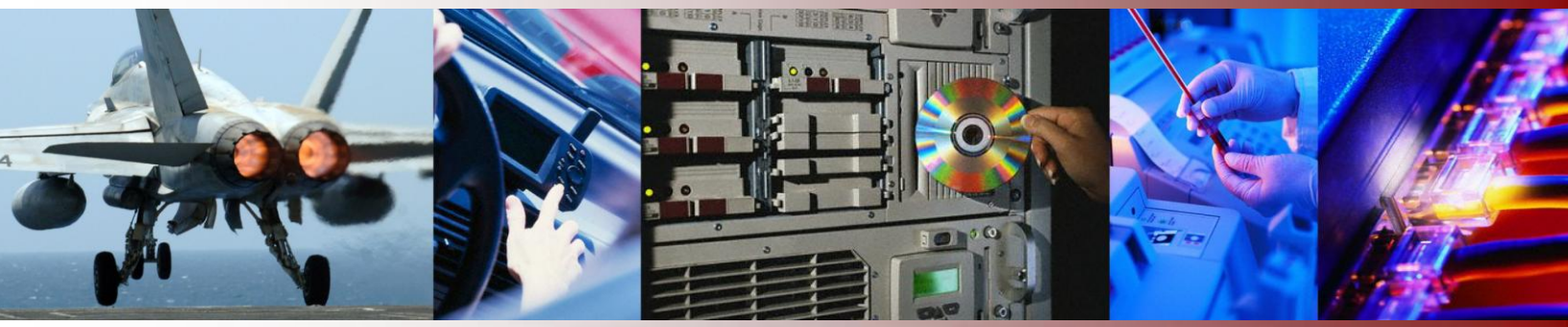
Raima® Database Manager PRODUCT FAMILY

BUSINESS WHITE PAPER



High Availability Using Raima Database Manager Server

A Raima Inc. Business Whitepaper

Published: January, 2008
Author: Paul Johnson
Director of Marketing

Abstract

High Availability (HA) is a broad term that can mean many things to just as many people. For the most part however people agree that it should describe the overall accessibility of data to the user. This paper highlights how Raima Database Managers provide developers with the ability to obtain high availability.

This article is relative to the following versions of RDM:

- ✓ RDM Server: All

Contents

Introduction

High Availability (HA) is a broad term that can mean many things to just as many people. For the most part however people agree that it should describe the overall accessibility of data to the user. With this in mind, HA can encompass the ability of your system to be "always up" in a 24 x 7 manner; in the telecom industry, HA is measured in percentage availability (five nines or better). HA can also include how your system reacts to internal component failure (fault tolerance). And finally, it can describe the level of responsiveness of your system. A system that is "always up", but not very responsive (a query takes hours to complete) cannot truly be described as "Highly Available".

Why is high availability important?

- ✓ In today's "Always Up" business environment, where customers demand instant response, and expect your services at their convenience, the cost of downtime can conservatively run into the thousands of dollars per minute.
- ✓ In addition, access to data can be a company's lifeline; data is the primary ingredient of business intelligence and many of the activities that surround the business. Consequently, the moment access is lost, the business suffers and expensive resources become inefficient or unavailable.

In the following pages we will take a high level look at how Raima Database Server can help embedded application developers achieve high availability.

High Availability with RDM Server

High availability has always been a driving force in the development of the RDM products. The three primary product components that stand out as directly relevant to high availability are:

1. **Zero Database Administration**—in addition to providing high availability is also a key factor in the defining an embedded database.
2. **Replication**—including in-memory, on-disk, RDM Embedded to RDM Embedded, RDM Server to RDM Server, and coming in the next release RDM Embedded to RDM Server.
3. **Hot Online Backup**—provides high availability by providing the ability to back up data while at the same time providing active read/write accessibility to the same data.

Zero Database Administration

One of the high value characteristics of RDM products that makes them unique among database solutions is its ability to be embedded within an application, with database configuration and administration completely controlled by the application, so that end-users of RDM-based applications are presented with virtually no administrative or maintenance requirements. This considerably lowers the end-user's total cost of ownership (TCO), which is often an important consideration when purchasing software applications.

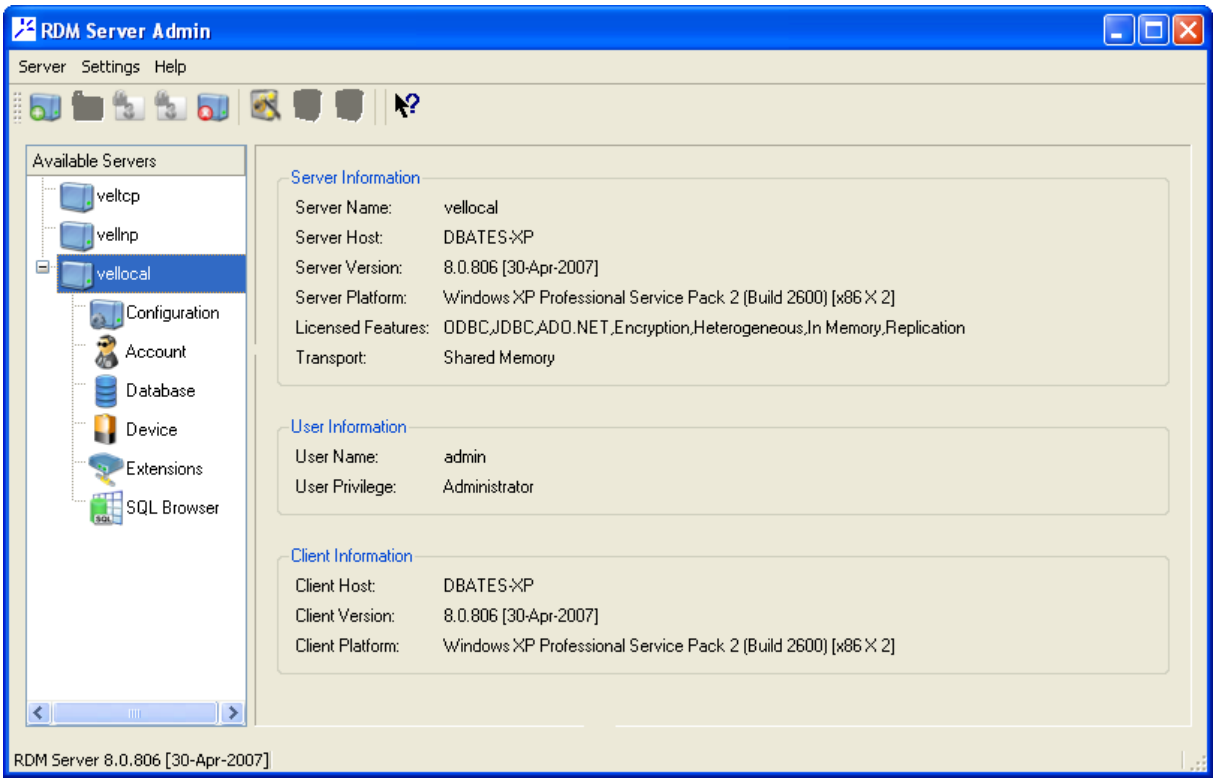
The RDM Server Admin API is a collection powerful and useful database administration utilities and functions. The Admin API is exposed to developers, allowing them to programmatically configure and run these database utilities and/or include them within their database application.

The RDM Server API offers a rich set of administrative functions to support application level administration of the server, allowing the database system to perform the administrative tasks automatically, without having to launch special administration utilities.

Administration Tools

RDM Server also supports ad hoc administrative activities by the end-user via an integrated administration utility for managing one or more server installations. The utility's graphical user interface makes it easy for the system administrator to:

- ✓ Create, modify, delete, and query server objects, including users, databases, Server Extensions, and database devices
- ✓ Start and shut down the server and perform backups
- ✓ Change all server and client configuration parameters and default settings
- ✓ Display server performance statistics
- ✓ Perform file management, including copying files to and from the server, deleting files on the server, and copying or moving files on the server



Client Recovery

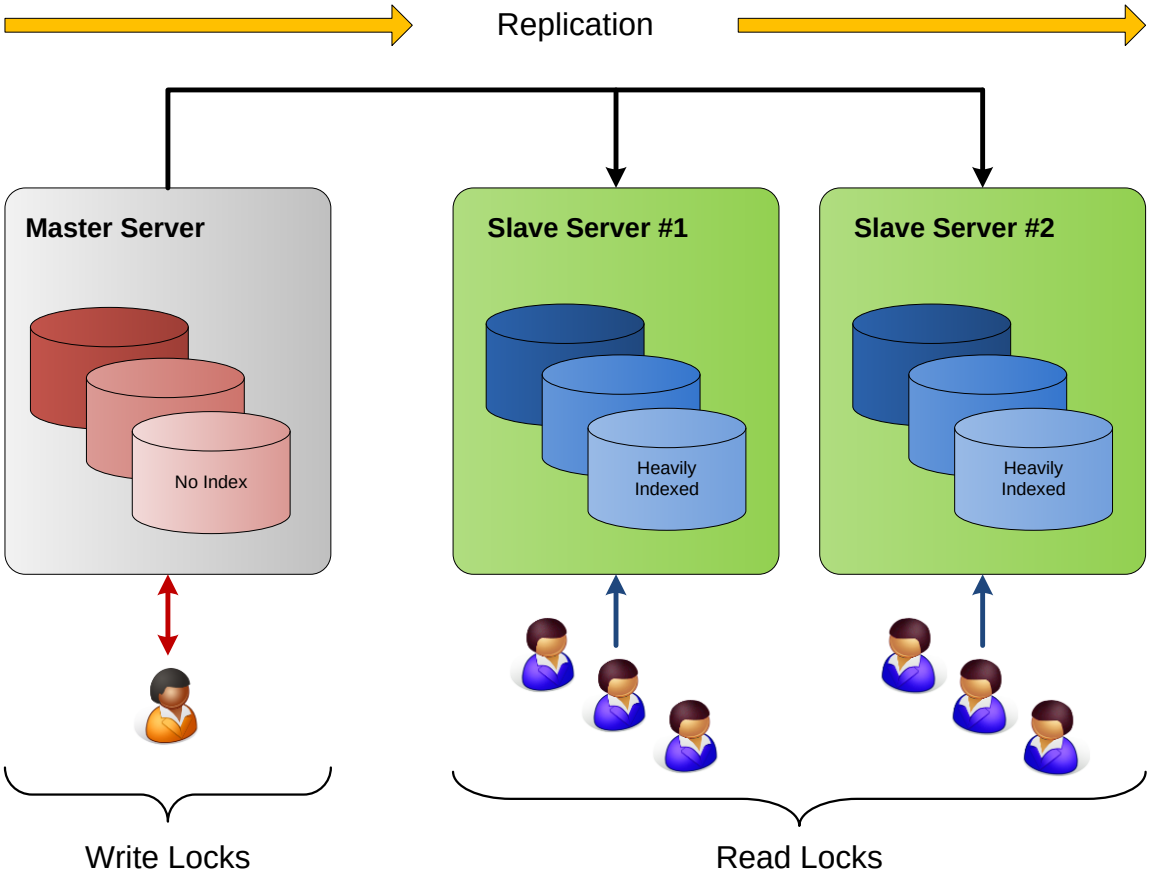
In a networked environment, failure can occur if a user disconnects from the network or turns off a PC during a database transaction. RDM Server monitors client activities and performs client recovery when a client process abnormally terminates.

When a client process terminates unexpectedly the server will detect its absence. The server will then automatically abort any active transactions, release locks, close the database files, and logs the client out.

Replication

Raima Inc. recognizes the importance of high availability, and is proud to have introduced a state of the art replication engine, guaranteeing out of the box 99.999% uptime. Included is support for multi-slave architectures, allowing for complex replication models, increasing server uptime, data redundancy and data availability. RDM Server 8.0 introduced native support for asynchronous and active-passive replication. The solution is simple and yet very effective: one server can act as the Master Server, receiving updates to the Master database, while one or more servers can take on the role of the Slave Servers, providing read-only access to the Slave databases.

Scalability through Replication



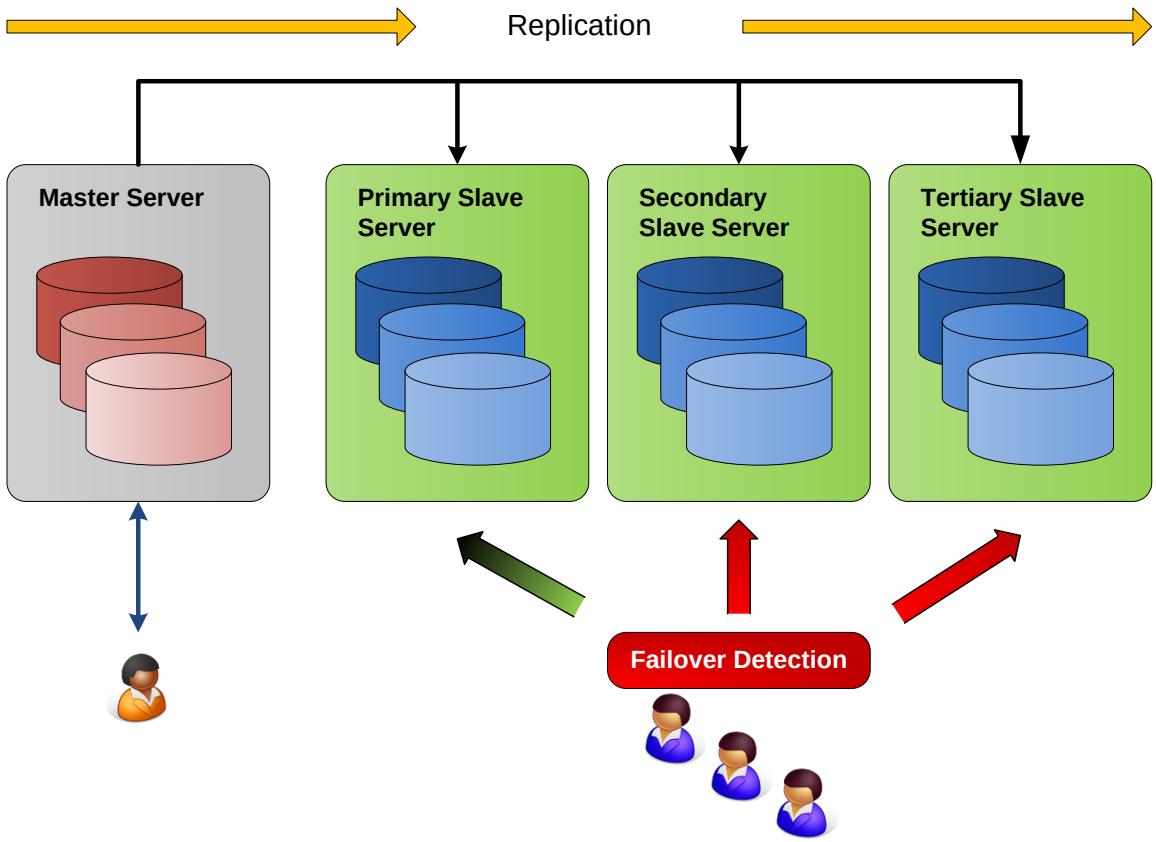
Using RDM Server, developers can utilize replication to scale out your solution to increase the availability and performance of your database. One way to achieve higher 'write' access on the Master server the system can be devoted to only handling 'write' operations leaving the Slave servers to handle the more frequent 'read' access. By distributing 'read' and 'write' locks in such a manner, developers can prevent users who only require 'read' access from being locked out due to a existing 'write' lock. Another technique to improve performance on the Master server is to define the Master schema without any indices; this will result in fewer 'writes' against the disk because there is no need to update index files. In addition, this will also result in a noticeably smaller overall footprint of the Master database.

Further performance improvement can be achieved on the Slave server by heavily indexing the Slave schema. One caveat with this option is when used in conjunction with the above mentioned method of not indexing the Master, developers will need to be aware that if the Slave machine fails, and **ActOnFail** is set to 1, the replication will simply do a file copy of the Master database to Slave database and not go through the process of applying each of the individual transactions to the Slave. As a result of the Master database not having any defined indices, no index files will be transferred from the Master to the Slave. To counter this problem developers have two options. They can either rebuild the Slave indices using the **s_keybuild** API or they can choose to set the **ActOnFail** parameter to 3 within the rdmserver.ini file.

Fault Tolerance through Replication

With RDM Server, it is possible to implement highly effective fault tolerance solution. There are two general possible failure scenarios that RDM replication addresses; the first is where a Slave server fails; and the other is when the Master server fails.

The first scenario, when a Slave server has failed, is the easiest to solve. A failed Slave can easily be detected when a client receives a failed query against a read only database. The application can be easily be architected in such a way that it will simply query an alternate Slave server database. Because the current release of RDM Server does not allow for Slaves to be dynamically added, this configuration needs to be defined beforehand.

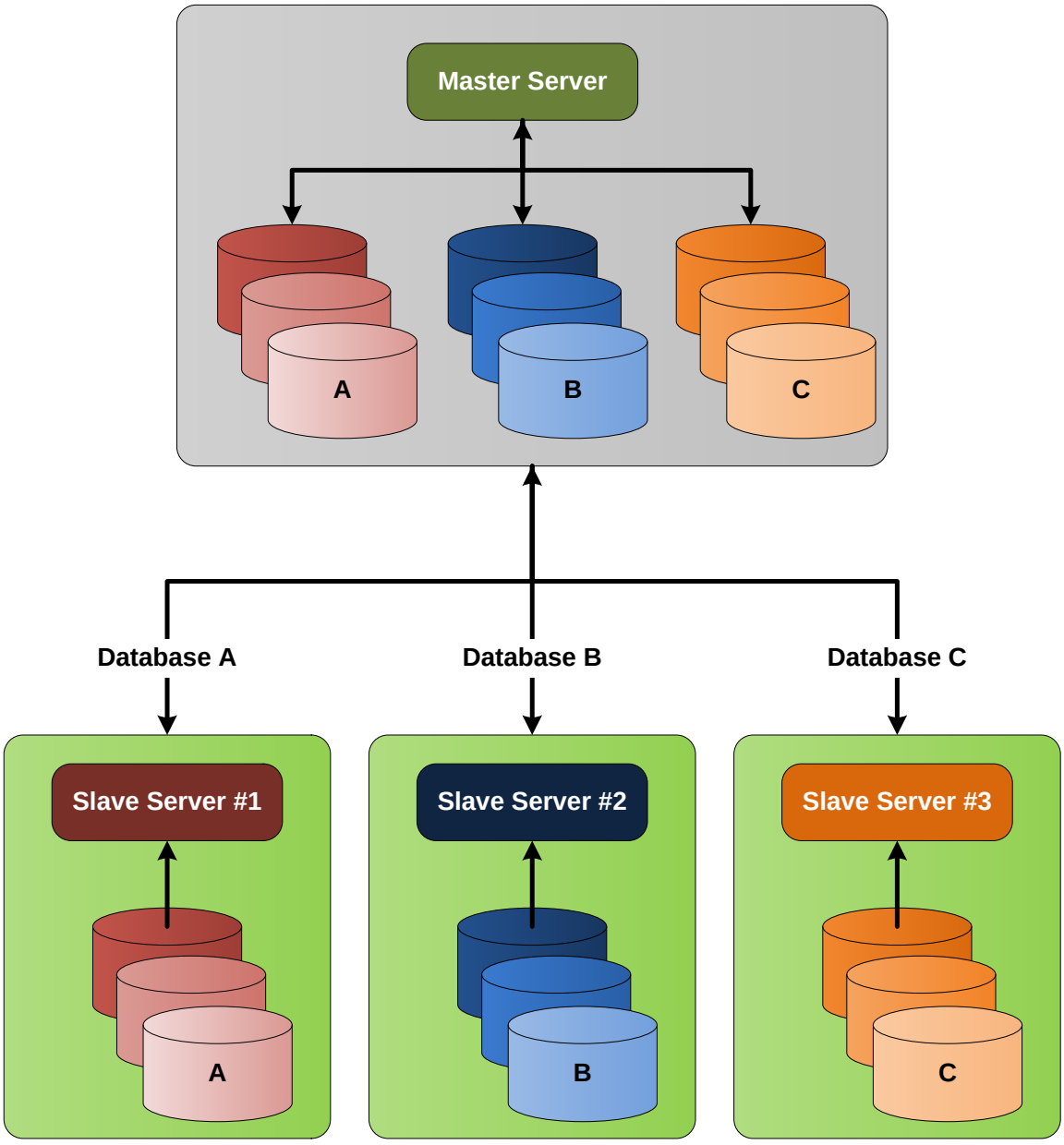


In the event that the Master server fails, the situation becomes slightly more difficult because it means that one of the Slaves servers needs to become the Master server. The first step is to identify which of the Slave servers will assume the role of Master server. Using the **s_iniSet** API, developers can simply reconfigure the chosen Slave server's rdmserver.ini file setting the 'role' parameter from 'Slave' to 'Master'. Depending on the

application and situation, other parameters such as the cache size may also need to be reset as well. The next thing to update using the **s_iniSet** is the new identity of the Master server within each of the Slave Server's rdmserver.ini files. Now that the roles of the Master and Slaves has been reset, all the machines next need to be gracefully brought down using the **s_shutdown** API and subsequently brought back up.

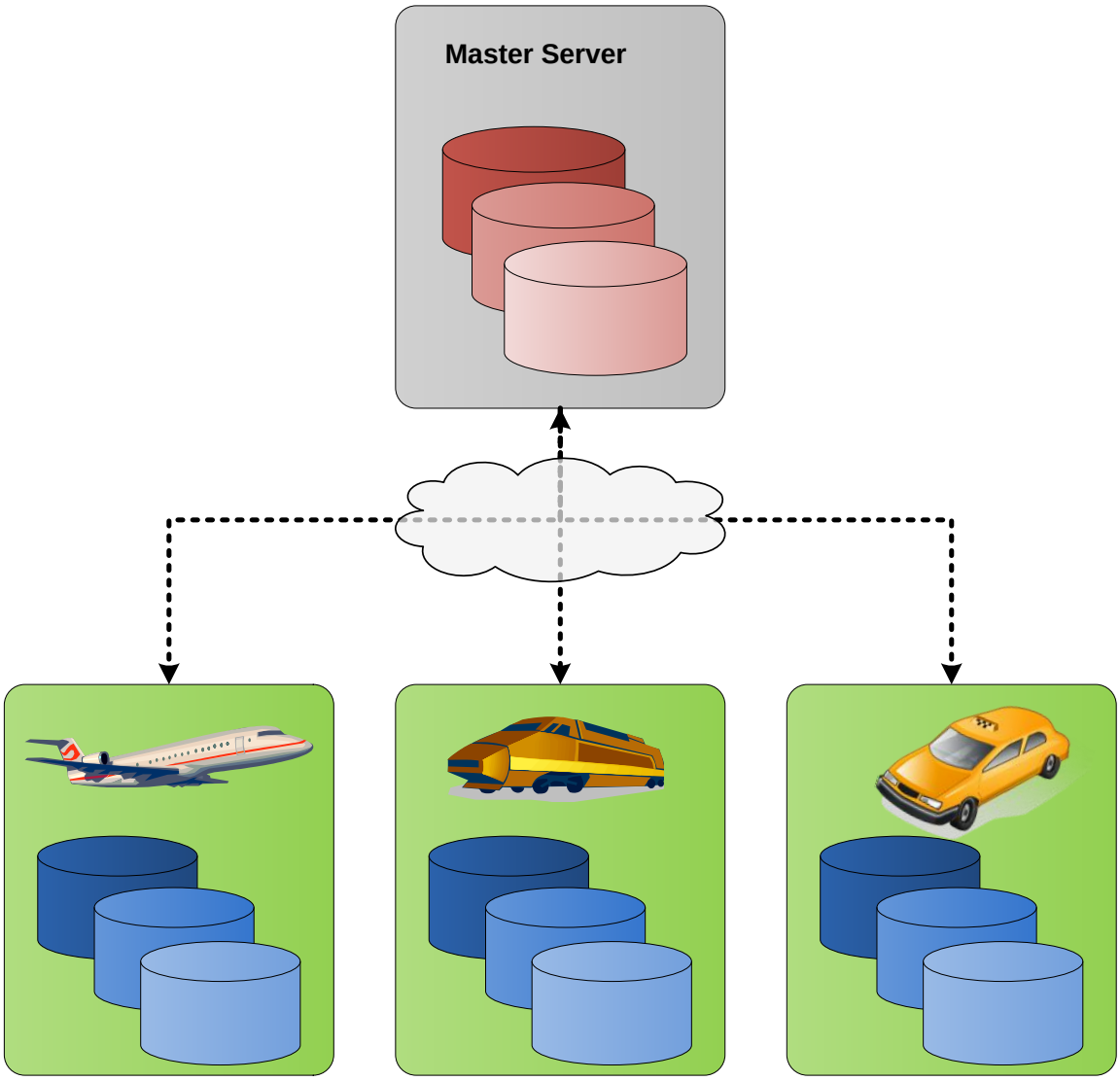
Security through Replication

With RDM Server it's possible to replicate specific databases on the Master server to targeted Slave servers. The use for such architecture could be security driven, where the requirement may be for specific databases to reside on certain servers. Or it could address performance issues, where an application expect certain databases to be access more often than others, and therefore wanting to located that database on a more powerful machine. Developers can easily define what Slave servers replicate what databases by setting the options in **RDMs_Slave** parameter within the rdmserver.ini file



Mobile Snapshots through Replication

With RDM Server's asynchronous replication solution, replicated databases don't need to be permanently connected to receive updates from the master. Through replication, developers can take a snap shot of your Master database and distribute it to devices in the field. As long as developers gracefully terminated the replication process, using by using the **s_repkill** API, your Slave database will be logically consistent and usable for read-only access. Once connected again the Master server and update your Slave database.



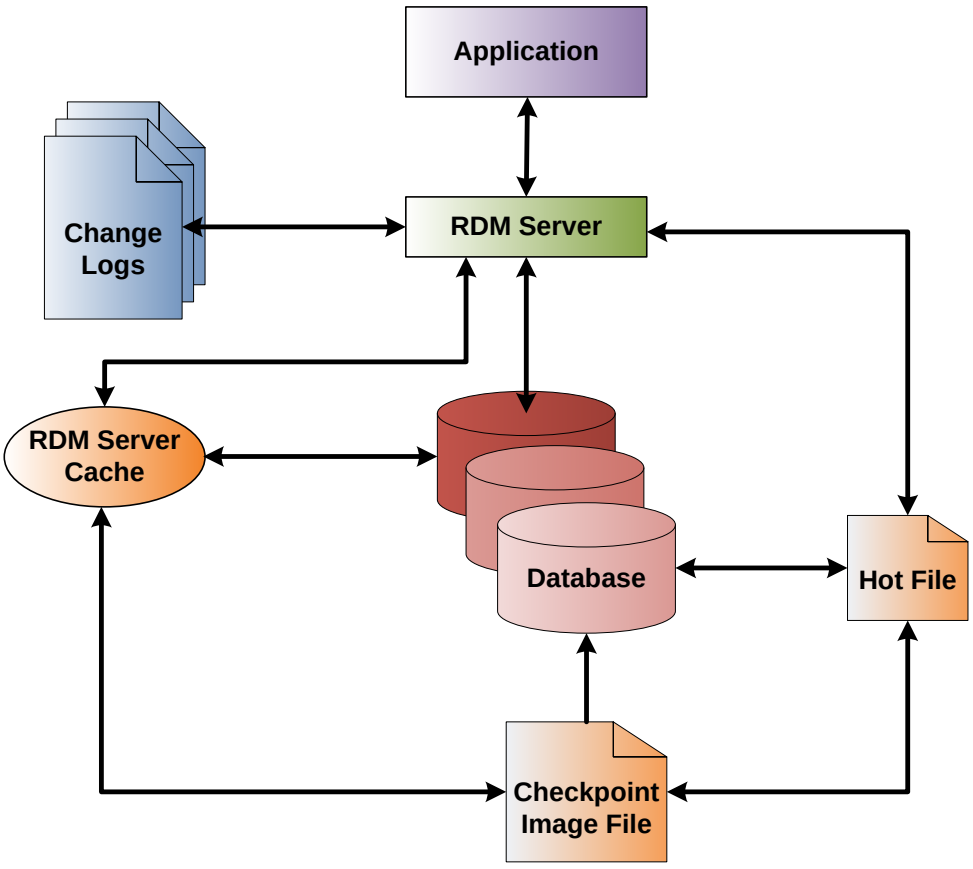
Hot Online Backup

High availability can also be implemented through Hot Online Backup. In today's dynamic business environment, where the expectation is that data should be accessible 24 x 7, Hot Online Backup offers the ability to back up one's database while still providing active read/write accessibility. Hot Online Backup is a superior solution to traditional Cold Backup in that it does not require any downtime of the system. RDM Server supports hot on-line backup, or backing up the database server while reads, writes, and other database activities continue. While in hot backup mode, the RDM Server database files are kept in a static, transaction consistent state, allowing any backup utility to back up the files. Alternatively, a developer can incorporate backup capabilities into an application, using the new hot backup administration functions. While in hot backup mode, RDM Server stores all changed database pages in a separate "hot" file. When hot backup mode has ended, the pages from the hot file are gradually migrated back to the database files as they are referenced.

Implementation

RDM Server implementation of Hot Online Backup is fairly simple; anyone with Administrator privilege can initiate backup either from RDSADMIN utility or programmatically using the s_ API's. When your system is in hot backup mode, you can expect the following conditions to apply:

1. Database Files are opened as shareable.
2. All updates to the database are written to a "hot" file.
3. No updates are made to the database files that will affect the logical consistency of the database.
4. You can expect to experience some performance degradation.



It is important to understand that RDM Server does not provide a utility to write the database file contents to a backup media device such as a tape drive. This is intentional in that it provides developers the ability to utilize any backup solution they want. Once hot backup mode begins, the actual backup can be performed manually or by using third-party backup software. When hot backup is terminated, the contents of the hot file are gradually migrated in to the database on a need-to basis. If developers wish to move all the changes from the hot file to the database immediately, they can do so by using the **s_backupFlush** API function.

The API's for Hot Online Backup are simple: **s_backupBegin** starts the backup; **s_backupEnd** terminates the backup process and **s_backupKill** that will unconditionally kill the backup process.

Conclusion

High availability is critical in today's environment and most applications are striving to achieve even the minimum requirements of five nines. RDM was designed from the ground up to always be on and if it went down it has always been able to get back on its feet. We called this from the beginning, in 1984 Zero Database Administration and it remains one of the key components of high availability within the RDM products. But as time has changed the requirements of within the database market so has RDM. Hot online backup was added to the product to increase HA functionality. More recently replication has been added to the RDM products providing even more HA functionality and flexibility. Developers looking to achieve high availability from a proven high quality embedded database at a reasonable cost need to consider Raima Database Managers.

Free Product Software Developer Kits Available at: http://www.raima.com/downloads/

Contact Information

Website: http://www.raima.com

WORLDWIDE

Raima Inc.
720 Third Avenue, Suite 1100
Seattle, WA 98104
Telephone: +1 206 748 5300
Fax: +1 206 748 5200
E-mail: sales@raima.com

EUROPE

Raima Inc.
Stubbings House, Henley Road
Maidenhead SL6 6QL, United Kingdom
Telephone: +44 1628 826 800
Fax: +44 1628 825 343
E-mail: sales@raima.com