SOLUTION BRIEF

Apache HBase* Software

Block Cache Testing in File Mode

Calculating the throughput when file mode is configured to reside on different devices.

"The HBase community delivers a new cost- and memory-flexible way to deal with falling out of DRAM-based heap memory in Java. When you fall out of memory where you fall out to matters."

Frank Ober
Solutions Architect, Intel Corporation

Scope

Apache HBase* software offers two modes of caching: L1 cache (on-heap), and a much larger L2 cache (BucketCache). This caching can be done in RAM (on-heap or off-heap) or in file. The following performance test uses BucketCache in file mode with SATA Hard Disk Drive (HDD), SATA Solid State Drive (SSD), and Intel® Solid State Drive Data Center Family for PCIe® (Intel® SSD DC P3700 Series). The goal of the test is to calculate the throughput received when configuring the file mode to reside on different devices.

Cluster Set Up

The cluster set up for the experiments consists of a two-node cluster as follows:

NODE 1	NODE 2
NameNode	HRegionServer
HMaster*	DataNode
Zookeeper*	

To avoid a network bottleneck caused by the data transfer, the client should run in the same node as HRegionServer.

The test system was configured with 32 logical cores with a total of 48 GB of DRAM memory.

- JDK version – 1.8
- HBase version – HBase - 0.98.12
- Java heap space – 10 GB
- File mode bucket cache size – 500 GB

Client Tool

YCSB*, a simple Java-based client load driver tool, was used to measure the performance.

Devices used in these experiments[1]:

NAME	TYPE	PRODUCT CODE	SERIAL NUMBER	CAPACITY	MODEL
Seagate	HDD	9YZ164-003	Z1N40A 1E	800GB	ST100NM0011
Intel® SSD DC S3700 Series	SATA SSD	G62042-201	BTTV246200T1800IGN	800GB	SSDSC2BA800G3
Intel SSD DC P3700 Series	PCIe SSD	H26406-304	CVFT4480008A2P0EGN	2TB	SSDPEDMD020T4

As suggested by the HBase community, to turn off Linux* kernel swap routines the swappiness in these systems was set to 0. This is especially useful in application memory managed systems.

Experiments

Each row had 10 columns and were 100 bytes in width. (Every row is ~1 kB+ in size.) Typical key value tests use the 1 kb average size, especially for standardized benchmarking purposes.

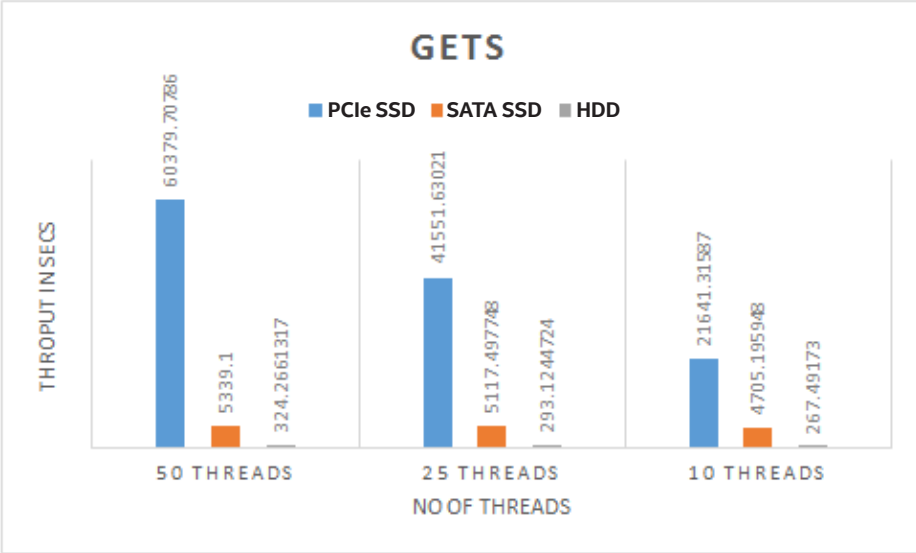
Tables were loaded with ~500 G of data and the BucketCache size is 500 G, with a goal of enabling the entire data to fit into the cache. Before the experiment the entire data was loaded to BucketCache.

A YCSB benchmark was run with a read work load of 10, 25, and 50 threads.

With this load, where the entire data set could not fit into the server's DRAM, a significant difference was observed in the performance throughput measured across various devices.

"Gets" are considered to be more random and the effect of OS caching is avoided by running a script that clears the OS page cache every five seconds in a pure read-based workload, which enables clear measuring of the reads happening directly from these devices rather than some of the data that may come from the OS page cache.

One more observation is that when the thread count was increased to 100, the database still maxed out at 60k "get" operations per second, which means that with 50 threads the device hit the maximum performance limit.



All the readings here are throughput measured as Operations per Second. A "get" is a simple read operation. (Higher values are better).

NO OF THREADS	PCIe SSD	SATA SSD	HDD
50 threads	60379.70786	5339.1	324.2661
25 threads	41551.63021	5117.497748	293.1245
10 threads	21641.31587	4705.195948	267.4917

Intel SSD Data Center Family for PCIe performs 81x to 189x compared to SATA HDDs.
Intel SSD Data Center Family for PCIe performs 4x to 11x compared to SATA SSDs.

Conclusion

Based on the experiments detailed in this Solution Brief, it is clear that the latest Intel SSD DC P3700 Series devices are superior storage devices for this configuration and NVM Express™ latency architecture. In order to effectively use these devices, HBase may have to create a layered caching principle with the hottest blocks (very frequently accessed) being in the DRAM. Java prefers to manage in DRAM first. Less frequently accessed blocks can be moved to Intel PCIe SSDs to avoid the performance lag of retrieving these blocks from HDFS altogether, and provides a more effective Layer 2 to DRAM than other storage devices have provided before. Watch for even lower latency storage components on the horizon with Intel® 3D XPoint™ technology.

For more information on Intel SSD Data Center Family for PCIe, visit **www.intel.com/SSD**

[1] The HBase software contributors who work for Intel and designed and ran these tests, used an older generation non-OEM Intel-based 2 socket server (S2600CP) with 64GB DDR3 memory and two Intel® CPU E5-2680 at 2.75GHz, 8 real CPU cores. I/O specific tests don't require the latest generation Intel processor family to show benefits of higher throughput as they leverage very few processor resources. NVMe™ is highly efficient in using processor resources as shown in this study: https://communities.intel.com/community/itpeernetwork/blog/2014/06/16/intel-ssd-p3700-series-nvme-efficiency

Printed in USA 160125/fo/ra ♻ Please Recycle 333614-001US