WHEN SQL USERS RUN WILD
RESOURCE MANAGEMENT FEATURES AND TECHNIQUES TO TAME APACHE IMPALA

Tim Armstrong – Software Engineer
IMPALA USE CASES

Affordable storage + easy ingest + fast analytics \(\Rightarrow\) workload growth

- Valuable data accumulates in central cluster
- Workloads from different groups grow organically
  - SQL generated by tools
  - Ad-hoc analysis with SQL
  - Data pipelines
OUTLINE

SQL queries run wild

Tranquilizers  (query guard rails)
Herders  (admission control)
Enclosures  (multi-cluster)
IN THE BEGINNING

- New clusters tend to be lightly loaded
- All queries get ample resources
  - CPU, Memory, I/O, Network
- SQL queries are happy and well fed
- Everyone gets their work done
THE FALL

Good things come to an end

- Overcrowding
- Disruptive queries
  - Batch jobs
  - Ad-hoc users
  - Data scientists
OUTLINE

SQL queries run wild

Herders (admission control)
Tranquilizers (query guard rails)
Enclosures (multi-cluster)
IMPALA ADMISSION CONTROL

Herding your queries

• Impala admission control helps:
  • Prevent overcrowding by queuing or rejecting queries
  • Divide resources between queries for efficient utilization
  • Limit memory consumption of disruptive queries
• Minimal overhead
• Queries still share hosts and some resources
• Admission control is essential for health of heavily-used clusters

Attribution: https://bit.ly/2FoRPYV
STATIC AND DYNAMIC RESOURCE POOLS
Two tiers of resource management in Cloudera Platform

Service Pools - CGroups

- HDFS 10%
- Kudu 10%
- HBase 10%
- Impala 30%

Admission Control Pools
- Ad-hoc
- BI
- ETL

Dynamic Resource Pools
- Support
- Data Science
- Priority

Soft limits – any service can use idle CPUs.
IMPALA ADMISSION CONTROL

- Admission control allocates cluster resources to queries
  - Range of policies supported
- Resource limits enforced during Impala query execution

```
select custkey, sum(total)
from orders
order by sum(total) desc
limit 10
```
CONFIGURING A SINGLE ADMISSION CONTROL POOL

- **Simplest recommended configuration for production clusters**
- Max running queries $\rightarrow$ # of CPU cores per host
- Max memory $\rightarrow$ use aggregate Impala memory for cluster
- Query memory limit $\rightarrow$ configure based on workload
SINGLE POOL CONFIGURATION EXAMPLE – PART 1

Max Memory: 384 Gi Per Cluster
- Maximum amount of aggregate memory available across the cluster to all queries executing in this pool. This should be a portion of the aggregate configured memory for Impala daemons, which is currently configured to be 384.0 GiB. Setting this to a non-zero value enables memory-based admission control.

Minimum Query Memory Limit: 2 Gi Per Host
- The minimum memory limit that can be set for a query in this resource pool. This limit is not enforced on the MEM_LIMIT query option when the Clamp MEM_LIMIT Query Option is not selected. This value must be less than or equal to Maximum Query Memory Limit and Max Memory.

Maximum Query Memory Limit: 20 Gi Per Host
- The maximum memory limit that can be set for a query in this resource pool. This limit is not
SINGLE POOL CONFIGURATION EXAMPLE – PART 2

Edit Resource Pool

Memory Limit is set.

Max Running Queries: 32
Maximum number of concurrently running queries in this pool. The default value is unlimited for CDH 5.7 or higher, see the documentation for lower versions. (optional)

Max Queued Queries: 50
Maximum number of queries that can be queued in this pool. The default value is 200 for CDH 5.3 or higher and 50 for previous versions of Impala. (optional)

Queue Timeout: 1 minute
The maximum time a query can be queued waiting for execution in this pool before timing out.

Default Query Options:
- `exec_time_limit_s` = 3600
- `scratch_limit` = 20GB

Save
MEMORY LIMITS

• Per-query value
  • E.g. “5GB of memory per host”
• Queries that exceed the memory limit are killed
• Impala works hard to keep queries under their memory limit:
  • Spill-to-disk, spill-to-network, reduced parallelism, etc.

• Different from Impala Daemon Memory Limit – which is per-host.
MEMORY ESTIMATES

- Estimate of *ideal* memory to run query as fast as possible
- Typically much higher than minimum memory required to run query

```sql
> explain select l_orderkey, count(*) from lineitem group by l_orderkey;
+-------------------------------------------------------------+
| Max Per-Host Resource Reservation: Memory=46.50MB Threads=4 |
| Per-Host Resource Estimates: Memory=140MB                   |
|                                                             |
| PLAN-ROOT SINK                                              |
|                                                           |
| 04:EXCHANGE [UNPARTITIONED]                                 |
|                                                           |
| 03:AGGREGATE [FINALIZE]                                    |
```
QUERY MEMORY ALLOCATION

• Memory-based admission control depends on per-query MEM_LIMIT
• Min/max memory limit in Impala 3.1+ and CDH 6.1+
  • Policy guardrails to prevent resource starvation or resource hogging
  • Within guardrails, Impala uses estimate of query size to determine MEM_LIMIT
• Default memory limit: pre-Impala 3.1 or CDH 6.1
• MEM_LIMIT can be overridden per query if needed
ADMISSION CONTROL INTERNALS

- Limit aggregate queries & memory per resource pool
- Limit total memory per host
CONFIGURING MULTIPLE POOLS

- Submission access control: control who can submit to which pool
- Placement rules: control where queries are routed by default
- Dividing resources between pools:
  - Set lower aggregate limits for lower priority pools
  - Ensure that important workloads have enough resources
### MULTIPLE POOLS CONFIGURATION EXAMPLE

#### Cluster 1

**Dynamic Resource Pool Configuration**

<table>
<thead>
<tr>
<th>Name</th>
<th>Max Memory</th>
<th>Max Running Queries</th>
<th>Max Queued Queries</th>
<th>Queue Timeout</th>
<th>Minimum Query Memory Limit</th>
<th>Maximum Query Memory Limit</th>
<th>Clamp MEM_LIMIT Query Option</th>
<th>Default Query Memory Limit</th>
<th>Access Control</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>root.bi</td>
<td>128 GiB</td>
<td>20</td>
<td>200</td>
<td>1 minute(s)</td>
<td>2 GiB</td>
<td>8 GiB</td>
<td>true</td>
<td>No Default</td>
<td>Custom</td>
<td>Edit</td>
</tr>
<tr>
<td>root.default</td>
<td>64 GiB</td>
<td>10</td>
<td>50</td>
<td>1 minute(s)</td>
<td>2 GiB</td>
<td>10 GiB</td>
<td>true</td>
<td>No Default</td>
<td>Inherit</td>
<td>Edit</td>
</tr>
<tr>
<td>root.priority</td>
<td>384 GiB</td>
<td>Unlimited</td>
<td>200</td>
<td>5 minute(s)</td>
<td>2 GiB</td>
<td>20 GiB</td>
<td>true</td>
<td>No Default</td>
<td>Custom</td>
<td>Edit</td>
</tr>
</tbody>
</table>

Displaying 1 - 4 of 4

---

**Impala Admission Control**

Each pool can support different limits, and can be configured to allow only a certain set of users and groups to access the pool.

3 running Impala Daemons are configured with a total of **384.0 GiB** of memory.
OUTLINE

SQL queries run wild
Herders (admission control)
Tranquilizers (query guard rails)
Enclosures (multi-cluster)
PROBLEMATIC QUERIES

• Accidents
  • Huge result sets:  
    SELECT * FROM large_table
  • Cross joins:  
    SELECT * FROM t1, t2

• Heavy hitting queries
  • Scans of huge historical tables:
    SELECT * FROM large_table ORDER BY x, y, z LIMIT 100
  • Joining multiple large tables
    SELECT * FROM large_table JOIN big_table ON ...

• Very complex queries, e.g. 100s of fragments
• Shoot the query first, ask questions later?
HOW TO SET GUARD RAILS
Dynamic Resource Pools in Cloudera Manager

- Options can be set for any resource pool, including the default
- Options can also be set per query
- http://tiny.cloudera.com/impala-rm-howto
RUNAWAY QUERIES

Ways to draw a line in the sand

- Limits on actual runtime & resource consumption
  - Estimated resource consumption can be inaccurate
- EXEC_TIME_LIMIT_S: time limit on execution (Impala 2.12/CDH 5.15)
  - e.g. “kill if running > 1 hour”
- SCRATCH_LIMIT: limit data spilled to disk per node
  - e.g. “kill if > 20GB spilled”
- SCAN_BYTES_LIMIT: limit data read from filesystem (Impala 3.1/CDH 6.1)
  - e.g. “kill if > 100TB scanned”
- Configure based on your workload and available resources
INTERACTIVE QUERIES

- NUM_ROWS_PRODUCED_LIMIT
- E.g. “kill queries issued from Hue that return > 10000 rows”
- Impala 3.2/CDH6.2
COMPLEX QUERIES

- Per-query limits on number of threads
  - Closely related to query fragments
    | Max Per-Host Resource Reservation: Memory=46.50MB Threads=4 |
- THREAD_RESERVATION_LIMIT
  - Limits threads on any host per query
- THREAD_RESERVATION_AGGREGATE_LIMIT
  - Limits total threads across cluster for query
- Both available in Impala 3.1/CDH 6.1
OUTLINE

SQL queries run wild
Herders (admission control)
Tranquilizers (query guard rails)
Enclosures (multi-cluster)
MULTI-CLUSTER
Fencing off workloads

- Shared data → no silos
- Overcrowded cluster → easily add more compute
- Problematic queries → isolated
- CM support in CDH 6.2
SEPARATE COMPUTE AND STORAGE

• Paradigm shift away from data locality

• **Pros:**
  • Hard isolation for CPU, memory, intra-cluster network
  • Easier to add compute capacity

• **Cons:**
  • Remote reads → network can become the bottleneck
  • Some services still shared, e.g. HDFS NameNode
CAN YOU EXPLAIN IT TO ME IN TERMS OF ANIMALS?

Compute (animals)  Network (army of overworked zookeepers)  Data (food)
ENGINEERING FOR MULTI-CLUSTER IN APACHE IMPALA

Work in flight

- Remote read benchmarking and optimization
- HDFS & S3 Client tuning
- File handle caching
- I/O caching
- Cache-aware query scheduling
- Read elimination – Parquet Column Indices
CONCLUSIONS

• Impala on CDH provides many options to tame your SQL
  • http://tiny.cloudera.com/impala-rm
• No silver bullet –policies depend on your needs and priorities
• Future CDP platform will have this, plus more!
• Engage with the Apache Impala community:
  • impala.apache.org/community.html
We’re living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you’re often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we're uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

**David Sanger**
The New York Times

David E. Sanger is the national security correspondent for the New York Times as well as a national security and political contributor for CNN and a frequent guest on CBS This Morning, Face the Nation, and many PBS shows.