Chapter 3. Server Regions - Replicated and Partitioned

3.1. Introduction

This lab will take you through the process of creating both replicated regions and partitioned regions in GemFire. An XML file will be used for the configuration, which is generally a best practice.

Concepts you will gain experience with:

- How to create a replicated region across multiple servers
- Testing the failover and refresh of data on recovered nodes
- Configuring partitioned regions in GemFire

Estimated completion time: 45 minutes

3.2. Quick Instructions

Quick instructions for this exercise have been embedded within the lab materials in the form of TODO comments. To display them, open the `Tasks` view (Window -> Show view -> Tasks (*not Task List*)).

3.3. Detailed Instructions

Instructions for this lab are divided into specific sections. Each section describes the steps to perform specific tasks.

If servers and locator are still running from the previous lab, be sure to stop them at this point. Remember, in order to shut the services down, you'll need to re-connect to the locator using the command **connect --locator=localhost[41111]**.

3.3.1. Creating Replication Regions

In this first section, you will work with the BookMaster region. The data that this region holds is a relatively small amount, is slow changing, and tends to be reference data. This is the type of data that is a good candidate for replication. We will walk through defining a region as replicated, watching shutdown of nodes, and the redundancy.

1. Browse to the server-regions project folder on the command line. This will be your starting point for this lab.
2. Use the STS to open the `serverCache.xml` file in the server-regions project folder. Add a region attribute to define the `BookMaster` region as a replicated region.
3. If the locator is not running, start it using `gfsh` and the command similar to the one used in the prior lab.
4. Start `server1` using a similar command as used in the prior lab. Do NOT start the second server yet.

> **Note**
>
> As of GemFire 8, the server startup command from within gfsh has changed and does not automatically include everything in the current CLASSPATH environment variable in the classpath of the servers when started. Most of this lab makes use of a special function that is registered with the server that must be on the server classpath. As a result, you will need to explicitly include the option: **--classpath=../target/classes/** to the server start commands.

5. In the STS, locate and run the `BookLoader` class to load 3 books into the `BookMaster` region.
6. Open the `ReplicationTest` class to understand how it looks for values in the `BookMaster` region.
7. Run the `ReplicationTest` to verify that the books were found.
8. Start the second server using the name `server2` so it will write log data to a different directory.
9. You can examine the region details by executing the gfsh command **show metrics --region=/BookMaster**. Note the member count and the number of entries for the cluster.
10. Stop server1 using the gfsh **stop server** command.
11. Re-run the `ReplicationTest` application to verify that the data still can be found in the remaining server (the new server 2 that was started).
12. Stop all the servers in preparation for the next section.

3.3.2. Creating Partitioned Regions

In this section, you will use the Customer region and try out different partitioning scenarios. You will be using three server instances this time so you can see the benefits of partitioning with redundancy.

1. Return to the `serverCache.xml` file and modify the `Customer` region to set the region type to partitioned.
2. If you stopped the locator along with the servers in the prior section, re-start the locator.
3. Start three servers, calling them `server1`, `server2`, and `server3`.

Note

If you specify the `--server-port=0` argument on each of these, the ports will be auto-assigned and registered with the locator.

4. Run the `CustomerLoader` class to load 3 customers into the distributed system.
5. You can observe the partitioning by issuing the following command.

```
gfsh> show metrics --region=/Customer --member=server1 --categories=partition
```

Note the reported values for `bucketCount`, `primaryBucketCount` and `configuredRedundancy`. You might try this for all three servers.

6. Now, stop all the servers (but not the locator).

3.3.3. Partitioned Regions with Redundancy

In the prior partitioned region configuration, if one of the servers stops for some reason, all the data stored in that partition is lost. In this section, we'll address that by adding a redundancy factor.

1. Go back to the serverCache.xml file and modify the Customer region attributes to add a redundancy of 1 (meaning there will be one primary and one redundant copy of every entry).

Tip

You can either do this by modifying the region shortcut or by inserting a `partition-attributes` element and specifying this. However, in a later step, you'll add a recovery delay value so you may want to take the extra time to type in the `partition-attributes` element now.

2. Save the file and re-start the servers. Re-run the `CustomerLoader` class to re-load the customers.
3. Repeat the show metrics command to see what has changed with the updated partitioned region configuration.
4. Now, stop `server3` and repeat the **show metrics** command for the remaining two servers. You'll notice that the `primaryBucketCount` value will have increased from 1 to 2 indicating that one of the redundant copies was promoted. Notice also that `numBucketsWithoutRedundancy` is not 0. This indicates that when the server was lost, the redundant bucket was promoted redundancy was not re-established for this or any redundant buckets that were on that server.
5. You can obtain even more detail using the special `PRBFunction` that has been registered with this project. T o make use of this functionality, run the `PRBFunctionExecutor` program found under the `com.gopivotal.training.prb` package in the STS. You'll see that a very extensive output is printed that displays every primary bucket and every redundant bucket for each server. You can see by which ones have a size > 0 the buckets that have entries. You should see output similar to the following for every server.

```
Member: HostMachine(server2:77234)<v2>:58224
 Primary buckets:
  Row=1, BucketId=2, Bytes=0, Size=0
  Row=2, BucketId=4, Bytes=0, Size=0
  Row=3, BucketId=9, Bytes=0, Size=0
  Row=4, BucketId=12, Bytes=0, Size=0
  Row=5, BucketId=13, Bytes=0, Size=0
              ....
  Row=20, BucketId=60, Bytes=0, Size=0
  Row=21, BucketId=61, Bytes=676, Size=1
```

6. Stop the servers once again

Congratulations!! You have completed this lab.

3.3.4. Partitioned Regions with Redundancy and Recovery Delay

This time, you will add a recovery delay so that after a period of time, redundancy will be re-established. This will address the issue identified in the prior section.

1. Go back to the `serverCache.xml` file and modify the partition-attributes element to define a recovery delay of 5 seconds.

Tip

If you used a region shortcut in the prior section, you'll need to add a partition-attributes element inside the region-attributes element for the `Customer` region

2. Save the file and re-start all the servers. Re-run the `CustomerLoader` class to re-load the customers.
3. Now, stop `server3` and repeat the **show metrics** command for the remaining two servers. If you run this command within 5 seconds of stopping `server3`, you'll likely see the `numBucketsWithoutRedundancy` is

still not 0. Wait a few more seconds and repeat the command. You should see that this value will be returned to 0. This indicates that redundancy has been re-established within the remaining servers.

4. Alternatively, you can use the `PRBFunctionExecutor` class to print out more detailed bucket listing as outlined in the prior section.
5. Stop the servers for the final time. Also stop the locator.

Congratulations!! You have completed this lab.
