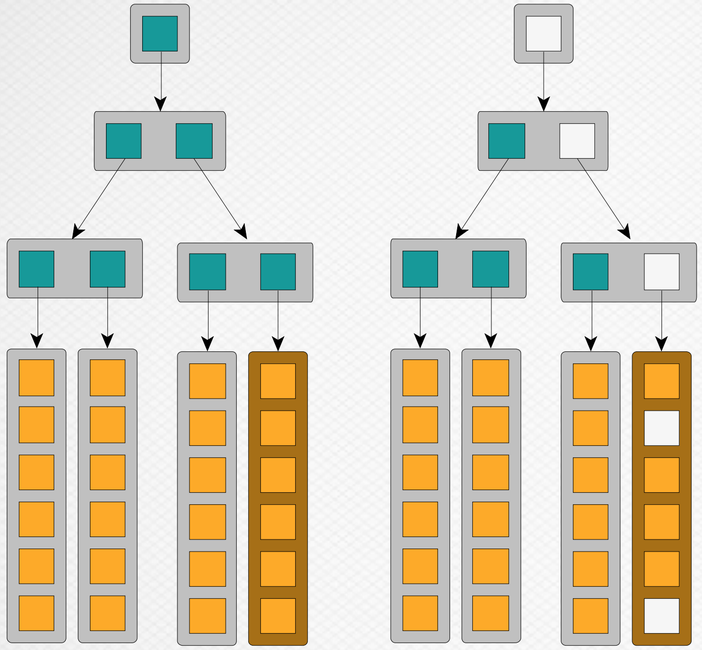


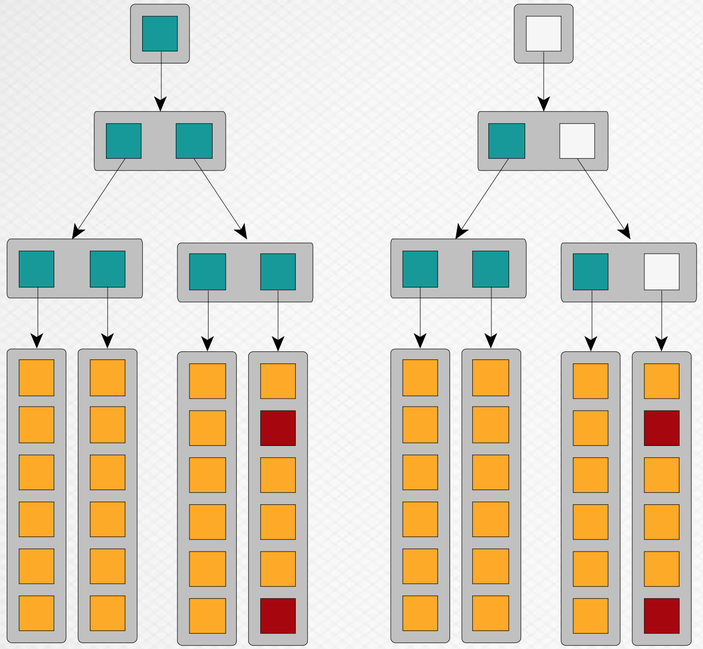
= hashes marked “dirty”











= keys to read-repair

Using Riak

Riak is the
ops-friendly
database

Riak is friendly to Developers, too!

Keys are namespaced

{"key": "val

into Buckets

key	value
key	value
key	value
key	value
key	value
key	value
key	value
key	value
key	value

Buckets aren’t “real”

“Namespace” that holds:

- Bucket Properties
- Metadata

Aitch Tee Tee Pee

GET /buckets/**bucket**/keys/**key**

PUT /buckets/**bucket**/keys/**key**

DEL /buckets/**bucket**/keys/**key**

Working With Ruby - Starting

```
$ gem install riak-client

book = {
    isbn:     '1111979723',
    title:    'Moby Dick',
    author:   'Herman Melville',
    body:     'Call me Ishmael. Some years ago...',
    owned:    3
}
```

Working With Ruby - Create

```
# Starting Client
client = Riak::Client.new(protocol:'pbc',
                          pb_port: 8087,
                          host:    '10.0.0.1')

books_bucket = client.bucket('books')

robj_moby = books_bucket.new(book[:isbn])
robj_moby.data = book
robj_moby.store
```

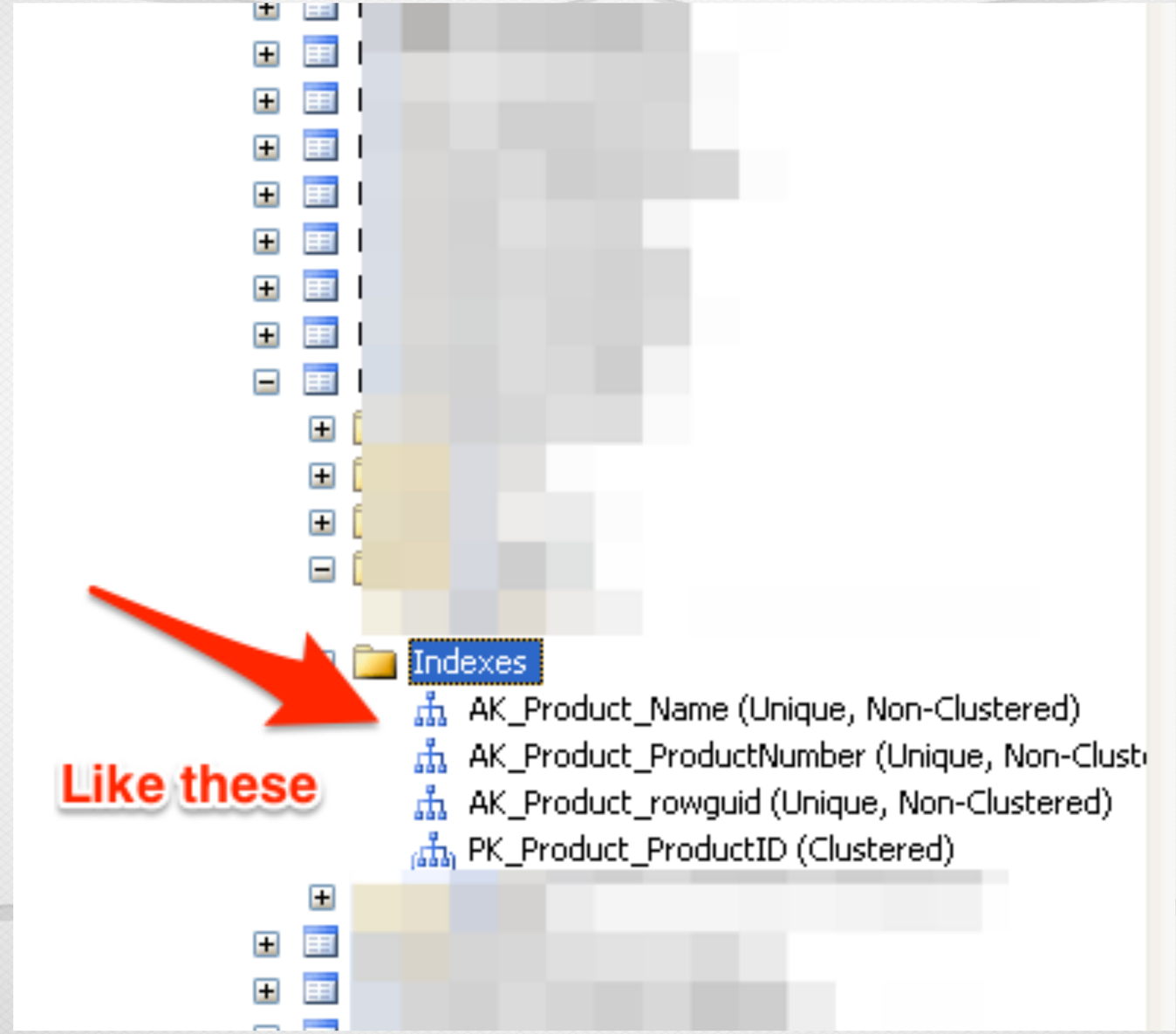
Working With Ruby - Read

```
fetched_book = books_bucket.get('1111979723')

puts fetched_book.raw_data

 => {  “isbn":"1111979723",
       "title":"Moby Dick”,
       "author":"Herman Melville”,
       "body":"Call me Ishmael. Some years ago…”,
       “copies_owned":3 }

fetched_book.data['isbn']
 => "1111979723"
```

Working With Ruby - Delete

```
fetched_book.delete

# or

books_bucket.delete('1111979723')
```

Secondary Indexes

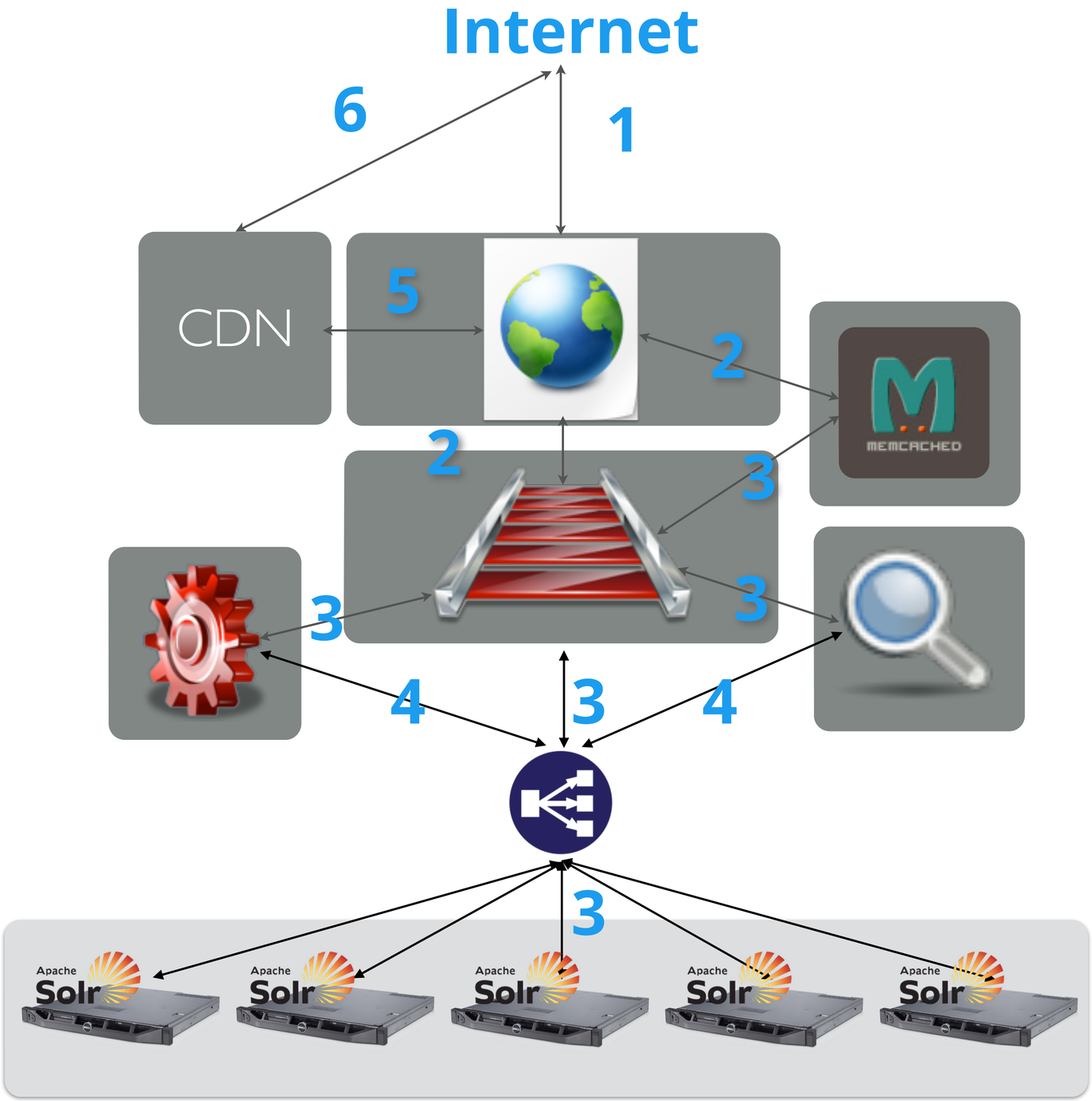


Working With Ruby - 2I

```
fetched_book.indexes['author_bin'] = ['Herman Melville']
fetched_book.store


books_bucket.get_index('author_bin','Herman Melville')

  => ["1111979723"]
```

etc...

RIAK SEARCH


Internet
6
1
CDN
5
2
2
3
3
3
4
3
4
3
Apache Solr
Apache Solr
Apache Solr
Apache Solr
Apache Solr

in riak.conf:

search = off ⟶ search = on

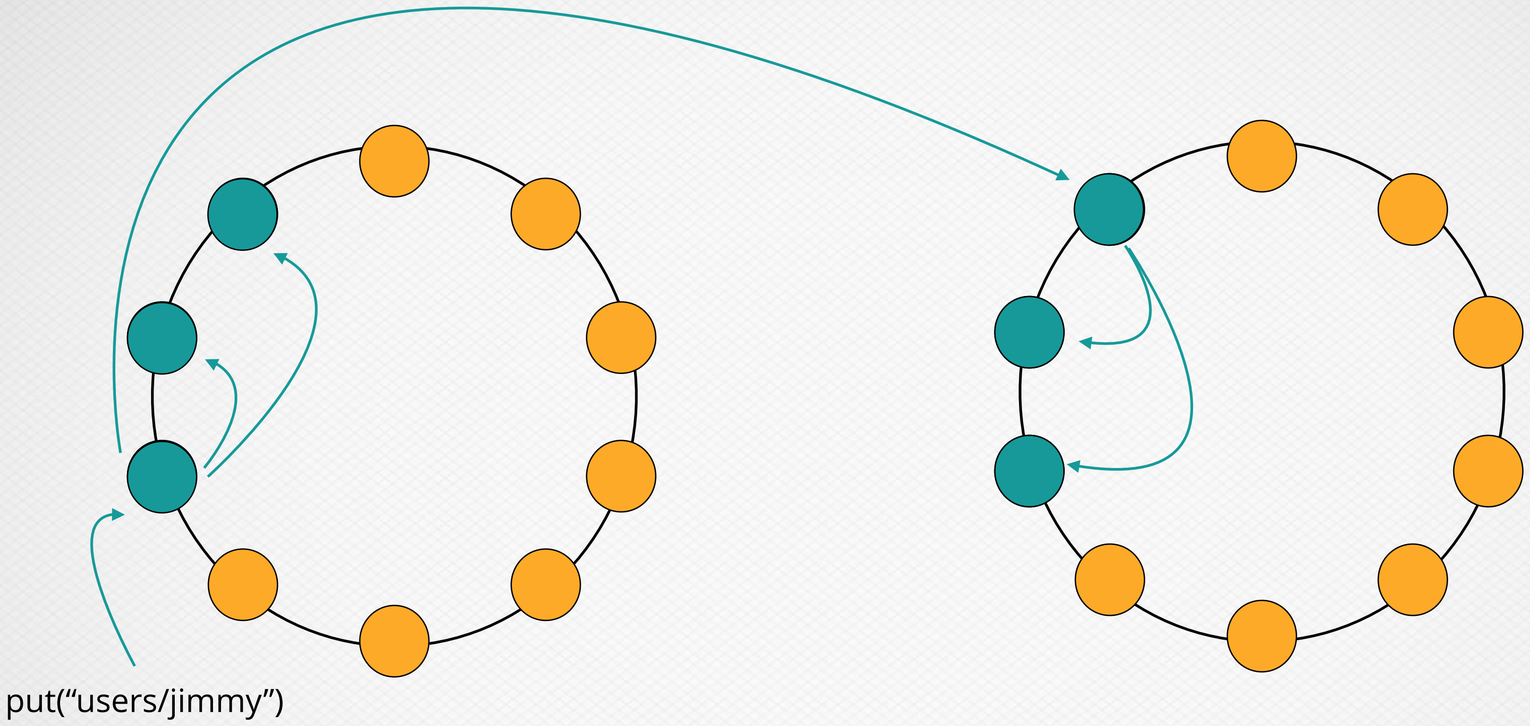
```
# create index
client.create_search_index('tweets_idx')

# assign index to bucket
bucket = Riak::Bucket.new(client, 'tweets')
bucket.props = {'search_index' => 'tweets_idx'}


# load some tweets into riak…

results = client.search('tweets", 'text:who')

```

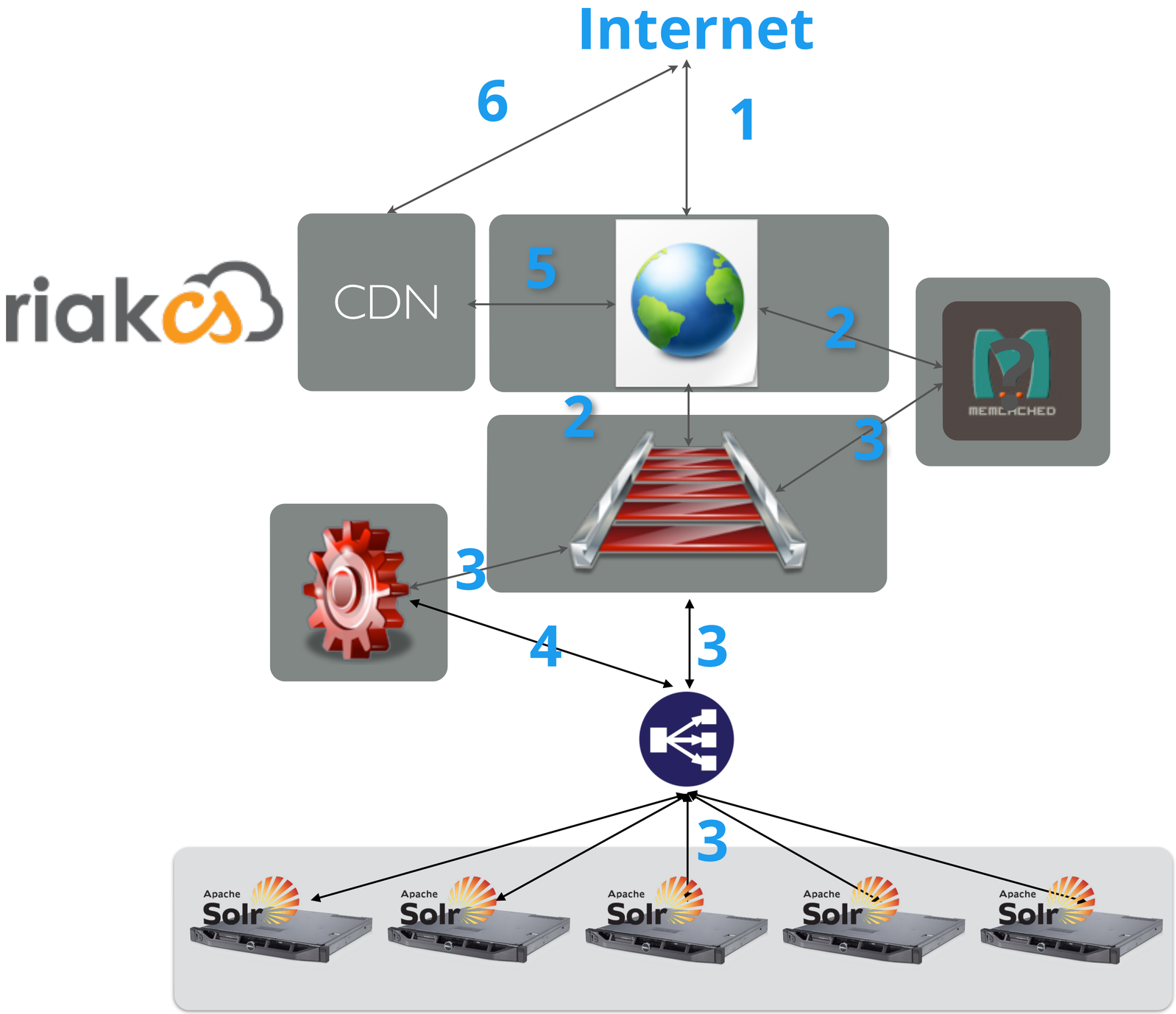
```
pp results['docs'].first

{'score'  => '2.5080490000000019489e+00',

 '_yz_rb' => 'tweets',

 '_yz_rk' => '421672897250226176',

 ...}
```

Multi-Datacenter Replication (E.G. - Nuke proofing)

Full Sync

Real-Time


put(“users/jimmy”)


Internet
6
1
riakcs
CDN
5
2
MEMCACHED
2
3
3
4
3
3
Apache Solr
Apache Solr
Apache Solr
Apache Solr
Apache Solr

More resources:

docs.basho.com

“Taste of Riak”

Q & A

Alex Moore - Client Services Engineer, Basho

@alexmoore

amoore@basho.com

http://alexmoore.io