High Availability and Performance Scalability of Real Time Database System

V. S. Phalake¹, K.D. Yesugade²

¹Bharati Vidyapeeth’s JNIOT.Pune-43, Maharashtra, India
²Bharati Vidyapeeth’s JNIOT.Pune-43, Maharashtra, India

Abstract - Today’s real-time systems (RTS) are characterized by managing large volumes of dispersed data making real-time distributed data processing a reality. Real time distributed systems were once the exception, constructed only rarely and with great difficulty by developers who spent significant amounts of time mastering the technology. Now, as modern software technologies have made distributed systems easier to construct. We will design software, which provides high availability and performance scalability for large distributed databases. It can provide transparent database clustering, partitioning, real time replications and real-time Database operations to any Java application through Java based Data Base Connector. It may works with any Java application without code modification and with any database engine after all we are planning to use java with OOPS and oops as reusability concepts. This will allow one to build systems that combine the functionality and attractive user environments of modern enterprise systems with delivery of high performance in those application components that need it.

Index Terms—Real time system, high availability, performance scalability, real time database, transparent database clustering.

I. INTRODUCTION

High availability and scalability are just as important for small and mid-size companies as they are for larger corporations. Traditionally, small businesses have had limited ability to deploy the same levels of high availability as their larger counterparts. They usually rely on single server systems and most of them have suffered through periods of down time, which have negatively impacted revenue and caused reductions in customer service and satisfaction. Scalability for small and mid-size companies has also been an issue.

Oracle has made great strides in reducing the complexity and cost of clustering by introducing Oracle Database Standard Edition (SE), which includes RAC at no additional cost.

Database replication and clustering are increasingly widespread solutions to scalability and high availability challenges in current information systems. Replicated database management systems are at the core of modern information systems, given the importance of achieving high availability, fault tolerance and prompt disaster recovery. Consistent database replication technology is key to shared-nothing database clusters, that are an increasingly popular solution to the scalability of multi-tier applications. Early approaches to database replication suffered from performance limitations that constrained the full use of deployed replicas. Database clusters provide superior scalability and high availability (HA) compared to single database instances.

A. Objectives

Design software, provide transparent database clustering; partitioning, real time replications and real-time Database operations to any Java application through Java based Data Base Connector. It may works with any Java application without code modification.

The goal of this dissertation is to provide cluster system, which simplifies implementation part. It will provide-

Availability:
MySQL Cluster provides a parallel server, fault-tolerant architecture that ensures your organizations. MySQL Cluster implements automatic node recovery to ensure an application automatically fails over to another storage node that contains a consistent data set, if one or more storage nodes fail.

Fast Automatic Failover:
MySQL delivers extremely fast failover time with sub-second response so your applications can recover quickly in the event of application, network or hardware failure. Plus, MySQL Cluster storage nodes are able to automatically restart, recover, and dynamically reconfigure themselves in case of failures without having to program advanced features into the application.

Linear Scalability:
To incrementally scale your system without a high initial hardware investment.

Easy to Administer:
Reducing your need to hire additional database administrators.

B. Goals

The goal of this paper is to provide cluster system, which simplifies implementation part. It will provides-

1. To control both the local database and remote database simultaneously and synchronization between heterogeneous databases handled by using NTP protocol.
2. To design such system so as to overcome the problems regarding scalability failure recovery, failover and to create geo-clustering.

© 2010 ACEEE
DOI: 01.IJRTET 04.02.154
3. To handle services to the web pages through JVM, RMI or through C, C++, Java, Perl etc.

II. LITERATURE REVIEW

Louis rillin explains database query caching technique, GlobeCBC, can be used to improve the scalability of Web applications. This paper addresses the availability issues in GlobeCBC. Even though high availability is achieved by adding more resources, proper algorithms must be designed to ensure that the clients receive consistent responses amidst failures of the edge and origin servers [3].

Clusters and distributed systems offer fault tolerance and high performance through load sharing, and are thus attractive in real-time applications. When all computers are up and running, we would like the load to be evenly distributed among the computers. When one or more computers-fail the must be redistributed.

The emerging edge services architecture promises to improve the availability and performance of Web services by replicating servers at geographically distributed sites. Our [1] experimental results show that by slightly relaxing consistency within individual distributed objects, our application realizes both high availability and excellent performance.

Lundberg & Klonowska, K.; Svahnberg, C. discusses, Clusters and distributed systems offer fault tolerance and high performance through load sharing, and are thus attractive in real-time applications. When all computers are up and running, we would like the load to be evenly distributed among the computers. When one or more computers-fail the must be redistributed [4]

III. ANALYSIS OF PROPOSED SYSTEM

A. Problem Study

Normally, databases and applications have to be restarted impacting availability and performance. It requires managing multiple systems whenever needed as well as Complex configuration issues can occur. In the event of a failover users must reconnect or be reconnected to the second node this can result in underutilized server resources. And systems can not handles different databases (SQL, ORACLE, SAP DB, DB) at same time. These problems we are going to solve through the clustering. High Availability & Performance Scalability for database systems is a transparent middleware solution offering clustering, load balancing and failover services for any database. The database is distributed and replicated over multiple nodes and Real Time Database balances the queries between them. Real Time Database handles node and network failures transparently. It also provides support for hot recovery, online maintenance operations and online upgrades.

B. Problem Definition

Software to develop middleware framework for real time database system so as to provide Real time clustering, Multi Node Access, Replication, Real time Recovery capability from the Heterogeneous database server.

C. Overview of solution

The developed System can help to resolve these problems by providing:

- Performance scalability-by adding database nodes and balancing the load among these nodes.
- High availability of the database tier- RTDB tolerates database crashes and offers transparent fail over using database replication techniques.

IV. DESIGN MODULE

Software design involves identifying and describing the fundamental software system abstractions and their relationships. The design process translates requirements into a representation of software.

Functional Diagram of Over All Expected System:

- **RTDB Controller principle**
  RTDB allows to build any cluster configuration including mixing heterogeneous databases. The main features provided by RTDB are performance scalability, fault tolerance and high availability.

  The system works with any JDBC-enabled database and access one or several databases. The system provides a generic JDBC driver to be used by clients. When client requests for data from database server the driver forwards SQL request to RTDB controller that balances them on a cluster of the database. The RTDB controller is instance running in JVM. The controller in our system is a java program, which has two different IP addresses and referred as localhost1 and localhost2. When database is going to access and one of the controllers fails then system automatically takes control over another controller. To maintain the duplication of up-to-date information and for real time data mirroring Octopus technology is used

- **The different components are**
  - **RTDB Driver**
    This is the JDBC driver that will be used by the client application talking to the database.(ie:Tomcat, JBOSS,...)
  - **RTDB Controller**
    This is the main component of RTDB. This is the RTDB controller that will handle and process requests coming from the RTDB driver.
**RTDB Console**

To administrate, monitor, view statistics, and recover failed database backend, you can use a regular text console.

**RTDB log**

The logs generated by the logging system can be found here. Also the RTDB report will be generated in this folder.

**RTDB xml**

The DTD and XLS files used by RTDB. The two main DTDs can be found there to validate controller configuration files and virtual database configuration files.

**RTDB bin**

All the scripts to start / stop applications are here.

**C. Back up and recovery Operations**

When configuring the virtual database, you must specify a backuper that will be used to backup and restore the database backends. You can also define several backupers in the same virtual database configuration. Each backuper must have its own Backuper element definition. The backuper configurations are included in the Backup element definition in the virtual database configuration file. A backuper configuration consists of the following:

- `BackupName` that will be used as a command argument of the backup CLC command when backing up a database.
- `className` that specifies the backuper implementation to be used.

**D. Sample Script to initialize the local host**

```plaintext
admin myDB
expert on
initialize localhost1
backup localhost1 init_dump Octopus /tmp/backup
TEST
enable localhost1
restore backend localhost2 init_dump
TEST
enable localhost2
show backend *
quit
```

**E. Sample sql script which are executed on database at startup**

```plaintext
create table demo(x int)
insert into demo values (1)
select * from demo
```

```plaintext
begin
insert into demo values (2)
rollback
select * from demo
begin
insert into demo values (2)
commit
setreadonly true
select * from demo
setreadonly false
drop table demo
quit
load driver org.hsqldb.jdbcDriver
Sql client jdbc:hsqldb:hsq1://localhost:9001
TEST
select * from ADDRESS
quit
quit
```

**F. Procedure to access the cluster**

1. Using driver manager class, load the drivers & try to make the connection with RTDB database server.
2. Driver makes the connection with RTDB and connects to database server on port 25322 or 25323 that contains myDB using username and password.
3. Driver manager uses following call to locate, load and link the named class `Class.forName("Driver");`
4. Use `PreparedStatement` to call SQL statements number of times with slight variations. The driver may then sends statement to the database.
5. The result of query returned into resultset object.

**V. CONCLUSION**

Many real time distributed database systems fail to meet their performance objectives when they are initially constructed. Others perform adequately with a small number of users but do not scale to support increased usage. This performance failures result in damaged customer relations, lost productivity for users, lost revenue, cost overruns due to tuning or redesign, and missed market windows. Here the developed system provides high availability and performance scalability for large distributed databases. It provides transparent database clustering; partitioning, real time replications and real-time Database operations to any Java application through Java based Data Base Connector.
REFERENCES

Research papers