Friday, January 1. 2010

Selfmade SSDs - or: the tale of thinking too complex ....

Frequent readers of my blog know that i'm using an Notebook via iSCSI to emulate a SSD. At the moment my SSD is an old Acer Aspire notebook. It works reasonably well, it's much faster than a hard disk and i can test the behavior of applications with SSD without buying one.

This idea haunted me since i saw the following figure in a Sun presentation:

Shortly afterwards looking at the slide it translated into this slide in my mind:

So i started to rework the mentioned notebook into a SSD of my tests. This way was at the end a nice example of thinking too complex about a problem at start. The device got simpler and simpler over the time. At first my SSD was a rather complex scripting to dump the content of an ramdisk to a file at shutdown and restore it from there after shutdown. All this stuff was done with dd, a little bit of checksumming and a lot of checking about error conditions, for example to force the disk offline as soon as the power failed.

While writing the tutorial for the COMSTAR iSCSI target another method came into my mind. Using a file-based backing store for the COMSTAR target and just copy it into the ramdisk right before usage and copying it back to hard disk at shutdown. But this procedure wasn't without problems, as you still had to copy this file from and to the SSD, so the scripting would necessary, too.

But then it came to my mind: I'm thinking too complex here. All the stuff i need is already included in Solaris. What do you really need to fake the storage part an SSD:

a memory area for aggressive write caching ... even when it's a sync write
a memory area for read caching ... at best in the same size of the "SSD"
a non-volatile storage to protect the contents of the "SSD" while the system is powered off.
a mechanism to load the content to the volatile memory at boot and to save it to the non-volatile memory at shutdown
a mechanism to save the content of the write cache at an regular schedule (not necessary, but nice)

When you think about it, all this is already integrated in a component of Solaris. The first requirement can be fulfilled by the separated ZIL on a ramdisk. The second can be fulfilled by the ARC of ZFS. The third is fulfilled by simply using the SSD (it load the data into the ARC) and by the function of ZFS to save the data protected by the ZIL into the pool when you export the pool. The fourth - optional - requirement is fulfilled by the regular transaction group commit runs.

Keeping this in mind creating the recipe for an homegrown SSD gets really easy, as the complex stuff is done by the ZFS mechanisms:

Create a pool on your system
create a ramdisk
write a SMF manifest for a transient service, that recreates it after a reboot
use the ramdisk as a separated ZIL device
create an emulated volume on the pool. Obviously the size should be in a reasonable relation to the size of the main memory. It doesn't make sense to create an 1 TB emulated volume, when you just have 4 GB in your server.
share it with COMSTAR to your favorite storage area network (most often iSCSI)
use it as an SSD from other systems

The "trick" is: As ARC and sZIL (due to the ramdisk) reside both in DRAM, your iSCSI zvol device is really an solid state device for all practical purposes. At the end this server provides all the components of a solid-state drive. At the end every disk is a custom build server speaking a storage networking protocol to move data.

Starting from here you could work with a short period for the transaction group on the SSD faking pool to ensure that the data is on nonvolatile disk after a short period of time. As there shouldn't be any read requests to the disk (it's completely in RAM or in the case of a faked SSD for sZIL purposes you never read from it anyways while normal operations). On the server you can work with a longer transaction group period to ensure that you habe only fewer, albeit larger write bursts (and thus freeing up IOPS for reads).

Of course this server shouldn't be used for productive use, as the write cache (the main memory) isn't battery protected. But it's relatively easy to complete the SSD. A professional ready-to-use SSD contains capacitors or batteries to power the SSD until the data is written to non-volatile memory. So it would be easy just to add an small rack-mountable 1 RU UPS to the self-made SSD. It just have to spend enough power to shut the system down immediately after the UPS signals the shutdown. However even without the battery backup it's en par with the current Intel SSD that doesn't have a battery protected write cache as well .

This shouldn't be an blog article to postulate, that server with Solaris and ZFS are better SSD. But they have advantages: You can build them from spare components, you can monitor them (Do you know a SSD you can dtrace internally to check what an application really does with the ssd ? ... even when you don't use a dtrace capable OS on your server). You can build an SSD with FC/SAS/iSCSI and IB at the same time and ... hey ... do you know an SSD capable of dedup, compression or synchronous/asynchronous replication (via AVS albeit it starts to get a little bit absurd here, as the latency introduced by the synchronous replication will result in latencies making the SSD somewhat useless.) ?It's easy to use it from several systems at the same time, and (albeit i didn't tested it) it should be possible to use is via FC/SAS locally and via iSCSI remotely to ease HA failovers in a cluster.

In any case it's sufficient to play around with SSDs to learn a thing or two ...

Posted by Joerg Moellenkamp in English, Oracle, Solaris, Technology at 12:29

Now add a SATA connector to it and put all this into a 2,5" enclosure so that I can stick it into my laptop, like I can do with a cheap-o off-the-shelf SSD for a few hundred bucks
Anonymous on Jan 1 2010, 13:39

There is a market emerging for those high performance accelerators, even at the home and small business level : e.g. "MARS & AMS Releases eSATA Dynamic SSD RAM Disk"
http://www.acard.com.tw/english/newstabpop.jsp?idno=101

Question : am I correct in believing that S.M.A.R.T. in the SDD is a prerequisite for OpenSolaris to be able to use it as either ZIL or L2ARC?

To my last information, ACARD's ANS-9010 RAM-SSD does not implement SMART (yet?). (That one has 2 SATA ports each 130,000 IOPS.)

P.S. Happy New Year!
Anonymous on Jan 1 2010, 14:04

Knew such a comment would appear in the moment i've clicked on save

Sorry, not for that use case ... Completely different purpose
Anonymous on Jan 1 2010, 14:16

this acard thing rocks... and you dont even need smart if you use it as l2arc only... it supports ecc, battery cache and even dump to compact flash, so it is the best choice I know for less than 1K bucks...
Anonymous on Jan 5 2010, 11:58

Here is yet another example that the industry at large is EXTREMELY active in this field :
http://silverstonetek.com/products/p_contents.php?pno=HDDBOOST&area=

"HDDBOOST is a unique product that combines the best qualities of traditional hard drive and solid state disk (SSD) into one virtual super storage solution." SATA SSD"

Unfortunately this one is so easy you don't get the chance to actually learn anything
Anonymous on Jan 5 2010, 18:47