MUSIC-Multi-site State Coordination Service

Project Name: MUSIC - Multi-site State Coordination Service

Project description:

To achieve 5 9s of availability on 3 9s or lower software and infrastructure in a cost-effective manner, ONAP components need to work in a reliable, active-active manner across multiple sites (platform-maturity resiliency level 3). A fundamental aspect of this is state management across geo-distributed sites in a reliable, scalable, highly available and efficient manner. This is an important and challenging problem because of three fundamental reasons:

- Current solutions for state-management of ONAP components like MariaDB clustering, that work very effectively within a site, may not scale across geo-distributed sites (e.g., Beijing, Amsterdam and Irvine) or allow partitioned operation (thereby compromising availability). This is mainly because WAN latencies are much higher across sites and frequent network partitions can occur.
- ONAP components often have a diverse range of requirements in terms of state replication. While some components need to synchronously manage state across replicas, others may tolerate asynchronous replication. This diversity needs to be leveraged to provide better performance and higher availability across sites.
- ONAP components often need to partition state across different replicas, perform consistent operations on them and ensure that on failover, the new owner has access to the latest state. The distributed protocols to achieve such consistent ownership is complex and replete with corners cases, especially in the face of network partitions. Currently, each component is building its own handcrafted solution which is wasteful and worse, can be erroneous.

In this project, we identify common state management concerns across ONAP components and provide a multi-site state coordination/management service (MUSIC) with a rich suite of recipes that each ONAP component can simply configure and use for their state-management needs.

Functionality:

At its core, MUSIC provides a scalable sharded eventually-consistent data-store (Cassandra) wherein the access to the keys can be protected using a locking service (built on Zookeeper) that is tightly coupled with the data-store. ONAP components can use the MUSIC API directly to store and access their state across geo-distributed sites. This API enables ONAP components to achieve fine-grained flexible consistency on their state.

MUSIC also provides a rich set of recipes (mdbc, prom, musicCAS, musicQ) that ONAP components can use to build multi-site active-active/active-passive /federated state-management solutions:

- **mdbc**: The most crucial recipe is a multi-site database cache (mdbc) that enable ONAP components that maintain state in a SQL database to avail the benefits of MUSIC without compromising their need to use transactional SQL DBs. These ONAP components can rely on existing db clustering techniques like MariaDB for transactionality and complex querying within a site. mdbc will intercept each of these read/write calls to the db cluster and mirror this state to other geo-distributed sites through MUSIC either synchronously or asynchronously (configurable at a table-level). For example, components like the ONAP Service Orchestrator that use MariaDB to maintain state can directly use this recipe with no change to their SQL code.
- **prom**: MUSIC provides a recipe for policy-driven ownership management (prom) of state for ONAP components to (1) partition state across replicas during both initial placement and during failures based on their individual policies (2) ensure correct transfer of state ownership across replicas during site failures and network partitions (3) ensure that the new owner has access to the most recent version of state (if needed).
- **musicCAS**: The distributed compare and set is a powerful primitive that will allow ONAP components to update shared state in an atomic manner. This is currently being used by the ONAP HAS (homing service) that is structured a job scheduler with multiple workers trying to pick up client-submitted jobs, while ensuring that only one of them actually performs the job.
- **musicQ**: Implementing a geo-distributed queue is a hard problem with many performance implications when data belonging to a queue is sharded across nodes. MUSIC provides a queue API that carefully structures the data to ensure good performance. ONAP HAS (mentioned above) uses this as its job queue.

Scope:

MUSIC is a shared service with recipes that individual ONAP components and micro-service can use for state replication, consistency management and state ownership across geo-distributed sites. MUSIC will make sure that the right service data is available at the right place, and at the right time to enable federated active-active operation of ONAP. For example, we envisage the use of MUSIC for multi-site state management in SO (to store Camunda state across sites), <SDN-C, AppC> (to store ODL related state across sites) , A&AI (to store its graph data) and most other ONAP components that need to manage state across sites.

Out of Scope:

While MUSIC is an optional solution for state-management of ONAP components across sites, OOM will continue to manage component level and platform level deployment, scalability, redundancy, resiliency, self-healing and high availability on top of Kubernetes across sites for ONAP.

Usage:

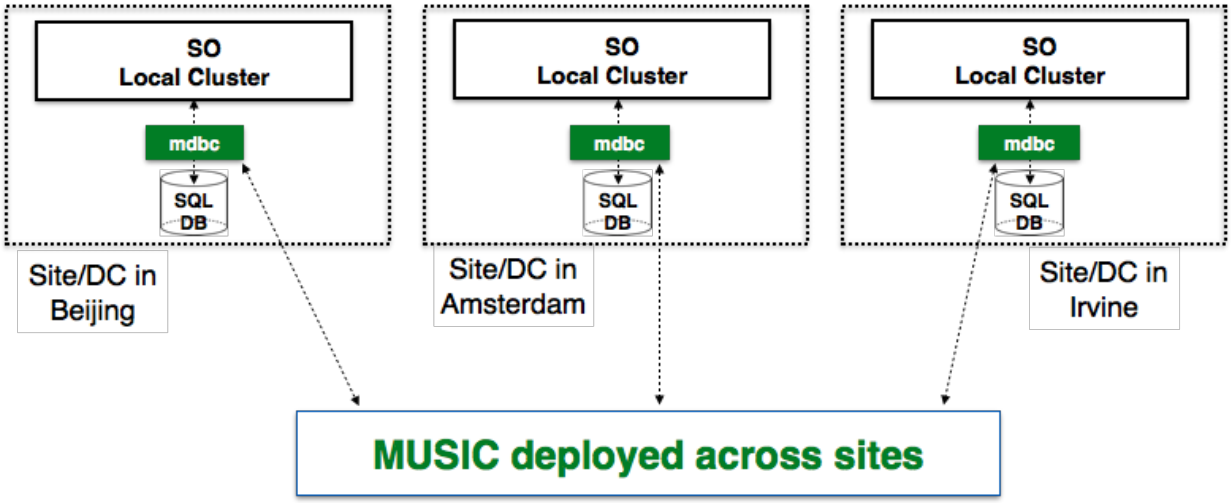
MUSIC and its recipes export a REST API apart from mdbc which is implemented as a jdbc driver to enable seamless integration with SQL-based ONAP components.

Architecture:

The figures below how MUSIC can be used in a general context and also provide a specific example of its potential usage in ONAP SO.

blocked URL

A specific example:



Seed Code Status:

- MUSIC and its recipes have all been open sourced in github:MUSIC.
- Here is an overview of MUSIC's REST API: https://github.com/att/music/blob/master/RestMusicOverview.pdf
- MUSIC and mdbc together can support the following databases: Cassandra, MySQL, MariaDB, H2.
- OOF-Homing Optimizer (HAS) uses MUSIC for its state persistence (as a queue) and as a highly available distributed messaging service. This is currently being run in production within ATT's ECOMP.

Architecture Alignment:

- How does this project fit into the rest of the ONAP Architecture?
  MUSIC will be available as a common service like DMaap or AAF as shown in the red, oblong box below:

blocked URL

- What other ONAP projects does this project depend on?

  Since OOM is responsible for the life-cycle of ONAP components, it will also need to manage the deployment of MUSIC.
- How does this align with external standards/specifications?

  MUSIC and its recipes export a REST API apart from mdbc which is implemented as a jdbc driver to enable seamless integration with SQL-based ONAP components.

- Are there dependencies with other open source projects?

  MUSIC depends primarily on Apache Cassandra, Zookeeper and the jdbc driver.

Other Information:

- link to seed code (if applicable)
- Vendor Neutral
  - if the proposal is coming from an existing proprietary codebase, have you ensured that all proprietary trademarks, logos, product names, etc., have been removed?
- Meets Board policy (including IPR)

Use the above information to create a key project facts section on your project page

Key Project Facts:

Facts	Info
PTL (first and last name)	Bharath Balasubramanian
Jira Project Name	Music
Jira Key	MUSIC
Project ID	music
Link to Wiki Space	MUSIC Project

Release Components Name:

Note: refer to existing project for details on how to fill out this table

Components Name	Components Repository name	Maven Group ID	Components Description
music	music	org.onap.music	This repo contains the code for a multi-site coordination service (MUSIC) and associated recipes that enables efficient, highly available state-management across geo-redundant sites.

Resources committed to the Release:

Role	First Name Last Name	Linux Foundation ID	Email Address	Location
PTL	Bharath Balasubramanian	bharathb	bharathb@research.att.com	Bedminster, NJ, USA
Committers				
	Bharath Balasubramanian	bharathb	bharathb@research.att.com	Bedminster, NJ, USA
	Viswanath Kumar Skand Priya	kspviswa-vz	viswanath.kumarskandpriya@verizon.com	Chennai, India
	Thomas Nelson Jr.	arthurdent3	tn1381@att.com	Bedminster, NJ, USA
	Greg Waines	gwaines	Greg.Waines@windriver.com	
Contributors				
	Srinivasa R Addepalli		srinivasa.r.addepalli@intel.com	
	Gil Hellman		gil.hellmann@windriver.com	
	Manoj Nair		manoj.k.nair@netcracker.com	
	Abbas Fazal		afazal@research.att.com	Bedminster, NJ, USA