Clustering/HA

Introduction, Helium Capabilities, and Potential Lithium Enhancements

Overview

- Introduction to Clustering
  - Goals
  - Terminology
  - Functionality
- Helium Clustering Capabilities
- Potential Lithium Enhancements
- APP HA Use Cases

"Design With Clustering in Mind."

Why Cluster?

1) **Fault Tolerance**
   - Replicate Data to Allow Recovery in the Event of a Failure
2) **Scale**
   - Distribute Work Across Set of Available Controllers
   - (Ideally) Keep Data and Execution of Operations on that Data on Same Controller

What is Needed For Clustering?

- Form and Maintain a Cluster of Controllers
- Divide and Replicate Data Among Members
- Fast & Safe Datastore API
- Remote Execution
- Cluster Status and Change Notifications
- Defined Network Partition Behavior
- Security

Discussion Outline

1) **Cluster Configuration**
2) **Datastore Clustering & Sharding**
3) **Autonomous Data Replication**
4) **Distributed Execution**

Focus: Clustering Concepts Over Code.

Akka

- Actor System (Framework)
- Use Akka-Provided Services and Custom Actors

1) Cluster Configuration

- Create/Delete Cluster
- Automatic Member Discovery and Management
- Track Member Reachability and Availability
- Cluster-Event Notifications
- Transport Config & Security
  - Ports
  - Authenticate Members
  - Encrypt Data Exchange
- Define “Cluster Leader” For Simplified Team Interaction
- Other Configuration (Failure Detection Interval, etc.)

Akka-Clustering

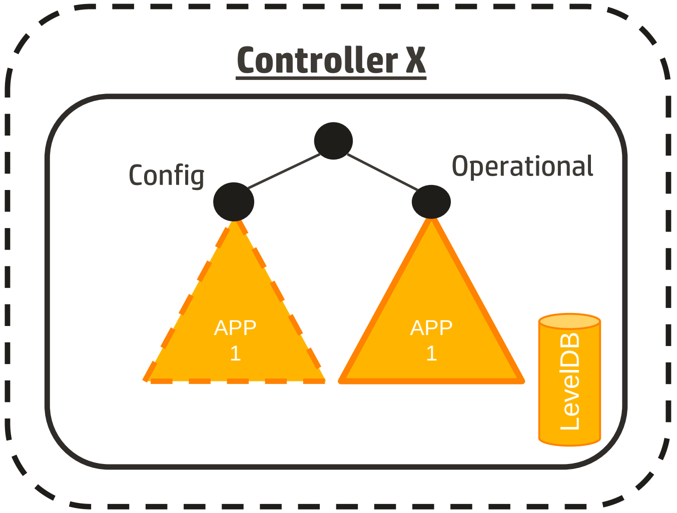
- Define Members + Discovery
- Tracks Membership (Failure Detector)
- Transport Config (Netty)

1) Cluster Configuration

Helium	**Lithium** (Potential Changes)
Static Team Configuration on Startup (akka.conf)	Programmatic Team Configuration Dynamic Team Formation
Akka-Specific Cluster-Member Event Notifications	Generalize Cluster Events
Local APP Deployment	Team-Wide APP Deployment (Apache Karaf Clustering?)
Odd # Node Clusters	Even # Node Clusters
	Team Leader (IP, Role, etc.)
Security Not Enabled	Member Authentication + Encryption

2) Datastore Clustering & Sharding

- Datastore:
  - Define "Shard"
  - Define "Sharding Strategy"
- Transparent Replacement for In-Memory Datastore (Distributed)
  - Transactions
  - Data Change Notifications
- Persistence (Optional)
- Scalability



Akka-Persistence

- Journaling / Snapshotting
- Configurable Plugins (Flat File, LevelDB)

2) Datastore Clustering & Sharding

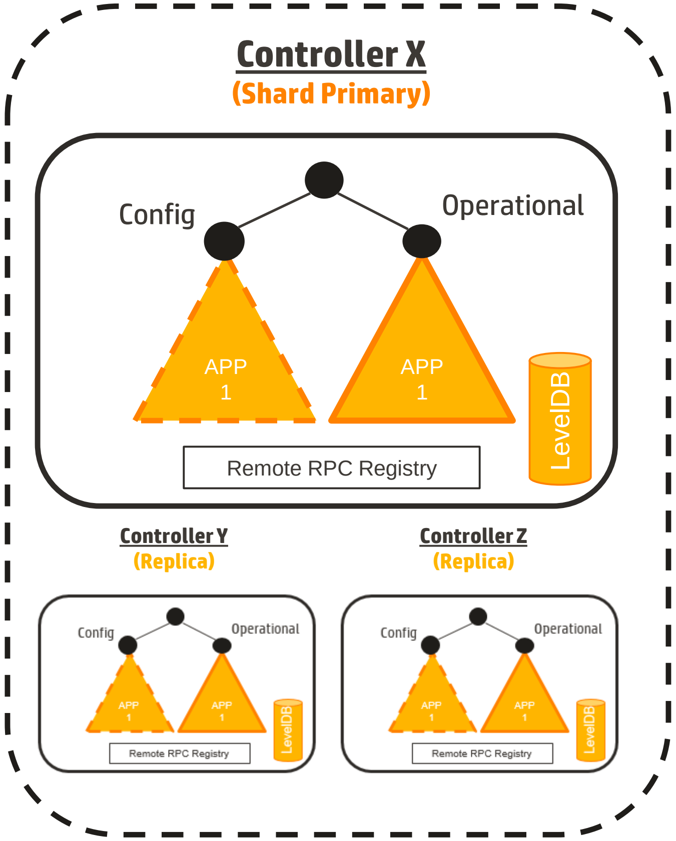
Helium	**Lithium** (Potential Changes)
Static Shard Configuration (module.conf, module-shards.conf)	Programmatic Shard Configuration
Top-Level Module Sharding Strategy	Finer-Grain Sharding (APP Defined)
Shard Persistence (enabled/disabled)	Finer-Grain Persistence Control (Node, Subtree, other?)
Single-Shard Transactions	Multi-Shard Transactions

3) Autonomous Data Replication

- Automatic Replication of Shards
  - Re-replication on Failures
- Dynamic Configuration:
  - Replication Level
  - Shard Placement
  - Consistency (R/W)
- Defined Network Partition Behavior (Quorum vs. Non-Quorum)

RAFT Consensus

- Actor-Based Implementation on Akka



RAFT Consensus in 60sec

- Goal: All Members of Cluster Agree on a Variable's Value.
  (e.g. Content of Operational Datastore Node(s))

1) **Leader Election**
2) **Data Replication**

- RAFT Elects and Maintains a Leader (Shard)
  - Leader Handles R/W Request and Replicates Data to Other Members
- ALL Reads/Writes Go Through Shard Leader (Strong Consistency)
  - On Write a Quorum of Nodes Must be Written Before Write Completes
- Minority (Non-Quorum) Partition Could Continue Operating
  - Reads Not Consistent With Majority
  - Writes Lost When Partition Healed

3) Autonomous Data Replication

Helium	**Lithium** (Potential Changes)
Static Shard Placement / Replication Levels (module-shards.conf)	Programmatic Shard Placement and Replication Levels
Don't Maintain Replication Level on Failure	Re-replicate to Maintain Configuration (config & team-size permitting)
	Special Replication Levels (Full, Quorum)
Strong Consistency (R/W Through Leader Only)	Adjustable R/W Consistency (Per Query?, Shard?)
Any Shard Can Be Elected Leader	Influence Shard Leader Election (Cluster Load Balancing)
Suspend on Non-Quorum Network Partition	Support (R, W, R/W) on Non-Quorum & Merge on Network "Heal"?

4) Distributed Execution

- Location Transparency For RPCs
  - Automatic Routing
  - Remote RPC Registry Updates
- Execute Operation on Data Owner

<u>Akka-Remoting</u>

- Location Independence
- Remote RPC Registry
(Custom Actors)

4) Distributed Execution

Helium	**Lithium** (Potential Changes)
Static Remote RPC Registry Update Interval (Gossip)	Programmable Update Interval

APP HA Use Cases

- High-Availability Modes of Operation:
  1) Active-Active
  2) Active-Passive (aka Active-Standby)

Discussion / Q&A

Resources

- **ODL Clustering DOCs:**
  - General Design
  - Detailed Design
  - Feature Summary
  - RAFT Consensus
  - Akka
- **Contacts:**
  - Moiz Raja (moraja@cisco.com)
  - Raghurama Bhat (ragbhat@cisco.com)
  - Harman Singh (harmasin@cisco.com)
  - Mark Mozolewski (mbm@hp.com)