ORACLE
SOLARIS

An Oracle Technical White Paper
September 2012

Detecting and Resolving Oracle Solaris LUN Alignment Problems

ORACLE

`

`

Overview

The Sun ZFS Storage Appliance offers block type volume access (LUNs) to client systems. These volumes can be configured using a block size that is optimized for a client's application I/O profile. It is critical that the start of the client partitions aligns with the blocks on the ZFSSA volumes. Often clients' partitioning configuration and formatting tools are still based on the old SCSI 512 bytes sector addressing model, which may cause partition misalignment, resulting in increased I/O response times and reduced I/O throughput.

This paper describes how to use the Oracle Solaris `format` utility to correctly set up partitions for a Oracle Solaris EFI-based partitioning schema for Sun ZFS Storage Appliance volumes using an 8K block size. Oracle Solaris provides system administrators with advanced DTrace Analytics monitoring to detect LUN alignment problems that can be alleviated with the Oracle Solaris VTOC or EFI format disk label utility. This paper shows how to use DTrace Analytics on an active system and resolve those LUN alignment problems by tuning the EFI disk label.

`

LUN Alignment Challenges with Advanced Storage Devices

Legacy disk drives accessed with the SCSI protocol present a logical block architecture (LBA) to the operating system that has historically relied on a 512B sector size for data storage, and the default behavior of operating system tools for formatting and accessing disk drives uses 512B aligned records to store metadata about the drive geometry as well as end-user data. Although this process works well for simple disk drives, it can produce suboptimal results for integrated storage systems that internally process data in larger allocation units.

For example, as shown in Figure 1, the default Oracle Solaris EFI disk label creates a slice that starts at sector 34. So if the OS does an 8kB access on the start of the file system, this translates to an access to the virtual sectors 34 through sector 49 and requires the storage system to read two 8kB allocation units, virtual sectors 32-47 and sectors 48-63, to process the requested data.

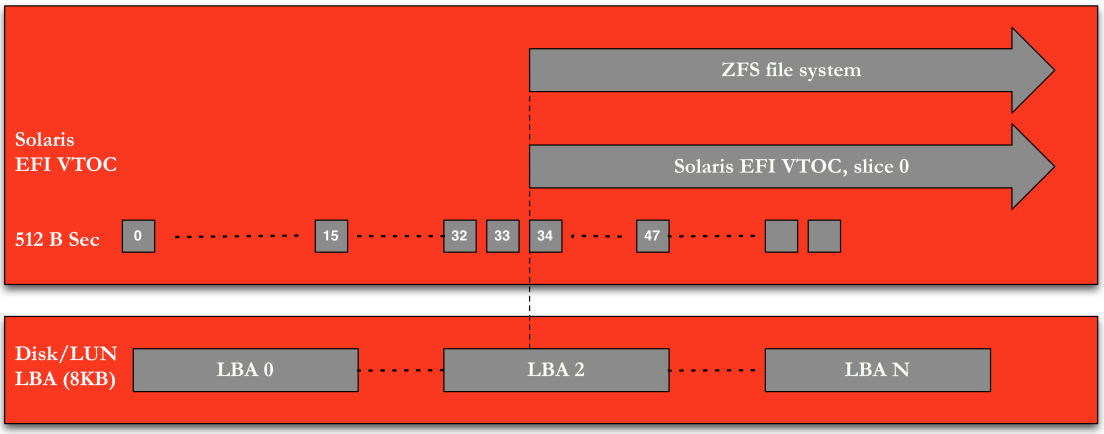

Figure 1. Default Oracle Solaris EFI disk label structure

Such misaligned I/O conditions reduce the efficiency with which the storage system can process I/O and consequently may increase response time at a fixed throughput or reduce throughput at a fixed response time.

Figures 2 and 3 show examples of the I/O response time and throughput characteristics associated with misaligned and aligned I/O on the same storage system. Resolving I/O alignment challenges presented by storage devices with allocation unit sizes greater than 512B requires that the system administrator begin storing user data on the first storage device allocation unit available after the disk label or volume table of contents (VTOC). Configuring the operating system to leave enough empty space between the end of the disk label and the first byte of user data ensures that the first byte of user data is written to the first byte of an allocation unit on the storage device.

For example, the Oracle Solaris EFI disk label is stored on the first 33 sectors of the LUN presented by the storage device, and the default partition table layout stores user data starting at sector 34 or 17kB into the LUN. To align I/O on an 8kB allocation unit, the starting sector of the data partition may be advanced to sector 48 (24kB), sector 64 (32kB), or any multiple of 16 sectors beyond sector 48. To accommodate larger allocation unit sizes, choose an appropriately larger starting sector: 128 for

`

64kB or 256 for 128kB. Figure 3 shows an example of how to use the Oracle Solaris `format -e` command to perform this alignment.

`

Detecting and Resolving LUN Alignment Problems

Detecting LUN alignment problems challenges the system administrator because of the complexity of the I/O processing stack between the application executing the I/O system call and the LUN processing the I/O. Complications include the presence of file systems and volume managers that may change the physical characteristics of the application-initiated I/O as well as knowledge of the partition table on the LUN.

To simplify detection of LUN alignment problems in these complex environments, Oracle Solaris DTrace Analytics provides the function boundary testing (fbt) provider to track I/O operations at the SCSI driver layer (sd). The code listed in the Appendix shows an example of how to monitor a running system to verify 8kB alignment of I/O at the LUN level. To illustrate the benefit of maintaining I/O alignment between the host-side data access and the allocation unit size of the storage device, this section shows a real example of client-reported I/O performance measurements and correlated DTrace Analytics output from misaligned and aligned cases, and an example of how to control data alignment using the Oracle Solaris format utility to tune an EFI disk label.

Figure 2 shows storage system performance for 8kB random read operations using a default EFI disk label that leads to client-side I/O being executed on 9kB boundaries. The workload shown in the test case is 64 outstanding 8kB read requests, and performance is quantified by a 3.9 ms I/O response time at a throughput of 15,900 IOPS. In practice, there is no obvious indication from this report that an I/O alignment problem is occurring except that I/O response time is longer and I/O throughput is lower than expected for the given workload running on the specific storage system.

```
# iostat -xnzcT d 1

Tue Jan 26 10:39:43 2010
     cpu
 us sy wt id
  1  5  0 94
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
    0.0    1.0    0.0    8.0  0.0  0.0    0.0    6.7   0   1 c2t0d0
 15639.6   42.0 124467.0    0.0  0.1 62.9    0.0    4.0  10 100 c0t6d0
Tue Jan 26 10:39:44 2010
     cpu
 us sy wt id
  1  5  0 94
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
 15887.3   44.0 126417.8    0.0  0.1 62.8    0.0    3.9   9 100 c0t6d0
Tue Jan 26 10:39:45 2010
     cpu
 us sy wt id
  1  5  0 94
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
 15437.0   38.0 122907.9    0.0  0.1 63.0    0.0    4.1   9 100 c0t6d0
Tue Jan 26 10:39:46 2010
```

Figure 2. I/O using misaligned EFI label

`

Figure 3 shows the output of the supplied DTrace Analytics script that lists the root-cause of the performance problem. The example shows that, although the host is performing 8kB I/O operations consistent with the known allocation unit size of the LUN, the operations are executed on 9kB boundaries. Consequently, for each host-initiated I/O against the LUN, the storage target needs to access two adjacent 8kB blocks: the first block for the first 7kB of the I/O and a second block for the remaining 1kB. This leads to inefficient processing of data, increases in response time, and degradation in throughput.

```
# ./align.d

5.000000284 seconds elapsed

   72627 IOs issued
   71682 IOs misaligned,        1 block offset     1011 byte offset
     484 IOs non-multiple of 8KB
       0 Percent non-8k IOs


           value  ------------- Distribution ------------- count
              -1 |                                         0
               0 |                                         4253
               1 |                                         6
               2 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 376259
               4 |                                         33
               8 |                                         27
              16 |                                         0
```

Figure 3. DTrace Analytics output using misaligned EFI label

Proper configuration of the starting location for data storage on the LUN can resolve the I/O alignment issue. On the Oracle Solaris operating system, the `format` command tool displays, creates and modifies partitions and labels disks. Figure 4 shows that the format output of the first available sector of an EFI disk label in Oracle Solaris is at 34, which is two blocks beyond 32kb being used for this example. Figure 4 also shows the steps to correct the 32kb block offset of the EFI label using `format` in expert mode with the `-e` option.

```
format -e /dev/rdsk/c0t6d0s0
selecting /dev/rdsk/c0t6d0s0
[disk formatted]

FORMAT MENU:
...
        partition  - select (define) a partition table
...
Format> partition
PARTITION MENU:
...
        print  - display the current table
...
partition> print
Current partition table (original):
Total disk sectors available: 268419037 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm                34      127.99GB          268419037
  1 unassigned    wm                 0           0                  0
  2 unassigned    wm                 0           0                  0
  3 unassigned    wm                 0           0                  0
  4 unassigned    wm                 0           0                  0
  5 unassigned    wm                 0           0                  0
  6 unassigned    wm                 0           0                  0
  7 unassigned    wm                 0           0                  0
  8   reserved    wm         268419038        8.00MB          268435421

partition> 0
Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm                34      127.99GB          268419037

Enter partition id tag[usr]:
Enter partition permission flags[wm]:
Enter new starting Sector[34]: 64
Enter partition size[268419004b, 268419067e, 131063mb, 127gb, 0tb]: $
partition> label
[0] SMI Label
[1] EFI Label
Specify Label type[1]: 1
Ready to label disk, continue? y

partition> print
Current partition table (unnamed):
Total disk sectors available: 268419037 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm                64      127.99GB          268419036
  1 unassigned    wm                 0           0                  0
  2 unassigned    wm                 0           0                  0
  3 unassigned    wm                 0           0                  0
  4 unassigned    wm                 0           0                  0
  5 unassigned    wm                 0           0                  0
  6 unassigned    wm                 0           0                  0
  7 unassigned    wm                 0           0                  0
  8   reserved    wm         268419038        8.00MB          268435421

partition> quit
```

Figure 4. Using the `format -e` command

The DTrace Analytics ouptut shown in Figure 5 verifies that correct I/O alignment is preserved between the application and the storage device. In this example, all 8kB read I/O operations are conducted on multiples of 8kB on the LUN presented by the storage system.

`

```
# ./align.d

5.000007182 seconds elapsed

   94878 IOs issued
       0 IOs misaligned,        0 block offset        0 byte offset
     629 IOs non-multiple of 8KB
       0 Percent non-8k IOs


           value  ------------- Distribution ------------- count
              -1 |                                         0
               0 |@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ 293087
               1 |                                         7
               2 |                                         7
               4 |                                         48
               8 |                                         49
              16 |                                         0
```

Figure 5. DTrace Analytics output using aligned EFI label

As a result of maintaining I/O alignment, system performance improves, as shown in Figure 6. In this example, I/O response time drops to 3.3 ms and throughput increases to 20,200 IOPS. This shows a 20% gain in system performance for the tested workload. In practice, performance gains may be greater or less, depending on the specific workload and the hardware configuration supporting the workload.

```
# iostat -xnzcT d 1

Tue Jan 26 10:41:11 2010
     cpu
 us sy wt id
  1  6  0 93
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
 20233.6   56.0 161002.7    0.0  0.1 62.8    0.0    3.1  11 100 c0t6d0
Tue Jan 26 10:41:13 2010
     cpu
 us sy wt id
  0  6  0 94
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
 19382.3   48.0 154315.7    0.0  0.1 62.6    0.0    3.2  11 100 c0t6d0
Tue Jan 26 10:41:15 2010
     cpu
 us sy wt id
  1  6  0 94
                    extended device statistics
    r/s    w/s   kr/s   kw/s wait actv wsvc_t asvc_t  %w  %b device
 20226.4   48.0 161068.9    0.0  0.1 63.0    0.0    3.1  10 100 c0t6d0
```

Figure 6. Verifying with the `iostat` command

`

Summary

Detecting LUN alignment issues with advanced storage devices like the Sun ZFS Storage Appliance can be achieved fairly quickly and easily using the built-in Oracle Solaris-tools to monitor I/O with iostat and DTrace Analytics, and once detected can quickly be corrected by calculating the correct beginning block sector based on the I/O block size and using the `format` command to write a proper EFI label. The result of this tuning is more efficient data processing by the underlying storage system, a reduction in I/O response time, and an increase in I/O throughput compared to untuned systems. Administrators seeking the most efficient use of storage resources should follow this best practice.

`

Appendix

DTrace Analytics Code to Detect LUN Alignment Problems

```
#!/sbin/dtrace –qs
# Script to detect IO misalignment for 8K based LUN volumes
BEGIN
{
        start = timestamp;
}
int issued;
int misaligned;
int misalignedBy;
int nonmult;
int tot_issued;
int tot_misaligned;
int tot_misalignedBy;
int tot_nonmult;
int tot_elapsed;
struct buf *bp;
struct sd_xbuf *xp;
fbt:sd:sd_core_iostart:entry
{
        issued++;
        bp = (struct buf *) arg2;
        xp = (struct sd_xbuf *)((bp)->b_private);
        misaligned += (xp->xb_blkno % 8) ? 1 : 0;
        misalignedBy += (xp->xb_blkno % 8 );
        @misalignedByHist = quantize(xp->xb_blkno % 8 );
        nonmult += (bp->b_bcount % 8192) ? 1 : 0;
        @a[bp->b_bcount] = count();
}
END
{
        elapsed = timestamp - start;
        tot_elapsed += elapsed;
        tot_issued += issued;
        tot_misaligned += misaligned;
        tot_nonmult += nonmult;
      tot_misalignedBy += misalignedBy;
      printf( "=======================================\n");
      printf( "Totals\n");
      printf( "=======================================\n");
        printf( "\n IO size Count\n" );
        printa( @a);
        printf( "\n IO misalignment Count\n" );
        printa( @misalignedByHist);
        printf( "%d.%09d seconds elapsed\n\n", tot_elapsed /
1000000000, tot_elapsed % 1000000000);
```

`

```
        printf( "%8d IOs issued\n%8d IOs misaligned, %8d block
offset %8d byte offset\n",
                tot_issued,
                tot_misaligned,
                (tot_issued==0 ? 0 : tot_misalignedBy /
tot_issued),
                (tot_issued==0 ? 0 : 512 * tot_misalignedBy /
tot_issued)
        );
        printf( "%8d IOs non-multiple of 8KB\n", tot_nonmult );
        printf( "%8d Percent non-8k IOs\n", (tot_issued==0 ? 0 :
(tot_nonmult * 100 / tot_issued)) );
}

tick-5sec
{
        elapsed = timestamp - start;
        printf( "%d.%09d seconds elapsed\n\n", elapsed/1000000000,
elapsed%1000000000 );
        printf( "%8d IOs issued\n%8d IOs misaligned, %8d block
offset %8d byte offset\n",
                issued,
                misaligned,
                (issued==0? 0: misalignedBy / issued),
                (issued==0? 0: 512 * misalignedBy / issued)
        );
        printf( "%8d IOs non-multiple of 8KB\n", nonmult );
        printf( "%8d Percent non-8k IOs\n\n", (issued==0 ? 0 :(
nonmult * 100 / issued )) );
        printa( @misalignedByHist );
        tot_elapsed += elapsed;
        tot_issued += issued;
        tot_misaligned += misaligned;
        tot_nonmult += nonmult;
        tot_misalignedBy += misalignedBy;
        start = timestamp;
        issued = 0;
        misaligned = 0;
        misalignedBy = 0;
        nonmult = 0;
}
```

`

References

For further information on the following topics, see the following resources:

- LUN-Partition Alignment
  - Partition Alignment Guidelines for Unified Storage blog, by David Lutz: https://blogs.oracle.com/dlutz/entry/partition_alignment_guidelines_for_unified

- Oracle Solaris Product Information
  - Oracle Solaris 11 http://www.oracle.com/uk/products/servers-storage/solaris/solaris11/overview/index.html
  - How to obtain a copy of Oracle Solaris http://www.oracle.com/technetwork/server-storage/solaris11/downloads/index.html
  - Oracle Solaris 11 Documentation http://www.oracle.com/technetwork/documentation/solaris-11-192991.html
  - Oracle Solaris Disk Management http://docs.oracle.com/cd/E23824_01/html/821-1459/disksconcepts-29477.html#scrolltoc

- Sun ZFS Appliances Product Information
  - Sun ZFS Storage Appliances http://www.oracle.com/us/products/servers-storage/storage/nas/overview/index.html
  - Sun ZFS Storage Appliance Documentation https://wikis.oracle.com/display/FishWorks/Documentation
  - Oracle Unified Storage Systems Documentation http://www.oracle.com/technetwork/documentation/oracle-unified-ss-193371.html

Oracle is committed to developing practices and products that help protect the environment

Detecting and Resolving Oracle Solaris LUN Alignment Problems
April 2010, v1.0; September 2012, v2.0
Author: Art Larkin, Application Integration Engineering
Contributing Authors: Jeff Wright, Peter Brouwer, Application Integration Engineering

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.like
Fax: +1.650.506.7200
oracle.com

SOFTWARE. HARDWARE. COMPLETE.