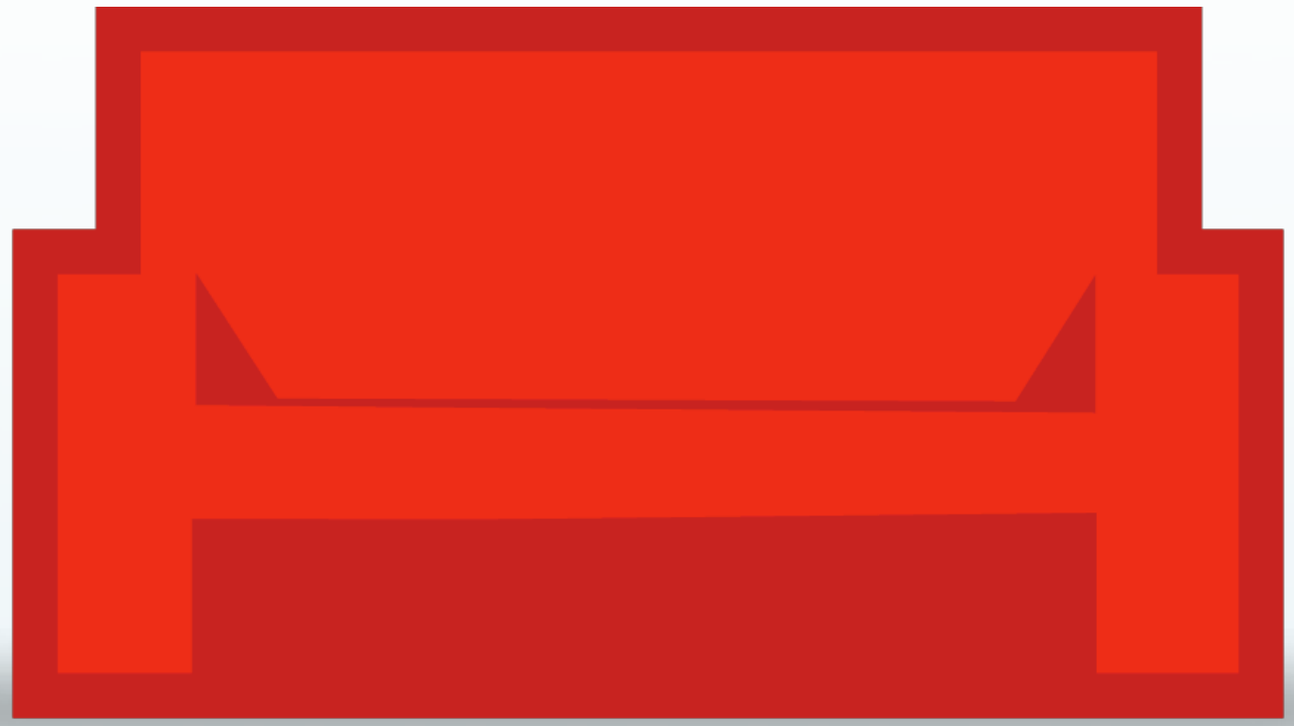


SCALING COUCHDB WITH BIGCOUCH

Adam Kocoloski
Cloudant

Erlang Factory SF Bay Area 2011

OUTLINE

- **Introductions**
- **Brief intro to CouchDB**
- **BigCouch Usage Overview**
- **BigCouch Internals**
- **Reports from the Trenches**

INTRODUCTIONS

Cloudant CTO

CouchDB Committer Since 2008

PhD physics MIT 2010

BigCouch

Putting the "C" back in CouchDB

Open Core: 2 years Development

Contact

adam@cloudant.com

kocolosk in #cloudant / #couchdb / #erlang

@kocolosk

COUCHDB IN A SLIDE

- **Document database management system**
- **JSON over HTTP**
- **Append-only MVCC storage**
- **Views: Custom, persistent representations of your data**
  Incremental MapReduce with results persisted to disk
  Fast querying by primary key (views stored in a B-tree)
- **Bi-Directional Replication**
  Master-slave and multi-master topologies supported
  Optional ‘filters’ to replicate a subset of the data
  Edge devices (mobile phones, sensors, etc.)

WHY BIGCOUCH?

CouchDB is Awesome

OPSCODE

mozilla LABS Raindrop

Ubuntu Linux

aptela

BBC

CERN

...But somewhat incomplete

Cluster
Of
Untrusted
Commodity
Hardware

"CouchDB is not a distributed database" -J. Ellis

"Without the Clustering, it's just OuchDB"

BIGCOUCH = HA CLUSTERED COUCH

- **Horizontal Scalability**

  Easily add storage capacity by adding more servers

  Computing power (views, compaction, etc.) scales with more servers

- **No SPOF**

  Any node can handle any request

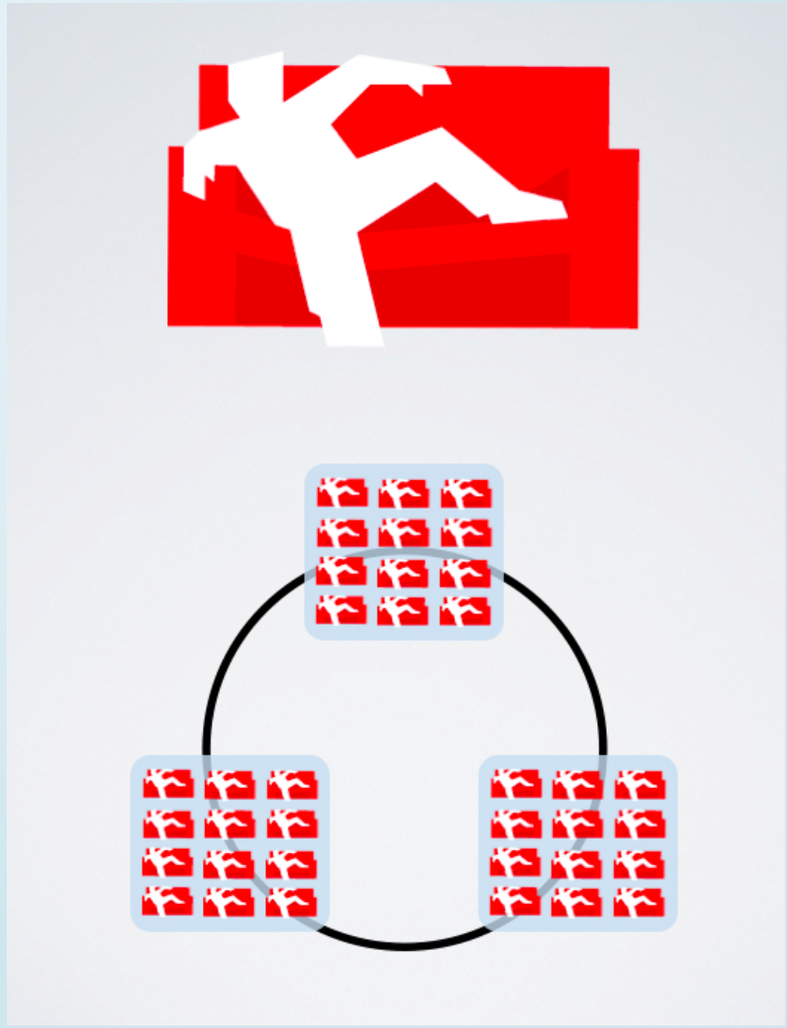
  Individual nodes can come and go

- **Transparent to the Application**

  All clustering operations take place “behind the curtain”

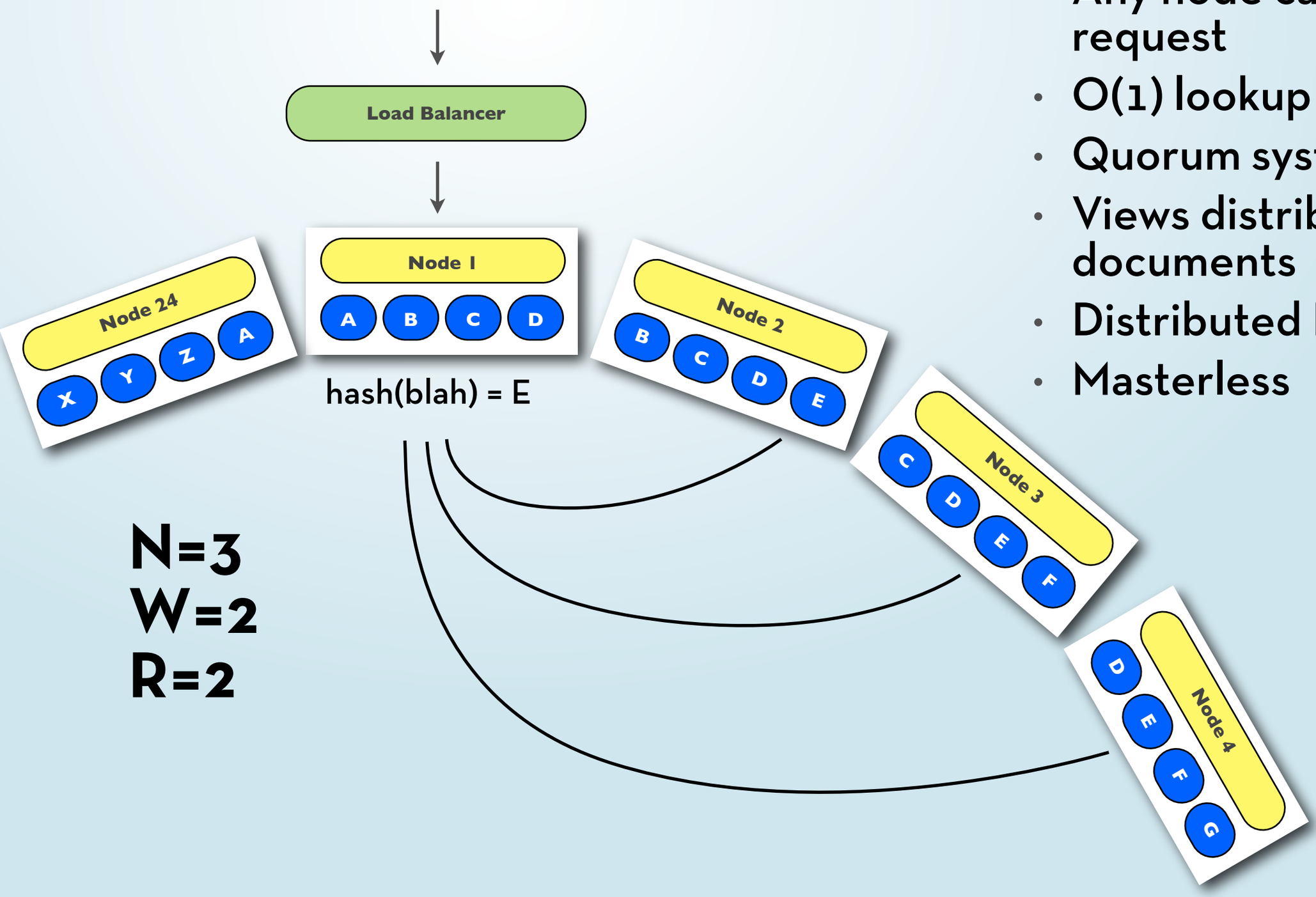
  ‘looks’ like a single server instance of Couch, just with more awesome

  asterisks and caveats discussed later



GRAPHICAL REPRESENTATION

PUT http://kocolosk.cloudant.com/dbname/blah?w=2



N=3
W=2
R=2

- Clustering in a ring (a la Dynamo)
- Any node can handle a request
- O(1) lookup
- Quorum system (N, R, W)
- Views distributed like documents
- Distributed Erlang
- Masterless

BUILDING YOUR FIRST CLUSTER

- **Shopping List**

3 networked computers

Usual CouchDB Dependencies

BigCouch Code

- http://github.com/cloudant/bigcouch

```
brew install erlang icu4c spidermonkey
brew ln icu4c
```

```
cd $CLOUDANT_SRC
./configure -p $PREFIX
make
sudo make install
```

BUILDING YOUR FIRST CLUSTER

Build and Start BigCouch

```
$PREFIX/bin/bigcouch
```

foo.example.com

bar.example.com

baz.example.com

Pick one node and add the others to the local “nodes” DB

```
curl -X PUT http://foo.example.com:5986/nodes/bigcouch@bar.example.com -d {}
curl -X PUT http://foo.example.com:5986/nodes/bigcouch@baz.example.com -d {}
```

Make sure they all agree on the magic cookie (rel/etc/vm.args)

```
curl http://foo.example.com:5984/_membership
```

QUORUM: IT'S YOUR FRIEND

- **BigCouch databases are governed by 4 parameters**

  Q: Number of shards

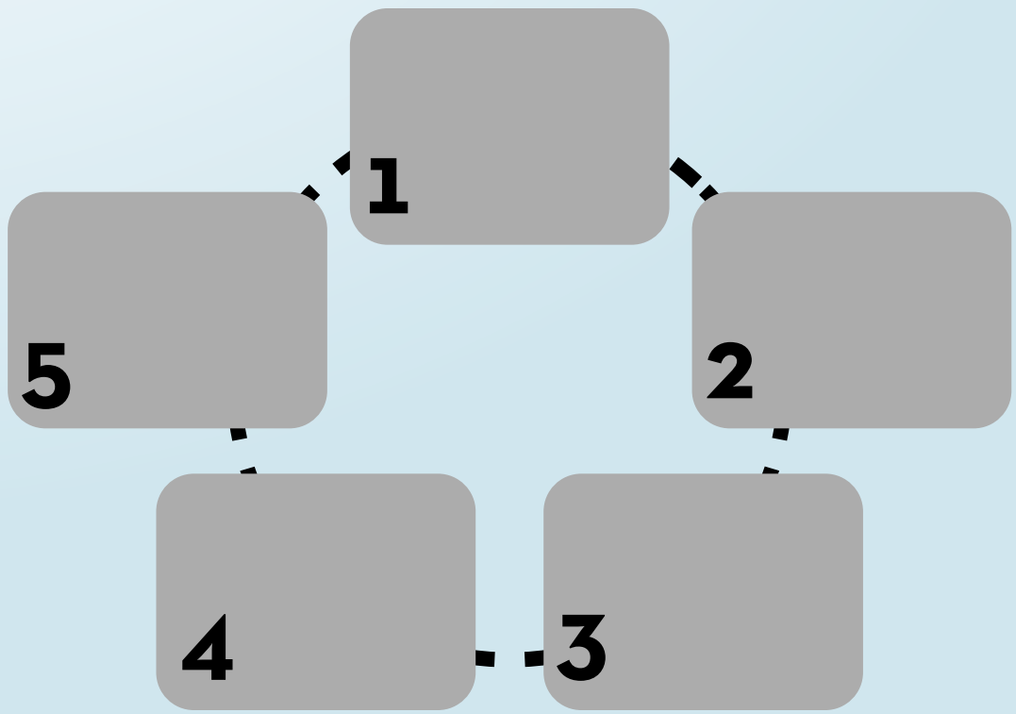
  N: Number of redundant copies of each shard

  R: Read quorum constant

  W: Write quorum constant

(NB: Also consider the number of nodes in a cluster)

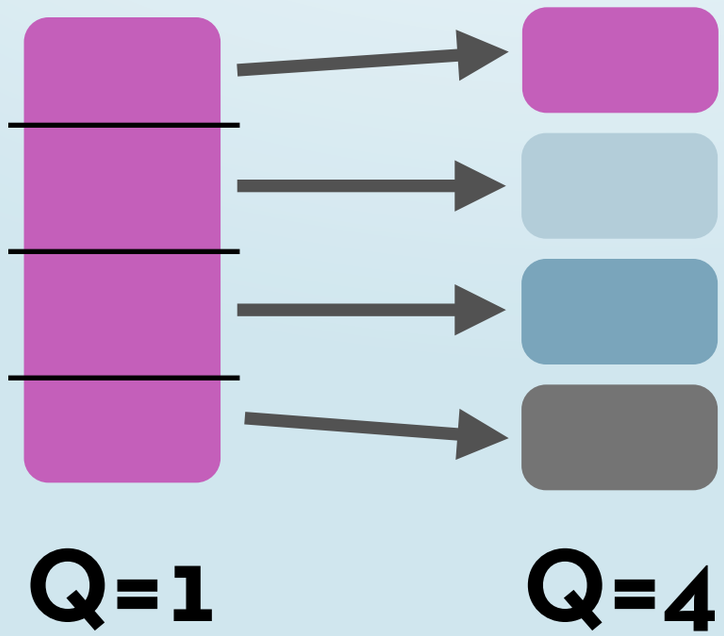
For the next few examples, consider a 5 node cluster

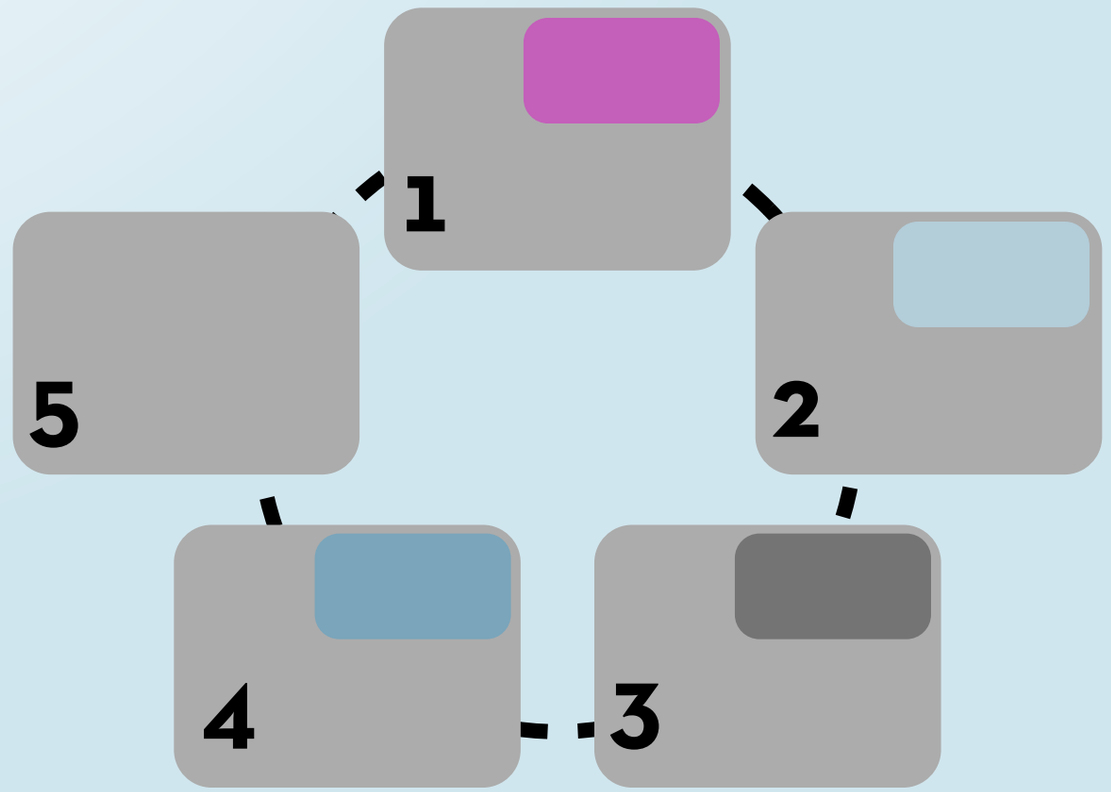


Q

- **Q: The number of shards over which a DB will be spread**

  consistent hashing space divided into Q pieces
  Specified at DB creation time
  possible for more than one shard to live on a node
  Documents deterministically mapped to a shard





N

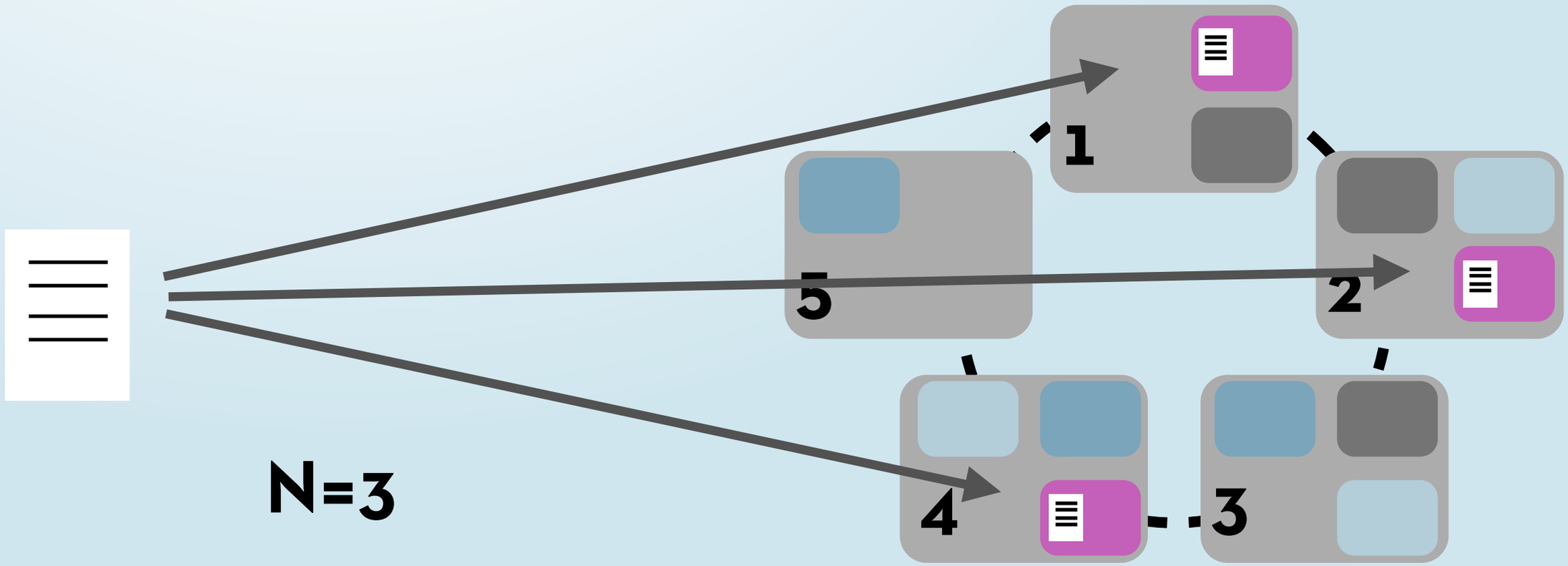
- **N: The number of redundant copies of each document**

  Choose N>1 for fault-tolerant cluster

  Specified at DB creation

  Each shard is copied N times

  Recommend N>2



W

- **W: The number of document copies that must be saved before a document is “written”**

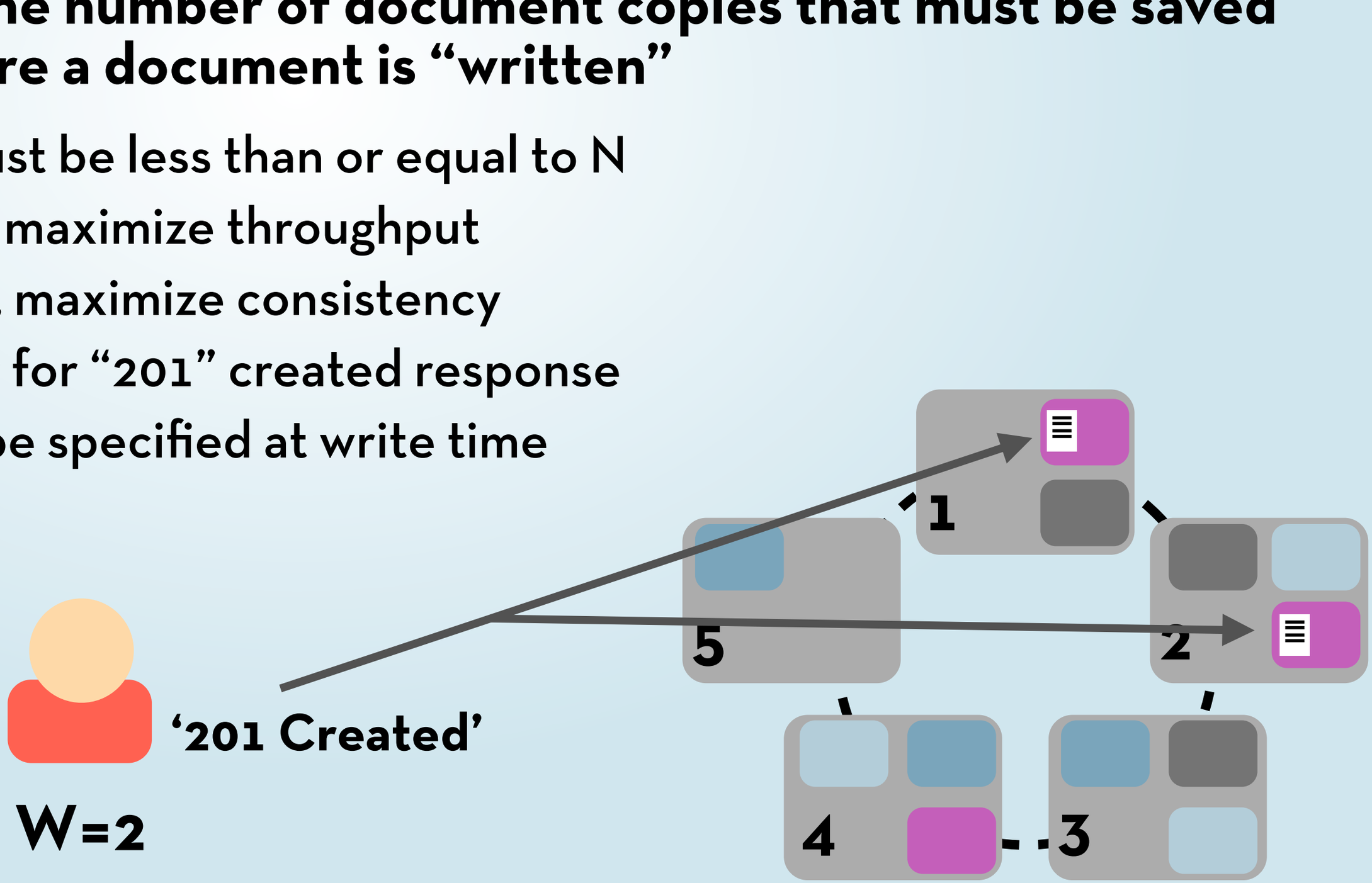
  W must be less than or equal to N

  W=1, maximize throughput

  W=N, maximize consistency

  Allow for “201” created response

  Can be specified at write time



R

- **R: The number of identical document copies that must be read before a read request is ok**

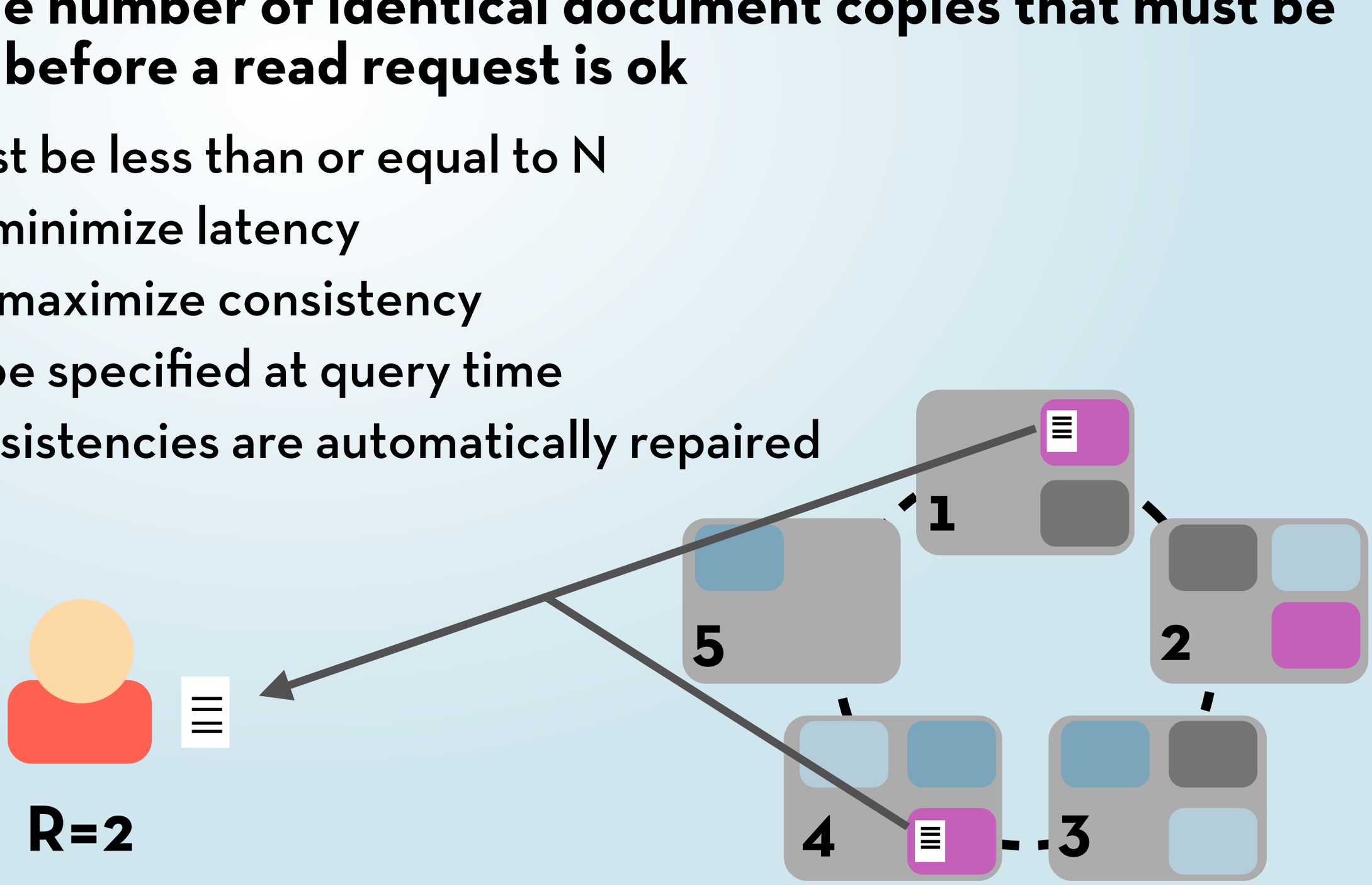
  R must be less than or equal to N

  R=1, minimize latency

  R=N, maximize consistency

  Can be specified at query time

  Inconsistencies are automatically repaired



VIEWS

- **So far, so good, but what about secondary indexes?**

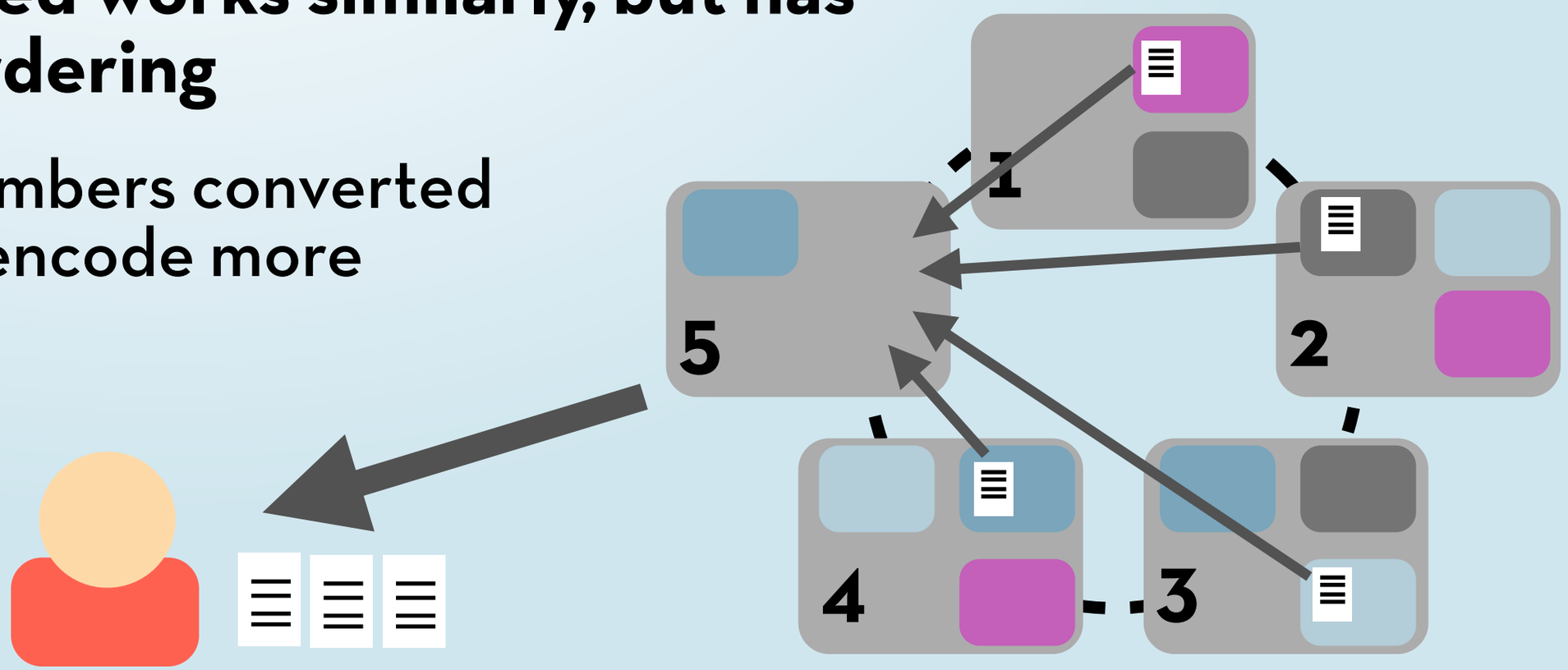
  Views are built locally on each node, for each DB shard

  Mergesort at query time using exactly one copy of each shard

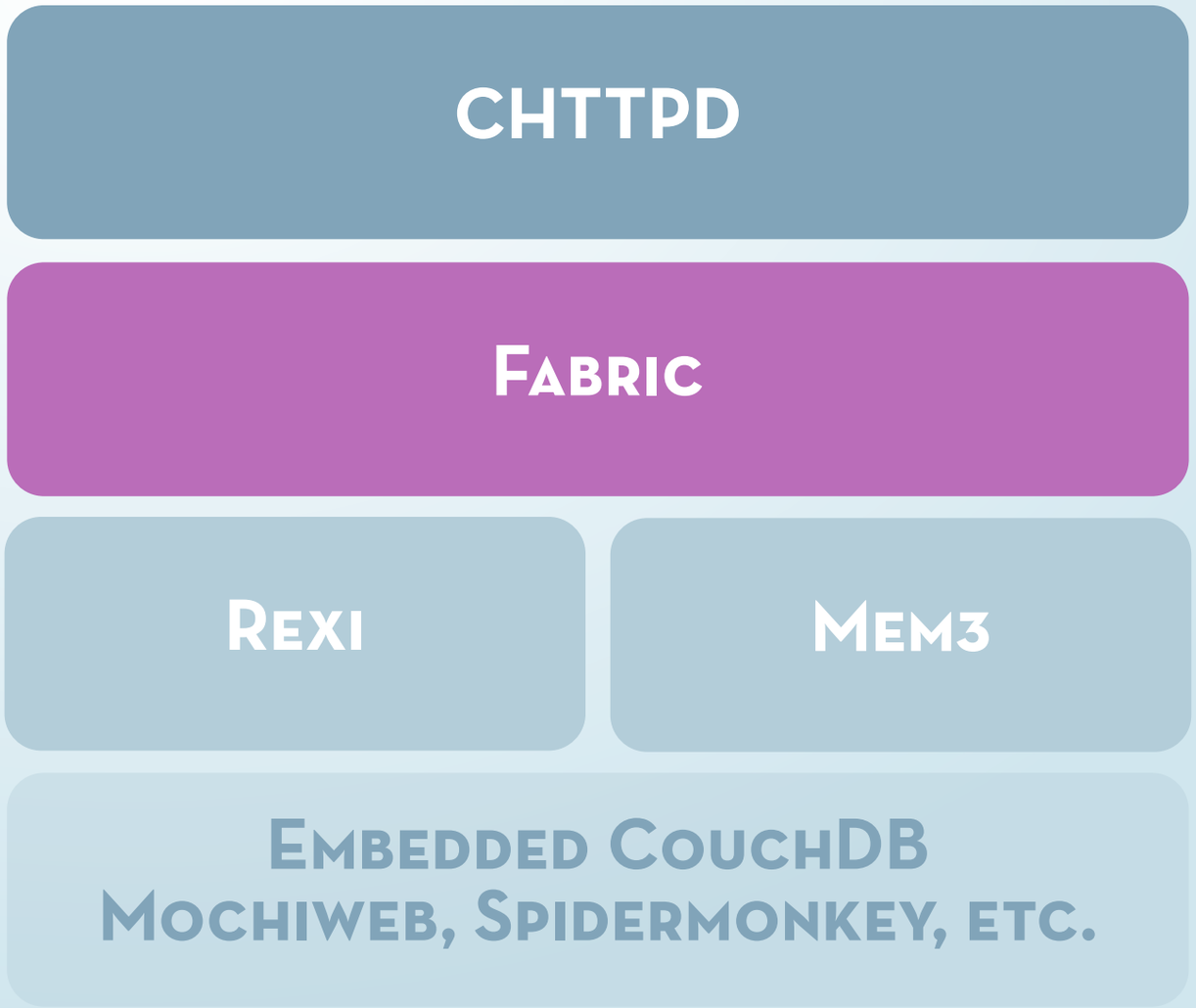
  Run a final rereduce on each row if a the view has a reduce

- **_changes feed works similarly, but has no global ordering**

  Sequence numbers converted to strings to encode more information

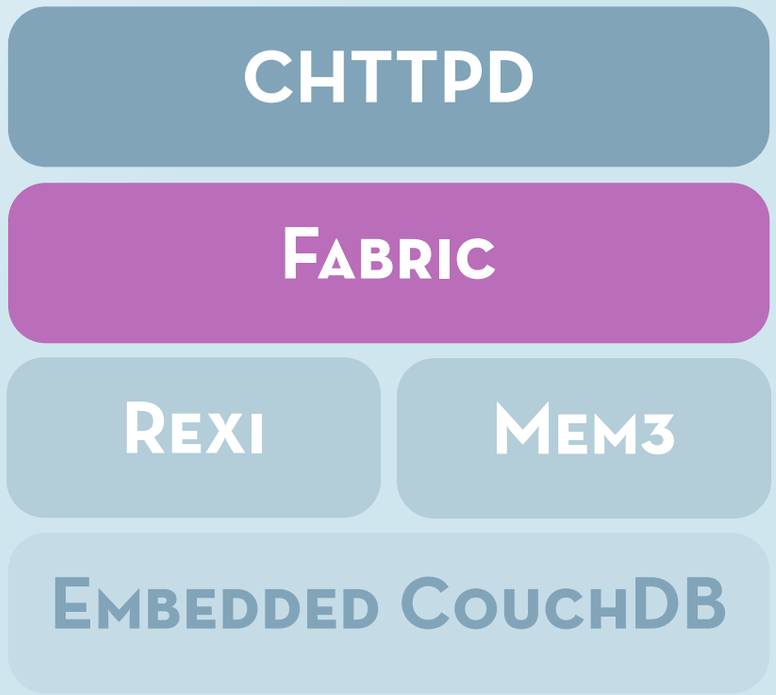


BIGCOUCH STACK



MEM3

- **Maintains the shard mapping for each clustered database in a node-local CouchDB database**
- **Changes in the node registration and shard mapping databases are automatically replicated to all cluster nodes**
- **Shard copies are eagerly synchronized**



REXI

- **BigCouch makes a large number of parallel RPCs**
- **Erlang RPC library not designed for heavy parallelism**

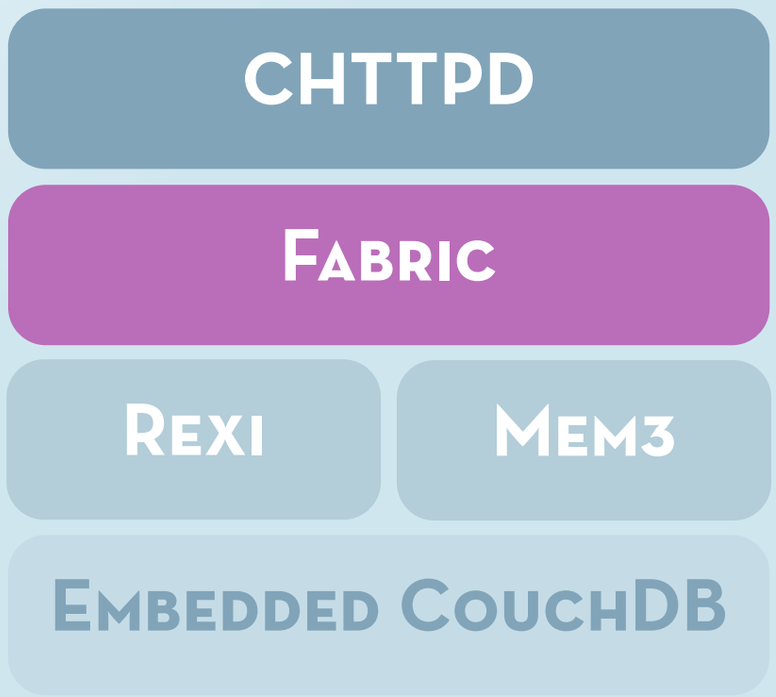
  promiscuous spawning of processes
  responses directed back through single process on remote node
  requests block until remote ‘rex’ process is monitored

- **Rexi removes some of the safeguards in exchange for lower latencies**

  no middlemen on the local node

  remote process responds directly to client

  remote process monitoring occurs out-of-band



FABRIC / CHTTPD

- **Fabric**

  OTP library application (no processes) responsible for clustered versions of CouchDB core API calls

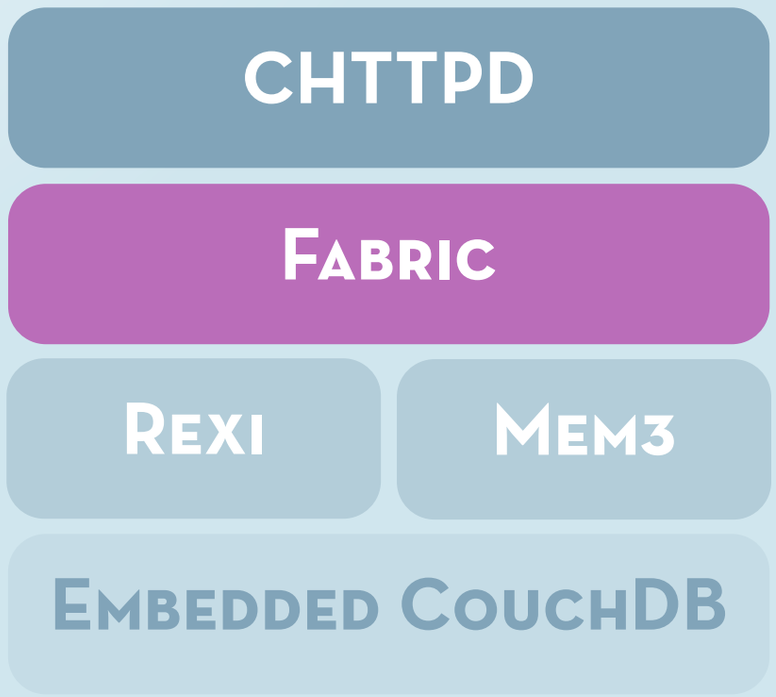
  Quorum logic, view merging, etc.

  Provides a clean Erlang interface to BigCouch

  No HTTP awareness

- **Chttpd**

  Cut-n-paste of couch_httpd, but using fabric for all data access



REPORTS FROM THE TRENCHES

- **code_change and supervision trees**
- **remote execution of fun expressions == recipe for badfun**
- **blocking !**

http://github.com/cloudant/bigcouch