Lift the Ceiling of Throughputs

Yu Li, Lijin Bin

{jueding.ly, tianzhao.blj}
@alibaba-inc.com

Agenda

- What/Where/When
  - History of HBase in Alibaba Search
- Why
  - Throughputs mean a lot
- How
  - Lift the ceiling of read throughputs
  - Lift the ceiling of write throughputs
- About future

HBase in Alibaba Search

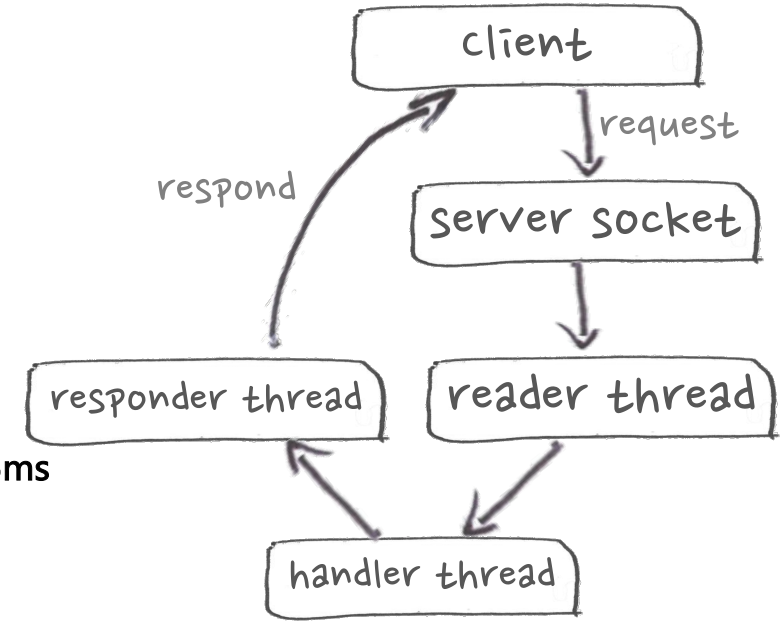
- HBase is the core storage in Alibaba search system, since 2010
- History of version used online
  - 2010~2014: 0.20.6→0.90.3→0.92.1→0.94.1→0.94.2→0.94.5
  - 2014~2015: 0.94→0.98.1→0.98.4→0.98.8→0.98.12
  - 2016: 0.98.12→1.1.2
- Cluster scale and use case
  - Multiple clusters, largest with more than 1,500 nodes
  - Co-located with Flink/Yarn, serving over 40Million/s Ops throughout the day
  - Main source/sink for search and machine learning platform

Throughputs mean a lot

- Machine learning generates huge workloads
  - Both read and write, no upper limit
  - Both IO and CPU bound
- Throughputs decides the speed of ML processing
  - More throughputs means more iterations in a time unit
- Speed of processing decides accuracy of decision made
  - Recommendation quality
  - Fraud detection accuracy

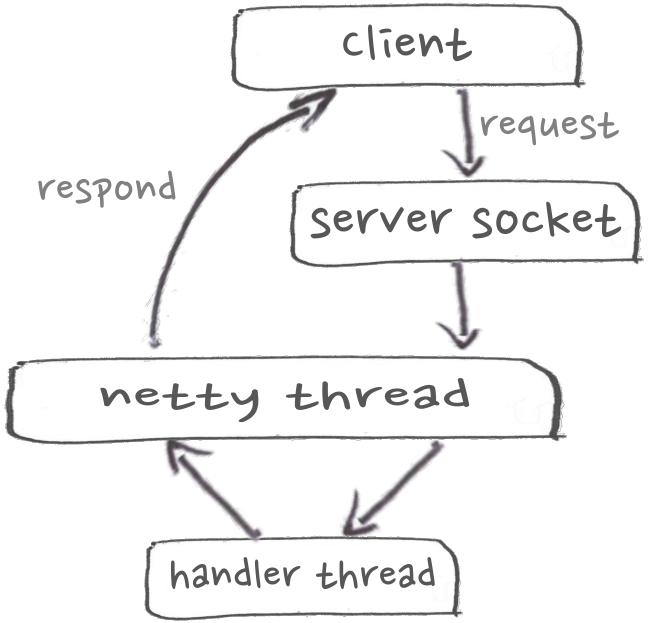
Lift ceiling of read throughput

- NettyRpcServer (HBASE-17263)
  - Why Netty?
    - Enlightened by real world suffering
      - HBASE-11297
    - Better thread model and performance
  - Effect
    - Online RT under high pressure: 0.92ms→0.25ms
    - Throughputs almost doubled



Lift ceiling of read throughput

- NettyRpcServer (HBASE-17263)
  - Why Netty?
    - Enlightened by real world suffering
      - HBASE-11297
    - Better thread model and performance
  - Effect
    - Online RT under high pressure: 0.92ms→0.25ms
    - Throughputs almost doubled



Lift ceiling of read throughput (con't)

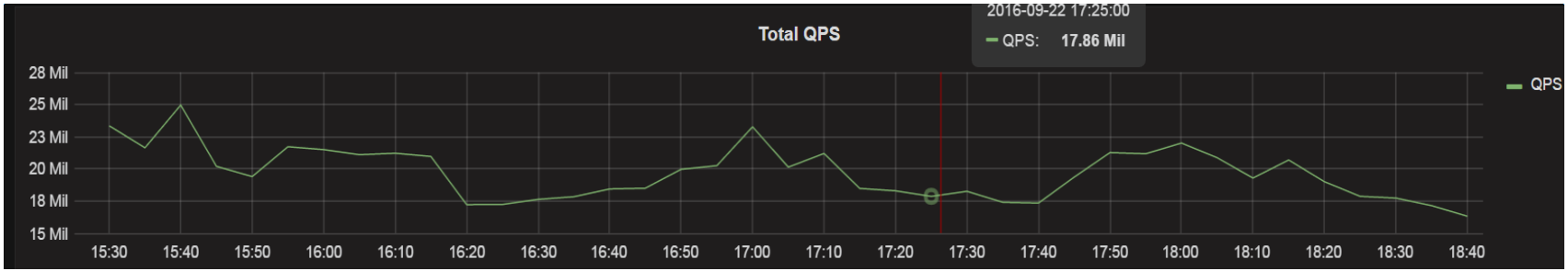
- RowIndexDBE (HBASE-16213)
  - Why
    - Seek in the row when random reading is one of the main consumers of CPU
    - All DBE except Prefix Tree use sequential search.
  - How
    - Add row index in a HFileBlock for binary search. (HBASE-16213)
  - Effect
    - Use less CPU and improve throughput， KeyValues<64B, increased >10%

Lift ceiling of read throughput (con't)

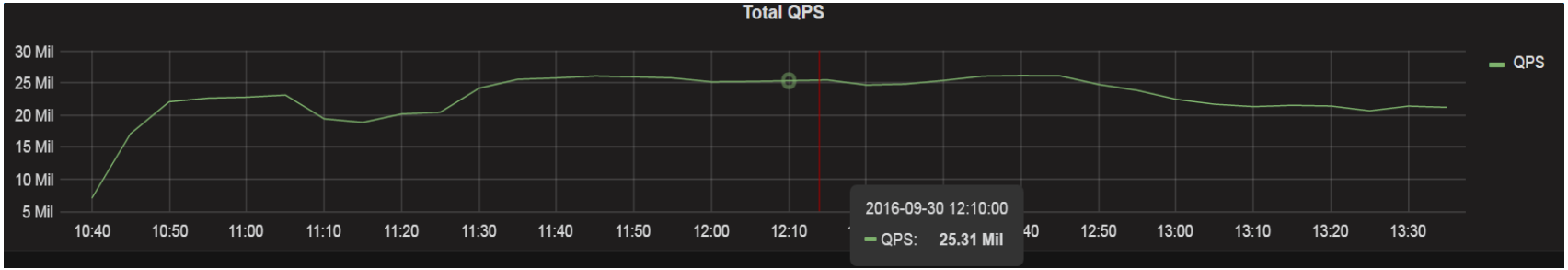
- End-to-end read path offheap
  - Why
    - Advanced disk IO capability cause quicker cache eviction
    - Suffering from GC caused by on-heap copy
  - How
    - Backport E2E read-path offheap to branch-1 (HBASE-17138)
    - More details please refer to Anoop/Ram's session
  - Effect
    - Throughput increased 30%
    - Much more stable, less spike

Lift ceiling of read throughput (con't)

- End-to-end read path offheap
  - Before



  - After



Lift ceiling of write throughput

- MVCC pre-assign (HBASE-16698, HBASE-17509/17471)
  - Why
    - Issue located from real world suffering: no more active handler
    - MVCC is assigned after WAL append
    - WAL append is designed to be RS-level sequential, thus throughput limited
  - How
    - Assign mvcc before WAL append, meanwhile assure the append order
      - Original designed to use lock inside FSHLog (HBASE-16698)
      - Improved by generating sequence id inside MVCC existing lock (HBASE-17471)
  - Effect
    - SYNC_WAL throughput improved 30%，ASYNC_WAL even more (>70%)

Lift ceiling of write throughput (cont'd)

- Refine the write path (Experimenting)
  - Why
    - Far from taking full usage of IO capacity of new hardware like PCIe-SSD
    - WAL sync is IO-bound, while RPC handling is CPU-bound
      - Write handlers should be non-blocking: do not wait for sync
      - Respond asynchronously
    - WAL append is sequential, while region puts are parallel
      - Unnecessary context switch
    - WAL append is IO-bound, while MemStore insertion is CPU-bound
      - Possible to parallelize?

Lift ceiling of write throughput (cont'd)

- Refine the write path (Experimenting)
  - How
    - Break the write path into 3 stages
      - Pre-append, sync, post-sync
      - Buffer/queue between stages
    - Handlers only handle pre-append stage, respond in post-sync stage
    - Bind regions to specific handler
      - Reduce unnecessary context switch

Lift ceiling of write throughput (cont'd)

- Refine the write path (Experimenting)
  - Effect (Lab data)
    - Throughput tripled: 140K → 420K with PCIe-SSD
  - TODO
    - Currently PCIe-SSD IO util only reached 20%, much more space to improve
    - Integration with write-path offheap – more to expect
    - Upstream the work after it's verified online

About Future

- HBase is still a kid – only 10 years' old
  - More ceilings to break
    - Improving, but still long way to go
    - Far from fully utilizing the hardware capability, no matter CPU or IO
  - More scenarios to try
    - Embedded-mode (HBASE-17743)
  - More to expect
    - 2.0 coming, 3.0 in plan
- Hopefully more community involvement from Asia
  - More upstream, less private

Q & A

Thank You!