Infrastructure - Task #3045

Task # 3116 (Closed): Metacat CN startup and initialization time needs to be reduced an order of magnitude

Reduce Hazelcast.init() startup time for populating hzIdentifiers

2012-07-02 16:43 - Ben Leinfelder

Status:	Closed	**Start date:**	2012-07-02
Priority:	High	**Due date:**	
Assignee:	Ben Leinfelder	**% Done:**	100%
Category:	Metacat	**Estimated time:**	0.00 hour
Target version:	Sprint-2012.27-Block.4.2		
Milestone:	None	**Story Points:**	
Product Version:	*		

Description

Initializing the shared Identifiers set takes hours in production. We believe this is due to HZ migrating "ownership" of the identifiers so that each node has an equal proportion. We don't really use this notion since all nodes are menat to have complete sets.
Would be nice if we could completely disable this or if it could run in the background and not prevent startup of the webapp.

Here is the relevant code in Metacat:

```
// Get a reference to the shared identifiers set as a cluster member
// NOTE: this takes a long time to complete
identifiers = Hazelcast.getSet(identifiersSet);
logMetacat.debug("got identifiersSet");

identifiers.addAll(loadAllKeys());
logMetacat.debug("Initialized identifiers");
```

History

#1 - 2012-08-03 01:14 - Chris Jones

- Parent task set to #3116

#2 - 2012-08-03 01:17 - Chris Jones

- Subject changed from HazelcastService.init() takes hours to complete to Reduce Hazelcast.init() startup time for populating hzIdentifiers

- Category set to Metacat

- Status changed from New to In Progress

- Target version set to Sprint-2012.27-Block.4.2

#3 - 2012-08-07 15:12 - Ben Leinfelder

- Status changed from In Progress to Closed

The current approach we are taking to alleviate this load is to process the differences between the shared hzIdentifiers set and the list of identifiers we have locally (for each server).
Here's a summary of the process, assuming 3 CNs are starting for the first time.
1. server A starts and sees an empty hzIdentifiers set so contributes all local ids to the shared set. This is quick because no other nodes need to be consulted and no migration events are taking place.
2. server B starts and sees that there are many identifiers already in the shared hzIdentifiers set and so performs a diff of those that are already shared and those that are not yet shared. There are two possible outcomes here: ideally, server B will not have any additional pids to add, but if it does it will only add those missing pids thereby skipping superfluous calls to hzIdentifiers.contains(). In the process of finding the pids that are missing from the shared map, we are also keeping track of the pids that the server B does not have locally (this will be used by the systemMetadata resynchronization process later).
3. server C starts and does the same thing that server B did, reaping the same efficiencies.