July 2022

DYNAMIC ONBOARDING PROCESS OF MAP DATA STRUCTURE ON HAZELCAST CLUSTER

Neeraj Deshmukh
Visa

Rohit Kesarwani
Visa

Aarushi Gupta
Visa

Prabu Kadapenthangal Venkatesan
Visa

TITLE OF THE INVENTION

"DYNAMIC ONBOARDING PROCESS OF MAP DATA STRUCTURE ON HAZELCAST CLUSTER "

Field of the Invention:

The present invention relates to creation and deployment of data structure on Hazelcast cluster more particularly to a method of dynamic onboarding process of map data structure on Hazelcast cluster.

Background:

Hazelcast is a distributed computation and storage platform for consistently low-latency querying, aggregation and stateful computation against event streams and traditional data sources. Hazelcast can process data on a set of networked and clustered computers that pool together their random access memories (RAM) to let applications share data with other applications running in the cluster. Using Hazelcast, you can store and process your data in RAM, spread and replicate it across a cluster of machines.

In Hazelcast, whenever a new onboarding or change in existing map configuration is required, the Hazelcast service should get restart to reflect the changes. hazelcast.xml file is used to deploy the map configuration on Hazelcast cluster. This file is maintained and managed by Application Support Engineer (ASE) team. Application team collaborates with ASE team to add or modify map configuration in hazelcast.xml file. The platform team reviews the modifications and participates in Production deployment. Once the production deployment is started, the VOCC team takes care of switching data center traffic in the production. The Middleware team implements the changes in Production. The Application Operations team performs the post validation in Production. Therefore, updating a map configuration requires a traffic switch along with support from multiple teams. Further, the middleware team takes about 2 to 3 hours to implement and validate the change in Production environment for each datacenter. Hence, there is a need for a system that can reduce the dependency on multiple teams and traffic switch to create / update a map configuration on Hazelcast cluster.

Summary:

In order to solve the above problems, the present invention provides a method to dynamically onboard the map data structure on Hazelcast cluster. To reduce the dependency on multiple teams for onboarding process of a map data structure, the present invention creates a repository

and maintains the hazelcast.xml file. The application team can add or modify the hazelcast.xml for new map onboarding or existing map configuration. Once the changes are done, the application team generates the RPM build. The platform team reviews the changes and deploy the RPM build in Hazelcast cluster. The dynamic onboarding of map on Hazelcast cluster starts upon installing the RPM build. In this way the present invention reduces dependency on multiple teams for new onboarding of a data structure in Hazelcast cluster .

The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

Description of Drawings:

The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and together with the description, serve to explain the disclosed principles. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The same numbers are used throughout the figures to reference like features and components. Some embodiments of device and/or methods in accordance with embodiments of the present subject matter are now described below, by way of example only, and with reference to the accompanying figures.

Figure 1 illustrates the basic working of Hazelcast cluster in accordance with some embodiments of the present invention.

Figure 2 illustrates the method for new onboarding the map data structure dynamically on Hazelcast cluster in accordance with some embodiments of the present invention.

Figure 3 illustrates the workflow process of onboarded map data structure on Hazelcast cluster in accordance with some embodiments of the present invention.

It should be appreciated by those skilled in the art that any block diagrams herein represent conceptual views of illustrative systems embodying the principles of the present subject matter. Similarly, it will be appreciated that any flowcharts, flow diagrams, state transition diagrams, pseudo code, and the like represent various processes which may be substantially represented

in computer readable medium and executed by a computer or processor, whether or not such computer or processor is explicitly shown.

Detailed Description:

In the present document, the word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any embodiment or implementation of the present subject matter described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments.

While the disclosure is susceptible to various modifications and alternative forms, specific embodiment thereof has been shown by way of example in the drawings and will be described in detail below. It should be understood, however that it is not intended to limit the disclosure to the particular forms disclosed, but on the contrary, the disclosure is to cover all modifications, equivalents, and alternatives falling within the scope of the disclosure.

The terms “comprises”, “comprising”, or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, device, or method that comprises a list of components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or device or method. In other words, one or more elements in a system or apparatus proceeded by “comprises… a” does not, without more constraints, preclude the existence of other elements or additional elements in the system or method.

In the following detailed description of the embodiments of the disclosure, reference is made to the accompanying drawings that form a part hereof, and in which are shown by way of illustration specific embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure, and it is to be understood that other embodiments may be utilized and that changes may be made without departing from the scope of the present disclosure. The following description is, therefore, not to be taken in a limiting sense.

Embodiments of the present invention are directed to methods for dynamically onboard the map data structure on Hazelcast cluster. The method includes creating the repository for xml file. The method also includes creating/updating the map xml and generating feature branch

RPM build for the xml file. The method further includes installing the feature branch RPM build in Hazelcast Develop cluster to onboard the map dynamically.

Figure 1 illustrates the basic working of Hazelcast cluster in accordance with some embodiments of the present invention. Figure 1 comprises of multiple servers (101a, 101b, 101c) each containing an application (102a,1012b, 102c). A database (104) stores the data used by the multiple applications present in the multiple servers. The applications (102a,1012b, 102c) retrieve the data from the Hazelcast (103). The Hazelcast (103) is a distributed memory cache that is present in the memory of a system. The working principle of Hazelcast (103) is similar to cache which stores the data for quick retrieval.

In an embodiment, the working principle of Hazelcast is explained through an example. The application (102a) retrieves the data (A) from the database (104). The data (A) is stored in the Hazelcast (103), and same data (A) can be accessed by the application (102b) and application (102c) without accessing the database for retrieval of data (A). The Hazelcast (103) provides distributed data along the applications for quick retrieval of data.

Figure 2 illustrates the method for new onboarding the map data structure dynamically on Hazelcast cluster in accordance with some embodiments of the present invention. The method includes creating a repository for hazelcast xml file at step (201), creating or updating map configuration in xml file at step (202), generating the rpm build for xml file at step (203), installing the rpm build in the Hazelcast cluster at step (204), performing the sanity test at step (205) and merging the changes and releasing the build of the xml file at step (206).

In an embodiment, the steps (201) – (205) are performed by the application team. The step (206) is performed by the platform team.

In another embodiment, the application team maintains the xml file in the created repository at step (201). In another embodiment, the application team creates a new map configuration of the xml file / modifies the existing map configuration of the xml file.

In another embodiment, the RPM build is generated for the created/modified xml file. The RPM build is a command-line driven package management system that install, uninstall, verify, query, and update the software packages.

In another embodiment, the rpm build is installed in the Hazelcast cluster and performs the sanity test. The sanity test is performed to check whether the changes / modification made in the code are working as expected or not. Once the sanity test is cleared, the modifications made are merged and build of the build new configured map is released.

In another embodiment, the released build is installed on the managed environment with the help of ASE team and the Middleware team.

Figure 3 illustrates the workflow process of onboarded map data structure on Hazelcast cluster in accordance with some embodiments of the present invention. The client (301) accesses the map using an application id. When the Hazelcast cluster (302) receives the request of accessing the map from a client (301), the hazelcast cluster (302) checks the map permission associated with the application id at the database (303) and returns the permission of access associated with the requested application id to the hazelcast cluster (302). If the map permission is associated with the application id in the hazelcast cluster (302) then the client (301) is authorized to perform the map operation. The Hazelcast cluster (302) creates a session for the client and sets the map permission at cluster level for the requested application id.

In another embodiment, the hazelcast cluster (302) onboards the map dynamically using App onboard rpm. In another embodiment, the application id is uploaded in the database (303) using credential uploader rpm.

Therefore, in this way even when a map configuration is updated, the app on board rpm dynamically onboards the updated map configuration on the Hazelcast cluster without restart of the cluster.

Advantages:

- By creating the repository for maintaining the xml file, the multiple team efforts, manual changes, and traffic switch are avoided in Production. Further application team can create or modify the xml file without depending on the ASE team.
- The cluster restart is not required upon each new map configuration by building the rpm for xml file.

Abstract:

The present invention provides a method to dynamically onboard the map data structure on Hazelcast cluster. To reduce the dependency on multiple teams for onboarding process of a map data structure, the present invention creates a repository for the xml file. The application team can create or modify the xml file for new map onboarding or existing map configuration. Once the changes are made, the application team generates the RPM build for the xml file. The platform team reviews the changes and deploy the RPM build in Hazelcast cluster. The dynamic onboarding of map on Hazelcast cluster starts upon installing the RPM build. In this way the present invention reduces dependency on multiple teams for new onboarding of a data structure in Hazelcast cluster.

#

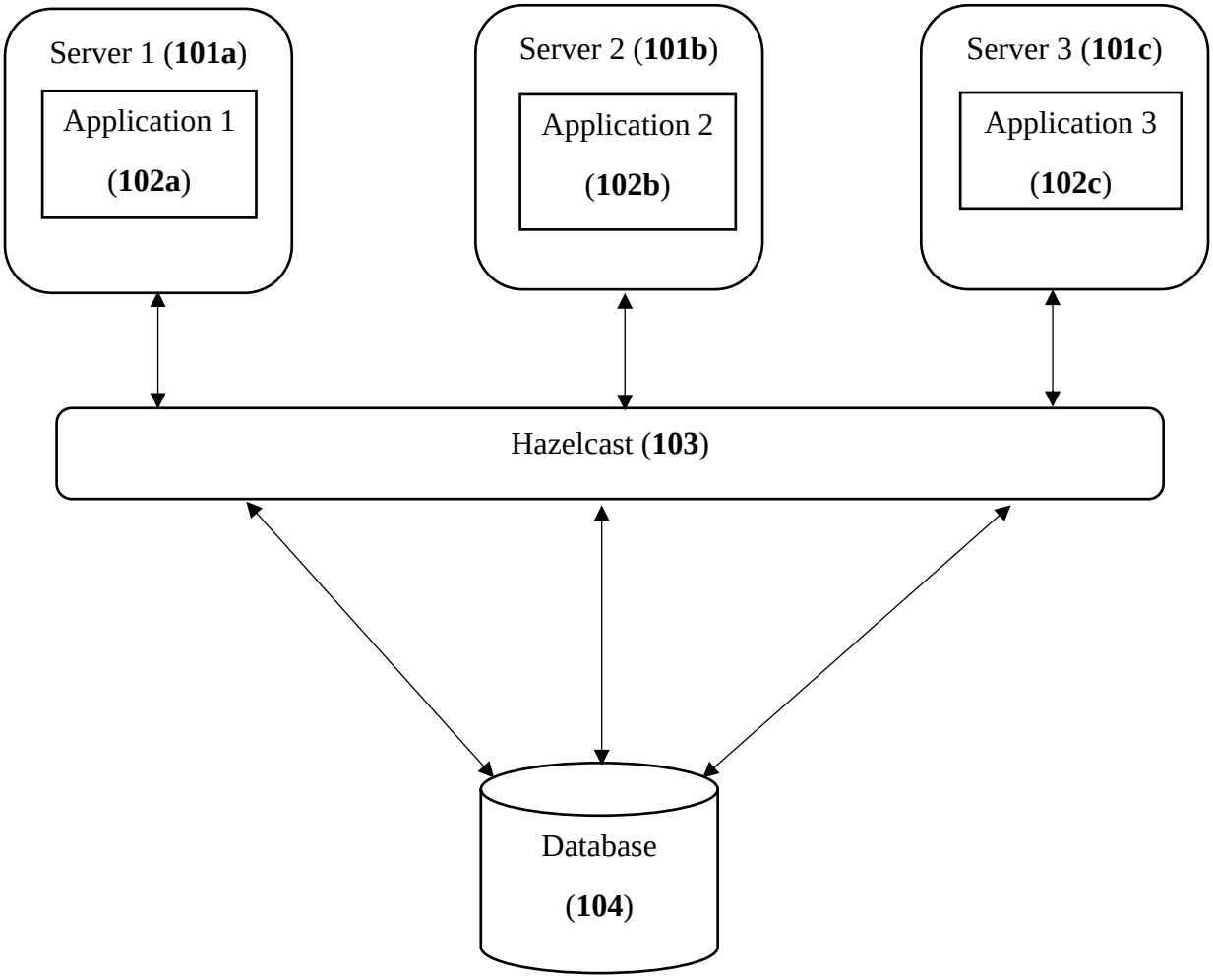
Server 1 (**101a**)
Application 1
(**102a**)
Server 2 (**101b**)
Application 2
(**102b**)
Server 3 (**101c**)
Application 3
(**102c**)
Hazelcast (**103**)
Database
(**104**)

Figure 1

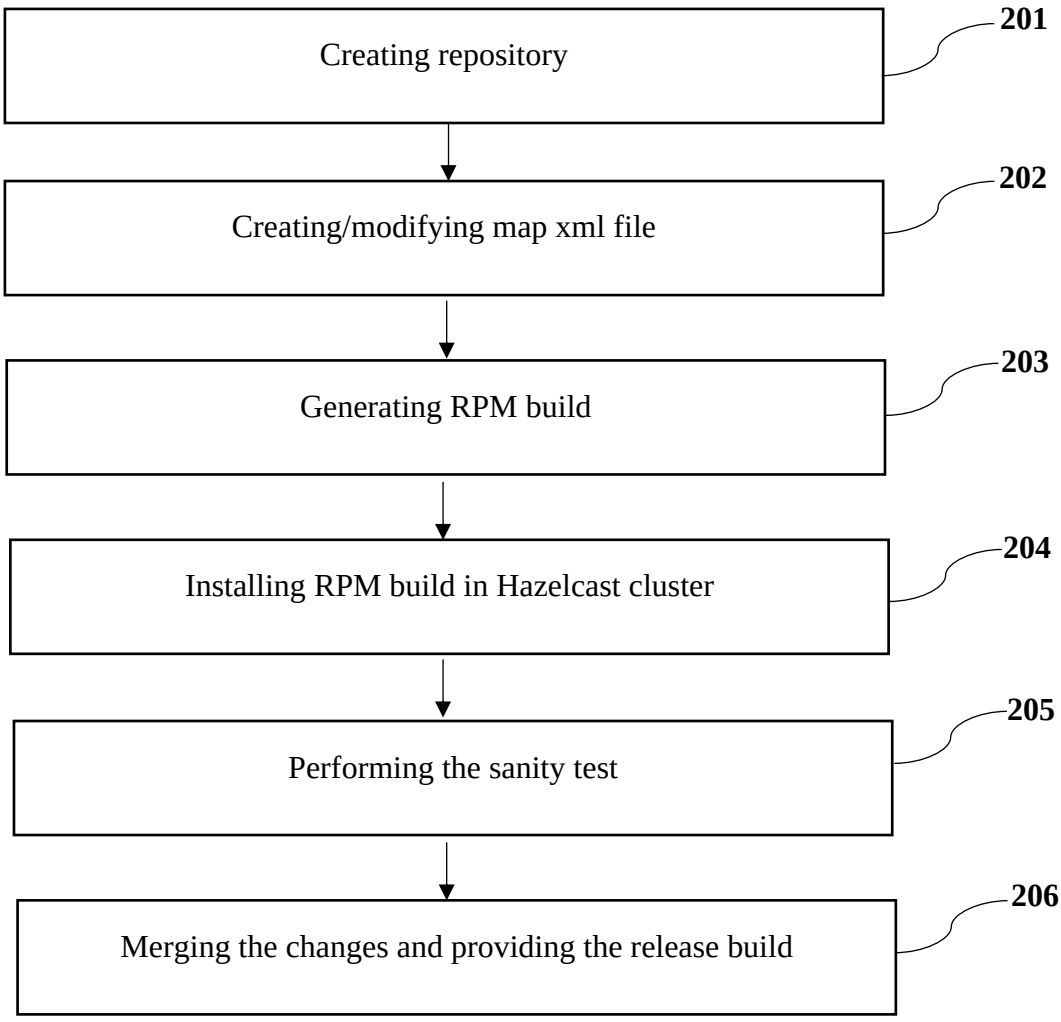
Creating repository
201
Creating/modifying map xml file
202
Generating RPM build
203
Installing RPM build in Hazelcast cluster
204
Performing the sanity test
205
Merging the changes and providing the release build
206

Figure 2

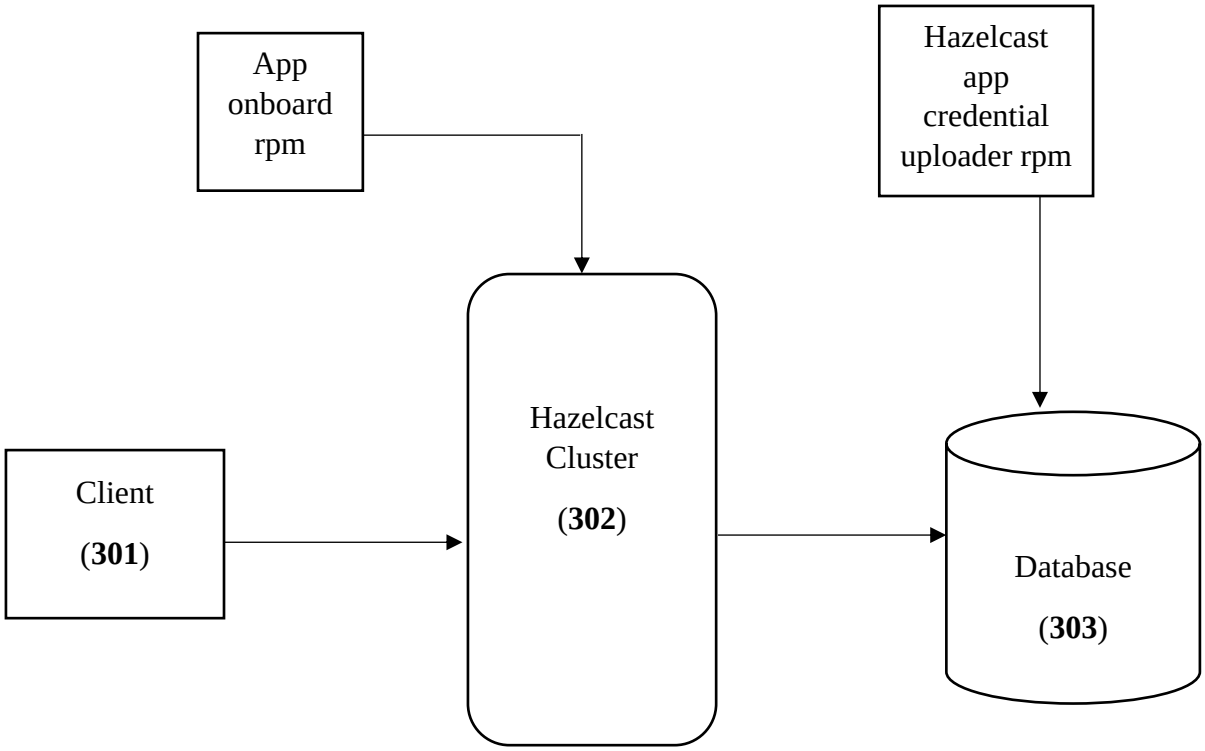
App onboard rpm
Hazelcast app credential uploader rpm
Hazelcast Cluster
(302)
Client
(301)
Database
(303)

Figure 3