AIX Mirror Write Consistency with Oracle Databases

By
Walter Orb
IBM/ERP Advanced Technical Support

Notices

Published in August 1999

Please send comments via E-mail to:

Walter Orb
Internet id: orb@us.ibm.com

Acknowledgments

Thanks to the following people for their contributions and support:

- John Aschoff, IBM Storage System Division Systems Performance Evaluation and Design
- Bill Britton, IBM AIX Performance Design Analysis
- Steve Dibbell, IBM RS/6000 ISV Technical Support
- Dale Martin, IBM Oracle Competency Center
- Gerald McBrearty, IBM AIX Filesystems Development
- Kenneth Rozendal, IBM AIX Kernel Development Service and Support
- Johnny Shieh, IBM AIX Filesystems Development
- Bruce Spencer, IBM RS/6000 Marketing
- Kothanda Umamageswaran, Oracle Parallel Server Development

Introduction

The objective of this paper is to describe the Mirror Write Consistency attribute of AIX mirrored logical volumes and it's implications for Oracle databases. The following applies only to systems where the AIX Logical Volume Manager functionality of software mirroring is used to protect database files against disk failures. Please refer to the standard AIX documentation for a more detailed description of the commands used and presented in this paper.

What is Mirror Write Consistency (MWC)

Mirror Write Consistency is an attribute of an AIX logical volume (LV) which can be set to either YES or NO. It's meaningless unless the logical volume is mirrored.

What problem does MWC solve

A write is in progress to a logical volume that has mirror copies. Before the write completes, power is lost so that the write never completes. When the system reboots, as with an un-mirrored logical volume, the data read from the area to which the unfinished write was addressed may be old data, new data or a mixture of the two. The difference that mirroring introduces is that if the data is read twice the two reads may give different results. This is because the two reads may read from different copies of the data. In many environments this is not a problem but some applications would be sensitive to this and this is the reason that MWC has been provided.

How does MWC work

Near the beginning (outer edge) of each disk AIX keeps the mirror write consistency cache record. This record tracks the last 62 Logical Track Group (LTG) writes to mirrored logical volumes. The size of a Logical Track Group is 128 KB.

When a write is performed to a mirrored logical volume which has MWC turned on, the LVM first checks the in memory version of the MWC record. If this results in new LTG entries being added to the MWC cache record, a parallel write of the MWC cache record is scheduled to each disk in a volume group, marking the new LTGs as out of sync. Once the write of the MWC record to one of the disks that holds a new LTG completes, LVM will schedule the actual write request to that disk. LVM will only send a write completion to the requesting application when the data is safely written to all mirror copies on disk.
The write of the MWC record involves a seek to the beginning of the disk. After the write of the MWC record completes, the actual mirror writes takes place which involves a seek back to the data area of the disk.

When the write of the mirrored data completes, LVM does not immediately update the MWC record on disk. This can be handled later since it is not dangerous to have areas marked as out of sync when really they are in sync. If the system should crash, when it reboots, the MWC record on each disk is examined and if any LTGs are found to be marked out of sync they are synchronized by reading one copy and writing it to the other. The copy to read is chosen randomly since each copy of the data is equally valid (or invalid).

Performance Implications of MWC

If MWC is turned on for a logical volume, performance of writes to that logical volume will be impacted because each write will result in a seek to the beginning of the disk and a write of the MWC cache record before seeking to the data to be written. If it is determined that MWC can be turned off, a performance gain can be realized for logical volumes with large amount of writes.
If the application is doing sequential writes and the record sizes are less than 128 KB, then MWC does not require a write (or seek) for every data record write. It only needs to update the MWC record once for each 128 KB written.
If a write cache is available on the path to the physical disk, the writes to the MWC record will most likely only go to the write cache. This will avoid the extra seek and the performance effects of MWC are minimal. IBM Enterprise Storage Server (ESS) and Versatile Storage Server (VSS) are disk subsystem with write cache. Some SSA adapters can be also configured with a nonvolatile fast write cache. These SSA adapters are currently not supported in HACMP configurations, but are planned to be supported at a later date.

Oracle Recovery

After a system failure Oracle has to do a crash recovery before the database is opened. During crash recovery the Oracle redo logs are read to identify data blocks for which a write request has been issued but a write confirmation had not been received prior to the crash. These dubious blocks are read from disk and checksum information stored in the data block is compared to checksum information in the redo log entry. If the checksum information does not match, the data block on disk is not correct. In this case, Oracle will rebuild the data block from redo log entries and rewrite the data block to disk. If the checksum process verifies that the data block is already consistent, the crash recovery process proceeds to the next dubious data block without rewriting the data block to the physical disk.

Potential problems with MWC off for filesystems or logical volumes with Oracle datafiles

Let's assume that an Oracle data file is allocated on a logical volume with 2 copies, MWC is turned off, and the 2 copies are indeed out of sync after a system crash. Oracle reads a dubious data block during crash recovery and the Logical Volume Manager reads by chance the copy with the good data, where the latest updates to the data block were already written to the physical disk. This data block will pass the checksum

processing and Oracle will skip to the next block. This means that the 2nd "bad" copy on disk is never updated. Now the two physical copies are still out of sync even after crash recovery. Once the database is opened there is a chance that during normal processing the "bad" copy is read, resulting in database inconsistencies or corruption. Therefore it is not possible to turn off MWC for Oracle datafiles without taking additional measures to provide synchronized copies of a logical volume before the Oracle crash recovery process is started.

Oracle Resilvering

Oracle is developing an enhancement to the crash recovery process called "resilvering", which allows for safely turning off MWC for Oracle datafiles without the risk of database inconsistencies or corruption. With the resilvering enhancement Oracle will determine during crash recovery whether a data file is allocated on a logical volume with multiple copies. In this case all copies of a dubious data block are read and go through the checksum processing. If one of the copies succeeds this test it is used to update any other copies failing the test. If all copies fail the test, the block will be rebuilt based on the redo log information and then written out thereby updating all copies of the logical volume. The resilvering process requires the capability to read individual copies of a mirrored logical volume, which is only supported for raw logical volumes. Oracle plans to support the resilvering enhancement with Oracle8i Release 8.1.6. Customers who plan to take advantage of the new resilvering function should contact Oracle support to confirm that the function is included once 8.1.6 has been released.

Until the resilvering enhancement becomes available, one of the methods described in the following chapters has to be used to synchronize a logical volume with MWC off, before Oracle is started after a system crash.

Alternative #1: Synchronize the whole logical volume before Oracle crash recovery

The AIX `syncvg` command can be used to synchronize the mirror copies of a logical volume. To synchronize all partitions of a logical volume the `–f` option must be used on the syncvg command. The `–l` option can be used to specify the logical volume name for synchronization, which also allows for running multiple `syncvg` commands in parallel. In addition the `–P` option allows for parallel processing within a single syncvg command. The optimum setting of this parameter is dependent on your environment and should be derived with a couple of tests. For example `-P 6` has been found to be a good starting point in many environments.

Let's assume you decide to turn off MWC for performance reasons on the filesystems /oracle/data10 and /oracle/data11 (logical volumes lvdata10 and lvdata11 respectively). You will then also have to turn off the automatic mount after reboot option of those filesystems (this is obviously irrelevant for database datafiles on raw logical volumes). For normal operations, these filesystems have to be mounted manually after each reboot.

After a system crash, the logical volumes must be manually synchronized before mounting the filesystems:

- Manually synchronize the logical volumes with the `syncvg` command

```
syncvg –f –P 6 –l lvdata10 &
syncvg –f –P 6 –l lvdata11 &
```

- Mount the filesystems after the `syncvg` commands complete

```
mount /oracle/data10
mount /oracle/data11
```

- Startup the database

The synchronization throughput rate of a single logical volume depends mostly on the data rates of the physical disk and the architecture of the disk attachment. If you synchronize multiple logical volumes in parallel, you might also reach the throughput rate of the disk adapter. Consequently the parallel synchronization jobs should be spread over as many adapters and disks as possible. It is advisable to keep the size of the filesystems small. For example if you assume a synchronization rate of 5MB/sec for a single logical volume, it would take about 7 minutes to synchronize a 2GB filesystem. If the biggest filesystem were 40 GB, then the minimum runtime would be at least 140 minutes. If the same data would be spread over 10 filesystems each 4GB of size, you might be able to synchronize the 40GB of data in a little over 15 minutes by running 10 `syncvg` jobs in parallel (assuming that these filesystem are spread over several adapters and there is no other contention while running the 10 jobs in parallel).

Alternative #2: Remove mirror copies before Oracle crash recovery

An alternative solution for providing a consistent view to the Oracle crash recovery process for logical volumes with MWC off is to remove the mirror copies of a logical volume before you mount the filesystem and startup Oracle. Then the mirror copies can be rebuilt while the database is up and running. This will allow for much sooner access to the database after a system crash, however there is the risk that (until all the mirror copies have been rebuilt and synchronized) some logical volumes would be unprotected against disk failures. In case of a disk failure you would have to restore and recover the lost datafiles.
The following is a description of a potential scenario using this method. The filesystem `/oracle/data10` on logical volume `lvdata10` is used as an example to illustrate some of the necessary commands. The chapter “Sample Korn shell script” shows how these steps could be potentially implemented in a shell script to automate most of the tasks.

1. Check the system for stale partitions (see the chapter “Check state of MWC and stale partitions” for details about the checkmwc script):

   ```
   checkmwc  >  mwc.out

   awk 'NR <= 2; /stale/' mwc.out
   ```

   Logical volumes with MWC off and stale partitions should be synchronized first (using the `syncvg` command), as in this case you can’t safely remove mirror copies.

2. Create a map of the current allocation of the logical volume and build a map file for the mirror copy of the logical volume. This step will enable you to rebuild the logical volume copy on the same disks exactly as before.

   ```
   lslv –m lvdata10 | tee lvdata10.map |  \
   awk '/hdisk/ { printf( “%s:%s\n”, $5, $4 ) }' > lvdata10.pp2.map
   ```

   `lvsapdata10.map` will then contain the current layout of the logical volume, `lvsapdata10.pp2.map` is a map file which will be used as input for the AIX mklvcopy command. The output files should look like:

   ```
   lvdata10.map:

   lvdata10:/oracle/data10
   LP    PP1  PV1               PP2  PV2               PP3  PV3
   0001  0126 hdisk20          0126 hdisk23
   0002  0127 hdisk20          0127 hdisk23
   0003  0128 hdisk20          0128 hdisk23
   0004  0129 hdisk20          0129 hdisk23
   ```

```
lvdata10.pp2.map:

hdisk23:0126
hdisk23:0127
hdisk23:0128
hdisk23:0129
```

You should verify the contents of these files before you proceed to the next step.

3. Remove the logical volume mirror copy using the AIX `rmlvcopy` command:

```
rmlvcopy lvdata10 1 hdisk23
```

4. Verify that the mirror copy was removed (check the value of the COPIES attribute in the output of the following command):

```
lslv lvdata10
```

5. Mount the filesystem:

```
mount /oracle/data10
```

6. Startup Oracle:
   At this time Oracle will run the normal crash recovery. After the completion of crash recovery, the database will be available for online use.

7. Create a stale mirror copy of the logical volume using the map file created in step 2:

```
mklvcopy –m lvdata10.pp2.map lvdata10 2
```

   This command will only allocate the physical partitions needed for the mirror copy, it will not synchronize the data. This allows us to use `syncvg –l` later to synchronize multiple logical volumes in parallel. You could use `mklvcopy` with the `–k` option to automatically synchronize the new partitions during the creation of the mirror copy, but you can run only one `mklvcopy` at a time (obviously it wouldn't make any difference in this example, however in a real scenario with multiple logical volumes with MWC off, the ability to run multiple `syncvg` commands in parallel would speed up the total synchronization process).

8. Verify that the create mirror copy was successful (check the value of the COPIES attribute again):

```
lslv lvdata10
```

9. Synchronize the new mirror copy of the logical volume:

```
syncvg –f –P 6 –l lvdata10
```

10. Verify that all mirror copies were created and that all logical volumes are synchronized (see step 1).

At this time the synchronization process is complete and the database is fully protected against disk failures again.

I/O subsystem performance considerations

It is recommended that MWC be turned on for most logical volumes to insure data integrity and minimize the data synchronization time after a system crash. If this leads to I/O performance problems, the traditional methods of database tuning should be applied first. This includes topics like application design, database

parameterization, datafile layout, etc. The ultimate goals are to avoid unnecessary I/O and then to spread the remaining I/O evenly over all available physical disks. The disk subsystem performance can be monitored with the standard tools like `iostat` and `filemon`. A disk should be investigated more closely if `iostat` constantly shows a disk utilization of more than 15-20%.

`Filemon` can be used to determine the most active logical volumes and datafiles on that disk, the results can then be compared with the Oracle v$filestat statistics for verification. The next step might be to move some heavily used objects from these datafiles out into their own tablespaces and datafiles, or to separate two actively used datafiles on their own disks.

If the database system still suffers from the write performance penalty associated with MWC, you might decide to turn MWC off for selected logical volumes. Remember that Oracle tries to do all writes and some reads (e.g. table scans) asynchronously to the application. Even though iostat shows a high disk utilization rate, this may not be adversely affecting end user or response time performance. If you plan to turn MWC off, first move datafiles with heavy write activity into their own filesystems. Then implement a procedure to synchronize the mirror copies of these filesystems in the event of a system crash. Then, you can safely turn MWC off.

It is recommended to leave MWC on for "read-only" and "read-mostly" filesystems as the MWC setting is irrelevant for reads.

Possible solutions for the necessary synchronization are the forced synchronization of a complete logical volume with the `syncvg -f` command, or to remove the mirror copies and rebuild them after Oracle crash recovery completes. A combination of both methods might also be feasible in certain environments.

To avoid the MWC penalty for I/Os to the Oracle redo logs, you should use multiple redo log members on non-mirrored logical volumes to protect the redo logs.

A common scenario that might provide a good compromise between online performance and downtime associated with a system crash could look like:

- Use multiple Oracle redo log members on non-mirrored filesystems to protect the logfiles
- Turn MWC off for filesystems with datafiles with Oracle rollback segments and use `syncvg -f` after a system crash before the startup off the database
- Turn MWC off for a few filesystems with heavy write activity and use the break/rebuild mirror method to synchronize the data after a system crash
- Leave MWC on for filesystems with datafiles which have light to moderate I/O activity or are predominantly read-only

Alternatives to LVM mirroring

LVM mirroring is an effective means of providing mirroring across storage subsystems to provide availability even in the event of an entire storage subsystem failure. However, if you are using LVM mirroring within a single storage subsystem, there may be alternatives that do not incur the MWC overhead and still avoid the need to resynch logical volumes after a system crash. Some of the possibilities include:

- **Hardware RAID 1 -** The IBM Fibre Channel RAID Storage Server supports hardware RAID 1 (mirroring). Since the mirroring function is performed in hardware, MWC is not required. The CPU overhead of performing LVM mirroring is also offloaded from the host server. This storage subsystem also supports hardware RAID 0 + 1 (striping and mirroring). Striping allows I/O activity for extremely active logical volumes to be spread across multiple physical disks.

- **Hardware RAID 5**– The IBM Advanced SerialRAID Adapter (feature code 6225) as well as the IBM Fibre Channel RAID Storage Server mentioned above support RAID 5 configurations. Also, the IBM Enterprise Storage Server (ESS) and Versatile Storage Server (VSS) are RAID 5 subsystems. RAID 5 provides protection against single disk failures by maintaining redundant parity information.

RAID 5 is often an extremely cost effective data protection solution but has a reputation for performing poorly on write intensive workloads. This reputation has to do with the RAID 5 “write” penalty. In order to write a block of data, the existing data and parity information is read from disk and the updated data and parity information is rewritten to disk, thereby requiring 4 I/Os (2 reads, 2 writes) operations to perform one logical write. Recent technology advances such as read/write cache, advanced sequential write algorithms, and the ability to evenly spread both read and write activity across multiple physical disks have improved RAID 5 performance to the point that it now performs as well or better than RAID 1 for many workloads, despite it’s historical reputation.

Sample Korn shell script

The following script should give some ideas about how to automate most of the break/rebuild mirror procedure. This script is by no means complete and has not been thoroughly tested. Most of the safety checks could and should be enhanced. Most of the manual checking could be automated as well.

```
#!/bin/ksh



function cont {

#this function asks whether to continue or not

   ANS=""
   until [[ "$ANS" = "cont" ]]
   do
      print "\nEnter cont to continue, quit to exit"
      read ANS?"cont/quit? "
      [[ "$ANS" = "quit" ]] && print "Bye bye" && exit 99
   done

   print " ...continuing...\n"
}


#Define array LV for logical volumes with MWC off
set -A LV "lvdata10" "lvdata11" "lvdata12" "lvdata13" \
          "lvdata14" "lvdata15" "lvdata16" "lvdata17"


#and another array with the according filesystems
set -A FS "sapdata10" "sapdata11" "sapdata12" "sapdata13
          "sapdata14" "sapdata15" "sapdata16" "sapdata17


 fs_prefix="/oracle/C21/"
   mwc_out="/tmp/mwc.out"


checkmwc > $mwc_out

print "\nCheck logical volumes"

for lv in ${LV[*]}
do
#check for stale partitions
   line=$( grep $lv $mwc_out )
   if [[ -z $line ]]; then
      print "$lv not found"
   else
      print $line |
         awk '$3 == 1    { printf( "%-16s%s\n", $1, "is not mirrored"    ) }
              $4 == "on" { printf( "%-16s%s\n", $1, "MWC is on"          ) }
              /stale/    { printf( "%-16s%s\n", $1, "has stale partions" ) } '
```

```
   fi
done

print "\nCorrect the above errors first (if any)"

cont


#Create lv maps

for lv in ${LV[*]}
do
   print "Create /tmp/${lv}.map and /tmp/${lv}.pp2.map"
   lslv -m $lv | tee /tmp/${lv}.map |
      awk '/hdisk/ { printf( "%s:%s\n", $5, $4 ) }' > /tmp/${lv}.pp2.map
done

print "Please check the created logical volume maps !!!"
cont

#remove mirror copies
for lv in ${LV[*]}
do
   PV=$( awk -F":" 'NR ==1 {print $1; exit }' /tmp/${lv}.pp2.map )
   if [[ -z $PV ]]; then
      print "Can't find physical volume for $lv"
      print "Please check map file /tmp/${lv}.pp2.map"
   else
      print "Remove mirror copy of $lv from $PV"
      rmlvcopy $lv 1 $PV
   fi
done

#verify that all copies are removed
print "\nPlease check that the Copies attribute is 1 for all lv's"
for lv in ${LV[*]}
do
   lslv $lv | awk '
        /^LOGICAL/  { lv     = $3 }
        /COPIES/    { copies = $2 }
       END         { printf( "%-15s Copies:  %s\n", lv, copies ) }'
done

cont

print "mount filesystems ..."
for fs in ${FS[*]}
do
   path=$fs_prefix$fs
   print "mount $path"
   mount $path
done

print "\n\nPlease startup Oracle"
print "Continue with this script after crash recovery is complete"

cont

#recreate mirror copies
for lv in ${LV[*]}
do
   print "Create mirror copy for $lv"
   mklvcopy -m /tmp/${lv}.pp2.map $lv 2
done

#check that all copies were created
print "\nPlease check that the Copies attribute is 2 for all lv's"
for lv in ${LV[*]}
do
   lslv $lv | awk '
        /^LOGICAL/  { lv     = $3 }
```

```
        /COPIES/    { copies = $2 }
        END         { printf( "%-15s Copies:  %s\n", lv, copies ) }'
done

cont

#synchronize the new partitions
for lv in  ${LV[*]}
do
   print "start syncvg for $lv"
   syncvg -f -P 6 -l $lv &
done

print "\n\nWait until all syncvg jobs are finished"
wait

#Verify that all mirror copies are synchronized

print "\nPlease check that the Copies and State attribute for all lv's"
for lv in ${LV[*]}
do
   lslv $lv | awk '
        /^LOGICAL/  { lv     = $3 }
       /LV STATE/  {  state  = $6 }
        /COPIES/    { copies = $2 }
       END          { printf( "%-15s Copies:  %-2s State: %s\n", lv, copies, state ) }'
done
```

Check state of MWC and stale partitions

The following script can be used to produce a report of all logical volumes in a system, it's associated filesystem, whether the logical volume is mirrored or not, the state of MWC and whether the logical volume is stale.

checkmwc:

```
#!/bin/ksh

print "LV              Filesystem                Copies MWC LV_STATE"
print "=============== ========================= ====== === ============"

export ODMDIR=/etc/objrepos

for lv in $(odmget -q'attribute=lvserial_id' CuAt | awk -F\" '/name/ {print $2}' )
do
   lslv $lv | awk -v lv=$lv '
                /LV STATE/ { state  = $6 }
                /COPIES/   { copies = $2 }
                /MOUNT/    { mount  = $3 }
                /MIRROR/   { mwc    = $4 }
                END        { printf "%-15s %-30s %-s %-3s %-15s\n",
                                    lv, mount, copies, mwc, state }'
done
```

A typically output of that script might look like:

```
LV              Filesystem                Copies MWC LV_STATE
=============== ========================= ====== === ===I========
hd5             N/A                            1 on  closed/syncd
hd6             N/A                            1 off opened/syncd
hd8             N/A                            1 off opened/syncd
hd4             /                              1 on  opened/syncd
hd2             /usr                           1 on  opened/syncd
hd9var          /var                           1 on  opened/syncd
hd3             /tmp                           1 on  opened/syncd
hd1             /home                          1 on  opened/syncd
paging00        N/A                            1 off opened/syncd
jfslog1         N/A                            2 on  opened/syncd
jfslog2         N/A                            2 on  opened/syncd
C21saptrans     /usr/sap/trans                 2 on  opened/stale
C21usrsap       /usr/sap/C21                   2 on  opened/syncd
C21sapmnt       /sapmnt/C21                    2 on  opened/syncd
C21stage_804    /oracle/stage/stage_804        1 on  opened/syncd
C21oracle       /oracle/C21                    2 on  opened/syncd
C21origlogA     /oracle/C21/origlogA           1 on  opened/syncd
C21origlogB     /oracle/C21/origlogB           1 on  opened/syncd
C21mirrlogA     /oracle/C21/mirrlogA           1 on  opened/syncd
C21mirrlogB     /oracle/C21/mirrlogB           1 on  opened/syncd
C21sapdata1     /oracle/C21/sapdata1           2 on  opened/syncd
C21sapdata2     /oracle/C21/sapdata2           2 on  opened/syncd
C21sapdata3     /oracle/C21/sapdata3           2 on  opened/syncd
C21sapdata4     /oracle/C21/sapdata4           2 on  opened/syncd
C21sapdata5     /oracle/C21/sapdata5           2 on  opened/syncd
C21sapdata6     /oracle/C21/sapdata6           2 on  opened/syncd
C21sapdata7     /oracle/C21/sapdata7           1 off opened/syncd
C21sapdata8     /oracle/C21/sapdata8           1 off opened/syncd
C21sapreorg     /oracle/C21/sapreorg           2 on  opened/syncd
```

You can redirect the output of this script to a file (e.g. mwc.out) and use the following awk commands to get various different reports:

- Report of all mirrored logical volumes

  ```
  awk 'NR <= 2 || $3 > 1' mwc.out
  ```

  ```
  LV              Filesystem                Copies MWC LV_STATE
  =============== ========================= ====== === ============
  jfslog1         N/A                            2 on  opened/syncd
  jfslog2         N/A                            2 on  opened/syncd
  C21saptrans     /usr/sap/trans                 2 on  opened/syncd
  C21usrsap       /usr/sap/C21                   2 on  opened/syncd
  C21sapmnt       /sapmnt/C21                    2 on  opened/syncd
  C21oracle       /oracle/C21                    2 on  opened/syncd
  C21sapdata1     /oracle/C21/sapdata1           2 on  opened/syncd
  C21sapdata2     /oracle/C21/sapdata2           2 on  opened/syncd
  C21sapdata3     /oracle/C21/sapdata3           2 on  opened/syncd
  C21sapdata4     /oracle/C21/sapdata4           2 on  opened/syncd
  C21sapdata5     /oracle/C21/sapdata5           2 on  opened/syncd
  C21sapdata6     /oracle/C21/sapdata6           2 on  opened/syncd
  C21sapdata7     /oracle/C21/sapdata7           2 off opened/syncd
  C21sapdata8     /oracle/C21/sapdata8           2 off opened/syncd
  C21sapreorg     /oracle/C21/sapreorg           2 on  opened/syncd
  ```

- Report of all mirrored logical volumes where MWC is off

```
awk 'NR <= 2; $3 > 1 && $4 == "off" ' mwc.out

LV              Filesystem                Copies MWC LV_STATE
=============== ========================= ====== === ============
C21sapdata7     /oracle/C21/sapdata7           2 off opened/syncd
C21sapdata8     /oracle/C21/sapdata8           2 off opened/syncd
```

- Report of all logical volumes with stale partitions

```
awk 'NR <= 2; /stale/' mwc.out

LV              Filesystem                Copies MWC LV_STATE
=============== ========================= ====== === ============
C21saptrans     /usr/sap/trans                 2 on  opened/stale
```