Doug Meil
August 1, 2011
HBase-4089 – Block Cache Report, 1st pass Design

General Statement of Need

Currently, RegionServer metrics have the total block counts, blockCache memory used and free, but there is no way to find out what is actually *in* the cache.

The proposal is a structure that would summarize the block cache contents on a per table and CF basis:

- total # of blocks in the cache
- total bytes in cache
- average time in the cache (bonus, but not required in first implementation)

This document is stating the general requirements and is presenting a 1st pass design. Not everything is figured out – comments welcome.

Use Cases for Output

There are several use cases for obtaining this information.

1. **UI**

    a. Having this structure available in the RegionServer's admin web pages would be useful.

2. **RegionServer API**

    a. Minimally, in order to satisfy the web-page use-case, a change to the RegionServer API would be required.

3. **Configurable Output and 3rd Party Operational Tools**

    a. Ganglia, etc.

        i. 3rd party tools like Ganglia would conceivably be interested in this information.

        ii. As an aside, though, I'm not entirely sure how it would handle it because the resulting information is not a single datapoint, but a list of complex structures.

1. E.g., would it require turning every table/CF summary statistic into a new metric?

b. JMX

   i. I'm assuming that folks would be interested in seeing this information exposed via JMX, but I have the same question about use (i.e., not a single "metric" but returning a list of complex structures).

   ii. For what it's worth, at Explorys we found that using Jconsole to connect to the cluster made HBase unstable, so we don't do it.

c. Custom Output

   i. The idea that Todd had raised on the dist-list was a configurable sink for the summary structure such that it could either be written to custom logs, or emailed, etc.

   ii. HBase Logs

      1. It is the opinion of the author that HBase should be "self sufficient" to a great extent in terms of diagnostic capabilities. For example, an HBase user should not be *required* to install additional tools to diagnose HBase.

      2. Can additional tools be helpful? Absolutely. But it should not be a requirement for every user to write a custom script or install a 3rd party framework to capture this information (capture is the important point here, not processing).

      3. Thus, the capability to periodically spool a blockCache summary report to the HBase logs (e.g., write output every XXXX), with a default of 5 minutes or something.

      4. If a user wanted to make a cursory manual inspection of what is changing over time, the logs will show the contents with no additional capture responsibilities.

d. Else?

Proposed Changes

1. **BlockCache**

    a. The interface *BlockCache* would provide a method:
    ```
    List<BlockCacheSummary> = getBlockCacheSummary();
    ```

2. **BlockCacheSummary**

    a. This class would have the following structure:

        i. Table
        ii. ColumnFamily
        iii. # of blocks
        iv. total bytes in memory

    b. Future features

        i. Average time in block cache. (nice to have, but not required in initial implementation. No time-in-cache information currently exists in *LruBlockCache*.)
            1. Conceivably, this could be supported by adding an attribute to *CachedBlock* for the time it was added to the cache.

3. **Summary Implementation**

    a. *LruBlockCache* and *SimpleBlockCache* would both implement `getBlockCacheSummary()`.

        i. This intended to be a point-in-time view of the block cache.

    b. LruBlockCache

        i. Iterate over the *ConcurrentHashMap* 'map' instance.

            1. The values are *CachedBlock* instances, which contain what we need.

        ii. Question: structure of block-name.

            1. This is an example of a block-name I got out of the cache from a unit test:
            2. `hdfs://localhost:51432/user/doug.meil/-ROOT-/70236052/info/18388013388958181870`

a. `-ROOT- = table`
b. `70236052 = region`
c. `info = CF`

3. The big number at the end looks like a StoreFile. Where are the 'blocks'?

4. More research needed here.

c. SimpleBlockCache

i. This class is only referenced from a test program, but it's not even a unit test.
   1. Does this belong in the code-base any more?

ii. It's referenced from a *RandomSeek's* 'main' with hard-coded paths…

```
Path path = new Path("/Users/ryan/rfile.big.txt");
long start = System.currentTimeMillis();
SimpleBlockCache cache = new SimpleBlockCache();
```

4. RegionServer API

a. Add `List<BlockCacheSummary> getBlockCacheSummary` to *HRegionInterface*

b. Add implementation of `getBlockCacheSummary` in *HRegionServer*

c. Question: RegionServer API effect on *BlockCacheSummary* class.

i. Everything that is returned by the RegionServer API implements *Writable*.
ii. Does that mean that *BlockCacheSummary* returned from *BlockCache* should implement *Writable*, or is there another class that represents the *BlockCacheSummary* that implements *Writable* that has the same information.

5. Web UI

a. This could either be jammed onto the existing RegionServer detail page (preferably at the top), or a new web-page linked from the RegionServer detail page could be created.

b. Because this is summary information (reminder: at table/CF level), it might still fit on the same page.

6. Configurable Output

a. This is where the hand-waving starts.

b. JMX

i. RegionServer statistics are exposed via *RegionServerStatistics* (which extends *MetricsMBeanBase*)

ii. Question: does this information go in *RegionServerStatistics*, or a new *BlockCacheSummaryStatistics* class?

1. It feels like the latter, but I'd like others to comment on this.

c. Ganglia

i. At the moment I'm not entirely sure how Ganglia integrates with HBase (e.g., JMX vs. direct RS API calls).

d. Custom Output

i. Serious hand-waving here.

ii. What if I wanted this information logged to a file every 15 minutes? Could HBase do this for me? Or do I have to write it all myself?
1. Or what if I wanted it sent over another protocol

iii. This seems like an odd question to ask only in the context of this ticket, but this is related to a similar output question in HBASE-4147 (StoreFile read report)

iv. I think a more general pattern for metrics/statistics output is needed.

v. Straw-Man Proposal

1. Background thread that has a configurable poll interval to obtain *BlockCacheSummary*

    a. In the context of HBASE-4147, this background thread would be responsible for all periodic stats output.

2. And has pluggable output formatters.
    a. E.g.,
    b. MyBlockCacheSummaryFormatter extends BlockCacheSummaryFormatter
        i. formatOutput(BlockCacheSummary summary)
        ii. // your code goes here.

3. But some formatters are shipped out of the box, such as the log-file one.