Starting Small and Scaling Out

Oracle NoSQL Database 11g Release 2 (11.2.1.2)

Oracle White Paper
April 2012

Oracle NoSQL Database

ORACLE

Executive Overview

Oracle NoSQL Database is a highly available, distributed key-value database, whose performance scales linearly as more storage nodes (SN) are added to the cluster. Horizontal scale-out and predictable latency of the Oracle NoSQL Database enables system architects to design enterprise systems for the acquisition and organization of big data.

It is simple to design an architectural solution for an application using Oracle NoSQL Database when the throughput, latency, and availability requirements are known and the hardware is available to deploy the solution. However, customers often want to deploy the applications on limited hardware now and expand it later in order to minimize upfront cost commitments.

This paper discusses how to deploy highly available Oracle NoSQL Database clusters on limited hardware, and how to scale-out horizontally later when more hardware becomes available.

Introduction

The Oracle NoSQL Database Admin Guide[1] provides tools to determine how many shards and replicas are needed to support a given data set. In the current release of NoSQL Database (11gR2.1.2), the number of shards and replicas is static. Once the KV-cluster is deployed and initialized (with N shards and M replicas per shard) you cannot add more storage nodes or replicas to the cluster. Elastic scalability will be added in a future release.

So what if you want to deploy and initialize a KV-cluster but don't have all the hardware that you need right now? One option would be to deploy N x M KV-cluster (where N = number of shards and M = number of replicas per shard) where multiple storage nodes share resources on some of the physical systems. These storage nodes could later be moved to separate physical systems as they become available. Note that shared resources among replication nodes will not provide optimal performance and should not be used in production. But in the absence of the required hardware, the next best thing is to deploy the desired topology on the available hardware and expand it later.

The rest of this paper discusses a couple of techniques that can be used to deploy production topologies on limited hardware and the step-by-step details of how the SNs can be migrated from shared resources to dedicated physical hardware.

[1] http://docs.oracle.com/cd/NOSQL/html/AdminGuide/installplanning.html

Deployment Techniques

Two techniques that can be used to deploy a KV-cluster on a limited set of physical hardware are, using virtual machines and configuring multiple replication nodes per physical host.

Virtual Machines (VM)

It is a common practice for many enterprises to buy high-end, multi-core, SMP machines with large amounts of RAM (10s to 100s of GB) and high throughput storage, and then use these machines to host multiple VM environments. It is certainly possible to deploy storage nodes to these VMs, using a 1:1 mapping between VMs and SNs. Once the dedicated hardware becomes available, it is possible to attach these VM instances to dedicated systems from your hardware pool. This technique of migrating VMs is simple and well understood by the user community and therefore not the focus of this paper.

Multiple Replication Nodes (RN) per physical host

The second option is to deploy multiple storage nodes on the same physical node. When additional hardware is made available, it is possible to simply run a 'Storage Migration Plan' to relocate the SN to a different physical system. We will be exploring this technique in the following sections, including real examples.

System Configuration

Let's use the following formula (as mentioned in the "Identify the Number of Replication Groups[2]" section of the Admin Guide) to arrive at a rough initial estimate of the number of replication groups (or shards) that we would need to support the workload we expect:

```
RG = (((((avg key size * 2) + avg value size) * max kv pairs) * 2) +
    (avg key size * max kv pairs) / 100 ) /
    (node storage capacity)
```

For example, a database sized to hold a maximum of 1.5 billion key value pairs, having an average key size of 10 bytes and an average value size of 1K, with 1TB (10^12) of storage available at each node, would require three shards:

```
((((10*2)+1000) * 1.5*(10^9)) * 2) + (10 *1.5* (10^9))/100) / 10^12 = 3 Shards (or RG)
```

[2] http://docs.oracle.com/cd/NOSQL/html/AdminGuide/store-config.html#num-rep-group

In order to support load balancing and availability, as recommended by the documentation, we select a replication factor of 3. Therefore, we plan to create a 3x3 configuration (three shards with three copies of the data -- one master and two replicas for each shard) to deploy our application to production. However, in order to minimize the startup cost, let's deploy the 3x3 topology on three physical systems (instead of nine) and add six more physical systems (in batches) as the workload increases to the level the system was originally designed for.

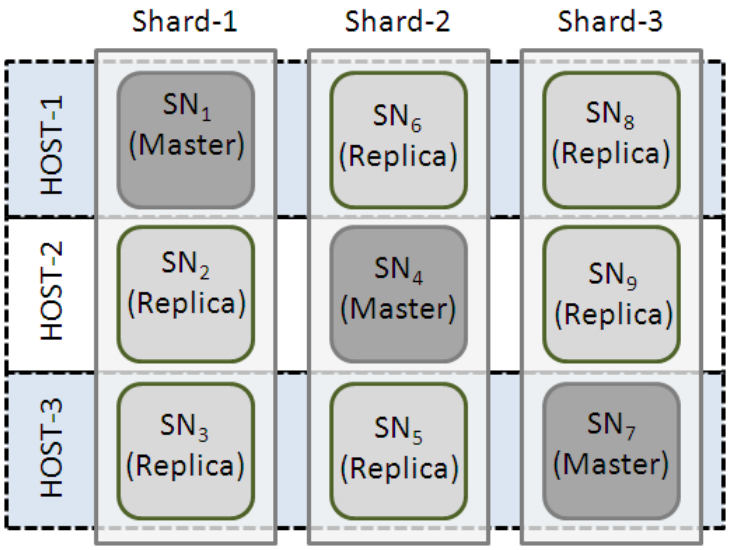

Figure 1: This depicts how the master and replica nodes should be configured at the end of the deployment process.

We need to deploy three storage nodes on each host (total of nine SNs on three hosts) and this will be achieved by creating three different KVROOTs and starting the Storage Node Agent (SNA) on different ports so that there is no conflict on any kind of resources. In the example below, we have shared the KVHOME for all three SNs on the same host.

1. Log on to Host1, Host2, and Host3 and create boot configuration on each one with three different KVROOTs and PORTs:

```
$java -jar KVHOME/lib/kvstore.jar makebootconfig -root kvroot1 -port 5000 -admin 5001 -host <host> -
harange 5010,5020
$java -jar KVHOME/lib/kvstore.jar makebootconfig -root kvroot2 -port 6000 -host <host> -harange 6010,6020
$java -jar KVHOME/lib/kvstore.jar makebootconfig -root kvroot3 -port 7000 -host <host>-harange 7010,7020
```

2. Start all three storage node agents (SNA) on Host1, Host2, and Host3:

```
$nohup java -jar KVHOME/lib/kvstore.jar start -root kvroot1 &
$nohup java -jar KVHOME/lib/kvstore.jar start -root kvroot2 &
$nohup java -jar KVHOME/lib/kvstore.jar start -root kvroot3 &
```

We now have a total of nine storage node agents (SNA) running on three physical hosts. Next, we will create a script that will deploy a 3x3 KV-cluster in such a way that the master in each shard gets deployed to a different physical host.

We recommend that you deploy two more admin processes on different physical nodes so that the admin console is also highly available. Make sure that the storage nodes that you select for deploying additional admin process run on the physical host other than the current running master admin process. In our case, we will select SN4 (Host2) and SN7 (Host3) for deploying admin replicas.

With that logic in mind, we have created a CLI script (called 3x3script.txt) that will assign the master nodes of each shard to a unique host and the replicas of that shard to the other two hosts.

```
### Begin Script ###
configure mystore
addpool BostonPool

plan -execute -name "Deploy Boston DC" deploy-datacenter "Boston" "Savvis"

#First Shard with a Master (running admin server) and two replica
plan -execute -name "Deploy n01" deploy-sn 1 host1 5000
plan -execute -name "Deploy admin" deploy-admin 1 5001  #admin will be deployed to SN1
plan -execute -name "Deploy n02" deploy-sn 1 host2 5000
plan -execute -name "Deploy n03" deploy-sn 1 host3 5000

#Second Shard with a Master and two replica
plan -execute -name "Deploy n04" deploy-sn 1 host2 6000
plan -execute -name "Deploy admin2" deploy-admin 4 6001 #admin replica on SN4
plan -execute -name "Deploy n05" deploy-sn 1 host3 6000
plan -execute -name "Deploy n06" deploy-sn 1 host1 6000

#Third Shard with a Master and two replica
plan -execute -name "Deploy n07" deploy-sn 1 host3 7000
plan -execute -name "Deploy admin3" deploy-admin 7 7001 #admin replica on SN7
plan -execute -name "Deploy n08" deploy-sn 1 host1 7000
plan -execute -name "Deploy n09" deploy-sn 1 host2 7000

joinpool BostonPool 1
joinpool BostonPool 2
joinpool BostonPool 3
joinpool BostonPool 4
joinpool BostonPool 5
joinpool BostonPool 6
joinpool BostonPool 7
joinpool BostonPool 8
joinpool BostonPool 9


plan -execute -name "Deploy the store" deploy-store BostonPool 3 900
quit
### End Script ###
```

Run the following command from one of the host systems in order to execute the script:

```
$java -jar KVHOME/lib/kvstore.jar runadmin -port 5000 -host host1 -script 3x3script.txt
```

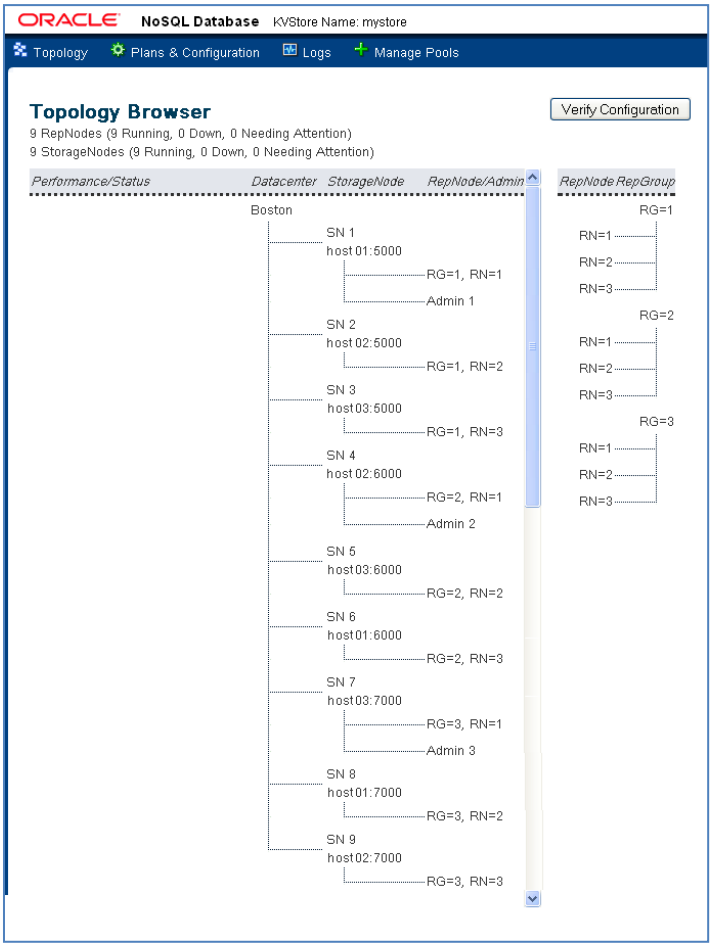

Figure 2: Topology view from administrative console showing three master nodes and three admin processes running on three different hosts and replicas of each shard on a host other than the master node.

This topology ensures high availability, with no single point of failure, given that each shard and the admin service are deployed across all three physical hosts. Of course, this topology is not recommended for a production system, since it involves having three storage nodes share the

same physical system, which will lead to suboptimal response times and less predictable performance.

Storage Node Migration

As shown above, it is possible to deploy multiple storage nodes on the same physical host. However, in a production environment it is recommended that you assign dedicated computing resources to each storage node in order to optimize throughput and system predictability.
Let's now assume that your new hardware has arrived and that you are ready to migrate the storage nodes to the new physical machines. In order to minimize impact on a running system, it is recommended that you migrate the replicas (which you can do gradually) and leave the master in place.

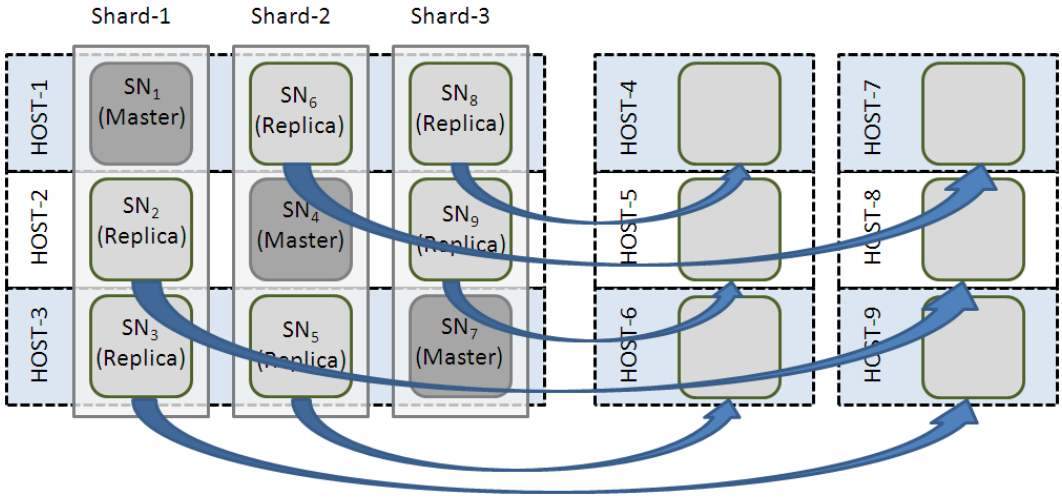

Figure 3: Shows the migration plan from multiple SNs per host to a single SN per host.

Let's walk through the storage-node migration steps, which are basically the same steps as described in the Admin Guide for "Replacing a Failed Storage Node [3]". We will start with SN2, which is the first storage node that we will migrate.

1. Prior to migrating the storage nodes, make sure that:
    - The target hosts where SNs will be migrated to are in the same subnet and are ready for the production environment
    - The old storage node is shut down before migration begins on the SN

```
$ java -jar KVHOME/lib/kvstore.jar stop -root KVROOT
```

[3] http://docs.oracle.com/cd/NOSQL/html/AdminGuide/replacefailedsn.html

2. On the new physical node, create a "boot config" configuration file using the makebootconfig utility. Do this on the hardware where your new storage node will run. You only need to specify the -admin option (the Admin Console's port) if the hardware will host the Oracle NoSQL Database administration processes.

   To create the "boot config" file, issue the following commands:

```
$mkdir -p KVROOT    (if it doesn't already exist)
$ java -jar KVHOME/lib/kvstore.jar makebootconfig -root KVROOT  -port 5000  -admin 5001 -host
<hostname>
 -harange 5010,5020
```

3. Start the Storage Node Agent software on the new node:

```
$nohup java -jar KVHOME/lib/kvstore.jar start -root KVROOT &
```

4. Deploy the new storage node to the new node. You can use an existing administrative process to do this, using either the CLI or the Admin Console. To do this using the CLI:

```
$java -jar KVHOME/lib/kvstore.jar runadmin -port <port> -host <host>
kv-> plan -execute deploy-sn "replace sn" <dc_id> <hostname> <reg port>
kv->
```

5. Add the new storage node to the Storage Node pool. (You created a Storage Node pool when you installed the store, and you added all your storage nodes to it, but it is otherwise not used in this version of the product.)

```
kv-> show pools
AllStorageNodes: sn1 sn2 sn3 sn4 sn5 sn6 sn7 sn8 sn9 sn10
BostonPool: sn1 sn2 sn3 sn4 sn5 sn6 sn7 sn8 sn9

kv-> joinpool BostonPool 10
AllStorageNodes: sn1 sn2 sn3 sn4 sn5 sn6 sn7 sn8 sn9 sn10
BostonPool: sn1 sn2 sn3 sn4 sn5 sn6 sn7 sn8 sn9 sn10
```

6. Create and execute a plan to migrate from the old storage node to the new one. The syntax for this plan is:

```
migrate-storagenode  <old SN ID>   <new SN ID>   <new reg port>
```

   Assuming that you are migrating from Storage Node 2 to 10 on port 6000, you would use:

```
kv-> plan -execute migrate-storagenode 2 10 6000
```

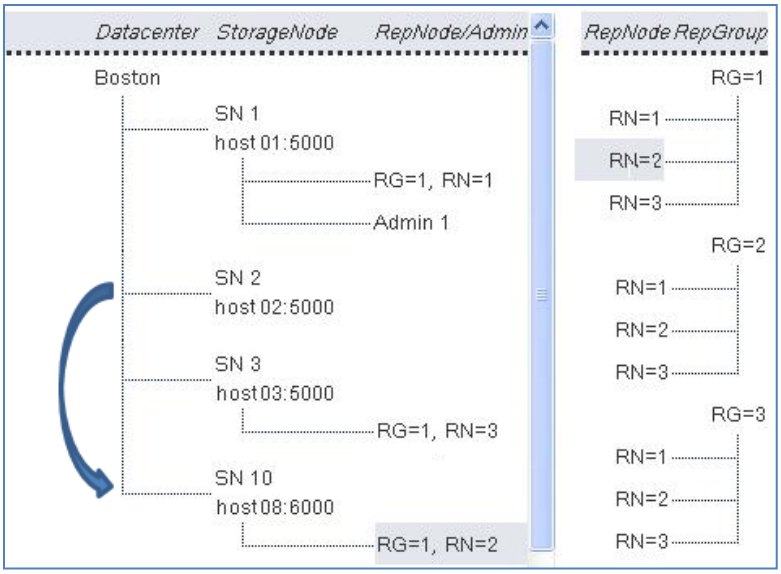

Figure 4: Topology diagram showing replication-node (RN=2) migrated from SN_2 to SN_{10}

Repeat steps 1 through 6 for each of the remaining storage nodes that need to be migrated (SN_3, SN_5, SN_6, SN_8, and SN_9), leaving the master nodes (SN_1, SN_4, SN_7) in place. At the end of this process we should have both a highly available and highly performant cluster up and running.

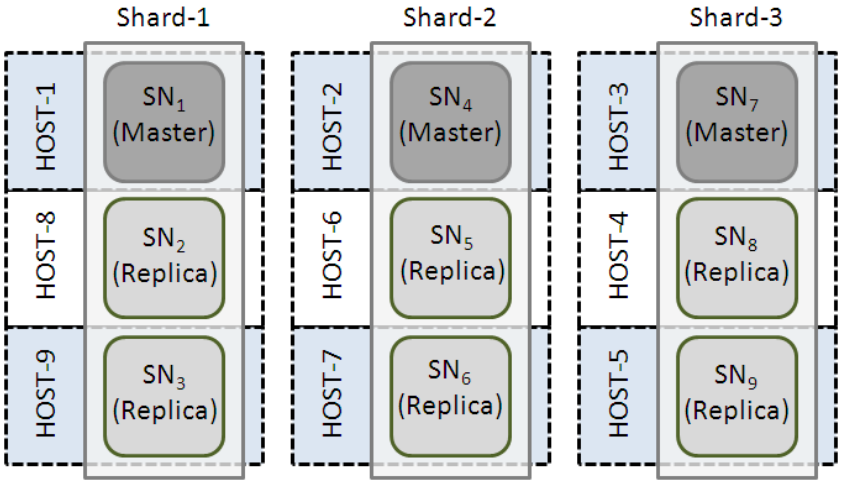

Figure 5: Block diagram showing how the 3x3 KV-cluster would look after performing migration on all the shared SNs.

Conclusion

In Release 1 of the Oracle NoSQL Database, the number of storage nodes and the number of shards are static. Several customers have asked how they can deploy a full KV-cluster topology on a limited set of initial hardware while planning to utilize the full hardware deployment later. This paper demonstrates that it is possible to accomplish this by planning out the topology based on the throughput, latency, and availability requirements using the tools available in the Admin Guide, deploying it on the limited hardware first, and then migrating it to the new physical systems once they become available.

Starting Small and Scalling Out
April 2012
Author: Anuj Sahni, Dave Segleau
Contributing Authors: Oracle NoSQL Database Team

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

0109