Intra-region scanners

The HBase patch that has been developed under HBase-12031 JIRA allows efficient implementation of multiple parallel scanners over a single HBase region.

Design

There is an internal lock inside HFileReaderV2/V3 which prevents multiple scanners from operating efficiently on a same region. Only one scanner at a time can operate in a sequential (fast) mode, all others do positional reads (slow) which is sub-optimal from performance point of view. The positional read operations are inefficient because of a small HFile block size (usually less than 64Kb). We can not increase HFile block size w/o sacrificing small read/scan performance, but we can do read-ahead and prefetch some data in advance. This is a main design idea: do positional reads with read-aheads. The default read-ahead buffer size is large enough (4MB) to make positional read performance comparable to sequential read mode. To eliminate possible OOME in RS we do not allow more than **hfile.readahead.active.buffers read ahead buffers per Region Server.** When this number is exceeded, the new Scanner operates in a regular mode (read without prefetching). The feature is disabled, by default. To enable it set **hfile.readahed.enabled to true.**

Sequential scans vs. skip scans

What happens if we do mostly skips, not sequential reads? In this case, if we can not detect that we run a skip scan mode, we will see a significant performance hit, because of a useless data prefetches. The simple next offset predictor **MovingAverageWithoutOutliersPredictor** has been developed to detect skip scan mode of operation. It calculates (predicts) total number of HFile blocks which will be read from read ahead buffer. If this number is greater or equals to **hfile.readahead.min.blocks** than scan operates in a read-ahead mode (with prefetches), if less than read-ahead is disabled (no data prefetching).

Applying patch

1. Go to $HBASE_HOME directory
2. Run: patch -p1 < path_to_patch_file

Parallel scanners configuration

The following are new configuration options (need to be specified in hbase-site.xml):

- **hfile.readahead.active.buffers -** Number of read-ahead buffers per RS. Default: 50
- **hfile.readahead.buffer.size -** Read-ahead buffer size. Default: 4Mb.
- **hfile.readahed.enabled** - Enable feature. Default: false.
- **hfile.readahead.min.blocks** - Minimum number of blocks found in a read-ahead buffer. Default: 8

Performance benchmarks

Two tests with BLOCK_SIZE = 16K and 4M respectively

Test description:

- HBase 0.98.3
- Create table with one CF and 3 columns; MAX_FILE_SIZE = 100G (to avoid region split), MAX_VERSIONS= 10
- Set split policy to ConstantSizeRegionSplitPolicy.
- Load 2M rows (3 col and 10 versions each) - total 60M cells
- Major compact
- Make sure we have 1 Region, 1 store file
- sudo purge (Mac OSX) to flush file cache
- Run two RegionScanner's in parallel. First scanner scans first half, second - second half of the region
- Measure scan time.

Results:

1 thread:

16K block size - 66 sec (~ 50MB/sec combined)

4M block size - 9.5 sec (> 300MB/sec combined)

4 threads:

4M block size - 7.0 sec (> 430MB/sec combined)

Scanner during compaction test

We run in parallel sequential scanner and major compaction on a table.

Test description:

- HBase 0.98.6-SNAPSHOT
- Create table with one CF and 3 columns; MAX_FILE_SIZE = 100G (to avoid region split), MAX_VERSIONS= 10
- Set split policy to ConstantSizeRegionSplitPolicy.
- Load 2M rows (3 col and 10 versions each) - total 60M cells
- sudo purge (Mac OSX) to flush file cache
- Major compact
- Make sure we have 1 Region, 1 store file
- Start major compaction and one RegionScanner in parallel.
- Measure scan time.

Results:

W/o patch : 75 sec

With patch: 19 sec

With patch and w/o compaction: 13 sec