Linux* Storage Stack performance

Kristen Carlson Accardi
Matthew Wilcox

Open Source Technology Centre
Legal Information

Intel is a trademark of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2008, Intel Corporation. All rights are protected.
Facts and Speculation about Solid State Drives (SSD)

1/5th the power consumption of a mechanical disk

200X (+) the performance

Same price?

Better performance, low power, competitive pricing make SSDs "disruptive technology"

By 2010, SSDs will:
- be 20% of the laptop market
- have “significant penetration” into the data center

SSDs will place much higher demands on storage stack than traditional disks.
- The Zeus IOPS - 52,000 IOPS
- Mtron - 78,000/16,000 IOPS

These numbers are going to increase rapidly over time as more players arrive in this huge potential market.
Problem

As IOPS increases, CPU overhead, per I/O, becomes significant and a bottleneck.

Latency issues in the Linux* storage stack could make software a bottleneck.

Storage stacks are optimized for seek avoidance
   - CPU time spent avoiding seeks is wasted.

SSDs are still fairly expensive and uncommon, making it hard for the Linux community to measure and optimize for them.
“SSD” for everyone

• Simulate SSD with a RAM driver
  • The first step to reducing latency is finding out how bad it is. The only way to keep latency low is to allow everybody to measure the latency, and avoid changes to the kernel which would increase latency.
  • The ‘rd’ ram disc driver does not behave like a driver for real hardware, and is not a good simulator for our purposes.
• Put relevant measurement data together in a tool that's easy to use.
Test Setup

Fake Drivers

- The ‘scsi_ram’ and ‘ata_ram’ drivers are, respectively, scsi and ata drivers for the Linux kernel, which simulate really fast discs by storing data in memory.
- These drivers will allow us to measure latency all the way down into the ATA layer.

Real Measurements

- The ‘iolat’ tool generates random disk I/O while simultaneously profiling the kernel.
- It reports the number of IOPS (I/O operations per second) that it achieves and where the kernel is spending its time.
Linux storage stack

- User Application (iolat)
- File System (ext2)
- Block
  - rd
  - sd
  - SCSI midlayer
    - scsi_ram
    - libata
    - ata_ram
The `scsi_ram` driver is designed to behave like a driver for a real SCSI card. It accepts SCSI commands and, instead of sending them to a piece of hardware, it queues them to a thread. The thread, typically running on a different CPU, copies data to or from an array of pages, then reports success.

The `ata_ram` driver is similar to the `scsi_ram` driver. The ATA command set is different from the SCSI command set, and the interface to `libata` is different from the interface to the SCSI midlayer, but the design of the driver is virtually unchanged.

Both drivers have options to help pinpoint performance issues. For example, the actual data copies can be disabled, removing that factor from the performance profile.
Iolat details

Generate Traffic and measure IOPS
- Random reads and writes
- Single large test file
- Size of read/write configurable
- Compare to “reference” data

Profile Kernel
- Uses /proc/profile

Classify functions profiled
- Hand classified, stored in a text file

Generates Reports
- IOPS measurement
- Classification report
More Tester details

Types of tests:
- Read
- Write
- Mixed reads and write

Can do Direct I/O, and Cached I/O
- Cached I/O tests will fdatasync() every 10 iterations.
- Direct I/O tests wait for previous I/O to complete before submitting next I/O. No batching or merging can occur in the driver.
### IOPSMeeasure version 0.3 (C) 2068 Intel Corporation

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Direct</th>
<th>bytes</th>
<th>IOPS</th>
<th>+/-</th>
<th>Avg. req time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small_direct_read</td>
<td>Yes</td>
<td>4096</td>
<td>90044</td>
<td>%3</td>
<td>0.10899</td>
</tr>
<tr>
<td>Small_direct_write</td>
<td>Yes</td>
<td>4096</td>
<td>85578</td>
<td>%2</td>
<td>0.11536</td>
</tr>
<tr>
<td>Medium_direct_read</td>
<td>Yes</td>
<td>131072</td>
<td>22893</td>
<td>%0</td>
<td>0.45172</td>
</tr>
<tr>
<td>Medium_direct_write</td>
<td>Yes</td>
<td>131072</td>
<td>13302</td>
<td>%0</td>
<td>0.96110</td>
</tr>
</tbody>
</table>

```plaintext
3352  scsi_ram_read     scsi_ram_driver  28.3545
3127  scsi_ram_write    scsi            14.9986
2451  scsi_request_fn   block           11.5580
838   scsi_dispatch_cmd scheduler       9.4730
748   blk_end_io        mm              8.3916
581   blk_done_softirq  fs              8.3420
458   __make_request    data_copy       2.1940
458   __blockdev_direct_IO primatives    1.3418
370   bio_alloc_bioset  elevator        0.6260
354   *unknown*         *                Unclassified 14.7205
350   __end_that_request_first
277   get_request       
206   __find_get_block  
198   __bio_add_page    
159   generic_make_request
150   __might_sleep     
```
Each layer subtracts performance

scsi_ram much slower than rd

ata_ram 10% slower than scsi_ram
  – Medium direct reads 50% slower??

Neither scsi nor ata layers can handle SSD IOPS
IOPSMerasure version 0.3 (C) 2008 Intel Corporation

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Direct</th>
<th>bytes</th>
<th>IOPS</th>
<th>+/-</th>
<th>Avg. req time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small_direct_read</td>
<td>Yes</td>
<td>4096</td>
<td>89944</td>
<td>%3</td>
<td>0.10843</td>
</tr>
<tr>
<td>Small_direct_write</td>
<td>Yes</td>
<td>4096</td>
<td>85774</td>
<td>%2</td>
<td>0.11414</td>
</tr>
<tr>
<td>Medium_direct_read</td>
<td>Yes</td>
<td>131072</td>
<td>22704</td>
<td>%1</td>
<td>0.40915</td>
</tr>
<tr>
<td>Medium_direct_write</td>
<td>Yes</td>
<td>131072</td>
<td>13010</td>
<td>%2</td>
<td>0.69613</td>
</tr>
</tbody>
</table>

9241 scsi_ram_read
8809 scsi_ram_write
7068 scsi_request_fn
2357 scsi_dispatch_cmd
2080 blk_end_io
1500 blk_done_softirq
1372 __blockdev_direct_IO
1359 __make_request
1020 __end_that_request_first
998 bio_alloc_bioset
909 *unknown*
829 get_request
604 __bio_add_page
531 __find_get_block
440 __might_sleep
429 generic_make_request
<table>
<thead>
<tr>
<th>Test Name</th>
<th>Direct</th>
<th>bytes</th>
<th>IOPS</th>
<th>+/-</th>
<th>Avg. req time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small_direct_read</td>
<td>Yes</td>
<td>4096</td>
<td>82022</td>
<td>%-11</td>
<td>0.12420</td>
</tr>
<tr>
<td>Small_direct_write</td>
<td>Yes</td>
<td>4096</td>
<td>77907</td>
<td>%-11</td>
<td>0.13057</td>
</tr>
<tr>
<td>Medium_direct_read</td>
<td>Yes</td>
<td>131072</td>
<td>12321</td>
<td>%-46</td>
<td>0.53577</td>
</tr>
<tr>
<td>Medium_direct_write</td>
<td>Yes</td>
<td>131072</td>
<td>12436</td>
<td>%-6</td>
<td>0.53186</td>
</tr>
</tbody>
</table>

### System Call Effective User Time (ms)

- `1440 scsi_request_fn` 22.5076
- `1170 ata_ram_read` 21.0077
- `1050 ata_ram_write` 12.5522
- `892 scsi_dispatch_cmd` 11.9045
- `432 blk_end_io` 7.4762
- `281 blk_done_softirq` 6.9612
- `178 __blockdev_direct_IO` 1.5829
- `160 kmem_cache_free` 1.1760
- `157 get_request` 0.4297
- `152 kmem_cache_alloc` 0.3483
- `130 __end_that_request_first` 0.2869
- `110 follow_page` 0.1294
- `109 bio_alloc_bioset` Unclassified 13.6374
- `93 blk_recalc_rq_segments`
Focus performance work on SCSI layer rather than Block Layer

scsi_ram is much slower than rd on small direct reads and small direct writes test

Profile data indicates a much greater % of time spent in block layer, and that the scsi layer adds significant overhead over just the block layer.
Next Steps

Investigate reducing SCSI layer overhead by:
- Digging down in the profiles to find hot spots
- Optimizing host lock acquisition

Take Performance analysis down to ATA layer with ata_ram
- libata uses the SCSI layer, then translates to ATA commands
- Many SSDs will interface as SATA devices

Investigate using different elevators
- Existing elevators are optimised for avoiding seeks. This is wasted work when seeks are cheap.

Move libata away from SCSI
- If it were to interface directly to the block layer, we could avoid the SCSI-to-ATA translation layer.
  • Need to be careful with drivers that support SAS and SATA drives
Backup