Support Persistent Memory as HDFS Cache Backend

Feilong He <feilong.he@intel.com>, Rakesh Radhakrishnan <rakeshr@apache.org>, Sammi Chen <chirnyee@gmail.com>, Uma Gangumalla <uma.gangumalla@intel.com>, Jian Zhang <jian.zhang@intel.com>

Background

Centralized Cache Management feature [4] was introduced into HDFS in HDFS-4949. With the help of HDFS cache, cluster memory can be utilized more easily by high level application frameworks, such as Spark, MR, Hive, Pig, and Impala. To accelerate workload, these frameworks can explicitly ask HDFS to cache dataset and schedule tasks with considering DRAM cache locality.

However, more and more applications require memory to hold as much data as possible to speed up their workloads. And for a HDFS cluster, memory is shared by OS, HDFS and other applications. To alleviate the pressure of memory resource, Persistent Memory (PM), also known as storage class memory (SCM), emerged in the industry.

Persistent Memory is a new class of memory technology with performance characteristics that falls between DRAM and flash. The below diagram highlights where PM fits in the storage media hierarchy. It has a much lower price per capacity unit compared with DRAM and much lower IO latency compared with SSD [3].

For flash and HDD, data access is at block and page levels. PM, like DRAM, is accessed on a cache line base, meanwhile it can offer a relatively larger space to cache data at a relatively lower cost in comparison with DRAM.
HDFS Cache Workflow with PM as Cache Backend

As the below figure shows, the basic workflow keeps unchanged if PM is used as cache backend. There is no difference for end user to use the current HDFS cache directives.

PM has non-volatile characteristic. But to simply the implementation, in our initial design, PM will not keep cache when DataNode restarts. It behaves similarly as DRAM.
HDFS PM Cache Details

1) Configuration

To enable Persistent Memory as cache backend, please configure the following property in hdfs-site.xml.

`dfs.datanode.cache.pmem.dirs`

This property specifies the cache directory on PM. For multiple directories, they should be separated with “,”, e.g. “/mnt/pmem0, /mnt/pmem1”. A secondary directory “hdfs_pmem_cache/” will be created to store cache data. The default value of this property is empty. If this property is configured, the PM volume capacity will be detected. So there is no need to configure `dfs.datanode.max.locked.memory` which just works for DRAM. The configuration for this property will be shaded if PM volume was configured.

How to enable PM on your OS is out of the scope of this document. Please refer to http://pmem.io/ and https://github.com/pmem/pmdk, or consult your device vendor [1, 2].

In our design, user should choose either DRAM or PM for Centralized Cache Management on a DataNode. So a block can be cached either to memory or to PM. If PM volume is specified, PM cache is enabled and DRAM cache is disabled. It is acceptable that some DataNodes are configured with PM and some are not. Lazy Persist Write will be disabled if PM cache is enabled. We will consider to support Lazy Persist Write on PM in the future.

There are two implementations for PM cache. The default one is pure Java based implementation and the other is native implementation which leverages PMDK library to
improve the performance of cache write and cache read. Which implementation is available to user is decided in build phase. A build option makes the difference.

To enable PMDK based implementation, please follow the below steps.

i. Install PMDK library. Please refer to the official site http://pmem.io/ for detailed information.
ii. Build Hadoop with PMDK support. Please refer to "PMDK library build options" section in `BUILDING.txt` in the source code.
iii. To verify that PMDK is correctly detected by Hadoop, run the `hadoop checknative` command.

2) Response to cache directives

HDFS PM backed read cache supports all the current HDFS cache directives, which is consistent with memory backed read cache. Once DataNode receives a DNA_CACHE command, it will try to pull corresponding blocks from HDD to PM. When a DNA_UNCACHE command is received, DataNode will remove the cache from PM.

If multiple PM directories are configured, a round-robin policy is used to select an available volume for caching a block. The selection policy can be extended to consider space usage of PM in the future.

If a PM volume is full, an exception will occur and block will fail to be cached to this volume. Consistent with DRAM cache, PM cache doesn’t have cache eviction mechanism.

3) Data read

When DataNode receives a data read request from a client, if the corresponding block is cached into PM, DataNode will instantiate an InputStream with the block location path on PM (pure java implementation) or PM cache address (PMDK based implementation). Once the InputStream is created, DataNode will send the cache data to the client.

Next Step

As PM has non-volatile characteristic, HDFS PM Cache can be enhanced by supporting persistent read cache. Thus, the cache warmup time can be significantly reduced in DataNode restarts scenario.

Meanwhile, in our plan PM will be enabled in HDFS Lazy Persistent Write feature, which can bring large write cache capacity and reduce data loss risk.

Reference