Ensuring Effective MySQL Backups on Enterprise Production Systems

By Thomas Weeks ©
Lead Systems Engineer
tweeks@rackspace.com
For MySQL User Conference, 2005-04-19
About Me

• Thomas Weeks
  Lead Systems Engineer

• BS-EET / Telecom, RHCE

• President of XCSSA.ORG
  (X-otic Computer Sys. Of S.A. TX)

• Co-Author with Chris Negus of
  “The Linux Troubleshooting Bible”
About Rackspace

- **Vital Statistics**
  - Founded in 1998
  - Based in San Antonio, TX
  - 100% Focused on Managed Hosting
  - 600+ Employees
  - 5 Data Centers: Texas (x3), Virginia, & London, UK
  - 6,700+ Customers
  - Net Income Positive Since February 2001 - $100+ Million in Revenue
  - 97% of Our Customers Would Refer us to a Business Colleague

- **Industry Leadership**
  - The only MySQL Certified Hosting Provider
  - Redhat Advanced Hosting Provider
  - More Red Hat Certified Engineers than any other hoster on the planet
  - 100% Network Uptime for 4+ Years Running – Cisco Powered Network™
  - Customers include Atari, Best Buy, EMI Records, Miller Brewing, Hershey’s, Motorola & National Geographic

United States: www.rackspace.com | 800.961.2888
Europe: www.rackspace.co.uk | 0800 085 3973
What We're Covering...

- Types of System Wide Backups
- Types of Database Backups
- MySQL Database Backup Agents and Tools
- Database Backup Strategies
- Production Backup and Restoration Issues
What We're Covering...

- Types of System Wide Backups
- Types of Database Backups
- MySQL Database Backup Agents and Tools
- Backup Strategies
- Production Backup and Restoration Issues
Types of System Wide Backups

In UN*X environments, OS level or system wide backups are classified into three general categories:

- **Differential Backups**
  - Backs up changes since the last *full* system backup

- **Incremental Backups**
  - Backs up changes since the last backup

- **UN*X Backup Levels**
  - Numeric backup “Levels” indicating backup depth
  - 0=Full Backup
  - 1-9=Backup that backs up incremental change since next lower level number
Types of System Backups
Differentials

Differential Backups simply get everything since the last full backup.

**Diffs Pros**
- Less tapes than nightly full backups
- Full restore only takes two tapes
- Higher data assurance
- Restores faster than Inc.

**Diffs Cons**
- Backups slower than Inc.
- Much redundant data wasting tape
- Expensive tape costs

© Wiley, 2004
Types of System Backups

Incremental

Incremental Backups get everything that has changed since the last incremental backup.

### Incremental Pros
- Most efficient for tape use
- Fastest backup method
- No redundant data

### Incremental Cons
- Restores multiple tape sets
- Restores take much longer
- Lower data assurance (no redundant data)
Types of System Backups
UN*X Backup Levels

Level 0 is a “full backup”, and levels 1-9 are incremental levels that get everything since the next lower level backup.

**Level 0-9 Pros**

- Most flexible
- Allows balance between most efficient and most effective
- Allows control over overall backup strategy
- TOH based level algorithms included in most commercial backup suites

**Level 0-9 Cons**

- Complex to set up manually
Types of System Backups
Level 0-9

Here is an example of using a modified “Tower of Hanoi” backup strategy to balance efficiency, cost, and data assurance. This can be accomplished with backup packages such as UN*X/Linux dump, Legato, Amanda, etc.

Weekend Level-0 Backup w/Modified TOH

- = New Data
- = Previous Data
- = Level 0

new/changed files (since last backup)
previous data level copied again
## Types of System Backups
### Backup Media Types & Costs

<table>
<thead>
<tr>
<th>Media</th>
<th>$ Drive</th>
<th>$ Med.</th>
<th>Nat/Comp</th>
<th>$/GB*</th>
<th>$/10TB**</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVD-RW</td>
<td>$100</td>
<td>$1.50</td>
<td>4.7GB</td>
<td>$0.39</td>
<td>$4,000</td>
</tr>
</tbody>
</table>
| Pro: Good for server config backups, DBs, and recovery data and tools  
Con: Still expensive, easy to outgrow  
**DVD-RW**: Good for server config backups, DBs, and recovery data and tools  
**Con**: Still expensive, easy to outgrow  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

<table>
<thead>
<tr>
<th>Hard Disk</th>
<th>$150</th>
<th>N/a</th>
<th>150GB</th>
<th>$1.00</th>
<th>$10,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVD-RW</td>
<td>$100</td>
<td>$1.50</td>
<td>4.7GB</td>
<td>$0.39</td>
<td>$4,000</td>
</tr>
</tbody>
</table>
| Pro: Good for server config backups, DBs, and recovery data and tools  
Con: Still expensive, easy to outgrow  
**Hard Disk**: Good for server config backups, DBs, and recovery data and tools  
**Con**: Still expensive, easy to outgrow  
**Includes price of the drive. This figure is not a linear comparison but just a ball park figure to show much your first 10TB of backups on this drive & media will cost you. You really need to account for your GB/month needs and project it over the course or a year to get a real world comparison for your scenario.**

<table>
<thead>
<tr>
<th>DDS3</th>
<th>$400</th>
<th>$4</th>
<th>12/24GB</th>
<th>$0.20</th>
<th>$2,400</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDS3</td>
<td>$400</td>
<td>$4</td>
<td>12/24GB</td>
<td>$0.20</td>
<td>$2,400</td>
</tr>
</tbody>
</table>
| Pro: Cheap price/GB, good for small jobs, easy to find media  
Con: Outdated, limited growth potential  
**Outdated**: Cheap price/GB, good for small jobs, easy to find media  
**Con**: Outdated, limited growth potential  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

<table>
<thead>
<tr>
<th>DDS4</th>
<th>$500</th>
<th>$8</th>
<th>20/40GB</th>
<th>$0.24</th>
<th>$2,900</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDS4</td>
<td>$500</td>
<td>$8</td>
<td>20/40GB</td>
<td>$0.24</td>
<td>$2,900</td>
</tr>
</tbody>
</table>
| Pro: Cheaper of high end AIT drives, good for small-medium servers  
Con: Comparative ROI of media over time is expensive  
**AIT-2**: Cheaper of high end AIT drives, good for small-medium servers  
**Con**: Comparative ROI of media over time is expensive  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

<table>
<thead>
<tr>
<th>AIT-2</th>
<th>$1,100</th>
<th>$45</th>
<th>50/100GB</th>
<th>$0.53</th>
<th>$6,400</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIT-2</td>
<td>$1,100</td>
<td>$45</td>
<td>50/100GB</td>
<td>$0.53</td>
<td>$6,400</td>
</tr>
</tbody>
</table>
| Pro: Good middle road tape with growth potential. Good for changers  
Con: Media still expensive/GB  
**Outdated**: Good middle road tape with growth potential. Good for changers  
**Con**: Media still expensive/GB  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

<table>
<thead>
<tr>
<th>AIT-3</th>
<th>$3,000</th>
<th>$50</th>
<th>100/200GB</th>
<th>$0.29</th>
<th>$5,900</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIT-3</td>
<td>$3,000</td>
<td>$50</td>
<td>100/200GB</td>
<td>$0.29</td>
<td>$5,900</td>
</tr>
</tbody>
</table>
| Pro: Cheaper end of AIT drives, good for small-medium servers  
Con: Comparative ROI of media over time is expensive  
**AIT-3**: Cheaper end of AIT drives, good for small-medium servers  
**Con**: Comparative ROI of media over time is expensive  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

<table>
<thead>
<tr>
<th>LTO-1</th>
<th>$2,800</th>
<th>$35</th>
<th>100/200GB</th>
<th>$0.21</th>
<th>$4,900</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTO-1</td>
<td>$2,800</td>
<td>$35</td>
<td>100/200GB</td>
<td>$0.21</td>
<td>$4,900</td>
</tr>
</tbody>
</table>
| Pro: Great cost/GB, good for for mid-high end servers, & for streaming  
Con: Newer tech, hard to find media, not good for start/stop  
**LTO-1**: Great cost/GB, good for for mid-high end servers, & for streaming  
**Con**: Newer tech, hard to find media, not good for start/stop  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

<table>
<thead>
<tr>
<th>LTO-2</th>
<th>$3,800</th>
<th>$75</th>
<th>200/400GB</th>
<th>$0.22</th>
<th>$6,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTO-2</td>
<td>$3,800</td>
<td>$75</td>
<td>200/400GB</td>
<td>$0.22</td>
<td>$6,000</td>
</tr>
</tbody>
</table>
| Pro: Fair costs, good future growth direction for changers, & for streaming  
Con: Expensive drive, newer tech, hard to find media, not good for start/stop  
**LTO-2**: Expensive drive, newer tech, hard to find media, not good for start/stop  
**Price per GB of the media is probably the most important figure to look at. But keep in mind that newer more expensive tape prices (such as AIT-3 and LTO) often come down quickly 2-3 years after introduction. Plan ahead.**

© Wiley, 2004
What We're Covering...

- Types of System Backups
- Types of Database Backups
- MySQL Database Backup Agents and Tools
- Backup Strategies
- Production Backup and Restoration Issues
Types of Database Backups

Problem:
There are not many backup suites out there that know how to properly backup all MySQL table types, and guarantee backups of only fully committed transactions; not to mention just how to get good copies of your table space in general. Most real solutions rely on native tools, utilities and creative methodologies.

Types of MySQL Backup Solutions:
Of the various backup solutions available, they can mostly be generally categorized into two camps:

- Dump Backups
- Raw Backups

Those two each have on line and off line variants.
Types of Database Backups
Dump Backup Types

A dump backup, usually using `mysqldump`, is done by dumping individual rows, tables or DBs out to a filesystem level flat file, or directly to a backup device.

**Dump Pros:**
- *Simple*: to do and integrate into existing pre-backup scripts
- *Open*: Backups can be manually edited or grep'ed for data
- *Flexible*: Good for migrating between other SQL engines
- *Universal*: Good for all table types (and for converting types)
- *Remote*: Can be easily configured to dump over the network

**Dump Cons:**
- *Fat*: Can require > space/backup than DB files themselves
- *Slow*: Much slower backup & restores than RAW based
- *Limited*: to smaller DBs
Types of Database Backups
Raw Backup Types

**Raw backups** work by grabbing the DB files at the binary filesystem level. **Hot online raw backups** are done by locking, flushing, and cloning the DB files in some way, or via LVM/snapshots. **Warm onlines** are done by breaking a replicated slave and backing it up. **Cold off line backups** are done by simply shutting off the DB and copying.

**Raw Pros:**
- *Faster:* since working at the filesystem level
- *Hefty:* can handle larger databases
- *Powerful:* Can be implemented with online LVM snapshots
- *Friendly:* to most table types and normalized designs

**Raw Cons:**
- *Limited:* to in place restores or same DB table/ver. migrations
- *Static:* Backups are not as easily editable or customizable
What We're Covering...

- Types of System Backups
- Types of Database Backups
- MySQL Database Backup Agents and Tools
- Database Backup Strategies
- Production Backup and Restoration Issues
Backup Agents and Tools
Dump and Raw Tools

Dump Based Backup Tools:
Since the dump method of backups is considered a universal legacy method of doing database backups, there is really only one tool that does this in MySQL.

- `# mysqldump --opt database > database.sql` (online)

Raw Based Backup Tools:
When looking at raw backup methods and tools, there are several ways to grab the binaries out from under the database, each with its own issues.

- `# mysqlhotcopy database /tmp` (online)
- LVM/snapshots or `mysqlsnapshot` (online)
- Backup agents (InnoDB, Arkia, True Image, etc)("hot" online)
- Slave replication backups (warm online)
- Service stop and copy (off line)
Types of Database Backups
Which to use?

Q: Which database backup method should you use?

A: First ask yourself these simple pre-requisite questions:
  Q: Do I require online backups?
    - If No: Use offline backups, dump, or hotcopy solution
    - If Yes: Ask next question:
  Q: Can I afford any interruption (even brief) in write access?
    - If Yes: Use dump, hotcopy, or snapshot solution
    - If No: Next...
  Q: Am I using InnoDB table types?
    - If Yes: Good, use ibbackup or snapshots for “hot backups”
    - If No: Convert to InnoDB table types w/ibbackup or snapshots
Backup Agents and Tools

Dump: Using `mysqldump`

The most time-tested way of doing online MySQL DB backups, the `mysqldump` command automatically locks and flushes your tables, and saves the output to a SQL flat file that you can easily restore from. Good for all table types.

**Dump Examples:**

- **Local backup of three tables from the mydb1 database:**
  
  ```
  # mysqldump --opt mydb1 users addresses cars>mydb1-tables.sql
  ```

- **Local backup of two databases:**
  
  ```
  # mysqldump --opt --databases mydb1 mydb2>mydb1-n-2.sql
  ```

- **Local backup of all databases:**
  
  ```
  # mysqldump --opt --all-databases > all-your-dbs.sql
  ```

- **Remote "pull backup" of all dbs on host dbserver.example.com**
  
  ```
  # ssh root@dbserver.example.com "mysqldump --opt --all-databases" > dbserver-backup.sql
  ```
Backup Agents and Tools
Dump: Tips Using `mysqldump`

- For backup automation, create a dedicated mysql backup/restore user (or use root) at the OS level and put their mysql based username/password in their `~/.my.cnf` file.

- Grant your backup/restore user (in mysql.user) permissions to at least `SELECT`, `RELOAD`, `FILE` (for backups), and `INSERT`, `CREATE`, `DROP`, `INDEX` (for restores).

- Enable binary logging (in `/etc/my.cnf` add `bin-log` under `[mysqlld]`) to retain at least two full backups worth of binary logs and perform daily flushes (or flush daily via backup).

- `--opt` is on by default in =>4.1 and includes `--add-drop-table --add-locks --all --extended-insert --quick --lock-tables`

- For getting inter-DB referential integrity when backing up all databases, use `--all-databases` along with `--opt`
For incremental dump based backups and guaranteeing all DBs and their binary logs are in sync, use both --opt and --flush-logs (with log-bin enabled). Only this will allow "point-in-time recoveries".

For InnoDB table types (only), use the --single-transaction (MySQL>=v4.0.2). This will allow for "Hot Backups" with full r/w ("lock-less") access during the dump. As of 4.1.8, use with the --master-data option for point-in-time recoveries.

For large DBs (<v4.1), use --quick and --extended-insert, or in =>4.1 just use --opt (default).

In Windows use the --result-file=db.sql option instead of >

For faster restores on single, very large DBs, dump with the -T option to save out to a separate paired table.sql (structure) and table.txt (TSV data) files. Then for fast restores use:

```
# mysql < table.sql && mysql -e LOAD DATA INFILE table.txt
```
Backup Agents and Tools

Raw: Using `mysqlhotcopy`

If you're only using ISAM or MyISAM table types, `mysqlhotcopy` is recommended over dump. It's faster, simple, has smaller output, and yields "drag and drop" restores.

**Raw `mysqlhotcopy` Examples:**

- **Local backup of just the users table from the mydb1 DB:**
  ```
  # mysqlhotcopy --allowold mydb1./users/ /root/mysql-backups/
  ```

- **Local backup of whole mydb1 DB to /root/mysql-backups/:**
  ```
  # mysqlhotcopy --allowold --regexp=mydb1 /root/mysql-backups/
  ```

- **Local backup of all DBs and keep old copy:**
  ```
  # mysqlhotcopy --keepold --allowold --regexp=".*" /root/mysql-backups/
  ```

- **Remote backup and DL of all DBs on dbserver.example.com:**
  ```
  # ssh root@dbserver.example.com "mysqlhotcopy --allowold --regexp='.*' /root/mysql-backups/" && scp -rp root@dbserver.example.com:/root/mysql-backups/* /root/mysql-backups
  ```
Backup Agents and Tools
Raw: Tips Using mysqlhotcopy

- Use `--allowold` to move the old backup out of the way (instead of overwriting) until the new backup is verified good.

- Use `--keepold` to keep the previous backup after current one succeeds. Backup DB directories receive the `_old` suffix.

- To skip the backing up the full index files (.MYI), use the `--noindices` option.

- If running binary logs, use the `--flushlog` option to flush them after the tables are locked.

- For automating password-less and secure network backup/downloads, set up ssh key based authentication.*

* For detailed directions on setting up key based ssh authentication, just google for `+ssh-keygen +bsdcow`. 
Backup Agents and Tools
Raw: LVM Snapshots

Any disk based table type can be backed up with snapshots. The external tools that can be used to accomplish this includes LVM/filesystem level snapshots, the tool `mysqlsnapshot`, as well as commercial backup tools such as True Image and Veritas Filesystem.

- If backing up InnoDB table types, LVM/filesystem snapshots are fine without flush locks. The files on the disk are consistent and can be rebuilt from the *undo* and *redo logs*. Thus, InnoDB LVM/filesystem snapshots can be easily scripted or put in a cronjob:

- **Filesystem LVM Snapshots (XFS on LVM2) w/InnoDB tables:**
  
  ```
  # xfs_freeze -f /var/lib/mysql \\
  && lvcreate -L 500M -s -n snap /dev/vgmysql/prod \\
  && mount -o nouuid,ro /dev/mapper/vgmysql-snap1 \\
  /var/lib/mysql-snap \\
  ;xfs_freeze -u /var/lib/mysql \\
  && târ czvf /dev/tape /var/lib/mysql-snap \\
  && umount /var/lib/mysql-snap \\
  && lvremove -f /dev/vgmysql/snap
  ```
Backup Agents and Tools
Raw: Snapshot Gotchas

If backing up or scripting the backup of InnoDB table types, any snapshots are fine since the files on disk can always be considered consistent.

However, when performing LVM/filesystem snapshots on My/ISAM tables, you must first issue a FLUSH TABLES WITH READ LOCKS before starting the snapshot and then an UNLOCK TABLES after the snapshot is complete:

- **Any Snapshots with My/ISAM table types:**
  
  
  | FLUSH TABLES WITH READ LOCK |
  | do snapshot here             |
  | UNLOCK TABLES               |

After the snapshot is complete, then you can mount the snapshot and back it up to your backup media.
The `mysqlsnapshot` perl script* is designed as way of easily backing up My/ISAM tables (only) on the same filesystem, especially for creating slaves. It will tar up all the DBs, bin-logs, and even gzip them by individual DB names.

**For using mysqlsnapshot (My/ISAM only) and taring to tape:**

```bash
# mysqlsnapshot -u root -p pass -s /root/mysql-snap/ -f -n --split \
&& tar cvf /dev/tape /root/mysql-snap \
&& echo All Done :\)
```

* To download the `mysqlsnapshot` script, go here:
http://jeremy.zawodny.com/mysql/mysqlsnapshot/
Here are a couple of popular commercial Raw backup tools:

- **Filesystem Image & Incremental snapshots**
- **Like "Ghost" on steroids**
- **Partnered with MySQLAB**
- **Windows & Linux**

- **Designed for MySQL by creator of InnoDB tables**
- **Backups:**
  ```
  # ibbackup /etc/my.cnf /etc/ibbackup.cnf
  ```
- **Restores:**
  ```
  # ibbackup --restore /etc/ibbackup.cnf
  ```
- **For "Hot Backups" on all filesystems and tables**
What We're Covering...

- Types of System Backups
- Types of Database Backups
- MySQL Database Backup Agents and Tools
- Database Backup Strategies
- Enterprise Backup and Restoration Tips
Database Backup Strategies
Replication Based Backups

Many people who are already running replication servers just find it easier to go ahead and configure one of the replication slaves as a backup server also.

Tip:
Just be sure that you always save your replication files like master.info file, your relay logs, relay index, etc. You'll need them if you have to reconfigure after a restore.
Database Backup Strategies
Centralized Network Backups

1. Centrally Controlled Remote DB Copies
   
   ```
   # ssh root@$servername "mysqldump --allowold --regexp='.*' /root/mysql-backups/
   ```

2. Centralized Pull of DB Copies
   
   ```
   && scp -rp root@$servername:/root/mysql-backups/* /root/$servername-backups$(date +%Y-%m-%d)
   ```

3. Centralized Backup to Tape
   
   ```
   && mt --rewind ; tar czvf /dev/tape /root/*/backups$(date +%Y-%m-%d)
   ```

Centralized Backup Server

If you have many database servers, and only one backup server, it will often behoove you to centrally automate your DB server backups:
1) Centrally trigger the MySQL dump or raw copy to disk
2) Pull all of your database content to your backup server
3) Backup all new data to tape
MySQL Hot Backup API?

Going beyond MySQL v4.x, MySQL is developing a full online MySQL Backup API. This is currently being tested for MySQL 5.1, and when implemented will handle “Hot Backups” on all table types!

Watch for news of inclusion of a libmysqlbackup.so library by Brian Aker!
What We're Covering...

- Types of System Backups
- Types of Database Backups
- MySQL Database Backup Agents and Tools
- Backup Strategies
- Enterprise Backup and Restoration Tips
Enterprise Backup and Restoration Tips

- Always **test your backups** via restores after even after minor changes.
- Monitor backup success/failures with success notifications.
- Rotate tape pools weekly, retaining a copy of your last full.
- Always buy media before you need it! Build servers with TCO figures including annual tape/media replacement figures.
- Keep emergency boot media and special backup hardware drivers physically with the server.
- Clean tape drives regularly and always keep extra cleaning cartridges on hand.