The Impala Cookbook

From Cloudera's Impala Team

Updated Jan. 2017

*Currently an Apache Incubator project

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage in Impala
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

cloudera

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage in Impala
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

cloudera

© Cloudera, Inc. All rights reserved.

Physical and Schema Design - Outline

- Schema design best practices
  - Datatypes
  - Partition design
  - Complex types
  - Common questions
- Physical design
  - File format – when to use what?
  - Block size (option)

cloudera

© Cloudera, Inc. All rights reserved.

Physical and Schema Design - Data Types

- Use Numeric Types (not strings)
  - Avoid string types when possible
  - Strings => higher memory consumption, more storage, slower to compute (80% slower compared to numeric)
    - E.g date string “20161031” or unixtime “1479459272”, switch to bigint
- Decimal vs Float/Double
  - Decimal is easier to reason about
  - Currently can't use Decimal as partition key or in UDFs
- Use String only for:
  - HBase row key – string is the recommended type!
  - Timestamp – ok to use string, but consider using numeric as well!
- Prefer String over Char/Varchar (except for SAS)

cloudera

© Cloudera, Inc. All rights reserved.

Partition Design: 3 Simple Rules

1. Identify access patterns from the use case or existing SQL.
2. Estimate the total number of partitions (better be <100k!).
3. (Optional) If needed, reduce the number of partitions.

cloudera

© Cloudera, Inc. All rights reserved.

Partition Design – Identify Access Patterns

- The table columns that are commonly used in the WHERE clause are candidates for partition keys.
  - Date is almost always a common access pattern and the most common partition key.
- Can have MULTIPLE PARTITION KEYS! Examples:
  - `Select col_1 from store_sales where sold_date between ‘2014-01-31’ and ‘2016-02-23’;`
  - `Select count(revenue)from store_sales where store_group_id in (1,5,10);`
  - `Partition keys => sold_date, store_group_id`

cloudera

© Cloudera, Inc. All rights reserved.

Partition Design – Estimate the #partitions

- Estimate the number of distinct values (NDV) for each partition key (for the required storage duration). Example:
  - If partitioned by date and need to store for 1 year, then NDV for date partition key is 365.
  - num store_group will grow over time, but it will never exceed 52 (one for each state).
- Total number of partitions = NDV for part key 1 * NDV for part key 2 * … * NDV for partition key N. Example:
  - Total number of partitions = 365 (for date part) * 52 (for store_group) ~= 19k
- Make sure #partitions <=100k!

cloudera

© Cloudera, Inc. All rights reserved.

Partition Design – Too Many Partitions?

- Remove some “unimportant” partition keys.
  - If a partition key isn’t as routinely used or it doesn’t impact the SLA, remove it!
- Create partition “buckets” and specify bucket id in downstream select queries.
  - Use month rather than date.
  - Create artificial `store_group` to group individual stores.
  - Technique: use prefix or hash
    - Hash(store_id) % store_group size => hash it to store_group
    - Substring(store_id,0,2) => use the first 2 digits as artificial store_group
    - e.g. select … from tbl where store_id = 5

      and store_group = store_id % 50; //Assume 50 store groups

cloudera

© Cloudera, Inc. All rights reserved.

Nested Types: Known Issues

- The maximum size of a single row fed into a join build cannot exceed 8MB (buffered-tuple-stream block size). For complex types this means:
  - Size of all materialized collections (post projection and filtering) cannot exceed 8MB for certain plan shapes (expected to be rare).
- IMPALA-2603: Incorrect plan is generated for inline view referencing complex types.
  - Inline view has several relative table refs that refer to different ancestor query blocks within the same nesting hierarchy.

cloudera

© Cloudera, Inc. All rights reserved.

Schema Design – Common Issues

- Number of columns - 2k max
  - Not a hard limit; Impala and Parquet can handle even more, but…
  - It slows down Hive Metastore metadata update and retrieval
  - It leads to big column stats metadata, especially for incremental stats
- Timestamp/Date
  - Use timestamp for date;
  - Date as partition column: use string or int (20150413 as an integer!)
- BLOB/CLOB – use string
- String size - no definitive upper bound but 1MB seems ok
  - Larger-sized string can crash Impala!
  - Use it sparingly - the whole 1MB string will be shipped everywhere

cloudera

© Cloudera, Inc. All rights reserved.

Physical Design – File Format

- Parquet/Snappy
  - The long-term storage format
  - Always good for reading!
  - Write is very slow (reportedly 10x slower than Avro).
- Snappy vs Gzip
  - Snappy is usually a better tradeoff between compression ratio and CPU.
  - But, run your own benchmark to confirm!
- For write-once-read-once tmp ETL tables, consider text instead because:
  - Writing is faster.
  - Impala can perform the write.

cloudera

© Cloudera, Inc. All rights reserved.

Physical Design – Block Size

- Number of blocks defines the degree of parallelism:
  - True for both MapReduce and Impala
  - Each block is processed by a single CPU core
  - To leverage all CPU cores across the cluster**, num blocks >= num core**
- Larger block size:
  - Better compression and IO throughput, but fewer blocks, could reduce parallelism
- Smaller block size:
  - More parallelism, but could reduce IO throughput
  - Can cause metadata bloat and create bottlenecks on HDFS NameNode(NN RPC overhead 40K-50K/s), Hive Metastore, Impala catalog service

cloudera

© Cloudera, Inc. All rights reserved.

Physical Design – Block Size

- For Apache Parquet, ~256MB is good and no need to go above 1GB.
- Don't go below 64MB except when you need more parallelism!
- (Advanced) If you really want to confirm the block size, use the following equation:
  - Block Size <= p* t * c / s
    - p – disk scan rate at 100MB/sec
    - t – desired response time of the query in sec
    - c – concurrency
    - s - % of column selected
- Regularly compact tables to keep the number of files per partition under control and improve scan and compression efficiency

cloudera

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – The Basics

- Memory is used by:
  - Hash join – RHS tables after decompression, filtering and projection
  - Group by – proportional to the #groups
  - Parquet writer buffer – 256MB per partition
  - IO buffer (shared across queries)
- Memory held and reused by later queries
  - Impala releases memory from time to time in 1.4 and later

© Cloudera, Inc. All rights reserved.

Catalog memory usage

- Metadata cache heap memory usage can be calculated by
  - num of tables * 5KB + num of partitions * 2KB + num of files * 750B + num of file blocks * 300B + sum(incremental col stats per table)
  - Incremental stats
    - For each table, num columns * num partitions * 400B
- Catalog-Update topic size should be no more than 2GB
  - http://<statestore host>:25010/topics

© Cloudera, Inc. All rights reserved.

Memory Usage – Estimating Memory Usage

- Use Explain Plan
  - Requires statistics! Memory estimate without stats is meaningless.
  - Reports per-host memory requirement for this cluster size.
    - Re-run if you've re-sized the cluster!

```
+---------------------------------------------------------------------+
| Explain String                                                      |
+---------------------------------------------------------------------+
| Estimated Per-Host Requirements: Memory=26.51MB VCores=2            |
|                                                                     |
| 07:AGGREGATE [MERGE FINALIZE]                                       |
| |  output: sum(count(*))                                            |
| |                                                                   |
```

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – Estimating Memory Usage (Cont'd)

- EXPLAIN's memory estimate issues
  - Can be way off – much higher or much lower.
  - group by's estimate can be particularly off – when there's a large number of group by columns.
    - Mem estimate = NDV of group by column 1 * NDV of group by column 2 * … NDV of group by column n
  - Ignore EXPLAIN's estimate if it's too high!
- Do your own estimate for group by
  - GROUP BY mem usage = (total number of groups * size of each row) + (total number of groups * size of each row) / num node

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – Finding Actual Memory Usage

- Search for “Per Node Peak Memory Usage” in the profile.
- This is accurate. Use it for production capacity planning.

```
Execution Profile b8414c34981f3ec9:a52048d52a00fb1:(Total: 9s754ms, non-child: 0ns, % non-child: 0.00%)
  Per Node Peak Memory Usage: alan-OptiPlex-790:22000(11.77 MB)
    - FinalizationTimer: 0ns
```


© Cloudera, Inc. All rights reserved.

Memory Usage – Finding Actual Memory Usage (Cont'd)

- For complex queries, how do I know which part of my query is using too much memory or causing an Out-Of-Memory error (i.e. hitting the mem-limit)?
  - Look at the "Peak Mem" in the ExecSummary from the query profile!

ExecSummary:

Operator	#Hosts	Avg Time	Max Time	#Rows	Est. #Rows	Peak Mem	Est. Peak Mem	Detail
07:AGGREGATE	1	77.817ms	77.817ms	1	1	48.00 KB	-1.00 B	MERGE FINALIZE
06:EXCHANGE	1	14.534us	14.534us	1	1	0	-1.00 B	UNPARTITIONED
03:AGGREGATE	1	1s025ms	1s025ms	1	1	84.56 KB	10.00 MB	
02:HASH JOIN	1	8s300ms	8s300ms	366.55M	280.88M	7.48 MB	525.91 KB	INNER JOIN, PARTITIONED
|--05:EXCHANGE	1	22.500ms	22.500ms	183.59K	183.59K	0	0	HASH(s2.ss_sold_date_sk)
| 01:SCAN HDFS	1	1s791ms	1s791ms	183.59K	183.59K	288.00 KB	16.00 MB	tpcds_parquet.store_sales s2
04:EXCHANGE	1	19.189ms	19.189ms	183.59K	183.59K	0	0	HASH(s1.ss_sold_date_sk)
00:SCAN HDFS	1	1s782ms	1s782ms	183.59K	183.59K	408.00 KB	16.00 MB	tpcds_parquet.store_sales s1

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – Hitting Mem-limit

- Top causes (in order) of hitting mem-limit even when running a single query:

1. Lack of statistics
2. Lots of joins within a single query
3. Big-table joining big-table
4. Gigantic group by

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – Hitting Mem-limit (Cont'd)

- Lack of stats
  - Wrong join order, wrong join strategy, wrong insert strategy
  - Explain Plan tells you that!

```
+-----------------------------------------------------------------------------+
| Explain String                                                              |
+-----------------------------------------------------------------------------+
| Estimated Per-Host Requirements: Memory=10.00MB VCores=2                    |
| WARNING: The following tables are missing relevant table and/or column statistics. |
| dmart.rate                                                                  |
```

  - Fix: Compute Stats <table>
  - For huge table that compute stats takes too long to finish, you can manually set table/column stats

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – Hitting Mem-limit (Cont'd)

- Lots of joins within a single query
  - `select…from fact, dim1, dim2,dim3,…dimN where …`
  - Each dim table can fit in memory, but not all of them together
  - As of CDH 5.4/Impala 2.2,
    - Impala might choose the wrong plan – BROADCAST
    - Impala sometimes require 256MB as the minimal requirement per join!
  - FIX 1: use shuffle hint
    - `Select … from fact join [shuffle] dim1 on … join [shuffle] dim2 …`
  - FIX 2: pre-join the dim tables (if possible). few join=>better perf!

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage - Hitting Mem-limit (Cont'd)

- Big-table join big-table
  - Big-table (after decompression, filtering, and projection) is a table that is bigger than total cluster memory size.
  - CDH5.4/Impala 2.0 and later will do this (via disk-based join).
  - (Advanced) For a simple query, you can try this advanced workaround – per-partition join
    - Requires the partition key be part of the join key

```
Select … from BigTbl_A a join BigTbl_B b where a.part_key =
b.part_key and a.part_key in (1,2,3)
union all
Select … from BigTbl_A a join BigTbl_B b where a.part_key =
b.part_key and a.part_key in (4,5,6)
```

cloudera

© Cloudera, Inc. All rights reserved.

Memory Usage – Disk-based Join/Agg

- Disk-based join/agg should be your last resort to deal with hitting mem-limit.
- Rely on disk-based join/agg if there is only one join/agg operator in the query. For example:
  - Good: select a.*, b.* from a, b where a.id=b.id
  - Good: select a.id, a.timestamp, count(*) from a group by a.id, a.timestamp
  - OK: select large_tbl.id, count(*) from large_tbl join tiny_tbl on (id) group by id
  - Bad: select t1.id, count(*) from large_tbl_1 t1, large_tbl_2 t2 where t1.id=t2.id group by t1.id
  - Bad: select a.*, b.*, c.* from a, b, c where a.id=b.id and b.col1=c.col2;
- Always set the per-query mem-limit (2GB min) when using disk-based join/agg!

© Cloudera, Inc. All rights reserved.

Memory Usage - Additional Notes

- Use explain plan for estimate; use profile for accurate measurement
- Data skew can use uneven memory/CPU usage
- Review previous common issues on out-of-memory
- Even with disk-based join in Impala 2.0 and later, you'll want to review the Query Tuning steps to speed up queries and use memory more efficiently.

cloudera

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

© Cloudera, Inc. All rights reserved.

Hardware Recommendations

- 128GB (assigned to Impala) or more for best price/performance
- Spindles vs SSD
  - Spindles are more cost effective
  - Most workload is CPU bound; SSD won’t make a difference at all
- 10Gb network

© Cloudera, Inc. All rights reserved.

Cluster Sizing – Objective and Keys

- Objective:
  - The recommended cluster size should run the desired workload within a given SLA throughout the projected life span of the cluster.
- Keys:
  - Workload - defines the functional requirement
  - SLA – defines the performance requirement
  - Projected life span – how will things change over time?

cloudera

© Cloudera, Inc. All rights reserved.

Cluster Sizing - SLA

- Query Throughput
  - How many queries should this cluster process per second?
  - This is the more meaningful measurement of “computing power”
- Query Response Time
  - How fast do you need the query to run?
  - Typically, single query response time isn’t too meaningful because there are always multiple queries running concurrently!
  - Some use cases require very fast response time e.g powering a web UI.
- Will more people be running queries over time? This means higher query throughput!

cloudera

© Cloudera, Inc. All rights reserved.

Cluster Sizing - Workload

- From the workload, you'll want to know:
  - How much memory do you need?
  - How much processing power do you need? (i.e. how complex is the workload?)
  - How much IO bandwidth do you need?
- The bigger the cluster, the more total memory, CPU, and disk-IO bandwidth you have.
- But usually, the network bandwidth is fixed.

© Cloudera, Inc. All rights reserved.

Cluster Sizing – Memory Requirements

- How much memory do you need?
  - Any huge group by expression?
    - Per Node Mem >= number of distinct group * row size +(number of distinct group * row size) / num node
    - Number of distinct group: hard to guess; just get a rough ballpark.
    - Row size: # columns involved in the query *column width
    - Column width for int 4 byte, bigint 8 byte, etc. For string columns, take some rough guess.
    - Increasing the cluster size won't help much to reduce the per-node mem requirement.

© Cloudera, Inc. All rights reserved.

Cluster Sizing – Connection Requirements

- The larger the cluster, the more intra connections needed between impalads. With high concurrency and/or very complex queries, this could cause a connection storm and query failures.
  - Impala caches established connections and re-uses them. If the workload is executed again, the existing connection pool will be leveraged to satisfy each connection request.
- On a Kerberos secured cluster, each impalad needs to authenticate with every other impalad. Requires: N*N KDC tickets. This could overwhelm a single KDC server.
  - If the cluster has more than 200 nodes, consider using more KDC servers to balance load.

cloudera

© Cloudera, Inc. All rights reserved.

Cluster Sizing – Workload Complexity (Cont’d)

- (Advanced) If you’re ready to dive deep into workload analysis…
  - Typically, you can assume the following processing rate (3 columns, parquet format with snappy):
    - Scan node 8~10m rows per core
    - Join node (single join predicate) ~10m rows per sec per core
    - Agg node (single agg)~5m rows per sec per core
    - Sort node - ~17MB per sec per core
    - Parquet writer 1~5MB per sec per core
- From a query in the workload, based on the #join/agg you can estimate #rows passing through the operator node then estimate the effect of partition pruning.
- Using the above processing rate, you can estimate how much cpu time it took to process a query.

cloudera

© Cloudera, Inc. All rights reserved.

Cluster Sizing – Summary

- Cluster sizing depends on SLA and workload. You need to know both!
- Memory requirement for doing big join/agg in memory:
  - Total Cluster mem >= the 2nd largest big table in the join after decompression, filtering, and projection
  - Per Node Mem >= number of distinct group * row size +(number of distinct group * row size) / num node

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

cloudera

© Cloudera, Inc. All rights reserved.

Benchmarking – Why Run One?

- Understand how Impala performs, how it scales, how it compares to the current system
  - Measures query response time as well as query throughput
- Understand how Impala utilizes resources
  - Measure CPU, disk, network and memory

cloudera

© Cloudera, Inc. All rights reserved.

Benchmarking Impala – Preparing the Workload

- Should be relevant to (or satisfy) the business requirement.
  - Don't run `select * from big_tbl limit 10` – it's meaningless.
- Should not be dictated on the query form.
  - You should be prepared to change the query/schema to deliver a meaningful benchmark.
  - Tune the schema/query!
- Stay close to the query that you're going to run in production.
  - If the result has to be written to disk, then write to disk and DO NOT send results back to the client.
  - Don't stream all the data to the client (i.e. data extraction).
- Use a fast client: You're benchmarking the server, not the client, so don't make the client a bottleneck.

cloudera

© Cloudera, Inc. All rights reserved.

Benchmarking – Avoiding Traps

- It’s easier to start with a smaller data set and simpler query. Trying to run a complex query on a huge data set on a small cluster is not effective.
- A data set that’s too small can’t utilize the whole cluster. Have at least one block per disk.
- Disable Admission Control when you’re doing benchmark!

cloudera

© Cloudera, Inc. All rights reserved.

Benchmarking – Preparing the Hardware

- Should be as similar to go-live hardware as possible
- Recommended: at least 10 nodes with 128GB each
- CAUTION: If the cluster is too small (i.e. 3 nodes), it's very hard to see the effect of scalability and identify potential bottlenecks

© Cloudera, Inc. All rights reserved.

Benchmarking – Measuring Single-Query Response Time

- Use Impala-shell (simple, easy to use) with the -B option. This disables pretty formatting so client won't be the bottleneck.
  - `Impala-shell –B –q "<your query>"`
- To simulate the effect of buffer cache, run it a few times to warm the buffer cache before measuring the result.
- To simulate the effect without buffer cache, clear the buffer cache by running:
  `echo 1 > /proc/sys/vm/drop_caches`

cloudera

© Cloudera, Inc. All rights reserved.

Benchmarking – Measuring Query Throughput

- Benchmark using JDBC (or use Jmeter!, make sure you set NativeQuery=1)
- Input parameter: list of hosts to connect to, workload queries, duration of the benchmark, and number of concurrent connections.
- Each connection will pick a host to connect to and keep running a query for the specified duration.
- Report QPS.
- Just keep increasing the number of connections until QPS doesn't increase – that will be the QPS of the system.

cloudera

© Cloudera, Inc. All rights reserved.

Benchmarking – General Performance Notes

- Gather query profiles and system utilization
  - You can get all these from CM or /var/log/impalad/profiles!
- Performance vs Hive - Impala will ALWAYS be faster after proper tuning. If not, something is wrong with the benchmark.
- Impala vs Hive-on-Tez/Spark SQL benchmark: https://blog.cloudera.com/blog/2016/02/new-sql-benchmarks-apache-impala-incubating-2-3-uniquely-delivers-analytic-database-performance/
- See the open TPC-DS toolkit to run your own! https://github.com/cloudera/impala-tpcds-kit

cloudera

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Admission Control

- **Recommended**: Static partitioning a fixed memory for Impala and use Admission Control
  - See the “Memory Usage” section for determining memory usage!
- The best public resource is this new Impala RM ‘how to’ guide: http://www.cloudera.com/documentation/enterprise/latest/topics/impala_howto_rm.html

cloudera

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Preventing a “Runaway” Query

- “Runaway" query = user submitted a "wrong" query accidentally that consumes a significant amount of memory
  - Limit the amount of memory used by an individual query using `per query mem-limit`:
    - Set it from Impala shell (on a per query basis):
      `set mem_limit=<per query limit>`
    - Set a default per-query mem limit:
      `-default_query_options=‘mem_limit=<per query limit>’`
    - On impala 2.6 and later, you can set default per-query mem limit at pool level

cloudera

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Admission Control

- How to approach Admission Control config:
  - Workload dependent – memory bound or not?
    - "Memory bound" means that you've used up the memory allocated to Impala before hitting limit on cpu, disk, network.
  - Memory bound – use mem-limit
  - Non-memory bound – use number of concurrent queries

cloudera

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Admission Control (Cont'd)

- Goals of Memory limit setting:
  - Prevents each group of users from overcommitting system memory
  - Prevents query from hitting mem-limit
  - (Secondary) Simulates priority by allocating more memory to important group

cloudera

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Admission Control (Cont'd)

- Step 1: Identify sample workload from each user "group", such as HR, Analyst etc
- Step 2*: For each query in the workload, identify the accurate memory usage by running the query. This is the memory requirement for the query.
- Step 3: Minimal memory requirement for each group = max (mem requirement from the query set) * 1.2 (safety factor). Also, set the per query mem-limit with this number.
- Step 4: You can divide the memory based on % too, but each group should have at least the min mem derived from Step 3.
- NOTE: sum(mem assigned to all groups) can be greater than total memory available. This is OK.

cloudera

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Admission Control (Cont’d)

- More on step 2
  - If the memory estimate from the explain plan is inaccurate:
    - FIX: Use per-query limit to override it, but that will require you to submit query through the shell.
    - FIX: Adjust the pool mem-limit accordingly; if it’s over the estimate, give it a higher mem-limit and vice versa.

cloudera

© Cloudera, Inc. All rights reserved.

Multi-tenancy Best Practices – Admission Control (Cont'd)

- Limiting the number of concurrent queries
  - Goal:
    - Avoid over subscription to CPU, disk, network because this can lead to longer response time (without improving throughput).
  - Works best with homogeneous workload
  - With heterogeneous workload, you can still apply the same approach, but the result won't be as optimal.

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics - Overview

- Given that your query runs, know how to make it faster and consume fewer resources.
- Always `Compute Stats`.
- Examine the logic of the query.
- Validate it with `Explain Plan`.
  - Runtime Filter
- Runtime Analysis
  - Use `Query Profile` to identify bottlenecks and skew
  - Codegen

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – More on `Compute Stats`

- `Compute Stats` is very CPU-intensive, but on Impala 1.4 and later is much faster than previous versions.
- Speed Reference: ~40M cells per sec per node + HMS update time (1 sec per 100 partitions). ~7million cell per sec per node if codegen is disabled (If table contains timestamp, char column that don't support codegen yet)
- Total number of cells of a table = num rows * num cols
- Only need to recompute stats with significant changes of data (30% or more)
- `Compute Stats` on tables, not view

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Incremental Stats Maintenance

- `Compute Stats` is slow and through 2.0, Impala does not support Incremental Stats
- Column Stats (number of distinct value, min, max) can be updated by `Compute Stats` infrequently (when 30% or more data has changed)
- When adding a new partition, run a count(*) query on the partition and update the partition row count stats manually via `ALTER TABLE`
- You can set column stats manually via `ALTER TABLE` as well

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Incremental Stats Maintenance

- Impala 2.1 or later supports "Compute Incremental Stats", but use it only if the following conditions are met:
  - For all the tables that are using incremental stats, Σ(num columns * num partitions) < 650000.
  - The size of the cluster is less than 50 nodes.
  - For each table, num columns * num partitions * 400B < 200MB

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Examing Query Logic

- Sometimes, the query can have redundant joins, distinct, group by, order by (very common during migration). Remove them!
- For common join patterns, consider pre-joining the tables.
- Use the proper join: Example:

```
select fact.col, max(dim2.col)
from fact, dim1, dim2
where fact.key = dim1.key and fact.key2 = dim2.key
```

  - The join on dim1 should be a semi-join!

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Validating `Explain Plan`

- Key points:
  - Validate join order and join strategy.
  - Validate partition pruning or HBase row key lookup.
- Even with stats, at times the join order/strategy might go wrong (particularly with layers of views). Can force join order by using STRAIGHT_JOIN hint

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Validating Join Order & Strategy

- Join Order
  - RHS is smaller than LHS
- Join Strategy - BROADCAST
  - RHS must fit in memory!

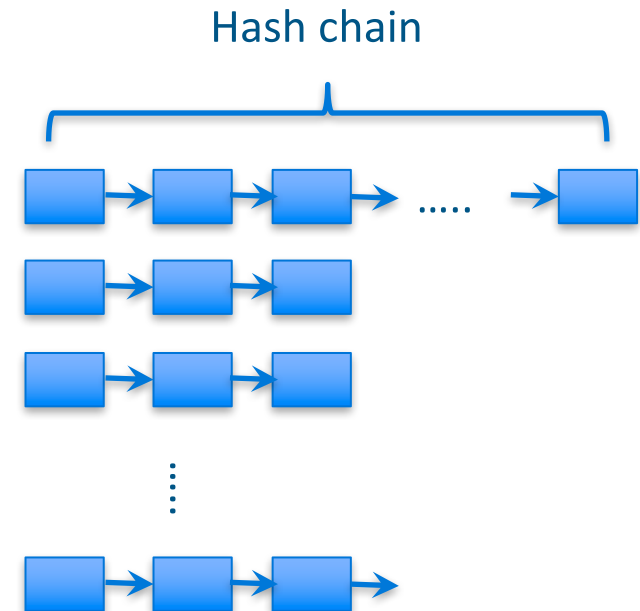
```
06:TOP-N [LIMIT=100]
11:AGGREGATE [MERGE FINALIZE]
10:EXCHANGE [PARTITION=HASH(b.int_col)]
05:AGGREGATE
04:HASH JOIN [INNER JOIN, PARTITIONED]
|--09:EXCHANGE [PARTITION=HASH(c.id)]
|  02:SCAN HDFS [functional.alltypes c]
03:HASH JOIN [INNER JOIN, BROADCAST]
|--08:EXCHANGE [PARTITION=HASH(a.id)]
|  00:SCAN HDFS [functional.smallTbl a]
07:EXCHANGE [PARTITION=HASH(b.id)]
01:SCAN HDFS [functional.BigTbl b]
```

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Common Join Performance Issues

- Hash Table
  - Ideally, the join key should be evenly distributed; only a few rows share the same join key from the RHS.
  - Is it a true foreign key join or more like a range join?
- Wrong join order – RHS is bigger than LHS table from the plan
- Too many LHS rows



© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Finding Bottlenecks

- Use `ExecSummary` from Query Profile to identify bottlenecks

ExecSummary:

Operator	#Hosts	Avg Time	Max Time	#Rows	Est. #Rows	Peak Mem	Est. Peak Mem	Detail
09:MERGING-EXCHANGE	1	4.394ms	4.394ms	7.30K	8.16K	0	-1.00 B	UNPARTITIONED
04:SORT	1	38.492ms	38.492ms	7.30K	8.16K	32.02 MB	8.00 MB	
08:AGGREGATE	1	8.397ms	8.397ms	7.30K	8.16K	458.25 KB	10.00 MB	MERGE FINALIZE
07:EXCHANGE	1	779.810us	779.810us	7.30K	8.16K	0	0	HASH(a.id)
03:AGGREGATE	1	161.736ms	161.736ms	7.30K	8.16K	466.25 KB	10.00 MB	
02:HASH JOIN	1	289.552ms	289.552ms	5.33M	5.33M	318.25 KB	20.91 KB	INNER JOIN, PARTITIONED
|--06:EXCHANGE	1	1.93ms	1.93ms	7.30K	7.30K	0	0	HASH(b.float_col)
| 01:SCAN HDFS	1	227.978ms	227.978ms	7.30K	7.30K	193.00 KB	160.00 MB	functional.alltypes b
05:EXCHANGE	1	816.252us	816.252us	7.30K	7.30K	0	0	HASH(a.float_col)
00:SCAN HDFS	1	228.362ms	228.362ms	7.30K	7.30K	193.00 KB	160.00 MB	functional.alltypes a

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Finding Skew

- Use `ExecSummary` from Query Profile to identify skew
  - Max Time is significantly more than Avg Time => Skew!

ExecSummary:

Operator	#Hosts	Avg Time	Max Time	#Rows	Est. #Rows	Peak Mem	Est. Peak Mem	Detail
08:EXCHANGE	1	113.775us	113.775us	31	227	0	-1.00 B	UNPART
07:AGGREGATE	81	359.985ms	436.922ms	31	227	3.44 MB	10.00 MB	FINALI
06:EXCHANGE	81	57.25us	515.364us	397	227	0	0	HASH(p
03:AGGREGATE	81	561.26ms	1s344ms	397	227	3.66 MB	10.00 MB	
02:HASH JOIN	81	3s730ms	18s695ms	184.02M	2.08M	3.04 MB	13.64 KB	INNER
|--05:EXCHANGE	81	27.471us	42.597us	26.74K	25.40K	0	0	HASH(p
| 01:SCAN HDFS	1	359.799ms	359.799ms	26.74K	25.40K	4.47 MB	16.00 MB	sds.md
04:EXCHANGE	81	130.853ms	1s608ms	184.03M	2.08M	0	0	HASH(s
00:SCAN HDFS	81	154.864ms	553.824ms	184.03M	2.08M	11.46 MB	88.00 MB	sds.ac

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Finding Skew (Cont'd)

- In addition to profile, always measure CPU, memory, disk IO and network IO across the cluster.
  - An uneven distribution of the load means skew!
- Cloudera Manager's charts can do that but only report at a minute interval
  - Use Colmux if your workload is short.

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Typical Speed

- In a typical query, we observed following processing rate:
  - scan node 8~10m rows per core
  - join node ~10m rows per sec per core
  - agg node ~5m rows per sec per core
  - sort node ~17MB per sec per core
  - Row materialization in coord should be tiny
  - Parquet writer 1~5MB per sec per core

```
 Query Timeline: 1s414ms
    - Start execution: 49.340us (49.340us)
    - Planning finished: 59.532ms (59.483ms)
    - Rows available: 987.346ms (927.813ms)
    - First row fetched: 1s019ms (32.521ms)
    - Unregister query: 1s412ms (392.159ms)
ImpalaServer:
  - ClientFetchWaitTimer: 415.244ms
  - RowMaterializationTimer: 7.795ms
```

- If your processing rate is much lower than that, it's worth a deeper look

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Improving Scan Node Performance

- HDFS Scan time – check out how much data is read; always do as little disk read as possible; review partition strategy.
- Column materialization time – only select necessary columns! Materializing 1k col is a lot of work.
- Complex predicates – string, regex are costly; avoid them.
- Order of predicates – Impala will try to order predicates by selectivity if stats are available. If stats are not available, you should order predicates by selectivity in your query.

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Aggregate Performance Tuning

- Needed when many rows going into aggregate
  - Reduce expression complexity in group by
- Complex UDA
  - No codegen for UDA yet. Try to rewrite UDA as built-in aggregate
- (Usually, not a big issue)

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Exchange Performance Issues

- Too much data across network:
  - Check the query on data size reduction.
  - Check join order and join strategy; wrong order/strategy can have a serious effect on network!
  - For agg, check the number of groups – affects memory too!
  - Remove unused columns.
- Keep in mind that network is at most 10GB.
- Cross-rack network slowness
- Query profile is usually not useful. Use CM or other system monitoring tools.

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Storage Skew

- HDFS Block Placement Skew
  - HDFS balances disk usage evenly across the whole cluster only. An individual table (or partition)’s data could be clustered in a handful of nodes.
  - If this happens, you’ll see that some nodes are busier (disk read and usually cpu as well) than the others.
  - This is inherent to HDFS: more pronounced if query data volume is tiny when compared to the total storage capacity.
  - By running a mixed workload that access data of a bigger set of tables, this type of hdfs block placement skew usually even out.

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Data Skew

- Partitioned Join Node Performance Skew
  - Join key data skew.
  - Each join key is re-shuffled and processed by one node.
  - If a single join key value account for a huge chuck of the data, then the node that process that join key will become the bottleneck.

- Possible workaround: use broadcast join but it uses more memory

© Cloudera, Inc. All rights reserved.

Query Tuning Basics –Expression Evaluation

- Impala evaluates expression lazily (i.e. only when the value is needed)
- Impala deals with inline views by substituting the select-list expressions from the inline view in the parent query block.
  - Expensive expressions inside the inline views might be executed multiple times

- Example

```
select concat(col1, col1, col1)
from (select regexp_extract(ori_col, '…..')  as col1
from tbl) x
```

- The regexp_extract will be evaluated 3 times by the coordinator!

© Cloudera, Inc. All rights reserved.

Query Tuning Basics –Expression Evaluation (Cont’d)

- To avoid redundant expression evaluation, we need to materialize the value.
- These operators materialize the expression: Order by, Union [all], Group by
- Example:

```
select concat(col1, col1, col1) from (select regexp_extract(ori_col,
‘…..’)  as col1 from tbl) x order by col1;
select concat(col1, col1, col1) from (select regexp_extract(ori_col,
‘…..’)  as col1 from tbl union all select null from one_row_tbl
where false) x;
```

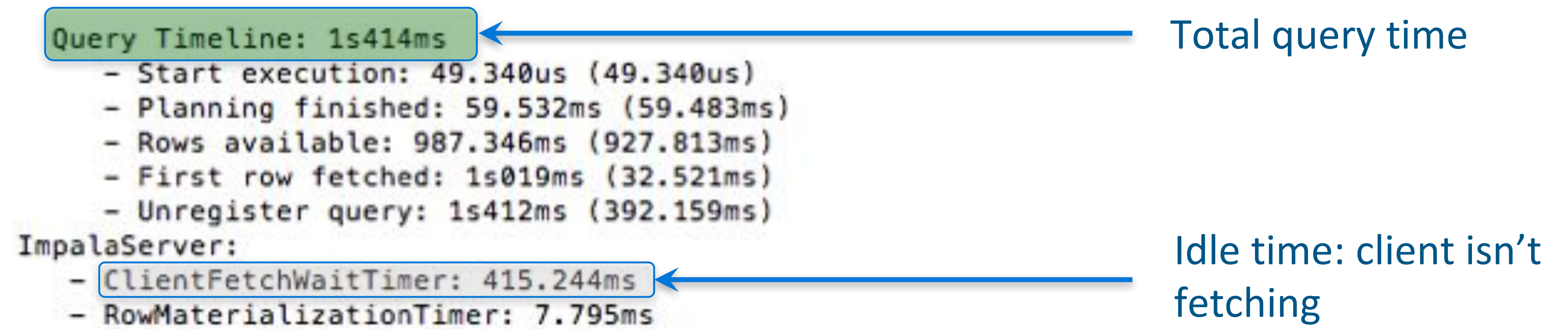
- Take away: Watch out for expensive expression inside an (inline) view!
- Take away: Due to lazy evaluation, it can affect all the exec nodes as well as the coordinator!

cloudera

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Client side Performance Issues

- Avoid large data extract.
  - It's usually not a good idea to dump lots of data out using JDBC/ODBC.
- For Impala-shell, use the –B option to fetch lots of data.



© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Excessive Planning Time?

- Usually due to Metadata Loading when running the query the first time after restart (or after invalidate metadata)
  - Try running the query again. The time should be short.

```
  Query Timeline: 1s414ms
     - Start execution: 49.340us (49.340us)
     - Planning finished: 59.532ms (59.483ms)
     - Rows available: 987.346ms (927.813ms)
     - First row fetched: 1s019ms (32.521ms)
     - Unregister query: 1s412ms (392.159ms)
ImpalaServer:
   - ClientFetchWaitTimer: 415.244ms
   - RowMaterializationTimer: 7.795ms
```

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Codegen

- In CDH 5.9 and before, Codegen has minimum cost (~500ms). for simple short query (< 2s), most of time could be spent on codegen parpare and optimization. Disable codegen could give more throughput.
  - This cost is reduced to 10-20% in CDH5.10+
- You can set query option to disable codegen
  - SET DISABLE_CODEGEN=1
- Some data types are not supported (CHAR, TIMESTAMP)

© Cloudera, Inc. All rights reserved.

Query Tuning Basics – Runtime filter and dynamic partition pruning (CDH5.7/Impala 2.5 or higher)

- Improve query performance for very selective equi-join by reducing IO, hash join, and network traffic.
  - If join is on partition column, Impala can use runtime filters to dynamically prune partitions on probe side, skip whole files. Not only can it improve performance, it can also reduce query resource usage (both memory and CPU)
  - If join is on non-partition columns, Impala can generate bloom filters. This can greatly reduce scan output of probe side and intermediate result.

cloudera

© Cloudera, Inc. All rights reserved.

Check how selective the join is

```
   ExecSummary:
Operator                 #Hosts   Avg Time    Max Time     #Rows  Est. #Rows   Peak Mem  Est. Peak Mem  Detail
---------------------------------------------------------------------------------------------------------------------------------
30:MERGING-EXCHANGE           1  211.877us   211.877us        52           5          0       -1.00 B  UNPARTITIONED
16:TOP-N                     15  148.816us   204.292us        52           5   12.00 KB      130.00 B
29:AGGREGATE                 15    2.462ms     2.795ms        52           5    2.34 MB      10.00 MB  FINALIZE
28:EXCHANGE                  15  259.949us   650.720us       728          51          0             0  HASH(a.ca_state)
15:AGGREGATE                 15  427.704ms   575.825ms       728          51   17.52 MB      10.00 MB  STREAMING
14:HASH JOIN                 15  352.894ms   584.100ms     3.13M       1.13M   14.00 MB      314.00 B  LEFT SEMI JOIN, BROADCAST
|--27:EXCHANGE               15   62.910us    87.360us        11          11          0             0  BROADCAST
|  26:AGGREGATE               1    8.992ms     8.992ms        11          11   10.35 MB      10.00 MB  FINALIZE
|  25:EXCHANGE                1   64.072us    64.072us        11          11          0
|  08:AGGREGATE               1   31.863ms    31.863ms        11          11    1.68 MB
|  07:SCAN HDFS               1    7.636ms     7.636ms   300.00K     300.00K    5.75 MB
13:HASH JOIN                 15    2s311ms     2s419ms    31.36M       1.13M    8.22 MB
|--24:EXCHANGE               15   13.602us    17.690us         1           1          0
|  23:EXCHANGE                1    8.183us     8.183us         1           1          0
|  22:AGGREGATE               1    1.316ms     1.316ms         1           1    2.27 MB
|  21:EXCHANGE                1    7.680us     7.680us         1         108          0
|  06:AGGREGATE               1    0.000ns     0.000ns         1         108    2.20 MB
|  05:SCAN HDFS               1   12.308ms    12.308ms        31         108    1.04 MB
12:HASH JOIN                 15    4s724ms     4s798ms     2.69B       2.73B    6.41 MB
|--20:EXCHANGE               15    2.346ms     3.762ms    73.05K      73.05K          0             0  BROADCAST
|  03:SCAN HDFS               1    5.785ms     5.785ms    73.05K      73.05K    2.09 MB      32.00 MB  tpcds_1000_parquet.date_dim
11:HASH JOIN                 15   11s504ms    12s678ms     2.75B       2.73B  396.38 MB     138.10 MB  INNER JOIN, BROADCAST
|--19:EXCHANGE               15  445.548ms   481.789ms     6.00M       6.00M          0             0  BROADCAST
|  00:SCAN HDFS              13   19.199ms    24.624ms     6.00M       6.00M   10.39 MB      32.00 MB  tpcds_1000_parquet.customer.
10:HASH JOIN                 15   20s971ms    26s141ms     2.75B       2.73B  148.37 MB      10.67 MB  INNER JOIN, BROADCAST
|--18:EXCHANGE               15    7.430ms     8.579ms   300.00K     300.00K          0             0  BROADCAST
|  04:SCAN HDFS               1   14.279ms    14.279ms   300.00K     300.00K   10.37 MB     120.00 MB  tpcds_1000_parquet.item i
09:HASH JOIN                 15   13s292ms    14s397ms     2.75B       2.77B  650.67 MB     100.71 MB  INNER JOIN, BROADCAST
|--17:EXCHANGE               15  650.933ms   685.678ms    12.00M      12.00M          0             0  BROADCAST
|  01:SCAN HDFS              13   27.058ms    41.049ms    12.00M      12.00M   20.80 MB     112.00 MB  tpcds_1000_parquet.customer
02:SCAN HDFS                 15  865.306ms     1s147ms     2.88B       2.88B  528.36 MB     176.00 MB  tpcds_1000_parquet.store_sa.
```

Large hash join output (~2.8B), the output of upstream one is much smaller (32M, reduced to only ~1%). This indicate runtime filter can be helpful.

© Cloudera, Inc. All rights reserved.

How it works

RF002 <- d.d_date_sk
Filter coming from Exch Node 20, Applied to Scan Node 02, reduce scan output from **2.8B** to **32M** and all the upstream hash join as well.

RF001 <- (d_month_seq)
Filter coming from Scan Node 05, Applied to Scan Node 03, Reduce scan output from **73049** to **31**

1

2

3

4

Why doesn't it work sometimes?

Final filter table:

ID	Src. Node	Tgt. Node(s)	Targets	Target type	Partition filter	Pending (Expected)	First arrived	Completed	Enabled
5	9	2	15	LOCAL	false	0 (15)	N/A	N/A	true
4	10	2	15	LOCAL	false	0 (15)	N/A	N/A	true
3	11	1	13	REMOTE	false	0 (3)	3s276ms	3s276ms	false
2	12	2	15	LOCAL	true	0 (15)	N/A	N/A	true
1	13	3	1	REMOTE	false	0 (3)	1s935ms	1s935ms	false
0	14	4	1	REMOTE	false	0 (3)	1s943ms	1s943ms	false

```
|  07:SCAN HDFS          1   6.700ms     6.700ms   300.00K  300.00K   3.70 MB   88.00 MB  tpcds_1000_parquet.item i
13:HASH JOIN            15   2s327ms     2s439ms    31.36M    1.13M   3.40 MB    5.00 B   LEFT SEMI JOIN, BROADCAST
|--24:EXCHANGE          15  13.272us    16.888us         1        1         0         0   BROADCAST
|  23:EXCHANGE           1   8.397us     8.397us         1        1         0   -1.00 B   UNPARTITIONED
|  22:AGGREGATE          1   8.342ms     8.342ms         1        1   2.27 MB   10.00 MB  FINALIZE
|  21:EXCHANGE           1  10.064us    10.064us         1      108         0             HASH((d_month_seq))
|  06:AGGREGATE          1   0.000ns     0.000ns         1      108   2.20 MB   10.00 MB  STREAMING
|  05:SCAN HDFS          1  22.849ms    22.849ms        31                                 tpcds_1000_parquet.date_dim
12:HASH JOIN            15   4s763ms     5s414ms     2.69B                                 , BROADCAST
|--20:EXCHANGE          15   2.481ms     3.947ms    73.05K
|  03:SCAN HDFS          1 972.009ms   972.009ms                                           _parquet.date_dim d
11:HASH JOIN            15  10s593ms    11s725ms     2.69B                                 , BROADCAST
|--19:EXCHANGE          15 277.472ms   314.535ms     6.00M
|  00:SCAN HDFS         13  16.104ms    20.280ms     6.00M                                 _parquet.customer...
10:HASH JOIN            15  22s428ms    25s443ms     2.69B                                 , BROADCAST
|--18:EXCHANGE          15   6.842ms     7.392ms   300.00K
|  04:SCAN HDFS          1 978.822ms   978.822ms   300.00K                                 _parquet.item i
09:HASH JOIN            15  12s937ms    13s782ms     2.69B                                 , BROADCAST
|--17:EXCHANGE          15 416.050ms   481.783ms    12.00M
|  01:SCAN HDFS         13 986.075ms   993.425ms    12.00M   12.00M  20.80 MB  112.00 MB  tpcds_1000_parquet.customer c
02:SCAN HDFS            15 857.578ms   950.789ms     2.75B    2.88B 510.79 MB  176.00 MB  tpcds_1000_parquet.store_sa...
```

Filter doesn't take effect because scan was short and finished before filter arrived.

HDFS_SCAN_NODE (id=3):

Filter 1 (1.00 MB):

- **Rows processed: 16.38K (16383)**
- **Rows rejected: 0 (0)**
- Rows total: 16.38K (16384)

© Cloudera, Inc. All rights reserved.

How to tune it?

- Increase RUNTIME_FILTER_WAIT_TIME_MS to 5000ms to let Scan Node 03 wait longer time for the filter.

  HDFS_SCAN_NODE (id=3)

  Filter 1 (1.00 MB)

  - InactiveTotalTime: 0
  - Rows processed: 73049
  - **Rows rejected: 73018**
  - Rows total: 73049
  - TotalTime: 0

- If the cluster is relatively busy, consider increasing the wait time too so that complicated queries do not miss opportunities for optimization.

cloudera

© Cloudera, Inc. All rights reserved.

Which file format is supported

- Partition filter
  - All file formats as long as it's a partitioned table and there are mappings for partition columns. E.g.
    - Select count(*) from t1, t2 where **t1.partition_col = t2.col1** and t2.col2="1234"
- Row filter
  - Only applied for parquet format

- Parquet can take most advantage of runtime filters.

cloudera

© Cloudera, Inc. All rights reserved.

Runtime filter Gotcha

- Runtime filters require extra memory on each host and extra work on coordinator. Example: on 100-node cluster, one 16MB filter
  - Memory cost, 16MB * 100 = 1.6GB for query
  - Network cost (only for GLOBAL mode), 16MB * 100 * 2 = 3.2GB network traffic on coordinator, and CPU time to merge and publish filters.
  - More filters, higher cost.
- Negative impact: Coordinator becomes the bottleneck

- Resolution: reduce # of filters by setting MAX_NUM_RUNTIME_FILTERS to a lower number or disable row filter by set DISABLE_ROW_RUNTIME_FILTERING=1.

cloudera

© Cloudera, Inc. All rights reserved.

Topic Outline

- Part 1 – The Basics
  - Physical and Schema Design
  - Memory Usage
- Part 2 – The Practical Issues
  - Cluster Sizing and Hardware Recommendations
  - Benchmarking Impala
  - Multi-tenancy Best Practices
  - Query Tuning Basics
- Part 3 – Outside Impala
  - Interaction with Apache Hive, Apache Sentry, and Apache Parquet

cloudera

© Cloudera, Inc. All rights reserved.

Interaction with Sentry, Hive, and Parquet

- Setup: Cloudera Manager 5.x does a good job; verify the dependency parents, such as Hive Metastore, HDFS.
  - Stability in HMS might affect Impala; check HMS health.
- File-based and Apache Sentry security
  - Even with Sentry, Impala needs to read/write all dir/files. No impersonation.

© Cloudera, Inc. All rights reserved.

Summary

- Approach cluster sizing systematically - SLA and workload
- Benchmark running technique and measurement – QPS
- Use Admission Control for multi-tenancy
- Tune your queries - identify and attack bottlenecks
- Cloudera Manager 5.0+ provides a tool for verifying whether many best practices have been followed

© Cloudera, Inc. All rights reserved.

Other Resources

- Impala User Guide: http://www.cloudera.com/documentation/enterprise/latest/topics/impala.html
- Impala website (roadmap, repository, books, more) & Twitter: https://impala.apache.org, @ApacheImpala
- Community resources:
  - Mailing list: user@impala.incubator.apache.org
  - Forum: http://community.cloudera.com/t5/Interactive-Short-cycle-SQL/bd-p/Impala

cloudera

© Cloudera, Inc. All rights reserved.

cloudera

Thank you

https://impala.apache.org

@ApacheImpala

© Cloudera, Inc. All rights reserved.