*LA-UR-11-05005*
*Approved for Public release*
*Distribution is unlimited.*

# U.S. Department of Energy Best Practices Workshop on File Systems & Archives San Francisco, CA September 26-27, 2011 Position Paper

**M'hamed Jebbanema**
Los Alamos National Labs
mjebb@lanl.gov

## ABSTRACT / SUMMARY

This paper will discuss a strategic and automated scripted solution to ensure High Performance Storage System (HPSS) metadata integrity, availability and recoverability in the event of a disaster.

## INTRODUCTION

An essential component of High Performance Storage System (HPSS) is the metadata and tools to manage and retrieve the metadata. Metadata can be described as the DNA of the storage system as metadata defines the elements of the transactional data and how they work together. Any loss of metadata proves to be disastrous to data integrity**.**

In addition to implementation of resilient and fault tolerance hardware and software best practices, we must guarantee the high availability and recoverability of metadata.

Since HPSS uses IBM DB2 as its metadata management system, manual and conventional DB2 standard backups and lack of logs archival monitoring tools dramatically increase administrator workload and probability of data loss.

A Perl language based comprehensive DB2RS (DB2 Recovery Solution) delivers a robust, configurable, and customizable recovery management with minimal efforts.

The Scheduler, Backup, Verifier and Checker(s) services work intrinsically in a holistic approach by using SQLite database as a central repository for all DB2RS activities.

This presentation will cover the design, and integration of DB2RS to ensure metadata integrity and recoverability when disaster occurs.

## Paper Content

In order to achieve transaction integrity and zero or little data loss , DB2RS makes automated scheduled local backups including logs to different media combined with an-offsite hosting similar backups in which to restore from in the event of a disaster.

**Goal:**

Develop a metadata backup/recovery solution that is simple, customizable, robust, and easy to maintain**.**

**Objective:**

Provides recoverable copy of databases.
Metadata can be recovered to any Point In Time (PIT).
Guarantees transactional consistency.

**Purpose:**

Develop scripts that would leverage DB2 integrated utilities and tools to automate all functions ensuring metadata recoverability.

## 1.Design

### 1.1 Logging

Proper DB2 log file configuration and management is an important key to data stewardship and operational availability.

All database changes (inserts, updates, or deletes) are recorded in the DB2 transaction logs. Transaction logs are primarily used for crash recovery and to restore a system after a failure.

DB2 does have the ability to perform dual logging on different volumes as well as different media thereby increasing redundancy for both active logs and archive logs.

We Configure DB2 log sets by implementing MIRRORLOGPATH and dual log archives where each active & archive log set on independent LUN that uses separate physical disks and different type media (TSM) (Figure 1&2).

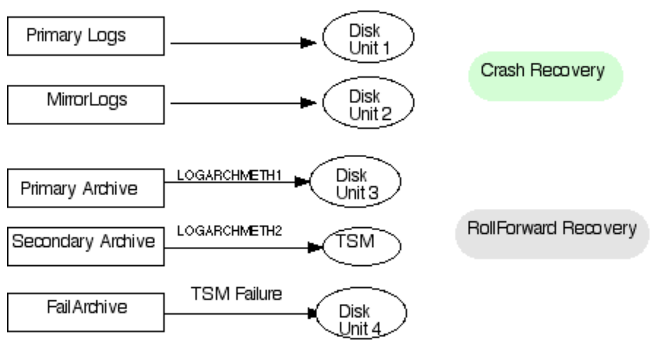
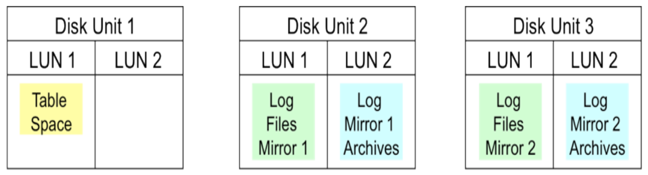


Figure 1. Logging Scheme



- Disaster recovery: Excellent. Short of fire, flood, etc., DB2 can always be recovered.
- Operational availability: Excellent, as good as you can get without going to high availability DB2s. Only interruption would be loss of Disk Unit 1 or simultaneous loss of Disk Unit 2 & 3

Figure 2: Metadata Hardware Fault Tolerance Implementation

## 1.2 Backup service

The scope of this design goes beyond the process of backing up objects to disks or tape but rather encompasses several functions that ensure the validity and integrity of the backups.

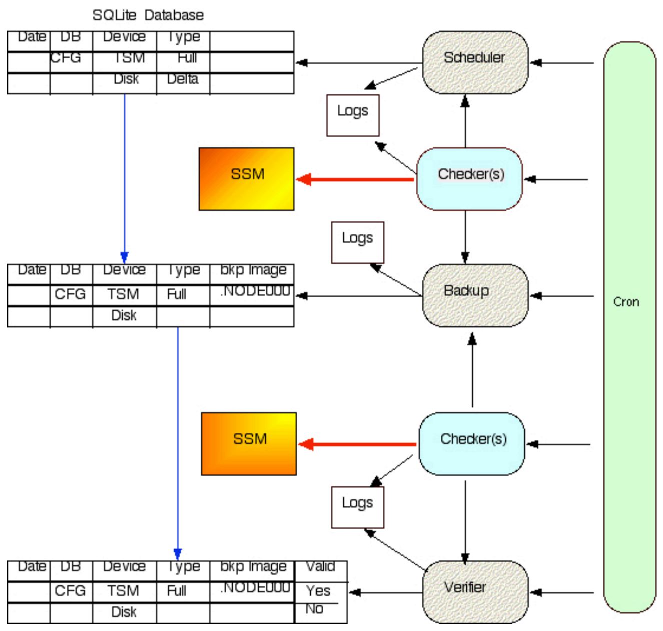


Figure 3: Overview of DB2RS

Perl Language was an evident choice to automate these tasks in order to take advantage of our locally developed Perl modules and for backward compatibility reasons.

Scripts were structured based on type of service or function and SQLite database (Example 1), an open source, self-contained, embeddable, zero-configuration was chosen as a central repository for all services activities.

Four types of services (Figure 3) were automated and can be run either via cron (Example 2) or manually.

Scheduler : Schedules backup in SQLite database.
Backup : Checks SQLite for scheduled backups and perform backup service.
Verifier : Validate the integrity of backups.
Checker & High Level Checker : perform diagnostics, sanity checks, monitoring and error reporting

Each service has multiple parameters entries in configuration (Example 3) file subject to customization based on disaster recovery requirements.

To prevent invalid data and allow synchronization among all services, applications state and locks were included in the initial design.

All services activities are captured in a centralized log and a man page was integrated into the code for quick reference. (Example 4)
Other useful utilities were coded and added to the mix to ensure completeness and automation of DB2RS.

Example 1: SQLite database service activities

```
20110816000601    CFG  TSM   FULL       1
20110816041003  Verified
20110817000306  SUBSYS1  TSM  INCREMENTAL  1
20110817051002  Verified
20110817000609  CFG      TSM  FULL         1
20110817045902  Verified
20110818000301  SUBSYS1  DISK  FULL         1
20110818045903  Verified
20110818000606  CFG      DISK  FULL         1
20110818041032  Verified
20110819000301  SUBSYS1  DISK  INCREMENTAL  1
20110819045902  Verified
```

Example 2: Services scheduled via cron

```
06 0 * * 1,2,3 schedulme -cron -db cfg -tsm -full -sessions 1
10,59 4,5 * * * bkp -cron > /dev/null 2>&1
02 6,7 * * * bkpv -cron > /dev/null 2>&1
0,41 * * * *  bkplogtrim -cron > /dev/null 2>&1
09 12 * * 4   checker -cron > /dev/null 2>&1
09 12 * * 1-3,5 checkerhl -cron -disk > /dev/null 2>&1
23 * * * *  ensure_archlogs -cron > /dev/null 2>&1
```

Example 3: DB2RS configuration file

```
[timemachine]
# bkp: should we make another bkp if one full bkp already exists
within bkpDiffHrs (integer hrs);
bkpDiffHrs = 20
#bkpv: looks back in SQLite for unverified bkp images earlier
than diffWkBkpv;  24hrs
diffWkBkpv = 86400
#checkerhl & checker: each service must have run successfully in
the last (n) days; 3 day
ChkDysServSuc = 259200
#chekerhl: calculate timestamp before which combo bkps and logs
should exists; 1wk
ChkhlWks   = 604800
# checker: all services Service must have run within; 8 hrs
ChkHrsSerbkp  = 64800
ChkHrsSerbkpv  = 28800
ChkHrsSersched  = 28800
#checker: check bkp service progress run..(avoid runaway and
hangs)..service must not be running
# for more than ChkHrsRunbkp; 2 hrs
ChkHrsRunbkp = 7200
# checker: if no bkp within ChkHrsBkpSched after being
scheduled; 20 hrs
 ChkHrsBkpSched  = 64800
 # checker: stats Failover paths for existence of logs (any files)
 within the last ChkHrsFailover; 1 hr
 ChkHrsFailover  = 3600
 # ChkHrsFailover  = 0 # test only; make sure checks paths
 immediately..no delays
 # checker: row created in Sqlite for each db combo int he last
 ChkWksSqlite; 1wk
#ChkWksSqlite  = 604800
 ChkWksSqlite  = 3600
[constantvars]
DB2DIR = /opt/ibm/db2/path
[db2]
databases = cfg, etc
[db2:cfg]
images = /usr/db2/image/path
archlogs = //path/path/etc..
failarchlogs = /usr/db2/path/path/etc….
imagespct = 60
archlogspct = 60
tsmstartdate = 20110202
```

Example 4: Man page

*ENSURE_ARCHLOGS(1)    User Contributed Perl Documentation   ENSURE_ARCHLOGS(1)*

*NAME*

*ensure_archlogs - Ensure that database has truncated logs recently*

*SYNOPSIS*

```
ensure_archlogs <options>
Within root's or instance owner crontab...
01 * * * * /lanl/hpss/path/db2rs/ensure_archlogs -cron
On the command line as root or instance owner...
# ensure_archlogs -force  Force a log archive right
  now
# ensure_archlogs -force=2h  Archive if not performed
   in last 2 hours.
# ensure_archlogs  Same, but use default intervals from
  the dbconfig
# ensure_archlogs -disable=1h   Disable scripts in cron
   for 1 hour
# ensure_archlogs -enable      Re-enable scripts in cron
# ensure_archlogs -help       Show the synopsis for  this
   script
# ensure_archlogs -man     Show the man page for this
   script
```

*DESCRIPTION*

*This should generally be run via cron.  It makes sure that DB2 has archived a log within a certain amount of time for  each HPSS database.  If it hasn't it tells it to do that with the "db2 archive log" command.  This limits the exposure of*

*Un-archived logs to a certain period of time while also allowing minimum impact on DB2. This should probably not run more frequently than one hour.*

## 1.3 Checker(s) and error reporting service

Applies to Logs and backups.

### 1.3.1 Checker (Example 5)

#### 1.3.1.1 Logs Checks:

Performs the following tasks:
Exclusive analysis utilizing log data mining list of events.
Disk & TSM logging failures detection.
FailArchive path check.
Immediate reporting and notifications to SSM.

#### 1.3.1.2 Backup Checks:

Performs the following tasks:
Sanity Checks
Diagnostic Checks
Immediate reporting and notifications to SSM.

Example 5 : Check output

```
info: No scheduled backup found at this time!
info: checker: No DB2 Logs in Failover paths...good thing
info: checker: Found all 4 scheduled combination backups in
Sqlite database..schedulme is working fine
info: checker: Last successfull bkp run at 20110817055901
info: checker: Last successfull bkpv Run at
20110817071029
info: checker: Last successfull schedulme run at
20110817000609
info: checker: bkp service has run within defined time (Hrs).
info: checker: bkpv service have run within defined time
(Hrs).
info: checker: schedulme service have run within defined
time(Hrs).
```

### 1.3.2 High-level Checker (Example 6)

Performs the following tasks:
Recent backup for each (database, device) pair completed successfully.
Recent TSM backup for each database must exist and verified
Ensure that TSM copies of all logs since the last verified TSM backup exists.
Immediate reporting and notifications to SSM.

Example 6: High Level Checker (Checkerhl) output

```
info: Found 56 logs for CFG DISK backup taken at
20110813041003 (2314 - 2369)
notice: Everything looks good for CFG DISK backup taken
at 20110813041003
info: Found 99 logs for SUBSYS1 DISK backup taken at
20110811045903 (3116 - 3214)
notice: Everything looks good for SUBSYS1 DISK backup
taken at 20110811045903
info: Found 96 logs for CFG TSM backup taken at
20110810041003 (2274 - 2369)
notice: Everything looks good for CFG TSM backup taken at
20110810041003
info: Found 146 logs for SUBSYS1 TSM backup taken at
20110808045903 (3069 - 3214)
notice: Everything looks good for SUBSYS1 TSM backup
taken at 20110808045903
info: Found all 4 backup combinations
```

## 1.4 Error Reporting to SSM (HPSS interface):

Exit codes are called to generate sub-class of errors based on custom binary scheme for multiple error reporting.
To simplify error reporting and monitoring, three (3) categories were considered to match HPSS error reporting style.

Minor
Backup, verifier, or scheduler service did not run within defined time (Hrs).
Backup have exceeded estimated allowable time to successfully complete backup.

Check ASAP (considered critical)
Failure to schedule expected pair (db,device) within the last (days).
TSM backups and associated logs are not found.
One or more services failed in the last (n) days.

MAJOR
Scheduled Backups are behind schedule.
Backup service did not run.
Backup failed (n) times as specified in cron.
Backup could have completed successfully but took longer than expected/estimated.
Failover DB2 log paths contain logs.
Filesystem(s) error.
Log(s) script failure –internal error-.
Log archiving failure: Disk or/and TSM.

## 2. CONCLUSIONS

DB2RS not only adds value to HPSS native metadata integrity monitoring and reporting tools but also ensures that our operational staff are monitoring the health and status of the metadata and thus reducing dramatically the risk of loss of data and respectively the time of recoverability.