Science & Technology Facilities Council

Technical Report

The effect of array verifies on I/O performance

J. I. Thorne

November 10, 2008

RAL-TR-2008-032

Library and Information Services
SFTC Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Didcot
OX11 0QX
UK
Tel: +44 (0)1235 445384
Fax: +44(0)1235 446403
Email: library@rl.ac.uk

ISSN 1358-6254

RAL-TR-2008-032

The effect of array verifies on I/O performance

J. I. Thorne[1]

ABSTRACT

There is a need to verify arrays on the RAL Tier1 disk servers to check for bad blocks and other inconsistencies, thereby reducing the risk of data loss. This can be achieved using the array verify commands on the RAID controllers. The impact this has on system I/O performance is be significantand has been determined. Recommendations are made on the implementation of array verifies within the RAL Tier1.

Keywords: array verifies, system performance, linux, hardware

---

[1] Science and Technology Facilities Council, Rutherford Appleton Laboratory,
Harwell Science and Innovation Campus, Didcot, Oxfordshire, OX11 0QX, England, UK.
Email: james.thorne@stfc.ac.uk

November 10, 2008

1 Introduction

Rutherford Appleton Laboratory (RAL) hosts the UK Tier1 for the LHC Computing Grid (LCG). The RAL Tier1 is tasked with providing many petabytes of storage space for the Large Hadron Collider (LHC) and other experiments. The large volume of data dictates that the cost per gigabyte is kept low while maintaining a reasonably low risk of data loss. This is achieved by using commodity storage servers ("disk servers") and a distributed storage solution developed at CERN called CASTOR.

There are four separate instances of CASTOR running at the RAL Tier1, each subdivided into disk pools. The separate instances isolate each experiment to prevent the whole service being affected when an experiment experiences problems. Each pool is set up with differing data policies; I/O to a particular pool is shared between all the disk servers within that pool.

The RAID controller cards used in disk servers at the RAL Tier1 are capable of running background array verifies. The majority of these controllers are manufactured by 3ware[1] and can execute verifies at five different priorities, ranging from 5 (fastest system I/O) to 1 (fastest array verify).

Most disk arrays at the RAL Tier1 are not verified on a regular basis. As failing and faulty drives are not detected in advance, there could be multiple disk failures when an array attempts to rebuild. Verifies can help to detect these faulty drives and alert systems administrators.

Verifies have a cost in terms of performance as there is contention between the verify process and system I/O. The size of this performance impact was not known within the RAL Tier1 and is determined in this study. Verifies were run at priority 5 (fastest system I/O) during the testing to measure the minimum possible impact of verifies. Only the 3ware controllers were tested as they make up the bulk of RAL Tier1 RAID systems.

2 Testing

2.1 Host setup

Two Intel-based Viglen disk servers were used for the testing. The systems were set up in a similar fashion to the CASTOR disk servers at the RAL Tier1. Each machine was installed with Scientific Linux[2] 4.5 on the RAID1 disk pair connected to a 3ware 9650SE-4LPML controller card. The remaining fourteen drives in each system were connected to a 3ware 9650SE-16ML, set up as a RAID6 unit and partitioned into three EXT3 file systems. Each of the file systems was created with the same options as the production CASTOR file systems:

```
   mke2fs -j -T largefile4 -m 1 -N 3000000 -O dir_index /dev/sda1 \
2>&1 > /tmp/makefs_sda1
```

The file systems were mounted under `/tst/test1`, `/tst/test2` and `/tst/test3`.

2.2 Tests

There were two tests; one to test the impact of verifies on system I/O (`run.sh`, appendix A) and one to test the impact of system I/O on verify performance (`loopy.sh`, appendix B). Both tests use the 3ware command line tool `tw_cli` to control verifies and the IOzone[3] benchmarking tool to measure and simulate system I/O. In both cases, verifies were run at the lowest priority that the RAID controller would allow. Each test was run multiple times and the mean I/O performance calculated. This was to reduce the effect of any system processes that may be scheduled to run during the test period.

2.2.1 run.sh

This test optionally starts an array verify on the host and then runs IOzone to test the I/O performance of the host during the verify. `run.sh` was run three times on each of the two test

[1] `http://www.3ware.com`
[2] `http://www.scientificlinux.org`
[3] `http://www.iozone.org`

machines, both with and without a verify running, to allow the impact of a verify to be determined. The impact should be small as the verify is running at a low priority.

2.2.2 loopy.sh

This test was designed to measure the reverse of `run.sh`, i.e. the impact of high I/O on verification times. The test optionally loops IOzone and then starts a verify. It allows measurement of the impact of system load on the duration of verifies. The impact could be quite large as the verify is running a at low priority; any I/O should have a big effect.

3 Results

3.1 run.sh

The I/O performance of the hosts was measured both with and without a verify (tables 1 and 2). The performance without the verify was good, with writes generally over 40 MB/s and reads over 300 MB/s. The results show that there is a substantial impact on performance when an array verify is running (table 2). The write performance drops by over 6% during verifies and the read performance by almost 14%.

Table 1: I/O performance without verify

Host	Run	Write KB/s	Rewrite KB/s	Read KB/s	Re-read KB/s
1	1	38704.98	53189.74	307741.80	318413.01
	2	43081.40	58493.44	377230.69	389717.27
	3	43520.21	58609.90	393361.73	384339.41
	Mean	*41768.86*	*56764.36*	*359444.74*	*364156.56*
2	1	45377.70	57647.80	368525.14	354157.20
	2	46400.36	57412.41	339026.87	352441.99
	3	45664.98	56657.19	348635.28	348694.68
	Mean	*45814.35*	*57239.13*	*352062.43*	*351764.62*
Mean		43791.61	57001.75	355753.59	357960.59

3.2 loopy.sh

The impact of system I/O on verify time was expected to be high and it was. Table 3 shows the results of the tests. The time taken for a verify on an idle machine was about four hours 28 minutes, measured on two separate hosts. When the machine is busy with I/O the verify duration increases 13 times to nearly two and a half days. Time constraints prevented more than one run of `loopy.sh` with IOzone in progress.

4 Recommendations

The impact of verifies on I/O performance is substantial (6% for writes, 14% for reads), however verifies must be carried out to allow early detection of failing or faulty disks. The performance impact on a CASTOR disk pool as a whole can be assumed to be the single server figures divided by the number of servers in that pool. The RAL Tier1 needs to choose an array verification policy that minimises the impact on performance of the service as a whole, rather than performance of individual disk servers.

Table 2: I/O performance during a verify

Host	Run	Write KB/s	Rewrite KB/s	Read KB/s	Re-read KB/s
1	1	39586.65	54747.02	307748.90	323638.83
	2	39962.23	54532.34	320739.49	323155.71
	3	39485.06	53815.69	309531.23	306180.93
	Mean	*39677.98*	*54365.02*	*312673.21*	*317658.49*
2	1	42957.65	54619.22	318718.26	302702.41
	2	42438.17	55027.07	289617.39	287093.33
	3	41554.87	52433.78	290835.13	287555.47
	Mean	*42316.90*	*54026.69*	*299723.59*	*292450.40*
Mean		40997.44	54195.85	306198.40	305054.45
Impact (%)		**6.4**	**4.9**	**13.9**	**14.8**

Table 3: Verify performance with and without I/O

I/O	Duration (days:hours:mins:secs)
No	0:04:28:59
No	0:04:28:28
Yes	2:11:31:03

It is possible to verify a few servers from a pool simultaneously with only a small drop in performance across the pool. This needs some sort of central control, preferably without too much administrator intervention.

One solution may be the Puppet[4] configuration management tool which has the facility to run scripts on remote machines. Puppet is already in use at the RAL Tier1 for configuring the CASTOR software on the disk servers. An alternative is to create a home-grown solution using the cron scheduler and open source components.

Any solution must prevent performance loss from exceeding an acceptable limit by restricting the number of concurrent verifies within a pool. The acceptable limit needs to be discussed and agreed with the experiments and may be different for each instance and pool.

5 Conclusions

Array verifies must be carried out to prevent data loss but a method for reducing the impact on performance is required. The RAL Tier1 needs a centralised place to schedule and manage array verifies that requires little administrator intervention once set up. This could be accomplished with the Puppet configuration management tool or a home-grown solution, both of which need further investigation. Acceptable levels of performance degradation for a disk pool need to be discussed and decided with the experiments before any implementation is carried out.

[4] `http://www.reductivelabs.com/trac/puppet`

A run.sh

```
#!/bin/sh
#
# $Id: run.sh,v 1.7 2008/10/23 15:03:15 jiant Exp $
#
# (c) Science and Technology Facilities Council
#
# A script to run iozone testing, and a verify if necessary (based on
# $TESTVERIFY environment variable).

# Where is iozone?  This value matches if you installed the RPM from
# http://www.iozone.org
IOZONE=/opt/iozone/bin/iozone

# start a verify on the data LUN if required
if [ $TESTVERIFY ]; then
  echo "Starting verify..."
  # disable scheduled verifies on the data array
  tw_cli /c1 set verify=disable
  # set verify to lowest *verify* priority (i.e. fastest I/O)
  tw_cli /c1 set verify=5
  # start a verify
  tw_cli /c1/u0 start verify
fi

# start IOzone throughput testing, one thread per partition on the data LUN.
echo "Starting IOzone on /tst/test..."
$IOZONE -s 16g -r 64k -i 0 -i 1 -R -b /root/verifytesting/iozone.‘hostname‘.xls \
-t 3 -F /tst/test1/iozone.tmp /tst/test2/iozone.tmp \
/tst/test3/iozone.tmp | tee /root/verifytesting/iozone.‘hostname‘.out
```

B loopy.sh

```
#!/bin/sh
#
# $Id: loopy.sh,v 1.10 2008/10/23 15:03:15 jiant Exp $
#
# (c) Science and Technology Facilities Council
#
# A script to run a  (based on
# $TESTIOZONE environment variable).

# Where is iozone?  This value matches if you installed the RPM from
# http://www.iozone.org
IOZONE=/opt/iozone/bin/iozone

# start a verify on the data LUN
start=$(date)
echo "Starting verify at $start..."
# disable scheduled verifies on the data array
tw_cli /c1 set verify=disable
# set verify to lowest *verify* priority (i.e. fastest I/O)
tw_cli /c1 set verify=5
# start a verify
tw_cli /c1/u0 start verify

# If required, loop IOzone until verify is done, otherwise, just sleep.
again=0
if [ $TESTIOZONE ]; then
  echo "Starting IOzone on /tst/test..."
  while [ $again -eq 0 ]; do
    # test whether verify is finished
    tw_cli info c1 u0 | grep VERIFY 2>&1 > /dev/null
    again=$?
    # start IOzone throughput testing, one thread per partition on the data LUN.
    $IOZONE -s 2g -r 64k -i 0 -i 1 -t 3 -F /tst/test1/iozone.tmp \
    /tst/test2/iozone.tmp /tst/test3/iozone.tmp 2>&1 > /dev/null
  done
else
  echo "Sleeping while verify runs..."
  while [ $again -eq 0 ]; do
    # test whether verify is finished
    tw_cli info c1 u0 | grep VERIFY 2>&1 > /dev/null
    again=$?
    # sleep for 5 minutes and then test for verify again.
    sleep 300
  done
fi
# output timing info
echo
echo "Started test at $start, finished test at $(date)."
```