GWD-R, GWD-I or GWD-C
GFS-WG or RG or CG name

Leitao Guo, China Mobile Research Institute
Xu Wang, China Mobile Research Institute
Meng Xu, China Mobile Research Institute
Wenhui Zhou, China Mobile Research Institute
August 15, 2008

Multi-Server Based Namespace Data Management of Resource Namespace Service

<u>Status of This Document</u>

This document provides information to the Grid Community about namespace data management of Resource Namespace Service (RNS). This document can be seen as the complementary to GFD 101, and the distribution is unlimited.

<u>Copyright Notice</u>

Abstract

This document describes a multi-server based namespace data management system for Resource Namespace Service (RNS), which is introduced in GFD 101 [MANUEL]. In this system, RNS can be accessed through a master node, which provides the standard interfaces and functions defined in GFD 101. When the master node receives the RNS request, it will dispatch the request to one of the slave nodes based on load balancing policy. Every slave node stores a copy of the namespace data and responses to the read request, such as list and query functions, while only one slave node responses to the write request, such as add, update and remove function. There is a data synchronization process to keep the data consistence on all the slave nodes. In this method, even though one or more slave servers failed, the RNS can also provide stable service. This multi-server based namespace data management system can improve the availability and scalability of RNS.

<u>Contents</u>

guoleitao@chinamobile.com

1. Introduction

The Resource Namespace Service (RNS) [MANUEL] encompasses a multi-faceted approach for addressing the needs of access to resources within a distributed network or grid by way of a universal name. RNS is a critical service in grid and distributed network, almost every operation on the data will access this service. At the same time, the namespace data are stored in the dynamic nodes, especially in grid environment, where nodes can come and go easily, so the quality of service of RNS can not be guaranteed. Meanwhile, if RNS is deployed only on one server, RNS would be the bottleneck in the network.

For the availability and scalability of RNS, this multi-server based namespace data management deployment is proposed. In this deployment, the namespace data will be stored on more than one server and all the nodes can provide the read function on the data, but only one node can provide the write function. However there is a synchronization process which synchronizes the data changed on the write node to all the other nodes to keep the data consistence.

2. Notational Conventions

The key words 'MUST," "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" are to be interpreted as described in RFC 2119 [BRADNER1]

3. Architecture of Multi-server Based Namespace Data Management

3.1   Basic Components

The architecture of the multi-server based namespace data management is shown in figure 1. There are two kinds of nodes, namely, master node and slave node. Generally, there is only one master node, but there can be more than one slave node. The slave nodes can be named as slave_0, slave_1, slave_2 and so one.
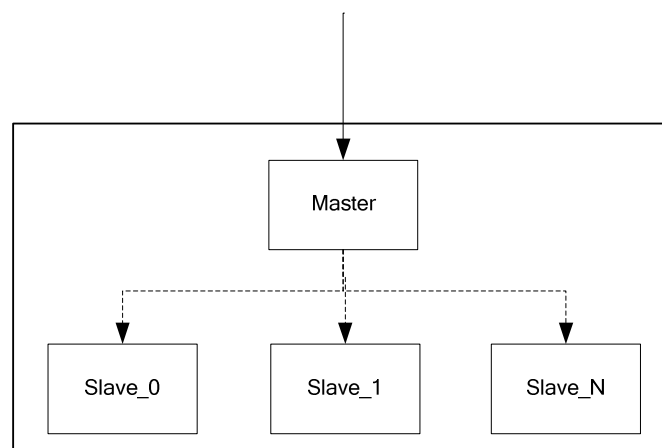

Figure 1

Master node doesn't store the namespace data, but provides the standard port types to the RNS caller. When the master node receives the requests, it just dispatches them to the slave nodes based on some load balancing policies.

Slave nodes   There are also 2 kinds of slave nodes, namely, slave_0 node and all the other slave nodes. Slave_0 node should response to the write request, such as add, update and remove functions, but all the slave nodes (including slave_0 node) should response to the read request, such as list and query functions. When slave nodes receive the request forwarded by

master node, the slaves will response to the requester directly by encapsulate the response message with the IP address of master.

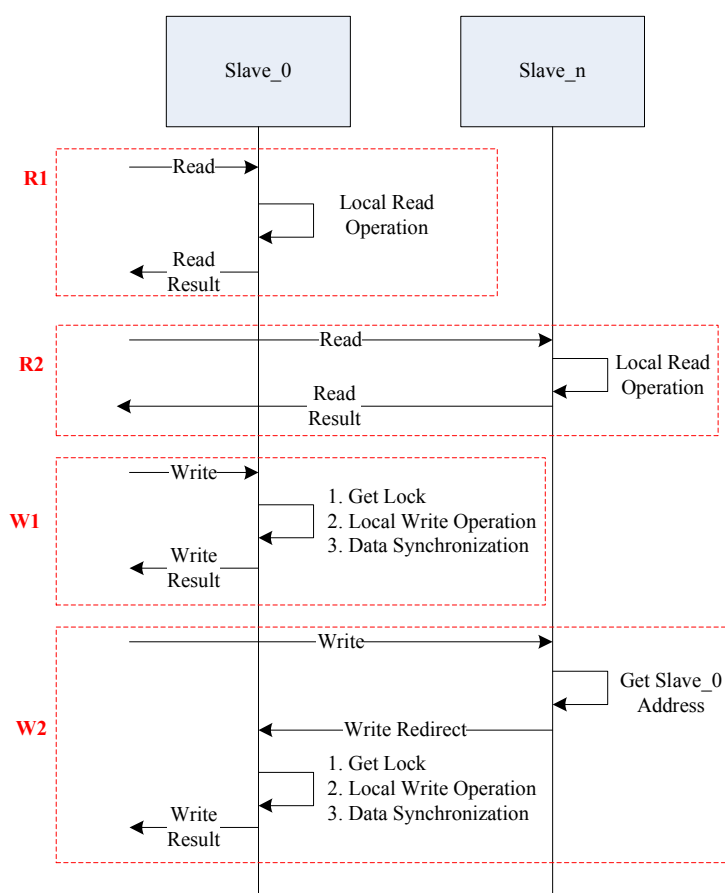### 3.2   Process Flow of Port Type Provided by RNS

Figure 2

The port type that RNS provided can be classifies into two groups, namely, read function and write function. According to the port types and slave node to which the requests are dispatched by master node, the process flow of port type provided by RNS can be divided into 4 cases, namely, R1, R2, W1 and W2, as shown in Figure 2.

➢   **R1**
In this situation, the read request is dispatched to slave_0 node. When slave_0 node receives the request, it just reads the local data and then response to the caller directly.

➢   **R2**
In this situation, the read request is dispatched to other slave node (not slave_0 node). Since the data on every slave node are consistent, the process flow is the same with R1.

➢   **W1**
In this situation, the write request is dispatched to slave_0 node which is the only slave nodes that can response to the write request.

Firstly, slave_0 get lock on the request data, so that all the slave nodes can not read that data. Then, it writes the local data and synchronizes the changed data to all the other slave nodes. After finishing synchronization, slave_0 node can release the lock and respond to the caller.

> **W2**

In this situation, the write request is dispatched to other slave nodes (not slave_0 node). Since slave_0 node is the only slave nodes that can response to the write request, the slave node just gets the address of slave_0 node from master or cached data, and then redirects the request to slave_0 node. The following process flow is the same with W1.

3.3   Data Synchronization

When slave_0 node receives a write request, it should get a lock on the requested dataset. Before the lock is released by slave_0 node after data synchronization, every slave node can't response to the read or write request on that locked dataset.

There are many data synchronization methods can be adopted, for example, slave_0 node can send the changed data to all the other slave nodes one by one, or it can only send the data to slave_1 node, then the slave_1 sends the data to slave_2 node and so on. Through this pipeline, the data can be passed to all slave nodes.

If some slave nodes return failure to the data synchronization, there will cause data inconsistence problem. In this situation, the slave_0 node should retry the data synchronization request before release the lock. If the slave node still returns failure after retry several times, the slave_0 node should report this information to master node. Once the master node receives this information, it should regard this slave node as dead, and exclude it when dispatching the request.

3.4   Join of the new node

If a new node joins to this multi-server based deployment, firstly the master assigns a name to this node, and then asks one slave node to synchronize the namespace data to this newly joined node. Then the new node can be served as a slave node.

3.5   Slave_0 Node Failure

If slave nodes other than slave_0 node failed, the system's availability will not be affected, because all the other slave nodes can also provide the read request on the namespace data. However, if the slave_0 node failed, the master should choose another slave node to take over the function of slave_0 node. There are a lot of research achievements on the election in a distributed computing system [GARCIA], so we can choose one of them. Once a slave node is granted the privilege of slave_0 node, the master will notify the address of slave_0 node to all the slave nodes.

4. Author Information

Leitao Guo, Xu Wang, Meng Xu
China Mobile Research Institute
53A, Xibianmennei Ave., Xuanwu District
Beijing 100053, P.R. China
guoleitao@chinamobile.com
wangxu@chinamobile.com
xumeng@chinamobile.com

5. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

6. Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

7. Full Copyright Notice

8. References

[MANUEL] Manuel Pereira, Resource Namespace Service Specification, GFD.101, March 2007.
[BRADNER1] Bradner, S. Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997.
[GARCIA] Garcia Molina, H. , Elections in a distributed computing system, IEEE Transactions on Computers, 51, 1, Jan 1982, 48-59.