HBase importing

Ryan Rawson, Systems Architect
Stumbleupon (@ryanobjc)

Problem

- I have 12 billion rows to get into HBase
- I have lots of small rows
- I have huge tables (hundreds of regions)

Attempt 1: map reduce w/write buffer

- Use map reduce to get 150x parallelism
- Use setAutoFlush(false) for write buffer
- Imports at about 100-200k ops/sec
- 5+ days to import (sad)

Attempt 2: no WAL

- Doesn't make things much faster
- New distributed coordination problem happens...

Why is my import slow?

- Behaviour:
  - Cluster goes to maximal speed, but then pauses nearly to 0
  - Speed goes back up
  - Never gets much faster than 100k/sec

Even worse

- A single threaded app can't insert much more than 1000-2000 ops/sec (ouch)

What's going on here?

- HBase cluster: 19 datanode & regionserver
- Table has 500-1500 regions
- But my write buffer is 20mb, why can't I get speed?

Put buffer code

```
Pseudo code:

- sort write buffer by row key
- foreach region worth of rows:
 - call regionserver.put(Put[])
 - if fail, relocate rows
        (due to split/reassign)
```

What's wrong with this?

- If there are 1000 regions, may need to do 1000 RPC calls to 19 servers
- RPC overhead = costly
- Calls are done serially!

Implications

- A single slow regionserver will cause all clients to pile up on that server and wait
- Need to wait for previous call to finish to do next

There must be a better way!

- If we have 19 regionservers, we should do 19 RPC calls
- Calls should be done in parallel as well

HBASE-2066

```
Pseudo Code:

(do this N times)
- bucket rows into K regionserver
    addresses
- create K callables to do RPC
- threadpool... GO!
- collect failures and retry
```

How much better?

- Single threaded putter used to do max 2k/rows/sec
- Now can do up to 16k/rows/sec (!)

Ultimate

- Combine no WAL, map reduce and 2066
- Instantaneous peak put of 1.1m rows/sec on cluster (!!!!)

Aggregate Import

- Two indexes, 12b rows each, < 48 hours
- That is 24b rows representing 1200 GB (post compression)

Some settings

```
hbase.regionserver.globalMemcache.upperLimit = 0.3

hbase.regionserver.globalMemcache.lowerLimit = 0.15

hbase.hstore.blockingStoreFiles = 25

hbase.hregion.memstore.block.multiplier = 8

hbase.regionserver.handler.count = 30

hbase.regions.percheckin = 30
```

When can I have it?

- HBase 0.21
- Requires RPC version bump - can't go into 0.20.3 (sorry)