Zmanda Recovery Manager
Guide to MySQL Backup & Recovery

By Dmitri Joukovski

Abstract

This paper introduces you to the Zmanda Recovery Manager (ZRM) for MySQL. ZRM is a comprehensive, certified, mission-critical enterprise backup and recovery solution designed to protect MySQL databases.

Table of Content

Introduction

MySQL has become the world's most popular open source database because of its ease of use, high performance, and high reliability. Regardless of how you use MySQL (website, forum, a custom MySQL application, Bugzilla bug tracking, or e-shop), you must ensure successful, secure and consistent backups with minimal impact on applications.

MySQL replication is a good mechanism to add robustness to a MySQL environment, but it is not a replacement for a comprehensive backup and archiving strategy: Replication only protects against hardware failure; it can not restore data lost due to user error (such as accidental table deletes) and malicious activity.

A robust backup and recovery solution is essential for ensuring data availability. Why fine-tune MySQL applications for high performance if you can not guarantee the complete protection of the critical data supporting them? There are several reasons why you should implement a robust backup and recovery solution for MySQL. The most common reasons include:

Disaster Recovery

While the computer industry has done much to increase hardware dependability, MySQL servers and storage hardware can still fail, resulting in permanent data loss. RAID (Redundant Array of Independent Disks) technology, although it improves dependability, is not completely fail-safe: Controllers still fail occasionally, and multiple drives can fail simultaneously.

Moreover, RAID storage cannot prevent data loss in natural disasters such as flooding, earthquake, or fire. Even the best storage arrays do not protect your data from disgruntled employees and outside hackers engaging in malicious activity.

Compliance Requirements

Depending on the industry, the law may require you to deploy a comprehensive backup solution to meet data retention requirements. In the United States and in Europe there are various regulations (SEC-17A, Sarbanes-Oxley, HIPAA, and many others) that require companies to retain data for various lengths of time. A good backup and recovery strategy is the first step towards regulatory compliance.

Protecting against User Error

Data-related mistakes are common. Keeping current backups lets you “go back in time” to restore a table or database to the point before a disastrous mistake was made.

Application Testing and Maintenance

Backing up your production MySQL server and then recovering its data to a test server efficiently accomplish two objectives at once: The backup and recovery procedures are verified, and you will have fresh production data to use in the test environment.

Requirements for a MySQL Backup Solution

After talking to dozens of DBAs over the years, we believe that the following requirements are essential for a successful MySQL backup solution.

Cross Engine and Platform Support

MySQL backups should work with every MySQL storage engine, including MyISAM, InnoDB, NDB and others. The solution should also back up multiple versions of MySQL running on Linux, Windows, Solaris, Mac OS X and other UNIX platforms. This enables MySQL DBAs to use optimal storage engine and platform for each application, without worrying about implication on backup.

Scale Up and Out

Often times you start with a single MySQL server and soon find yourself managing quite a few more. Even more challenging is that each database is growing in size exponentially. Your backup solution has to scale up and be able to backup Terabytes of data stored in a single database. It should also scale out from backup of a single MySQL server to a backup of multiple MySQL servers that could be geographically dispersed and be quite large.

Reduced or Zero Application Downtime

To accommodate data growth the backup solution should scale up with the size of the database without increasing lock times, thus minimizing the impact on users and applications.

A backup window is sometimes defined as the amount of time it takes to complete a backup. A better definition for backup window is the amount of time that the application is unavailable or degraded during backup. This is an important distinction, because some backup scenarios take the application offline for just a few moments, but can then spend hours moving the resultant backup to media. One such example is using snapshot technologies combined with traditional backup.

Relational Integrity Protection

A successful backup solution maintains the referential integrity of the database: any databases and tables that reference each other must be backed up together to ensure a consistent state after restoration.

Location of Backup Data

Storing backup data on the same disk or server as the original data is risky. Backup data should optimally reside in a physically separate location, preferably many miles from the production data.

Recovery Objective Requirements

Many DBAs focus too much on optimizing the backup process, thus ignoring an essential feature**– the ability to recover data after various mishaps as quickly as possible.**

The Recovery Time Objective (RTO) is the amount of time elapsed between the disaster and restoration of business functions. You should carefully consider how long your MySQL applications can be offline when designing your backup and recovery solution. For applications that cannot afford to be down for very long, you should consider MySQL replication with backup.

The Recovery Point Objective (RPO) is the time (prior to the disaster or error) to which you plan to recover your data. The most desirable RPO would allow restoring transactions to the point just prior to the mishap. ZRM for MySQL can recover data to any

point in time between two successful backups.

Ease of Recovery

Real life recoveries are done under stressful conditions. That is why simplified and bullet-proof recovery procedures make a big difference when you are the one responsible for recovery.

Application Recovery

You use MySQL server as a database for an application, for example SugarCRM or MediaWiki. To recover an application, you have to backup and recover not only the database and its binlog, but also all the necessary files that your application depends, for example the configuration files.

Scheduling and Automation

A backup solution should provide flexible, rule-based scheduling. A key feature is the ability to postpone and cancel backups based on administrator-defined thresholds. For example, you might want to postpone backup runs when the MySQL database is under heavy load.

Automation is also essential. Relying on human intervention to perform backups usually results in inconsistent execution. You should design your solution to work with as little human intervention as possible.

Automated backups must be monitored, preferably via customizable reports and automatic notifications. Backups are not very useful if you can't find the desired data quickly. At a minimum, your backup and recovery solution should list all backups performed. This list should contain what was backed up, when it was backed up, and the location of the backup media.

Backup Security

Access to backups must be secure. In addition to allowing encryption, the backup/recovery solution should control access to the tool itself, for example, allow only authorized personnel to perform recovery.

Zmanda Recovery Manager for MySQL

Zmanda Recovery Manager (ZRM) for MySQL was designed with extensive input from MySQL DBAs and meets all of the requirements outlined in the previous section. It:

- Is easy to install and use.
- Provides a complete backup and recovery solution for MySQL.
- Allows to backup and recover the files of an application that uses MySQL.
- Takes advantage of MySQL replication.
- Provides hot backup of live databases with minimal and measurable impact on users.
- Offers a full range of recovery options including point-in-time recovery.
- Allows global management of all backup operations for multiple MySQL servers.
- Allows flexible, automated backup scheduling.
- Provides real-time monitoring and automated reports to track backup processes.
- Ensures secure backup and recovery.

To start, we'll define a few terms and explain some of the technologies used by ZRM.

Backup Sets

A backup set is a collection of databases and tables from a single MySQL server that must be backed up on the same schedule using similar settings. ZRM lets you manage multiple backup sets so you can protect different servers and databases with settings appropriate for the respective applications.

When creating backup sets, remember to include all the databases and tables required to maintain the referential integrity and consistency of the application. Put all of the related databases in the same backup set.

Logical Backup

A logical backup saves the stream of SQL statements needed to recreate the database as it was in a particular point in time. The output is a text file, and is often compressed. Logical backups are possible regardless of the underlying backup engine (MyISAM, InnoDB, and others). Logical backups are also portable: they can be restored to a MySQL instance on another platform, or even a different database engine. But there is price to pay for such flexibility: Logical restores usually take longer than restoring from disk images, because all the MySQL statements must be replayed. Another disadvantage is the difficulty in predicting the size of logical backups. Depending on the type of data and your database schema, the size of the logical backup could be larger than the database itself. You can mitigate this problem by using compression; because logical backups are stored as text, they compress well.

Raw Backup

Raw backup provides a consistent copy of the database by storing a binary copy of the data files. The advantages of raw backups over logical backups are:

- Backup and especially recovery are much faster. For example, it not unusual to see that for the very same database with size of around 5 GB, the raw backup is 5 times faster than the logical backup, and recovery of the raw backup image is 20 times faster than recovery of the logical backup.
- Because it is simply a copy of the database files, a raw backup has a predictable size.
- It provides better scalability than logical backup, which could be important if your MySQL database is rather large, for example, 10-20 GB or more.

Raw backups can be recovered only to the same version of MySQL server running on the same operating system as the original data files.

For further details on the relative performance of different backup methods, register with the Zmanda Network at http://network.zmanda.com/ and read the white paper "Backup and recovery benchmarks for MyISAM and InnoDB engines with Zmanda Recovery Manager for MySQL". This paper provides backup and recovery benchmarks for MyISAM and InnoDB engines using all the methods supported by ZRM for MySQL, which are logical, raw and snapshot.

Hot and Warm Backup

Both raw and logical backups provide warm backup, meaning you don't have to shut down MySQL server for backup, but all tables are locked for writes during backup. That is why you should consider using the ZRM scheduling plug-in that allows delaying backups based on thresholds defined by you. For example, you can postpone a backup for an hour if more than 50

users are accessing the database right now.

Hot backup means you can back up a database without any impact on users. ZRM provides hot backup by using snapshots. Also, ZRM takes no lock on the database while performing incremental backups.

Using Snapshots for Backup

To perform backups with the least impact on database applications, consider backing up from snapshots. A snapshot provides a logical copy of a file system at a point in time. That logical copy can be used as a backup source, allowing recovery of the database to the point in time when snapshot was taken. There are many software and hardware snapshot technologies, and today these technologies are mature and robust. Fundamentally all of them work in a similar way. Let's take a look at one of the most common mechanisms: the copy-on-write snapshot (available through the Linux Logical Volume Manager and some file system types).

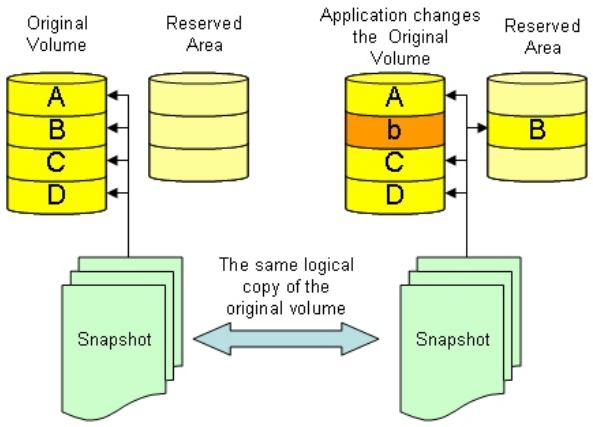

Figure 1. Logical Volume Manager copy-on-write snapshot

LVM copy-on-write creates a logical copy of an entire volume at an instant in time. A logical copy is a "picture" of the data on a volume. Since there is no movement of the actual bits stored on the source volume, the snapshot is almost instantaneous regardless of how much data is stored on the volume. It usually takes less than a second. The snapshot creates a new logical device in the **/dev** directory that stores changed data. When you mount and read a snapshot most of the data is actually read from the original volume.

When the application is about to change a block of data on the original volume, the LVM copies that block to the reserved area before making the change. This is why LVM snapshots are called "copy-on-write." Perhaps, "copy-before-write" is a more accurate description.

When the snapshot is read, the LVM reads data from both the original and the reserved area, providing you with the same data as you had on the original volume when snapshot was taken. Please note that you must set up a reserve area that is large enough to accommodate all changed blocks. Depending on how many blocks change, you can take snapshots of very large volumes with a relatively small reserved area. However, if the reserve is insufficient to store the changed blocks, the snapshot will fail. The result is similar to having a removable drive mounted and then ejecting the media without un-mounting it.

When correctly configured, snapshots reduce the read lock time to a few seconds or even milliseconds. To ensure consistency of data for MyISAM and other storage engines, ZRM issues a read lock for a moment (in most cases a second or two), which is acceptable in most application scenarios. Note that the lock is enforced only during snapshot creation; there is no lock during the subsequent data transfer when the snapshot is mounted and read. You can continue to run queries on your tables even during the momentary lock time.

Currently ZRM supports the following snapshot technologies:
- Linux LVM snapshot
- Veritas VxFS storage checkpoints
- Sun Solaris ZFS snapshots
- Network Appliance snapshots
- EMC CLARiiON SnapView snapshots
- Microsoft Windows VSS (Volume Snapshot Service aka Volume Shadow Copy Service) snapshots

Regardless of the snapshot type, ZRM performs the following steps to ensure recoverable and consistent backups:

- Freeze the database
- Flush the memory buffers for logical consistency of data on disk (you can not take a snapshot of memory – this technology works only for a disk)
- Process the snapshot, which actually includes several steps:

Action	Example of implementation for NetApp
Identify database volume	Identify filer and WAFL volume
Create snapshot	Snapshot using DATA ONTAP
Mount snapshot for reading (moving to a different location)	Mount WAFL on NFS mount point (move snapshot from the filer to ZRM server)
Un-mount snapshot	Un-mount WAFL snapshot
Delete snapshot	Delete using DATA ONTAP

- Unfreeze the database right after the "create snapshot" step above
- Manage the snapshot:
    - Scheduling
    - Moving to a different location
    - Monitoring and reporting

Be advised that some vendors (NetApp, for example) keep snapshots on the same filer as the original data files. While that is perfectly acceptable against user error, for disaster recovery you should move backup data off the original filer to another filer by using NetApp replication software.

ZRM for MySQL lets you choose where to store backups. You can back up to any disk that is writeable by ZRM. For example, you can backup to a local disk, a disk array, or to a NAS device. You should never keep your backup data on the same spindles as the original data, not even for a short period of time unless you copy it onto another server or medium and keep a version of the backup on the same disk for faster recovery. Quick snapshot option allows you to store snapshots on the same volume as the original data to enable almost instantaneous recovery from a snapshot.

Recovery Points

When data recovery is required, the first question you must answer is "what is the recovery point?" For example, suppose you have been using ZRM to execute full backups every day. On Friday you discover that the database is corrupted. Which backup should you use? Usually the answer will be the most recent backup before the corruption took place. In our example that might be Thursday's full backup.

The more frequently you backup, the more granular your recovery point can be. However, it is not feasible to perform a full backup on most databases on an hourly basis. As you will read in the Backup Levels section below, by using incremental backups you can achieve a more granular recovery point.

ZRM enables Continuous Data Protection[1] (CDP) by using LVM, VxFS, VSS, ZFS, EMC CLARiiON SnapView and NetApp snapshots in conjunction with MySQL transaction logs. When doing a recovery to a particular point in time, ZRM reads data from the snapshot and then replays MySQL transaction log from that point forward. This creates a point-in-time recovery that satisfies the SNIA definition of CDP.

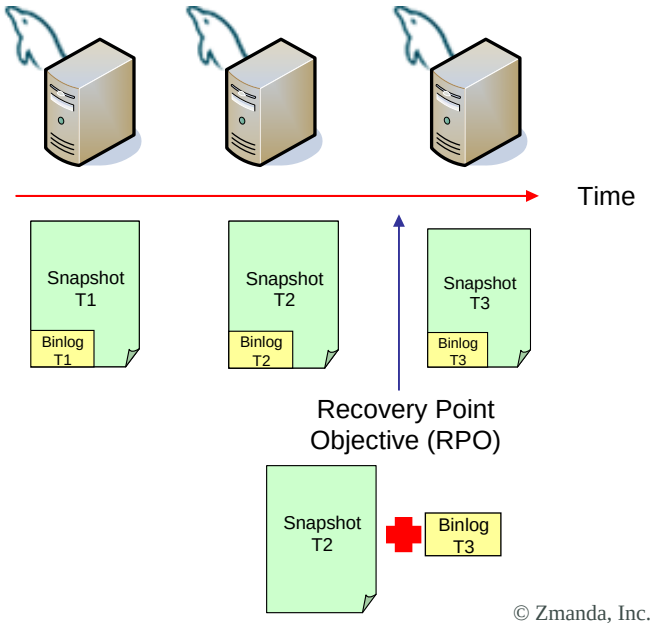

Figure 2. Point-in-time recovery using snapshots and binlogs with ZRM.

For each point in time T, ZRM creates a snapshot that includes a MySQL binary log. To recover MySQL database to a specific Recovery Point Objective (RPO) between T2 and T3, ZRM reads data from snapshot T2 and replays transactions from binlog T3 up to RPO providing the user with point-in-time recovery. Quick snapshot option in ZRM (activated via ZMC) enables instantaneous recovery of snapshots stored where the database files are, for example, on the same NetApp filer.

[1] Storage Networking Industry Association (SNIA) defines Continuous Data Protection (CDP) as "a class of mechanisms that continuously capture or track data modifications enabling recovery to previous points in time", see http://www.snia.org/education/dictionary/c/

Backup Levels

ZRM supports two backup levels – full (level 0) and incremental (level 1).

A full backup is a complete copy of the specified databases or tables. The copy can be logical, raw, or snapshot-based.

An incremental backup stores data changed since the last (full or incremental) backup by leveraging MySQL binary logs. When ZRM for MySQL performs an incremental backup, it flushes and saves the MySQL binary logs. Because logs can be flushed and saved without a read lock, it is a "hot backup", having no impact whatsoever on the live database.

The binary log saves all database events, thus providing a very fine granularity of recovery points. ZRM makes it easy to recover data from incremental backups even when multiple incremental restores are required to achieve the recovery point. Although incremental backups require that MySQL binary logs are enabled, in most cases this results in a performance hit of less than 1%.

ZRM Architecture

Before discussing the details of ZRM for MySQL, let's take a brief look at ZRM architecture.

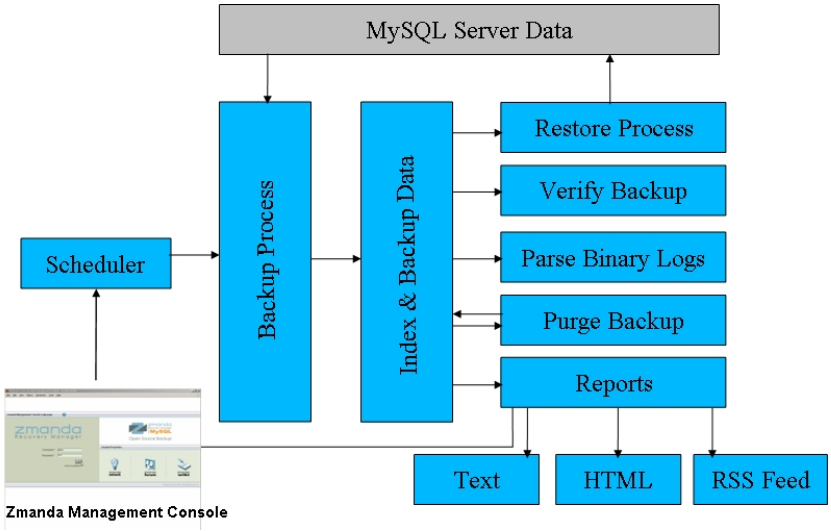

Figure 3. ZRM architecture.

ZRM architecture is modular and pluggable, which simplifies customization and addition of new features.

The ZRM scheduler starts all backup processes. Depending on the MySQL server, storage engine, and underlying storage technology (for example, whether snapshots are available), ZRM intelligently decides what kind of backup to perform. The primary consideration is impact on the database. ZRM bases its scheduling decisions on the following criteria:

- Are snapshots possible?
- Is a raw backup possible?

- Is replication available?
- Does the backup set include a MySQL Cluster database?

If all other options are not feasible or, for example, if snapshot fails because there is not enough space in the reserved area, ZRM performs a logical backup.

You can always override the ZRM recommendations. For example, instead of using a snapshot you can force ZRM to perform a logical backup.

ZRM will index all backups to track the metadata required for a successful restore. Every day ZRM checks if any backups are older than the retention policy (the number of days or months that backups should be kept). The expired backups are purged from the ZRM backup index and removed from the disk. A reporting engine provides customizable reports in either text or HTML format. For ZRM installed on Linux or Solaris reports can be distributed by e-mail or by RSS feed. For ZRM installed on Windows all backup information is logged in a Windows event log.

The Zmanda Management Console (ZMC)

The Zmanda Management Console (ZMC) is a browser-based user interface for setting up and managing all backup and recovery activities.

ZMC is integrated with the Zmanda Network, which provides certified ZRM binaries, white papers, demos, technical support and a knowledgebase to help you deploy MySQL backup solution. ZMC provides context-sensitive help and the most current information about ZRM from the Zmanda Network. After registering with the Zmanda Network you will be notified of new ZRM releases via RSS feed.

Figure 4. Zmanda Management Console top-level navigation.

Just as the MySQL Enterprise Monitor lets you manage MySQL server operation, ZMC lets you manage backup and restore operations for MySQL.

Ease of Installation

The ZMC Rapid Installer provides an easy method of installing ZRM and all of its components, without affecting the existing MySQL configuration on the target server. ZMC is built on certified LAMP and SAMP (Linux or Solaris, Apache, MySQL, Perl/PHP) stacks. What if you already have one or several of these components on the host where you plan to install ZRM? This is not a problem at all since the rapid installer keeps all required AMP components in a ZRM specific location (/opt/zmanda/zrm). In addition, those components that require network ports will be configured to use ports that do not conflict with those being used by your existing installations.

Comprehensive Backup and Recovery for MySQL

ZRM for MySQL is an enterprise-strength solution, covering all versions of MySQL and dependent technologies (storage engines, replication, cluster servers, etc.) regardless of OS

platform. This comprehensive coverage is described in the following sections.

Database engines

ZRM provides logical, raw, and snapshot-based backup for all database engines including NDB clusters. Note that some engines do not support all backup methods:

Storage Engine	Logical	Raw	Snapshot
MyISAM	Yes	Yes	Yes
InnoDB	Yes	No	Yes
NDB (cluster)	No	Yes	No
Archive	Yes	Yes	Yes
Falcon	Yes	Yes	Yes

You can mix storage engines (for example, InnoDB and MyISAM tables) within a single backup set. In such cases, ZRM will account for all storage engines in the backup set before deciding on the optimal backup mechanism.

Platform support

ZRM runs on Windows, Solaris 10 or a Linux-based server. ZRM installed on Windows can backup only MySQL servers running on Windows. ZRM installed on Solaris or Linux can back up MySQL databases running on any Windows, Linux or UNIX platform. In the following illustration, a ZRM server on Linux or Solaris backs up a MySQL server running on Windows, Linux and Solaris, each using different storage engines.

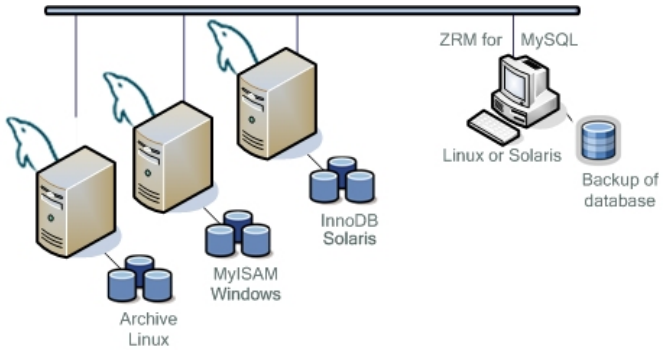

Figure 5. Global management of backups with ZRM.

Backup methods used by ZRM

The table below summarizes the technologies that ZRM supports for different operating systems:

MySQL server			
Linux	Solaris	Windows	OS-X and other UNIX
Logical – warm Raw - warm LVM - hot VxFS – hot NetApp – hot	Logical – warm Raw - warm ZFS - hot VxFS – hot NetApp – hot	Logical – warm VSS - hot	Logical – warm

ZRM also can also take full advantage of MySQL replication. If you set up MySQL replication,

you can use the replication slave as the backup source. Such backups have zero impact on your primary MySQL instance.

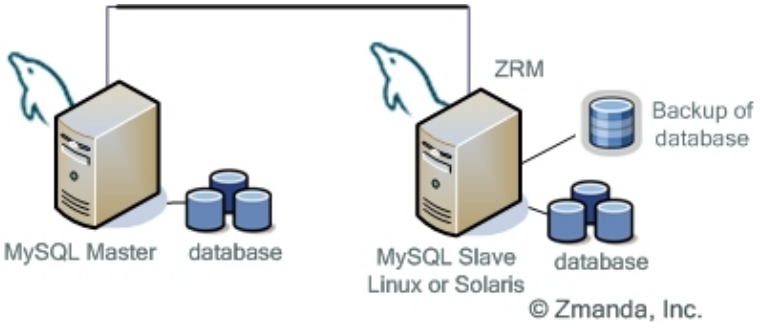

Figure 6. ZRM taking advantage of MySQL replication.

Your MySQL Master can be on any operating system, but for ZRM to take advantage of replication, your MySQL Slave has to be on either Linux or Solaris. ZRM installed on MySQL Slave will ask the MySQL Master to temporarily stop replication, perform raw backup and then continue replication.

Backup of MySQL Cluster

MySQL supports clusters for high-availability database applications, using the NDB database engine. ZRM performs raw backup of clusters using MySQL-ndb-tools and is the only product that provides consolidated backup solution for MySQL clusters.

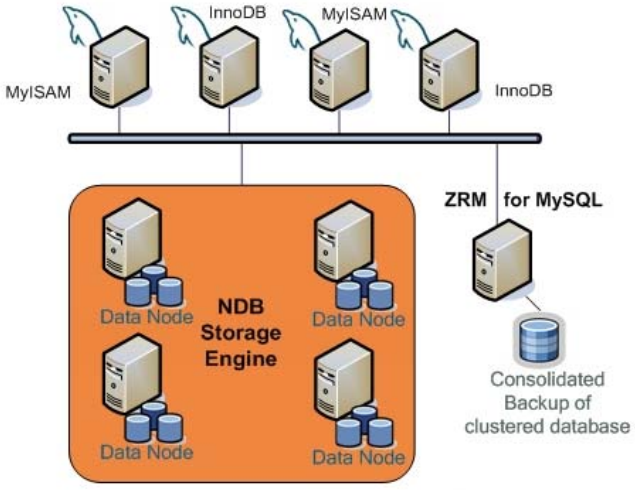

Figure 7. Backup of NDB cluster with ZRM.

ZRM automatically consolidates backup from each Data Node, which simplifies the recovery process. ZRM lets you recover data to a cluster with fewer Data Nodes than the original cluster. This important feature lets you recover from a disaster with less hardware than was available in the original cluster configuration.

Simplified Backup Configuration

To configure ZRM backups, all you need to do is specify:

- What to back up (specify server, database(s) and table(s) to back up)
- Where to store the backup (specify a destination)
- When to perform the backup (scheduling options for full and incremental backup)
- How to perform the backup (specify copy mechanism(s) to use).

ZRM user interface is organized into these tasks, and includes a corresponding page (**Backup What**, **Backup Where**, **Backup When**, and **Backup How**) to accomplish each task as described in the sections that follow.

The **Backup What** page simplifies backup source selection by automatically discovering all of the databases and tables on a given MySQL server. The following example shows a ZRM for MySQL set to back up a server that includes six databases.

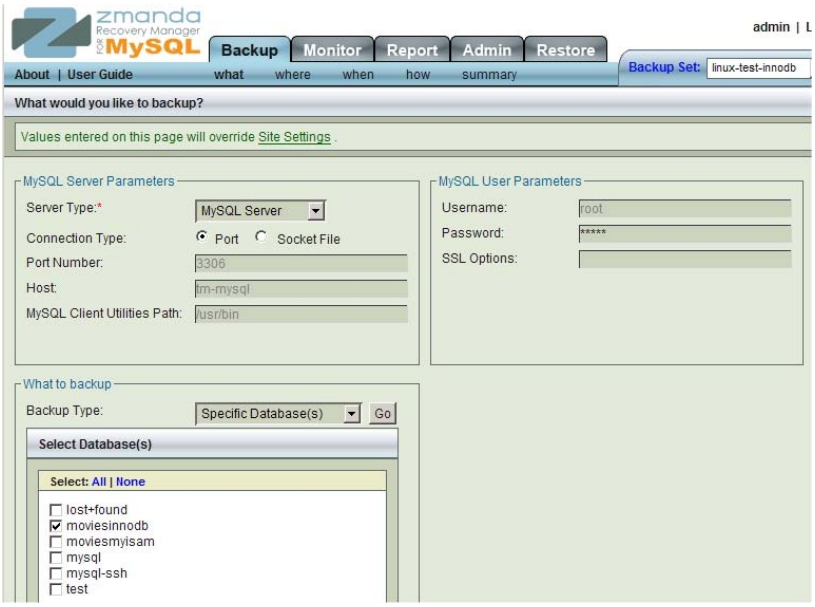

Figure 8. Screenshot of the Backup What page

You can choose to back up all databases, particular databases, or particular tables of a database.

The **Backup Where** page allows you to specify which disk directory will be used to store the backups for this backup set.

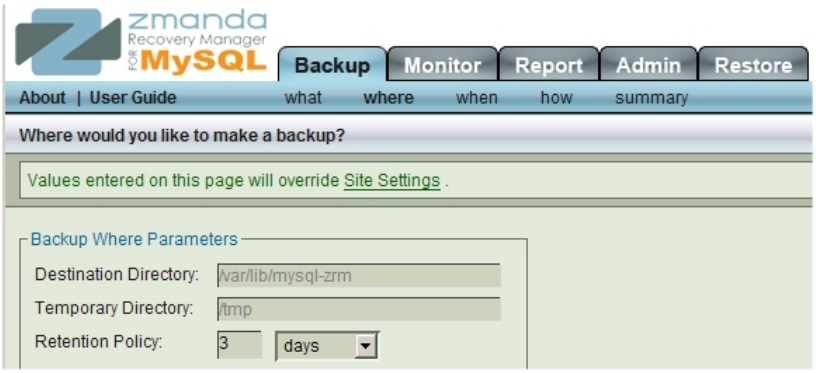

Figure 9. Screenshot of the Backup Where page.

You can store backups to any storage that is available to the server's OS: a local disk, Network Attached Storage (NAS), or a RAID array on a Storage Area Network (SAN) that you share between multiple servers and applications.

The **Backup When** page lets you specify both full and incremental backups for the same backup set. To do this, you create multiple schedules for the backup set. You can, for example, create one backup schedule for full backup at 2:00 AM on Mondays, and another one for incremental backups at 4:00 AM on Mondays, Wednesdays and Fridays (as shown below).

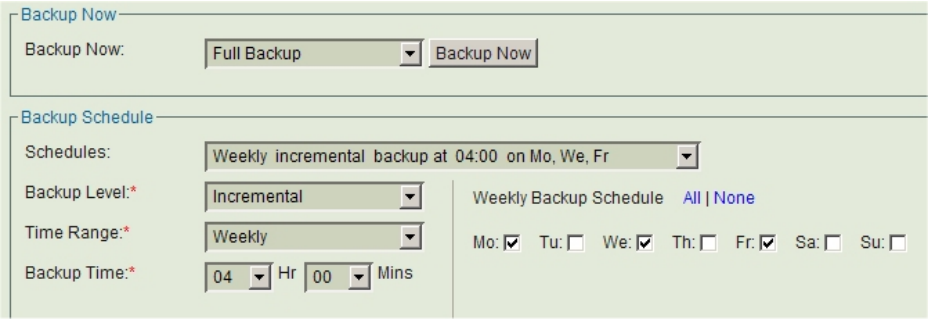

Figure 10. Screenshot of the Backup When page

You can also start either full or incremental backup immediately by clicking on "Backup Now" button. Immediate backup is very convenient when testing backup procedures. It is also a good idea to perform immediate full backup before you update the version of the MySQL server or the operating system.

The **Backup How** page specifies the mode of backup – logical or raw. It also lets you choose various options to control your backup: whether to use compression or encryption, pre and post processing scripts, how to remotely copy raw and incremental backups, and whether MySQL replication is used.

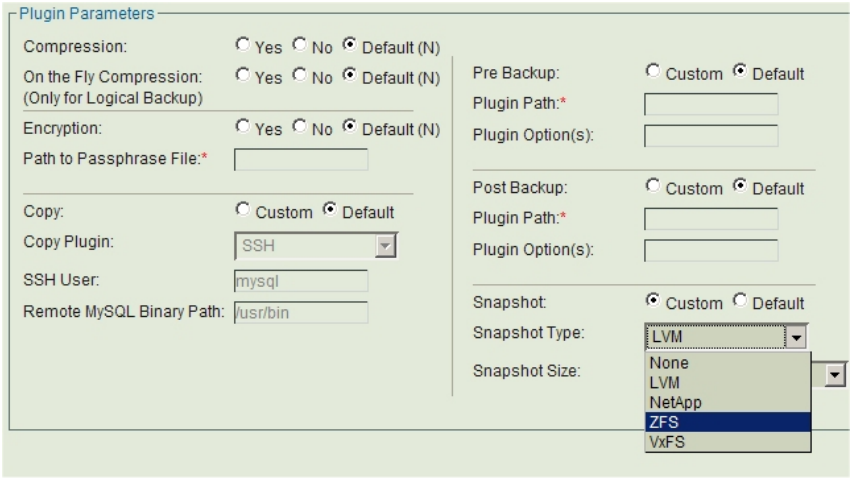

Figure 11. Screenshot of the Backup How page

Please note that you will see the configuration settings for snapshots only when you select Backup Mode as "Raw".

Compression and Encryption

ZRM supports compression and encryption of the backup images, using any utility supported by the OS. For example, for ZRM running on Linux or Solaris you can use **gzip** for compression (which is the default). ZRM installed on Windows takes advantage of the Encrypting File System (EFS) component of NTFS on Windows. In general, any utility that reads from standard input (**stdin**) and writes to standard output (**stdout**) can be used.

Reporting

The Zmanda Management Console lets you monitor your backups with a full-featured reporting tool. A calendar-based view shows the success or failure of backups on any given day using color-coded indicators: a red square indicates a failure, a green square indicates successful backup. You can drill down to the details for any backup.

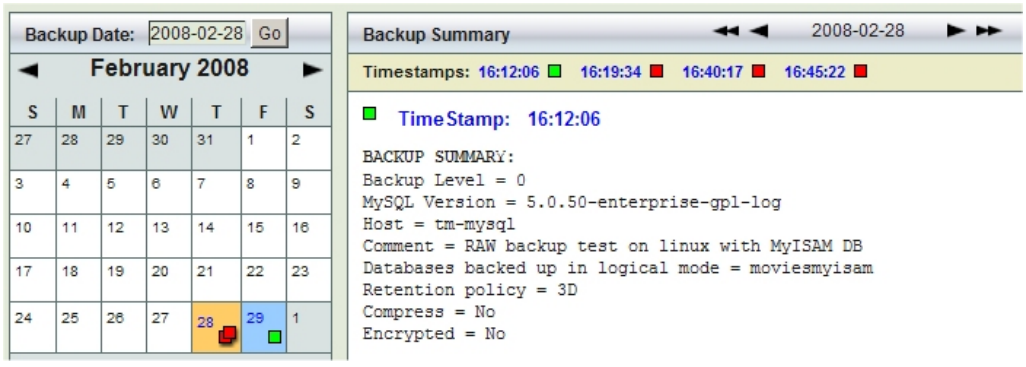

Figure 12. Calendar-based view of backup summary reports.

There are nine standard, pre-defined reports, all of which can be easily customized:

- Backup report

- Backup application performance report
- Backup status report
- Backup methods report
- Backup retention policies report
- Backup performance report
- Incremental backup report
- Backup replication report
- Cluster backup report

Some of these are designed to give you detailed status information of your backups. Others help you fine-tune your backup for performance. For example, the "Backup Performance" report, among other things, breaks down the amount of time spent performing a backup: for example, how much time was spent performing compression or encryption.

The "Backup Application Performance" report shows you the impact that the backup had on the MySQL dependent applications. This report helps you determine if you should alter the ZRM backup method being used to optimize application performance.

Backup Application Performance Report for linux-test-innodb

Backup Size	Flush Logs time	Read Locks Time	Time Taken
340.04 MB	00:00:00	00:00:03	00:04:02
	00:00:00	00:00:00	00:00:00
	00:00:00	00:00:00	00:00:00
146.37 MB	00:00:00	00:02:02	00:02:43
146.37 MB	00:00:01	00:02:30	00:03:05
146.37 MB	00:00:01	00:02:22	00:02:30
146.37 MB	00:00:01	00:02:25	00:02:30
	00:00:00	00:00:00	00:00:00

Figure 13. Backup Application Performance report.

The report above shows how backing up an InnoDB table with size of 340MB affected the database in question. It shows that read lock time was 3 seconds (we used MySQL running in a Virtual Machine for this white paper) and total backup time of 4 minutes and 2 seconds.

In addition to the pre-defined reports, you can create and save for future use highly-customized reports on more than 30 different backup and MySQL parameters.

Reports can be generated and delivered in a variety of ways. The output may be either text or HTML. They can be delivered as RSS feeds or as email. These options let you integrate ZRM reporting with the monitoring dashboard you are already using.

Database Events Viewer

The **Database Events Viewer** (Visual Log Analyzer) lets you visually browse the MySQL binary logs used for incremental backup, and select exactly what point in time you wish to restore.

The date and time of each database events is shown as an individual record. You can easily scroll through the entries, and search the logs with queries.

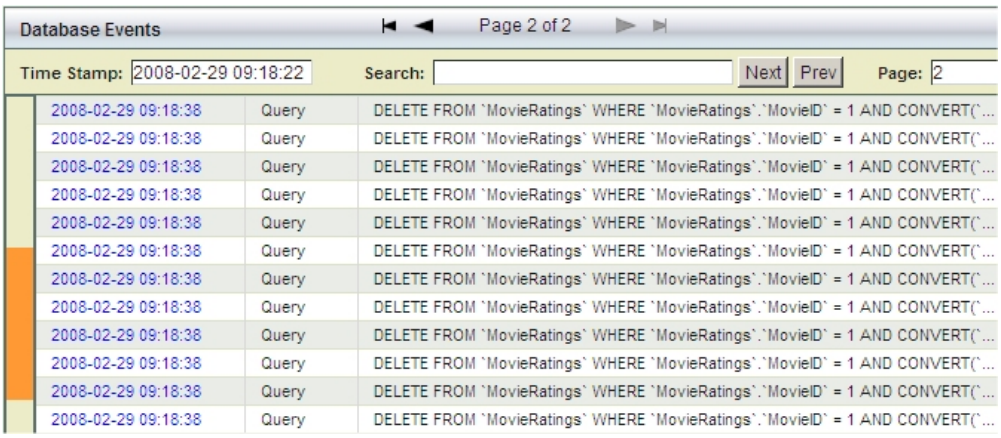

Figure 14. Database Event Viewer showing available recovery points

Using the search, you can locate any event that caused harm to your database. You can also jump to a given date and timestamp. To help you navigate through events, there is a special vertical orange-colored tool bar to the left of the date and time stamps in the database events list. As you move your cursor up and down this vertical bar a tool-tip displays the date and time of the events in that relative location. Clicking the tool-tip jumps to the given date and time.

All of this lets you choose a recovery point right down to the transaction level. **You can set the recovery point right from the database events viewer.** Just click on the date and time stamp link next to the last transaction you wish to recover. ZRM then directs you to the Recover screens, with all the appropriate values filled in to recover the database to that exact date and time.

The Database Event Viewer is a unique tool that has many uses outside of recovery planning. For example, you could use it to scan the logs for malicious or suspicious activity.

Simplified Recovery

The "R" in ZRM stands for Recovery, which of course is ZRM's primary objective: simple, dependable, and efficient recovery of lost data.

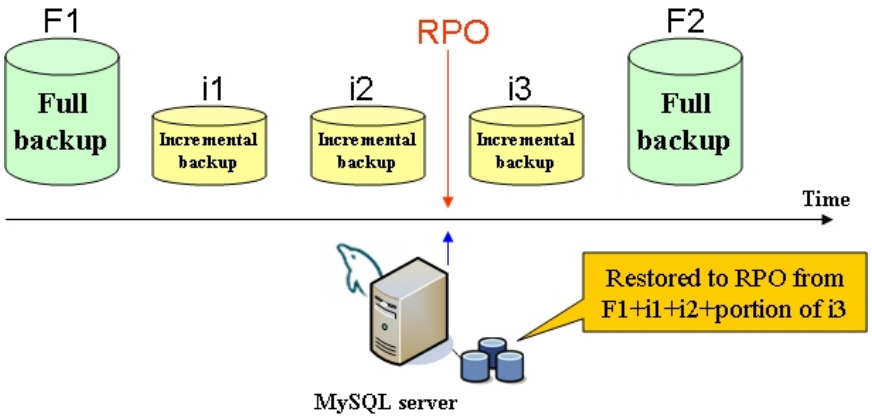

Figure 15. Restoration from full and incremental backups to Recovery Point Objective.

To restore data, simply select the backup set, the database or individual tables to recover. You can restore individual tables only if logical backups have been run. After selecting what to restore from, choose a recovery point objective (RPO). ZRM catalogs all full and incremental backups performed. When you request a point-in-time restore, ZRM automatically finds the full backup and all the incremental backups required to restore your database to the requested RPO. In the example above, to restore MySQL to the RPO requires restoring full backup F1, incremental backups i1, i2, and a portion of the binary log i3.

ZRM allows you to restore to a point in time regardless of whether logical, raw or snapshot based backup was executed; the only requirement is that you have set up incremental backups as well.

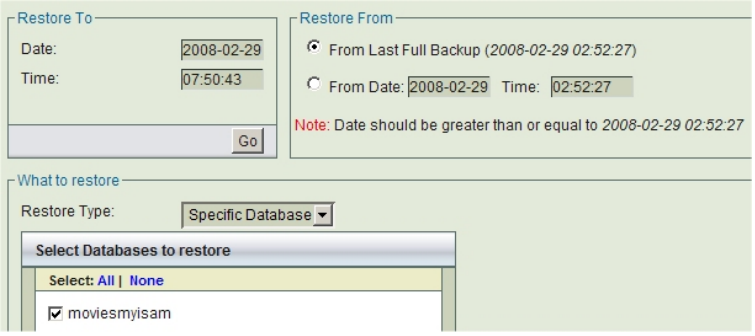

Figure 16. Screenshot of Restore What page.

Depending on how you determined the RPO, you can launch the restore page in several ways:

- Click one of the timestamp links included in any successful backup report.
- Identify the recovery point in the **Database Events Viewer** and click on timestamp of the given event.
- Open the **Restore What** page and restore from the last full backup or any other date on which a successful backup was completed.

You can restore to a MySQL server that differs from the original backup source. You can also use restores to instantiate new MySQL replication slaves.

This simplified restore workflow is the same regardless of storage engine and backup method used. When recovering mission-critical data, having a unified recovery procedure for all MySQL servers helps ensure efficient, dependable, and stress-free recoveries.

Simplified Administration

ZRM provides several functions to simplify MySQL backup administration. One of these is Role Based Access Control. You can define multiple ZRM users with Administrator or Operator permission levels. Administrators have full privileges to manipulate any backup set. Operators can only manipulate backup sets assigned to them. This is useful when distributing backup and recovery responsibilities among several people. This is also useful for a service provider looking to give control of backup and recovery of specific databases to heir respective owners.

ZRM also lets you set site-wide defaults that allow you to quickly create backup sets appropriate for your MySQL environment. The defaults can be overridden when defining a backup set, allowing ZRM users to defining backup sets by changing only those items that are unique to the specific backup set, thus saving configuration time.

Conclusion

Zmanda Recovery Manager will help you to globally manage all backup and recovery operations for MySQL servers from a centralized console. It will ease the sometimes complex backup and recovery burden that every DBA must carry and let you focus on other important tasks of MySQL management.

Where to Get Additional Information

For additional information, please register free of charge to the Zmanda Network http://network.zmanda.com where you can find demos, presentations and additional technical white papers about ZRM.