What I Learned This Month: Flash Express

Scott Chapman
American Electric Power

Last month I wrote about our ongoing adventures with the IBM DB2 Analytics Accelerator (IDAA). The good news on that front is that we did get a microcode fix that resolved the issues we were having. Unfortunately that level of code is not yet generally available and certified with the IDAA, but probably will be by the time you read this. If you have an IDAA in house, keep an eye on the Prerequisites and Maintenance page[1] and pay particular attention to the "Heads Up" notes there.

I did break away from IDAA for a little while to try out another new feature that should help improve performance: Flash Express. This is a new optional hardware feature available on the zEC12 machines. It also requires z/OS 1.13 with RSM Enablement Offering[2]. While we've had the hardware for a while now, we just got system programmer test systems upgraded to z/OS 1.13.

From a hardware perspective, one Flash Express feature consists of one pair of cards (for redundancy) containing a set of high speed Solid State Disks. I've seen different IBM documentation referring to the usable capacity of the pair as either 1.4TB or 1.6TB. I'm not sure why there's this discrepancy in the documentation, but our zEC12s show 1424GB available.

You may question putting relatively expensive SSD storage in the processor when you could buy SSD storage for your disk subsystem. There are a few good reasons to consider Flash Express in addition to having SSDs in your disk subsystem. Foremost is that having SSDs in the processor reduces the time it takes to get data from the storage. While it may seem like I'm splitting hairs, the speed of light issue becomes important when you consider it relative to the cycle time on the modern mainframe processors. No matter how close your disk subsystem is to your processor, it's going to take significantly more time (measured in processor cycles) to get to the disk and back vs. just going down to the Flash Express card in the PCIe drawer in the processor.

The other reason is that Flash Express is recognized by the operating system as a new class of memory: Storage Class Memory (SCM). Yes, it's sort of like we're back to the days of Central Storage and Expanded Storage. Just like when Expanded Storage was introduced, SCM is expected to be exploited by a number of different components. If you don't have SCM in your system you won't be able to take advantage of these new capabilities.

---

[1] See http://www-01.ibm.com/support/docview.wss?uid=swg27039487 for IDAA v4 and http://www-01.ibm.com/support/docview.wss?uid=swg27035960 for v3.

[2] This is a web deliverable, see: http://www-03.ibm.com/systems/z/os/zos/tools/downloads/#RSME

In particular, SCM's first use was for paging space. Not only is paging to SCM much faster than paging to disk, but if SCM is available[3], 1MB pages can be pageable as well. However, the software allocating the 1MB pages has to change to allocate the 1MB pages as pageable. The currently available version of Java 7 on z/OS optionally will do this. Over time I expect that there will be more and more things moving to pageable 1MB pages because using 1MB pages greatly improves the effectiveness of the Translation Look-aside Buffer, which improves overall system efficiency.

With the software and hardware in place, configuring Flash Express is very simple. From the HMC you assign an allocated and maximum amount of Flash Express that each LPAR can access. I did this when we first put installed our zEC12s so I wouldn't have to worry about it later.

From the operating system perspective, there is a new optional parameter in IEASYSxx that you can use to limit the amount of SCM reserved for paging, but I just let it default to "ALL" until we have something else to exploit it.

Once you IPL with a Flash Express allocation for the LPAR, the "D ASM" command will show the SCM space below your page datasets that are defined on disk. Until it fills up, SCM will be used for paging instead of the disk datasets. While you may be able to eliminate your disk paging datasets, I'm keeping ours around because our DR system does not have Flash Express.

Essentially, once you have Flash Express allocated to a z/OS LPAR, it's automatically used for paging, no additional work needed. Once I was told that the RSM Enablement Offering had been installed, I went to check what I'd need to do to enable it. Because I had already assigned an allocation to the system, I found the system was already paging to SCM. I like that simplicity!

But I wanted to see SCM really work under load and I wanted to see 1M large pages actually paging. On our smallest test system (it only has 4.5GB of memory allocated to it), I set up a series of batch jobs to run Java programs that would repeatedly fill up their allocated heap space. Pretty much any Java program will do this if it runs long enough, but I wrote some bad code to do it faster. At the busiest I had very active JVMs with a total of around 6GB of heaps running on a system with only 4.5GB of real memory.

As you can imagine, paging to SCM, while much faster than paging to disk, is still a performance impediment. While system responsiveness was definitely impacted, it survived notably better than I would have expected. If it had been a real production system my phone would have been ringing for sure, but it didn't fail, and it was still responsive enough that I was able to move around in the system, cancel and submit jobs, and look at things in RMF III.

---

[3] Initially it was thought that SCM was required before the system would allow 1MB pages to page. Customers have now discovered that this is not the case—1MB frames can be pageable even without SCM. But from a performance perspective, it's paging 1MB frames to disk is probably not a good idea.

Watching things in RMF III while the jobs were running was interesting: I haven't seen paging rates like that in a really long time, if ever! Similarly the RMF post-processor reports showed interestingly large numbers.

First I confirmed that I had some pageable 1MB pages:

1 MB FRAMES	FIXED			PAGEABLE		
	TOTAL	AVAILABLE	IN-USE	TOTAL	AVAILABLE	IN-USE
MIN	716	0	35	504	0	424
MAX	716	681	716	504	80	504
AVG	716	352	364	504	2	502

So I only had about a half a GB of pageable 1MB pages, but that's something. I used the RMF III panels to figure out which JVMs had received large pages and saw that they were in fact paging.

The Page Data Set activity report showed just how busy SCM really was:

P A G E D A T A S E T A C T I V I T Y

z/OS V1R13 SYSTEM ID OST1 DATE 06/04/2014 INTERVAL 15.00.401
RPT VERSION V1R13 RMF TIME 14.15.02 CYCLE 1.000 SECONDS

NUMBER OF SAMPLES = 884

PAGE DATA SET AND SCM USAGE

PAGE SPACE TYPE	VOLUME SERIAL	DEV NUM	DEVICE TYPE	SLOTS ALLOC	SLOTS USED MIN	SLOTS USED MAX	SLOTS USED AVG	BAD SLOTS	% IN USE	PAGE TRANS TIME	NUMBER IO REQ	PAGES XFER'D	V I O	DATA SET NAME
PLPA	PLP010	4F28	33903	179	179	179	179	0	0.00	0.000	0	0		SYS1.OST1.PAGE.PLPA0
COMMON	PLP010	4F28	33903	71999	25227	25227	25227	0	0.00	0.000	1	1		SYS1.OST1.PAGE.COMMON0
LOCAL	PLC010	4F20	33903	599399	0	0	0	0	0.00	0.000	0	0	Y	SYS1.OST1.PAGE.LOCAL0
LOCAL	PLC011	4F21	33903	599399	0	0	0	0	0.00	0.000	0	0	Y	SYS1.OST1.PAGE.LOCAL1
LOCAL	PLC012	4F22	33903	599399	0	0	0	0	0.00	0.000	0	0	Y	SYS1.OST1.PAGE.LOCAL2
LOCAL	PLC013	4F23	33903	599399	0	0	0	0	0.00	0.000	0	0	Y	SYS1.OST1.PAGE.LOCAL3
SCM	N/A	N/A	N/A	8389K	425311	1489K	1317K	0	90.61	0.000	950433	12.88M		N/A

As I was looking at the RMF reports I wondered: what does a "page" now mean when RMF indicates that so many pages were transferred or paged in or out? Is that a count of both 4K and 1M pages? I couldn't find a clear answer in the documentation, so I opened a question with IBM. It turns out that the counter for page movement is kept in 4K page equivalents. If a 4K page is moved, it's counted as 1. If a 1MB page is moved, it's counted as 256.

So when we see 12.88 million pages transferred in the report above, that means around 49GBs were transferred during the interval. Unfortunately we don't know how much of this was large pages vs. 4K pages. I opened an RFE[4] to suggest that IBM collect and report on 1M paging separate from 4K paging. Because I've just barely started to think about paging 1M pages, I'm not sure that there's a whole lot of value to this, but it seems like information that could be useful to know. For example, if an address space shows a page-in rate of 512 pages/sec, it seems that paging 2 1M pages is probably a different impact to the address space than 512 individual 4K pages.

The test system was the smallest LPAR on a zEC12 410 machine. During the test it consumed up to about 10% of the entire machine just to do the paging work. I expect that a production system and systems that are not running on sub-capacity machines may be able to page a lot more to Flash Express than what I was able to demonstrate in my test. Nonetheless, I believe my little test showed that Flash Express will support relatively high paging rates.

---

[4] See Request For Enhancement 54608, accessible from DeveloperWorks at http://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=54608 feel free to vote for it if you think it makes sense!

While it is still true that the best paging is no paging, my belief is that SCM will reduce the performance risk of having started tasks with relatively large memory allocations. In particular, I'm thinking about the Websphere servants used for development and test purposes that are usually idle, but can start actively using 100s of MBs of memory at any moment. As we make pageable 1M pages available for such address spaces, the TLB efficiency should increase, improving system performance as well.

As always, if you have questions or comments, you can reach me via email at ~~sachapman@aep.com~~scott.chapman@epstrategies.com.