An Oracle White Paper
March 2010

Oracle Berkeley DB Java Edition High Availability - Write Forwarding Using Java RMI

ORACLE

Introduction 2
Why is Write Forwarding Important? 2
Managing Write Requests in Applications 2
Background 3
  Replica-Based forwarding 3
  Remote Method Invocation (RMI) 4
Example Overview 4
Getting Started 4
Understanding How Write Forwarding Works 5
Disclaimer 7
Conclusion 8

Introduction

Oracle Berkeley DB Java Edition High Availability (JE HA) is an embedded data management system, designed to provide applications with JE's capabilities along with a "master read-write, replica read-only" replication paradigm. This means that all database writes/updates are performed on a single master node, and replica nodes are available only for read activity.

Why is Write Forwarding Important?

JE HA provides APIs to create a replicated environment and process read/write transactions. The application running on the node that is currently the master can process both read and write requests, while the applications running on the replicas can process only read requests.

When a replica node receives write requests the application throws an exception or returns an error. To avoid this scenario, applications running on replica nodes must be able to direct these requests to the master node. This is called write forwarding.

Write forwarding must be managed to ensure the replica nodes track the master node and then forward the write requests to the current master node.

Managing Write Requests in Applications

It is the responsibility of the application to provide the functionality to transport application requests to the master. There are two main ways by which an application can direct write requests to the master:

- Replica-based forwarding: Each application on a replica keeps track of the identity of the master and forwards any write requests it receives to the current master.
- Monitor-based routing: Each application runs a router application on a monitor node, to route write requests to the current master and load balance read requests across all active members.

This write-up explains with an example how each replica node can keep track of the master in the group and send all write requests to it using Java RMI (Remote Method Invocation). Write forwarding using Java RMI is based on the Replica-based forwarding method.

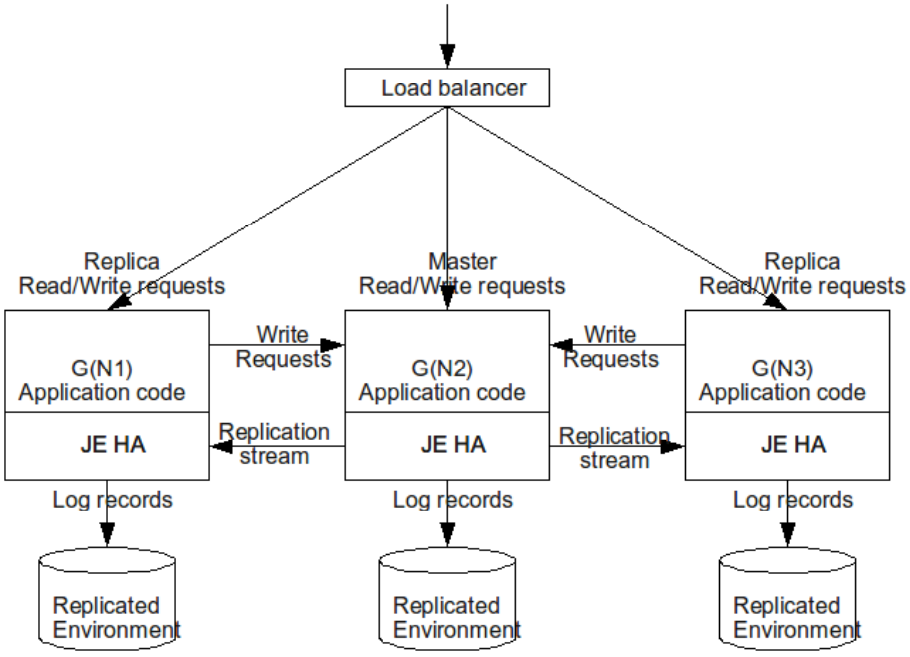
Background

Replica-Based forwarding

A JE HA application accesses its replicated environment through a replicated environment handle. The handle API enables the application to:

- Determine its current state. For example, the node can be queried as to whether it is currently the master or the replica.
- Associate a state change listener with the handle. The state change listener then receives events that can keep it informed of the current member that is serving as the master.

With these APIs, the application can determine whether it is currently the master. If it is not the master, it can arrange to forward the write request to the node that is currently the master, as illustrated in the figure.



Replica based Write forwarding

Remote Method Invocation (RMI)

The Java Remote Method Invocation (RMI) system enables an object running in one Java virtual machine to invoke methods on an object running in another Java virtual machine. RMI applications often comprise two separate programs, a server and a client. A typical server program creates some remote objects, makes references to these objects accessible, and a client program obtains a remote reference to one or more remote objects on a server and then invokes methods on them.

The applications must do the following:

- Locate remote objects. For example, an application can register its remote objects with RMI's simple naming facility, the RMI registry.
- Communicate with remote objects. RMI handles details of communication between remote objects.

Here, the master node functions as the server and the replica nodes function as clients. And, the invoked write method is executed in the master node as an RMI thread.

Example Overview

The StockQuotesRMIForwarding example is a mock stock ticker application that stores stock values in a replicated JE HA environment. The following commands are accepted:

- <stock> <number> : enter this stock price into the database
- print : print all the stocks and current prices held in the database
- quit : shut down

StockQuotesRMIForwarding is an extention of the StockQuotes example. The StockQuotes example executes print requests made at master and replica nodes and rejects update requests made at a replica's console. Whereas, the StockQuotesRMIForwarding example illustrates how RMI is used to track the master node, and forward write requests to the master.

This example is explained with three nodes, running as a group locally, on the same machine. The tasks that you execute are explained against the implementation for the functionality to work.

Getting Started

- Step 1. Starting Stock Quote Servers: Start the three stock quote servers supplying the right arguments. Refer to the Javadoc associated with the StockQuotes and the StockQuotesRMIForwarding example for more information on starting the servers.
- Step 2. Displaying Stock Information: Enter a command at the console established by the application at each node. The command can be Print, Update, or Quit. Refer to the doCommand function defined in the StockQuotes example.

  Master and replica nodes accept Print commands and the stock name and stock prices are displayed. Refer to the printStocks function defined in the StockQuotes example.

Understanding How Write Forwarding Works

When you update the stock price at the command prompt, the node forwards the write request, if it is not currently the master, or if the node changes status when a transaction is in progress. The update method is executed as an RMI thread in the master node.

- Step 1. Binding Remote Reference: Each node in this example is an RMI server and hosts an RMI registry. Hence, there are multiple RMI registries, one per each node, instead of a single common one. This implementation helps your JE HA application avoid a single point of failure. The object associated with the RMI binding makes available the entire high-level database write operations that are part of the application. When this node is the master, replicas use the remote methods to invoke write operations on it. The replicas play the role of RMI clients making remote method calls to the master to forward their write requests. An example code for setting up the RMI registry and binding the remote reference is as follows:

```
private StockQuotesRMIForwarding(String[] params)
throws Exception
{
super(params);
nodeRegistry = LocateRegistry.
               createRegistry(repConfig.getNodePort() +
               RMI_PORT_DISPLACEMENT);
writeServices = new WriteServicesImpl(System.out);
UnicastRemoteObject.exportObject(writeServices, 0);
nodeRegistry.rebind(RMI_NAME, writeServices);
}
```

- Step 2. Tracking Master Node: Each node uses the StateChangeListener to track the node that is currently the master. This Listener reacts to StateChangeEvent notifications and maintains the name of the current master. The following is an example function that is used to service StateChangeEvent notifications.

```
public void stateChange(StateChangeEvent stateChangeEvent)
 throws RuntimeException
      {
  switch (stateChangeEvent.getState())
  {
  case MASTER:
  case REPLICA:
  String masterName = stateChangeEvent.getMasterNodeName();
          masterInfo = new MasterInfo(masterName);
  System.err.println("Master: " + masterName +
                              " at " + new Date());
  break;
  default:
  masterInfo = new MasterInfo();
  System.err.println("Unknown master. " +
            " Node state: " +
            stateChangeEvent.getState());
  break; }  }
```

- Step 3. Forwarding Updates: The stock updates entered at an application that is currently running as a replica node, are forwarded as write requests to the master using RMI. The forwardStockUpdate function illustrates how an update request that is made to the replica, is forwarded to the master, if reachable. The stockUpdateLine is the command to forward and printStream is the stream that captures the output from the forwarded request.

```
private void forwardStockUpdate(Quote quote) {
    try {
        if (masterInfo.name == null) {
            System.out.println("Could not update:" +
            quote.stockSymbol +
            " Master is unknown. ");
        return; }
        masterInfo.reference.update(quote);
        System.out.println(repEnv.getNodeName() + " forwarded " +
                           quote.stockSymbol + " update to " +
                           masterInfo.name);
        } catch (RemoteException e) {
        if (e.getCause() instanceof ReplicaWriteException) {
            forwardStockUpdate(quote);
            return; }
        System.out.println("Could not connect to master: " +
                           masterInfo.name + " Exception: " + e);
        }  }
```

The masterInfo name is obtained from the state change function. The following internal class treats the master name and remote reference as a fixed pair:

```
private class MasterInfo {
    /* The node name of the master*/
    final String name;
    /* The remote reference to the Master with the above name */
    final WriteServices reference;
    public MasterInfo(String masterName) {
        this.name = masterName;
        ReplicationNode masterNode =
        repEnv.getGroup().getMember(name);
        Registry currentMasterRegistry;
        try {
            currentMasterRegistry = LocateRegistry.
                getRegistry(masterNode.getHostName(),
                                    masterNode.getPort() +
    RMI_PORT_DISPLACEMENT);
        reference = (WriteServices)currentMasterRegistry.
                lookup(RMI_NAME);
        } catch (RemoteException e) {
            throw new RuntimeException(e);
        } catch (NotBoundException e) {
            throw new RuntimeException(e);
        }}
    public MasterInfo() {
        name = null;
        reference = null; } }
```

- Step 4. Executing Stock updates: The following function, updateStock, is invoked from the replica on the master node.

```
void updateStock(final String line, final PrintStream
printStream)
throws InterruptedException {
final Quote quote = QuoteUtil.parseQuote(line);
if (repEnv.getState().isReplica()) {
    forwardStockUpdate(quote);
    return;
}
new RunTransaction(repEnv, printStream) {
    @Override
    public void doTransactionWork(Transaction txn) {
        dao.quoteById.put(txn, quote);
        System.out.println(repEnv.getNodeName() +
            " processed update request: " + line);
    }
    @Override
public void onReplicaWrite(@SuppressWarnings("unused")
       ReplicaWriteException replicaWrite) {
        forwardStockUpdate(quote);
    }
 }.run(false);
 }
```

Disclaimer

This example is intended to be illustrative. It favors concise code over completeness to avoid distracting from the primary intent. For example, it does not handle all cases of exceptions.

Conclusion

Oracle Berkeley DB Java Edition high availability (JE HA) offers "master read-write, replica read-only" replication paradigm. To forward the write requests received by the replica nodes, applications can use a Replica-based forwarding method using the Remote Method Invocation approach. The master node functions as the RMI server and the replica nodes function as RMI clients. And, the invoked write method is executed in the master node as an RMI thread. This approach helps avoid single point of failure and the use of load balancers.

You can download Oracle Berkeley DB Java Edition at:

http://www.oracle.com/technology/software/products/berkeley-db/je/index.html

You can post your comments and questions at the Oracle Technology Network (OTN) forum for Oracle Berkeley DB Java Edition at:

http://forums.oracle.com/forums/forum.jspa?forumID=273

For sales or support information email: berkeleydb-info_us@oracle.com

Find out about new product releases by sending an email to: bdb-join@oss.oracle.com

Oracle Berkeley DB JE HA -Write Forwarding
Using Java RMI
March 2010

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