[HBASE-13071] Hbase Streaming Scan Feature: Evaluation Results

February 22, 2015

Eshcar Hillel (eshcar@yahoo-inc.com)

Assumption: the application processing time can be balanced by the time it takes to retrieve the data. Then, with an *asynchronous* scanner the time an application waits for the next piece of data (row) should be equivalent to reading local data.

We used an extended version of ycsb to measure the accumulated latency of next steps in large scans.

Experimental Settings

We set up a small cluster of 3 hdfs nodes, with a single region server (and a master). All configuration properties are set to default.
We loaded a table with 1M records, each record with 10 fields of 100b, total of 1GB table.
We then ran 1000 big scans by a single thread, each scanning 100K records.

We extended ycsb to measure the time each next step of a scan waits for the result.
To simulate the time the application processes the results we added delays averaging 20us per next operation (this delays are not part of the latency we measure).
We experimented with different values of prefetch size (hbase.client.scanner.caching).
Specifically, we ran sync scanner with 100, 1000, and 10,000 prefetch size, and async scanner with 1000 and 10,000 prefetch size.

Experimental Results

The figure below depicts the accumulated latency of the x-th step at each experiment (averaging over the 1000 scans).
The following table summarizes the average latency over all 100K next steps, and their respective stdev.

Latency us (microseconds)	sync prefetch=100	sync prefetch=1000	sync prefetch=10000	async prefetch=1000	async prefetch=10000
AVG	113	41	31	19	8
STDEV	1,101	1,246	2,923	514	868

Note that simulating the delay (20us) at the application level gives the async scanner some advantage as it is working in the background to retrieve additional data.
For example, it is easy to see that for a sync scanner with prefetch size set to 10,000, there is a big leap every 10,000 steps, when the client is waiting for the results. Likewise, in sync scanner with prefetch size set to 1000, the leaps are smaller, yet visible.
The async scanners results are much smoother.
In addition, while the async scanner with prefetch size set to 10,000, pays a price at the first step which is similar to sync scanner, later on it behaves almost like a true streamer.

This last point gives rise to further optimizing the scan by automatically tuning the prefetch size, starting with a small number and gradually increasing it to the optimal size (both can be configured by the user).
This optimization is not included in our implementation.