BigCouch for WMAgent

Valentin Kuznetsov, Cornell University & Seangchan Ryu, FNAL

March 4th, 2014

Current status

- CouchDB is used for:
  - local WMAgents; ReqMgr, WMStats, WorkQueue
    - it is self-contained product capable of holding few millions of documents
- Beyond 3M docs we see large latency in views & compact procedure, silent crashes, etc.
- CouchDB code is 2-3 years old (we use 1.1.0 while official version is 1.5.0)

Proposed solution: BigCouch

- To resolve scalability, stability and maintenance issues it was proposed to evaluate BigCouch solution (Cloudant Inc.)
  - IBM announced to acquire Cloudant on Feb 24th.
- It is highly available, fault-tolerant, clustered version of Apache CouchDB
- It is standard OTP application based on Amazon Dynamo DB implementation.
- Refs: http://bit.ly/1gWtJ54, http://vimeo.com/21773600
- Transparent migration from CouchDB, i.e. no code changes

OTP in nutshell

- OTP stands for Open Telecom Platform developed by Ericsson and used in major mobile infrastructure, e.g. T-Mobile.
- It provides generic framework to manage and supervise processes in concurrent, fault-tolerant fashion
  - Every process is supervised and monitored
  - Framework provides ability for “hot” code swapping on live systems (i.e. no downtime for upgrade procedure), etc.
  - Process management (sync and async)
  - Provides generic server behavior, maintenance & deployment

Dynamo DB concept

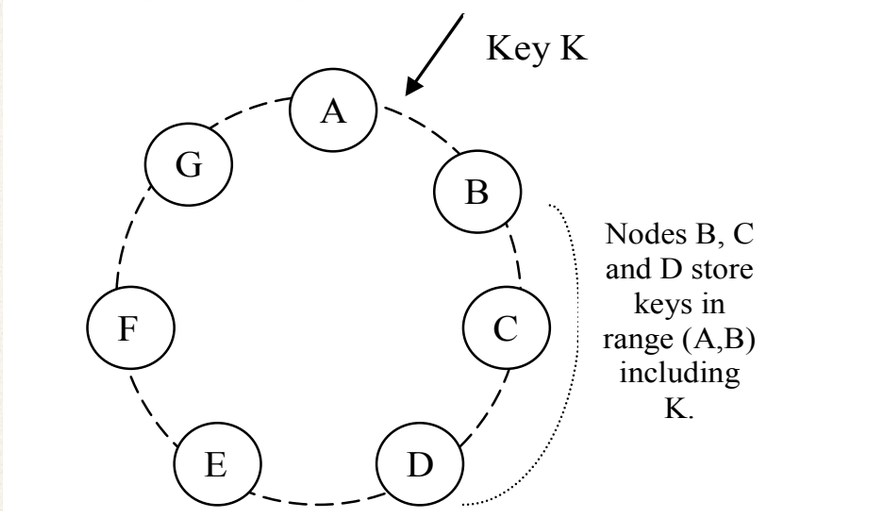


Figure 2: Partitioning and replication of keys in Dynamo ring.

Traditional replicated relational database systems focus on the problem of guaranteeing strong consistency to replicated data. Although strong consistency provides the application writer a convenient programming model, these systems are limited in scalability and availability [7]. These systems are not capable of handling network partitions because they typically provide strong consistency guarantees.

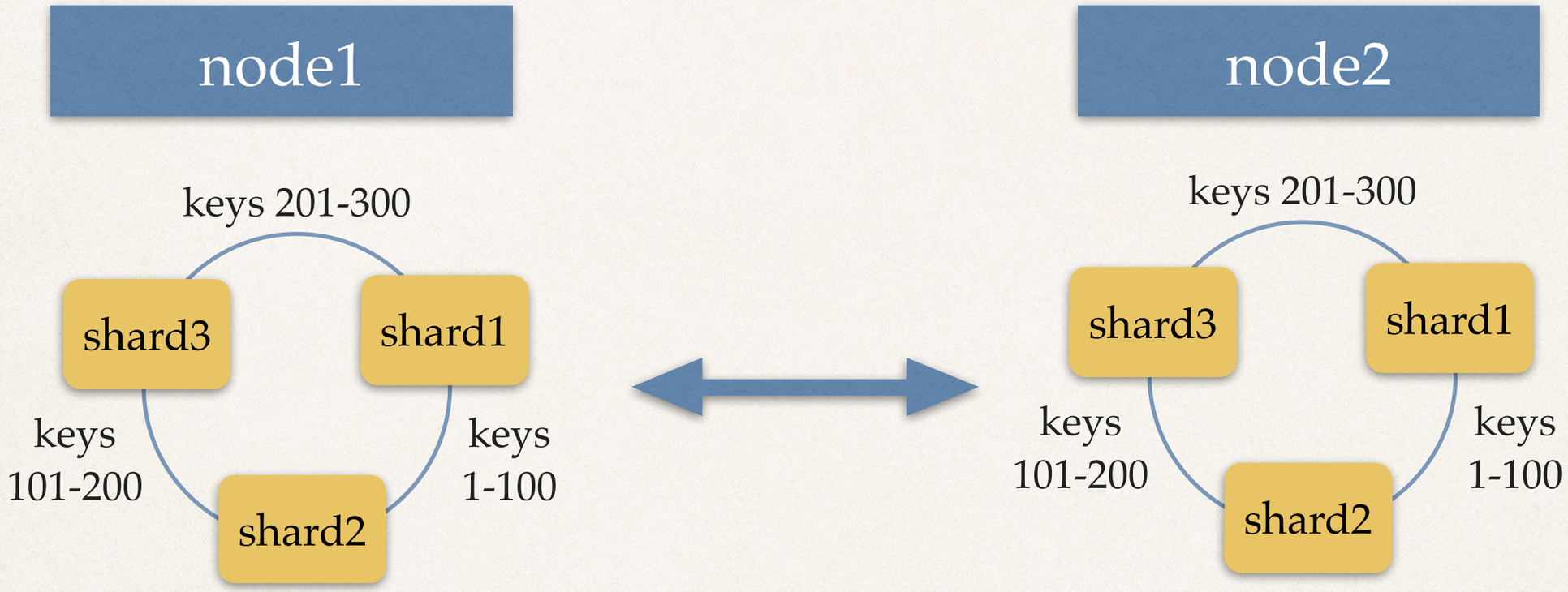
Table 1: Summary of techniques used in *Dynamo* and their advantages.

Problem	Technique	Advantage
Partitioning	Consistent Hashing	Incremental Scalability
High Availability for writes	Vector clocks with reconciliation during reads	Version size is decoupled from update rates.
Handling temporary failures	Sloppy Quorum and hinted handoff	Provides high availability and durability guarantee when some of the replicas are not available.
Recovering from permanent failures	Anti-entropy using Merkle trees	Synchronizes divergent replicas in the background.
Membership and failure detection	Gossip-based membership protocol and failure detection.	Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.

BigCouch status

- Initial spec was written by A. Melo, I extended and build BigCouch RPM
  - PR: https://github.com/cms-sw/cmsdist/pull/352
- Standard cmsweb deploy/manage scripts
  - PR: https://github.com/dmwm/deployment/pull/106
- BigCouch can run on single node or we can form a cluster
- We can substitute CouchDB with BigCouch without any code change and setup replication between the two

BigCouch setup



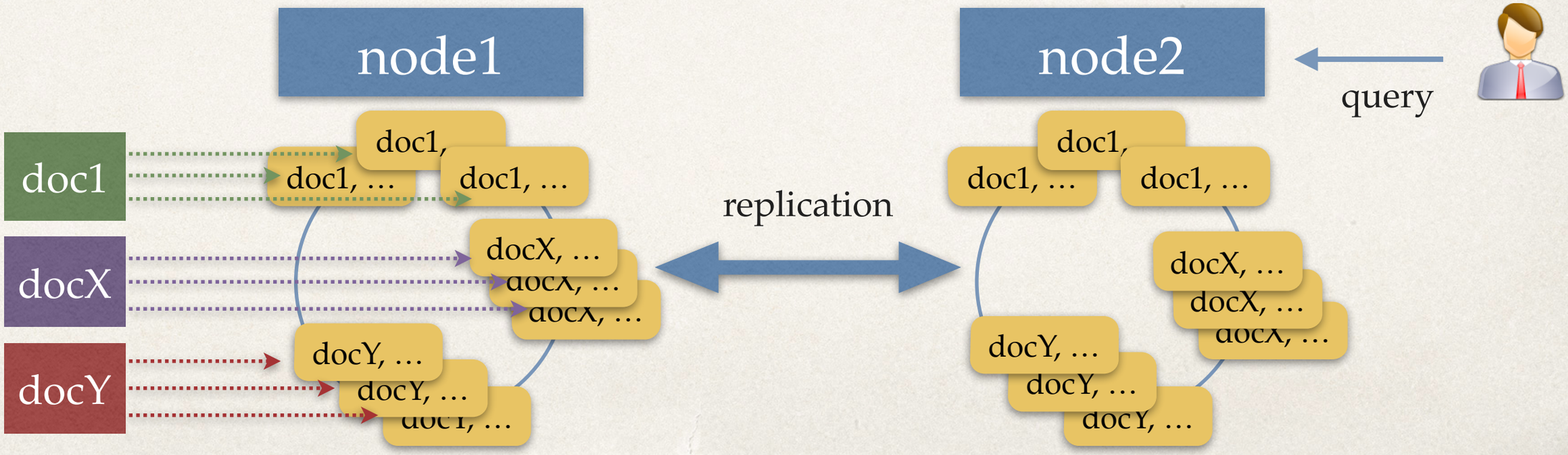
- Data are replicated between nodes
- Each shard keeps some portion of the data
- BigCouch provides consistent hashing
- Nodes can come and go and your requests will always answered

BigCouch setup, cont'd

- **Q**: control number of shards over which a DB will be spread, specified at DB creation
- **N**: number of redundant copies of each document, specified at DB creation
- **R**: read quorum constant, i.e. #docs copies that must be read before a read request is ok, can be specified at query time
- **W**: write quorum constant, i.e. #docs copies that must be saved before document is "written", can be specified at write time. W=1 maximize throughput, W=N maximize consistency

Create a database comprised of 32 partitions where each document is stored 3 times

curl -X PUT http://127.0.0.1:15984/test_db?n=3&q=32



BigCouch views

- BigCouch is not so differ from CouchDB for doc retrieval, i.e. it is based on Map-Reduce paradigm
- Views are built locally on each node, for each DB shard
  - large number of shards reduces latency on view performance
- Mergesort at query time using exactly one copy of each shard
- Run a final re-reduce on each row if view has reduce
- It is possible to see feeds via _changes

BigCouch cluster

- BigCouch provides ability to form a cluster form BigCouch nodes
- Configure firewall rules (via iptables) to support RPC communication between Erlang nodes
  - **`iptables -I INPUT -s $host -p tcp --dport 4369 -j ACCEPT`**
  - **`iptables -I INPUT -s $host -m state --state NEW -m tcp -p tcp --dport 9100:9105 -j ACCEPT`**
- Inform nodes about neighbors (admin port 15986)
  - **`curl -X PUT http://127.0.0.1:15986/nodes/bigcouch@host.com -d {}`**
- Inject a document and it will be replicated to participated nodes (db port 15984):
  - **`curl -X PUT http://127.0.0.1:15984/db/doc_1 -H content-type:application/json -d '{"a":1,"b":2}'`**
- Fetch document from any participated node, e.g.
  - **`curl http://127.0.0.1:15984/db/doc_1`**

BigCouch tests

- Setup BigCouches on two VM: das and das-dbs3
- Form a cluster and inject/read docs

curl http://127.0.0.1:15984/_membership


```
{“all_nodes":["bigcouch@188.184.23.191","bigcouch@188.184.24.166"],
"cluster_nodes":["bigcouch@188.184.23.191","bigcouch@188.184.24.166"]}
```


curl http://127.0.0.1:15984/db


```
{“db_name":"db","update_seq":[3,"abc"],"purge_seq":0,
“other”:{"data_size":3538185},"doc_del_count":0,"doc_count":3440,
"disk_size":11572040,"disk_format_version":5
"disk_format_version":5,"compact_running":false,"instance_start_time":"0"}
```

BigCouch tests, cont'd

- We replicated WMAgent CouchDB:5984 content into BigCouch:15984:
  - `curl -X PUT http://127.0.0.1:15984/wmstats`
  - `curl -H "Content-Type: application/json" -X POST http://127.0.0.1:5984/_replicator -d '{"target":"http://127.0.0.1:15984/wmstats","source":"wmstats"}'`
- We tested BigCouch by injecting 3M FWJR docs, 1M docs per 24 hours with 10 shell jobs looping over 100k docs
- We run standard views with idle DB:
  - initial view construction took ~20min
  - sub-sequent calls return results within a second

WMAgent & BigCouch

- Only little changes were required on WMAgent side
  - WMAgent RPM depends on BigCouch
  - Adopt deploy / manage scripts
  - New port range allocation
  - WMAgent configuration changes
- Setup and testing done on cmssrv101.fnal.gov

Migration plan

- Create official RPMs and update manage/deploy scripts **[DONE]**
  - Create wmagent-dev RPM with BigCouch dependency
  - Install WMAgents with CouchDB/BigCouch back-ends
- Measure performance of WMAgent with BigCouch and CouchDB back-ends and compare their performances **[~1 month]**
- Split data into hot/cold but support writing current schema, **[1-2 months]**
  - work with data-ops to define hot/cold data
- Setup BigCouch cluster on cmsweb, **[1 month, once we identify resources]**
  - work with HTTP group to define hardware requirement for BigCouch cluster

Migration plan, cont'd

- Replicate DBs: ReqMgr, WorkQueue, WMStats, Workload summary O(100GB), **[1 week]**

- Fix replicated databases, e.g. there are some content which is missing in ReqMgr, **[continuos process]**

- Stop services and adjust WMAgent settings to point to BigCouch, if required perform additional replication and clean-up, **[1 week]**

- Turn on WMAgent with BigCouch back-end