Home For Gypsies — Storage for NoSQL Databases
Atish Kathpal, NetApp
Agenda

1) Introduction on NoSQL
   - Master-less and Master-slave architectures
   - Data management provided by NoSQL DBs
   - How is Shared Storage relevant?

2) Backup and Restore for NoSQL DBs
   - Opportunity to leverage shared storage features
   - Challenges

3) Conclusions
NoSQL Databases

Overview

- Different from RDBMS
  - Tunable consistency semantics
  - Vertical v/s Horizontal scale

- When scale and data availability more important than consistency
  - Big data, web-scale apps - IoT, Mobile, Analytics
  - Trade-offs: CAP theorem [1]

- Open source, commodity nodes, DAS

Master-Slave Databases

NoSQL DB classification

- All writes to a partition, first written to the master node
  - Thus subsequent reads involving the Primary are always consistent
  - Loss of primary node leads to shard/partition-unavailability until new leader is elected
  - MongoDB, Redis fall in this category

https://www.slideshare.net/mongodb/sharding-v-final
Master-Less Databases

NoSQL DB classification

- Data is scattered across nodes using consistent hashing techniques
  - Writes streamed simultaneously to all nodes that the data hashes into
  - Eventual consistency: Unavailability of a destination node does not lead to write-failure, data is *eventually* replicated to the node when it becomes available, or gets hashed into a peer node
- Examples: Cassandra, CouchBase

NoSQL DBs – Built-in Data management

- Performance through horizontal scale-out
  - Commodity compute nodes with DAS
- Replication: high-availability and fault-tolerance
- Cluster Management, across data centers
- Inline compression, Encryption
- Developer friendliness
  - Support for different data formats and schemas
  - Integrations with analytics engines like Hadoop and Spark

What additional value can shared storage bring in?

In-memory objects can easily map into JSON-documents => flexible schema

Image source: https://www.mongodb.com

© 2017 NetApp, Inc. All rights reserved.
Shared-Storage value-adds

- **Independent scaling of compute and storage**

- **Consolidation of storage implies easier storage resource management**
  - Reduced cluster management costs
Shared-Storage key value-adds (continued)

- All Flash Arrays
  - Performance at *reduced cluster size*
  - Can run mixed workloads without performance impact

So shared storage can provide value as primary tier, however what about data protection and secondary storage?

Can support mixed workloads: NoSQL clusters, RDBMS, Hadoop, Data warehousing etc.

- Less space
- Less power
- Lower TCO
- Predictable performance

Legacy HDD storage arrays

All flash storage array
Backup and Restore

Relevance of Shared Storage
Why Backup/Restore NoSQL DBs?

Customers are directly ingesting critical into NoSQL

Security breach are on the rise e.g. ransomware attacks on MongoDB [2] and recent WannaCrypt exploits

“Fat-finger” errors eventually propagate to replicas

Sandbox deployments for test/dev

- Bring up shadow clusters of different cardinality (from production cluster snapshots)

Compliance and regulatory requirements

IDC, 2016 report [3] lists data-protection and retention as one of the top infrastructural requirements for NoSQL


Existing Open-source Utilities - Limitations

- Utilities like mongodump, mongorestore are inadequate
  - Operates on per-node basis
  - Copy based solution (expensive for TBs of data)
  - High restore times due to copy-in

- Cassandra backup utilities
  - Keeps a hard link of data files on disk (storage overhead)
  - Requires expensive repair steps upon restore

- Such utilities require separate automation to take cluster wide backups
  - Suffer from failure scenarios

Shared storage to the rescue? – Address pain points of copy-based backup and repair after restore
NoSQL DBs on Shared Storage

High-level, conceptual deployment architecture

Filesystem
(DB Journal, Logs, Data Files)

Filesystem
(DB Journal, Logs, Data Files)

Filesystem
(DB Journal, Logs, Data Files)

Ideas:
Backup: Leverage storage snapshots
Restore: Leverage cloning

Shared Storage Array
(Snapshots, Cloning, Compression, Deduplication, Encryption, Cloud Integrations)
NoSQL Data Protection – Challenges
Master-Slave Databases

Challenges:

- **Storage efficiency**
  - Redundant data captured in backup (replicas)

- **Replicas do not dedupe**
  - Unique ids per document per node
  - Compression, encryption

- **Fault tolerance**
  - Backup may capture unstable state of cluster
  - Can lead to higher RTO, due to new leader election during restore?

1. When primary fails, new leader is elected
2. Non-quorum data in failed primary is rolled back
NoSQL Data Protection – Challenges

Master-Less Databases

Challenges:

- **Storage efficiency**
  - Consistent hashing based data layout leads to replica overlap, as shown
  - Compression, encryption

Replication factor = 2

Write:
(K1, V1),
(K3, V3)

Write:
(K1, V1),
(K2, V2)

Write:
(K2, V2),
(K3, V3)

Cassandra N1

Cassandra N2

Cassandra N3

Shared Storage cluster
LUN1: K1, K3
LUN2: K1, K2
LUN3: K2, K3

No block based deduplication @ 4KB

NoSQL Data Protection – Challenges

Master-Less Databases

Challenges:

- **Fault tolerance**
  - Backup may capture stale data due to *eventual consistency*
- **Higher restore times**, since Cassandra will perform *repairs during restore*

Client, Update K1 (CL.ONE)

Update: (K1, V11, Tnew)

Ack

Update: (K1, V11, Tnew)

Cassandra N1

Cassandra N2

Cassandra N3

Snapshot of LUN2 will point to stale data

Shared Storage cluster

LUN1: (K1, V1, Told), … (K1, V11, Tnew)
LUN2: (K1, V1, Told), …
LUN3: …
NoSQL DB Protection – Challenge

NoSQL backup/restore challenges

- Flexible topology restores
  - Production cluster topology may have changed since backup
    - Commodity components may fail, cluster might be re-scaled
  - May need to restore to smaller/larger cluster for test/dev or analytics

- Challenge: Storage *needs* context of NoSQL DB cluster topology changes
Data Protection Summary and Solution Directions

Potential Solution directions

- **Cluster-consistency at scale**
  - Need to tolerate faults during backup and combat eventual consistency
  - Potential solution directions:
    - Take crash consistent snapshots
    - Post process crash consistent snapshots (in a sand-box) using NoSQL DB stack to reach an cluster-consistent state [4]

- **Space Efficiency**
  - Replica set data copies do not de-duplicate. Moreover data could be encrypted.
  - Potential solution direction: Remove replicas logically (application aware backup)

- **Topology Changes**
  - Cluster topology may change across backup and restore schedules
  - Storage snapshots do not have context about cluster topology
  - Use cases may require restore to a test/dev cluster of different cardinality
  - Solution direction: Save Cluster topology and storage mapping as part of backup

More details in to-be-published USENIX, HotStorage paper:


URL: https://www.usenix.org/conference/hotstorage17/program/presentation/kathpal
Conclusions

- Shared storage has relevance as backend storage for NoSQL DBs
  - Independent scaling of compute and storage
  - Storage consolidation => easier administration of resources
  - Flash based networked storage can meet challenges of performance, scalability and consolidation

- Data protection
  - Existing solutions have several inefficiencies like copy-based backup and have poor integrations with shared storage
  - Opportunity for shared storage to provide differentiation through efficient snapshots and clones
  - Need to address challenges of cluster consistent and storage efficient backup and flexible topology restores
Thank You.

atish-[at]-netapp.com

Acknowledgements: Priya Sengal, Gaurav Makkar, Parag Deshmukh