Aaron Toponce

{ 2012 12 03 }

How A ZIL Improves Disk Latencies

I just setup an SSD ZFS Intent Log, and the performance improvements have been massive. So much so, I need to blog about it. But first, I need to give you some background. I'm running a 2-node KVM hypervisor cluster replicating the VM images over GlusterFS (eventually over RDMA on a DDR InfiniBand link). One of the VMs is responsible for reporting outages, graphing latencies and usage, and other useful statistics. Right now, it's Zabbix, Munin and Smokeping. Other useful utilities will be added. You can see some of the pages at http://zen.ae7.st.

Due to the graphical intensive nature of the web applications, disk latencies were intense. I was seeing an average of 5 second latencies. Now, there are things I can do to optimize it, and work is ongoing with that, such as replacing Apache with Nginx, fgci, etc. However, the data store is a ZFS pool, and I have SSDs in the hypervisors (each hypervisor is identical). So, I should be able to partition the SSDs, giving one partition as a mirrored ZFS Intent Log (ZIL), and the rest as an L2ARC cache. So, first, I partitioned the SSDs using GNU parted. Then, I issued the following commands to add the partitions to their necessary spots:

```
# zpool add pool log mirror ata-OCZ-REVODRIVE_OCZ-33W9WE11E9X73Y41-part1 ata-OCZ-REVODRIVE_OCZ-X5RG0EIY7MN7676K-part
# zpool add pool cache ata-OCZ-REVODRIVE_OCZ-33W9WE11E9X73Y41-part2 ata-OCZ-REVODRIVE_OCZ-X5RG0EIY7MN7676K-part2
```

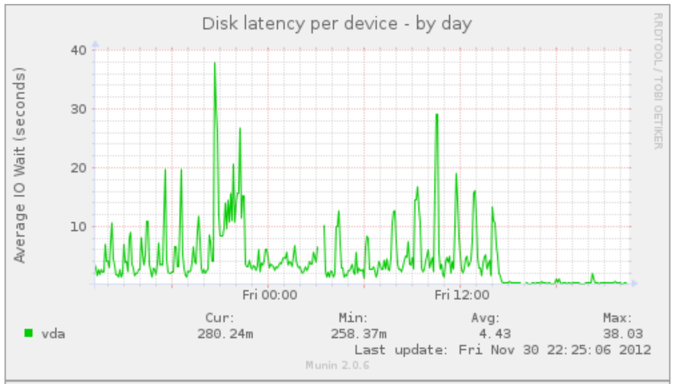
I used the disk ID as found in /dev/disk/by-id/ rather than the direct device, as it's guaranteed to be unique and persistent, even if the /dev/sd? assignment changes on boot. Because the cache drives are not persistent, they may not retain metadata about what they contain, so the pool will not add them if it can't find them.

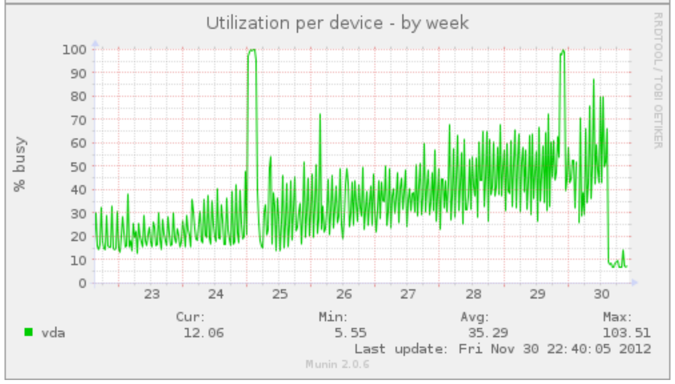
Now, I can see the status as follows:

```
# zpool status
  pool: pool
 state: ONLINE
 scan: scrub repaired 0 in 3h40m with 0 errors on Sun Nov 25 03:40:14 2012
config:

        NAME                                                  STATE     READ WRITE CKSUM
        pool                                                  ONLINE       0     0     0
```

```
	  raidz1-0                                          ONLINE       0     0     0
	    sdd                                             ONLINE       0     0     0
	    sde                                             ONLINE       0     0     0
	    sdf                                             ONLINE       0     0     0
	    sdg                                             ONLINE       0     0     0
	logs
	  mirror-1                                          ONLINE       0     0     0
	    ata-OCZ-REVODRIVE_OCZ-33W9WE11E9X73Y41-part1  ONLINE       0     0     0
	    ata-OCZ-REVODRIVE_OCZ-X5RG0EIY7MN7676K-part1  ONLINE       0     0     0
	cache
	  ata-OCZ-REVODRIVE_OCZ-33W9WE11E9X73Y41-part2    ONLINE       0     0     0
	  ata-OCZ-REVODRIVE_OCZ-X5RG0EIY7MN7676K-part2    ONLINE       0     0     0

errors: No known data errors
```

How did this improve my disk latencies? Well, as mentioned, I was seeing 5 second latencies on write for a specific VM, which sucked. After adding the ZIL, I'm now seeing 300 milisecond disk write latencies, and sometimes 200 milisecond. That's a full order of magnitude folks times two! See the images below:





The SSD ZIL was added around 14:30, as is evident in the graphs. Not only did disk latencies and utilization drop, but CPU usage dropped. As a result, load dropped. The fact of the matter is, the system is in a much more healthy state with respects to storage than it was previously. This is Big News.

Now, it's important to understand that the ZIL is write-intensive, while the L2ARC is read-intensive. Thus, the ZIL should be mirrored and on non-volatile storage, while the L2ARC can be striped and on volatile storage. When using SSDs, the ZIL can wear out the chips, killing the drive. So, unless your SSD supports wear leveling, you could be in trouble with old data (~5s) on disk. If you're not comfortable putting the ZIL on your SSD, then you could use a battery-backed DRAM disk. My SSD supports wear leveling, and is using the 32nm process, which gives me something around 4,000 P/E cycles. Because this SSD is 120 GB in total size, this should guarantee 120 GB x 4000 writes or 480,000 GB writes = 480 TB of written data. Further, it seems the ZIL is only holding about 5 MB of data at any time, even though I gave the partition 4 GB.

If this isn't convincing enough to use SSDs for a ZIL, and to use ZFS for your data storage, I don't know what to say.

Posted by Aaron Toponce on Monday, December 3, 2012, at 6:00 am. Filed under Debian, Linux, Ubuntu, ZFS. Follow any responses to this post with its comments RSS feed. You can post a comment or trackback from your blog. For IM, Email or Microblogs, here is the Shortlink.

{ 1 } Comments

1. Pousen | October 5, 2013 at 11:35 am | Permalink

   Thank you for sharing this information about ZFS in Linux!

   I was looking for something to help me and I found this!
   Awesome!

{ 1 } Trackback

1. Aaron Toponce : ZFS Administration, Part III- The ZFS Intent Log | December 13, 2012 at 6:06 am | Permalink

   [...] I blogged about this just a few days ago at http://pthree.org/2012/12/03/how-a-zil-improves-disk-latencies/. [...]