



LLAP: Sub-Second Analytical Queries in Hive

Gunther Hagleitner

```
hive (tpcds_bin_partitioned_orc_1000)> set hive.llap.execution.mode;  
hive.llap.execution.mode=all  
hive (tpcds_bin_partitioned_orc_1000)> set hive.llap.execution.mode
```

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	container	SUCCEEDED	1	1	0	0	0	0
Map 2	container	SUCCEEDED	31	31	0	0	0	0
Map 5	container	SUCCEEDED	7	7	0	0	0	0
Reducer 3	container	SUCCEEDED	1	1	0	0	0	0
Reducer 4	container	SUCCEEDED	1	1	0	0	0	0

VERTICES: 85/85 [=====>>] 100% ELAPSED TIME: 9.47 s

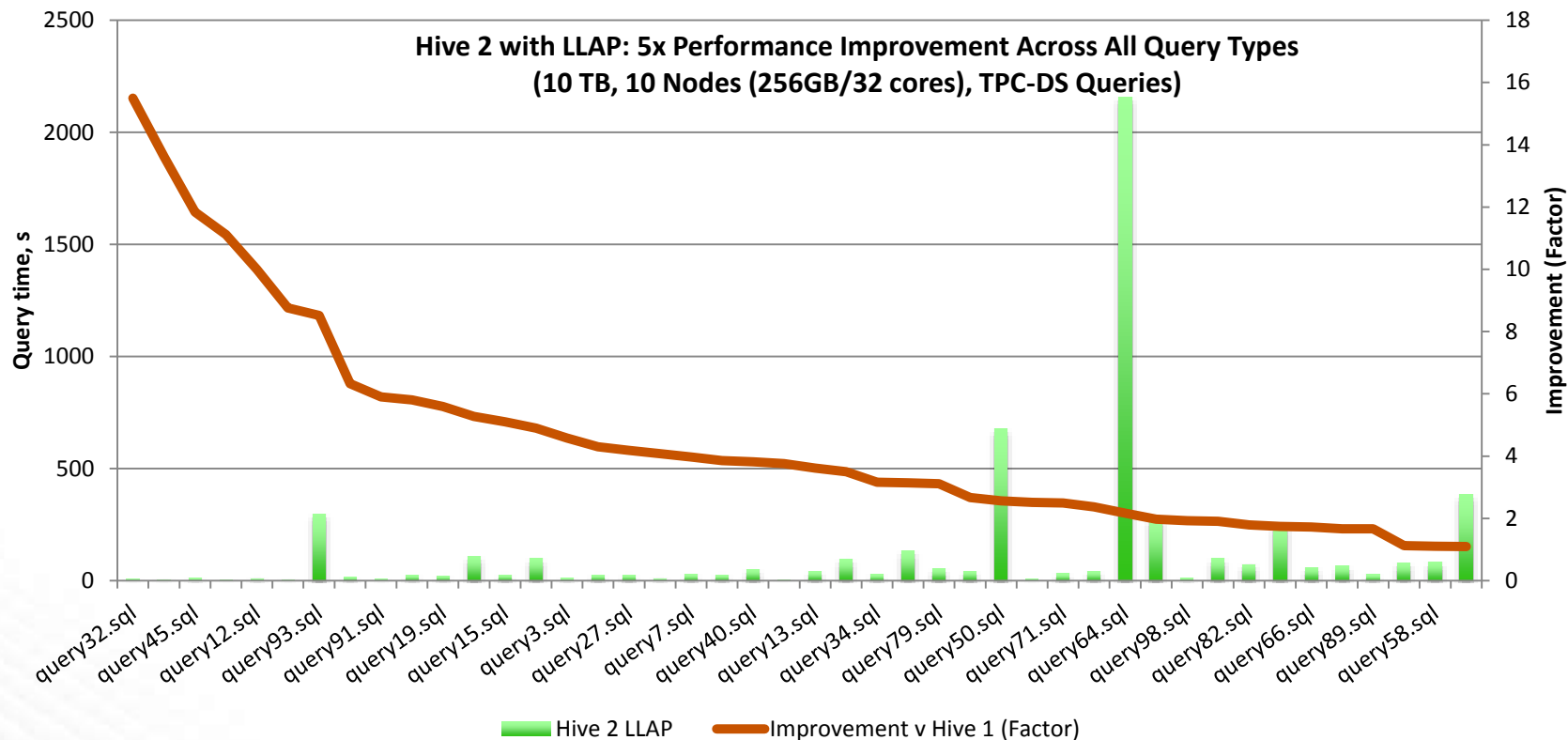
Status: DAG finished successfully in 9.47 seconds

VERTICES	MODE	STATUS	TOTAL	COMPLETED	RUNNING	PENDING	FAILED	KILLED
Map 1	llap	SUCCEEDED	1	1	0	0	0	0
Map 2	llap	SUCCEEDED	31	31	0	0	0	0
Map 5	llap	SUCCEEDED	7	7	0	0	0	0
Reducer 3	llap	SUCCEEDED	1	1	0	0	0	0
Reducer 4	llap	SUCCEEDED	1	1	0	0	0	0

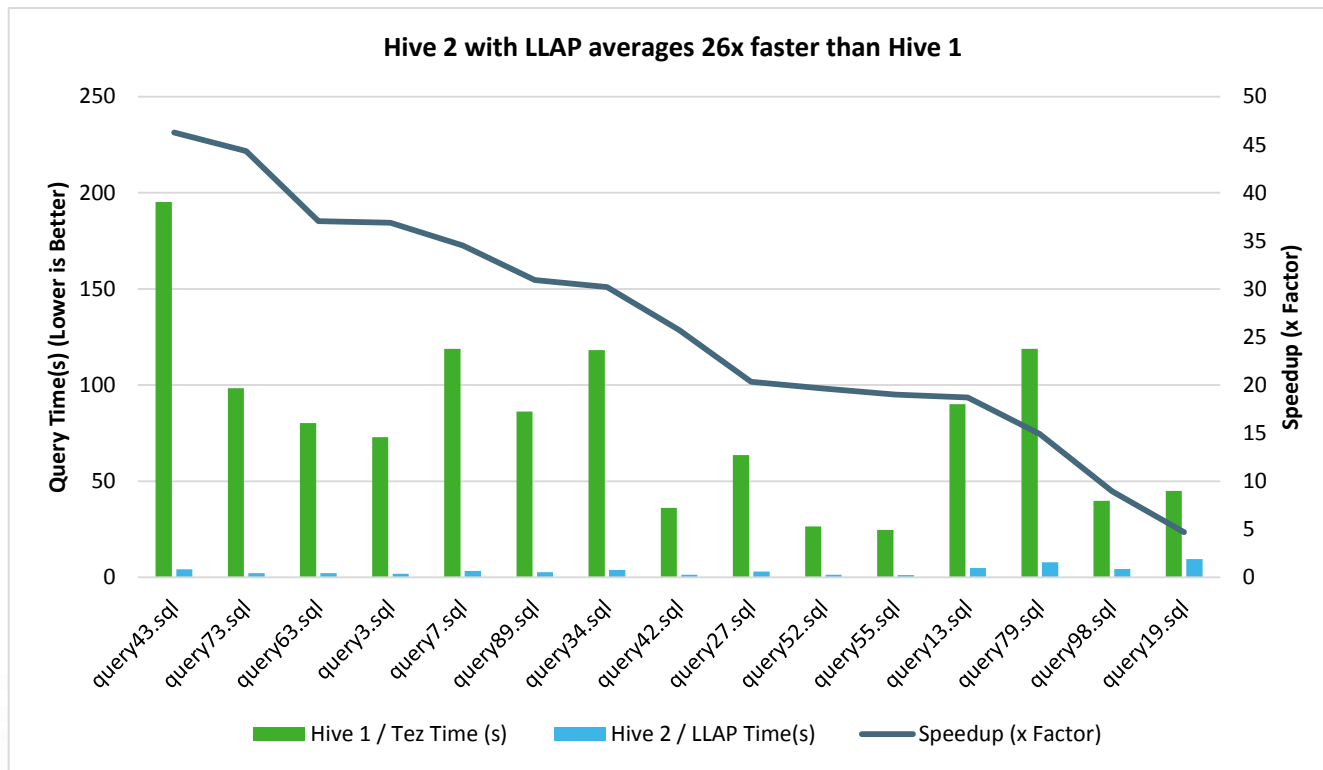
VERTICES: 85/85 [=====>>] 100% ELAPSED TIME: 0.89 s

Status: DAG finished successfully in 0.89 seconds

Hive 2 with LLAP: 5x Performance Boost at 10 TB Scale



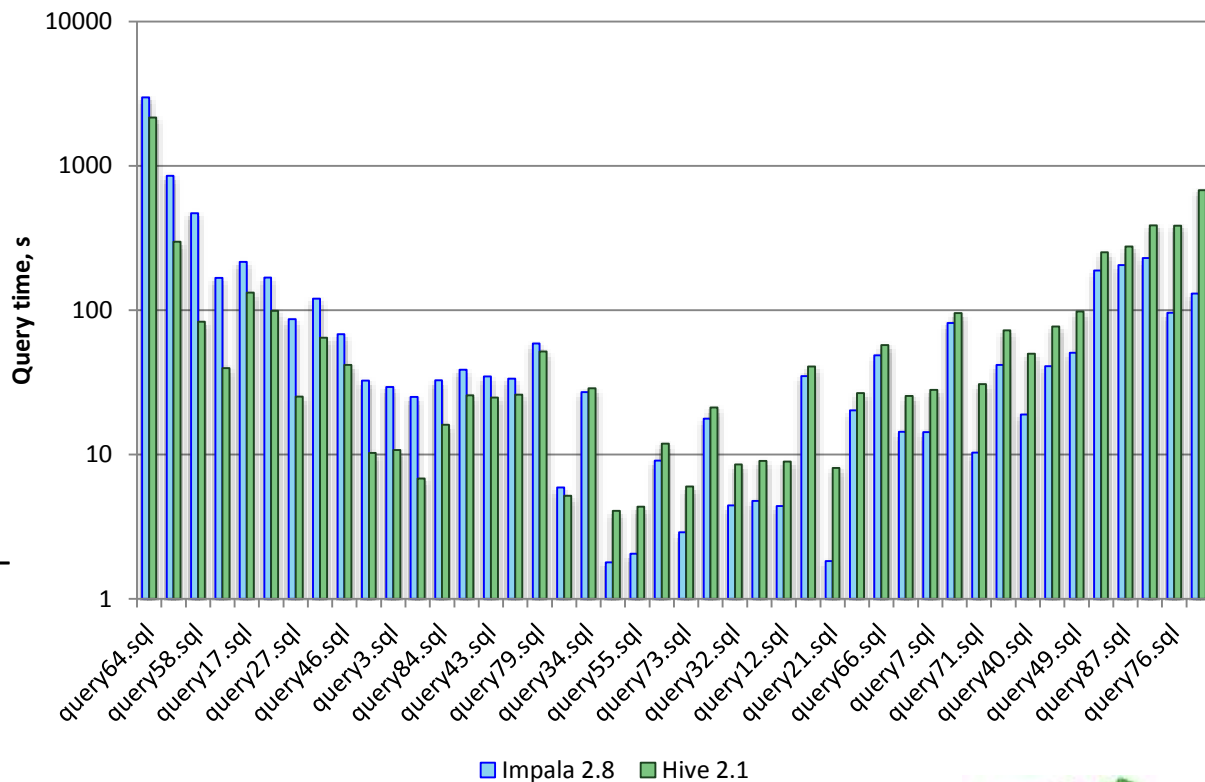
Hive 2 with LLAP: 25+x Performance Boost: Interactive / 1TB Scale



Apache Hive vs. Apache Impala at 10TB

Highlights

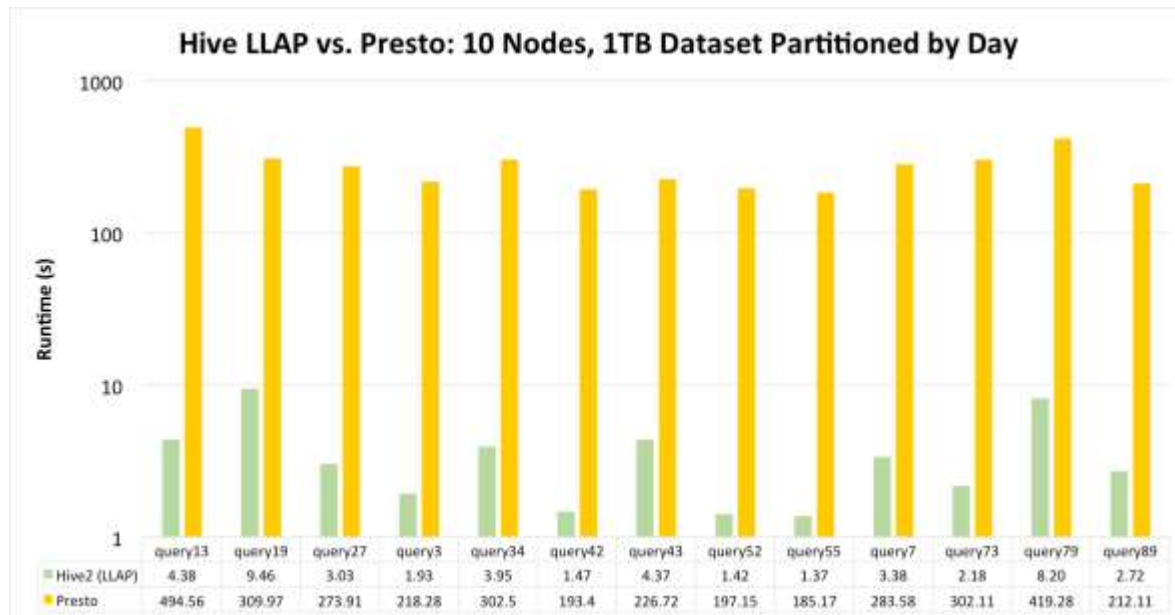
- 10TB scale on 10 identical nodes.
- Hive and Impala showed similar times on most smaller queries.
- Hive scaled better, completing larger queries quicker with lower end-to-end time for all queries



Apache Hive vs. Presto on a partitioned 1TB dataset.

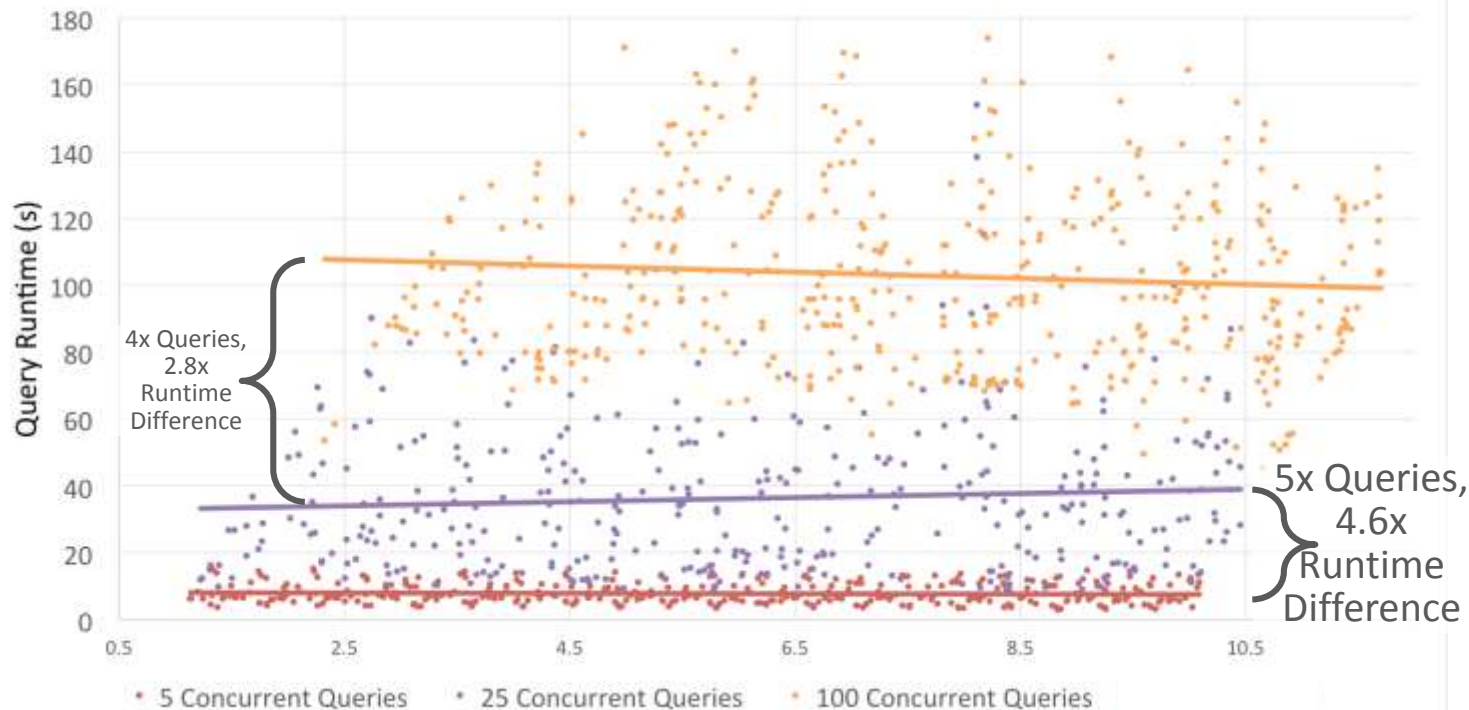
Highlights

- Presto lacks basic performance optimizations like dynamic partition pruning.
- On a real dataset / workload Presto perform poorly without full re-writes.
- Example: Query 55 without re-writes = 185.17s, with re-writes = 16s. LLAP = 1.37s.



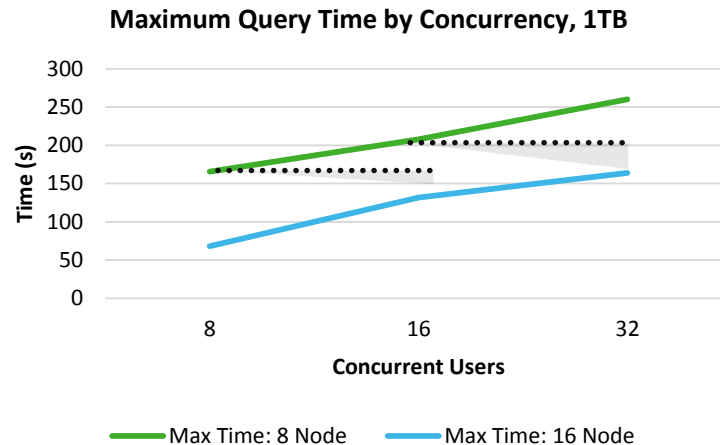
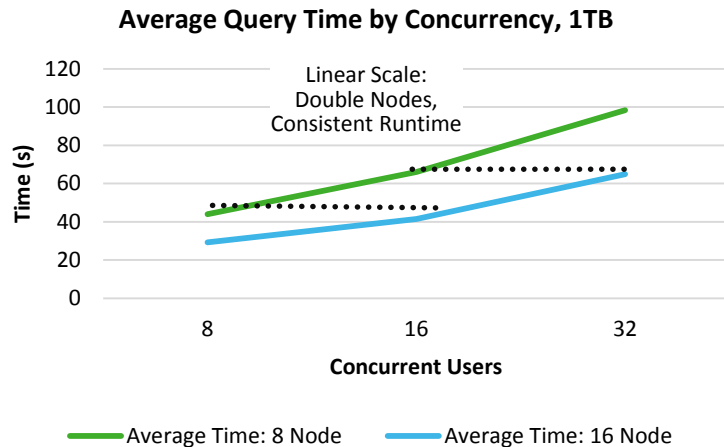
Hive LLAP: Stable Performance under High Concurrency

Interactive Query Runtimes: 10 Nodes, 1 TB Data



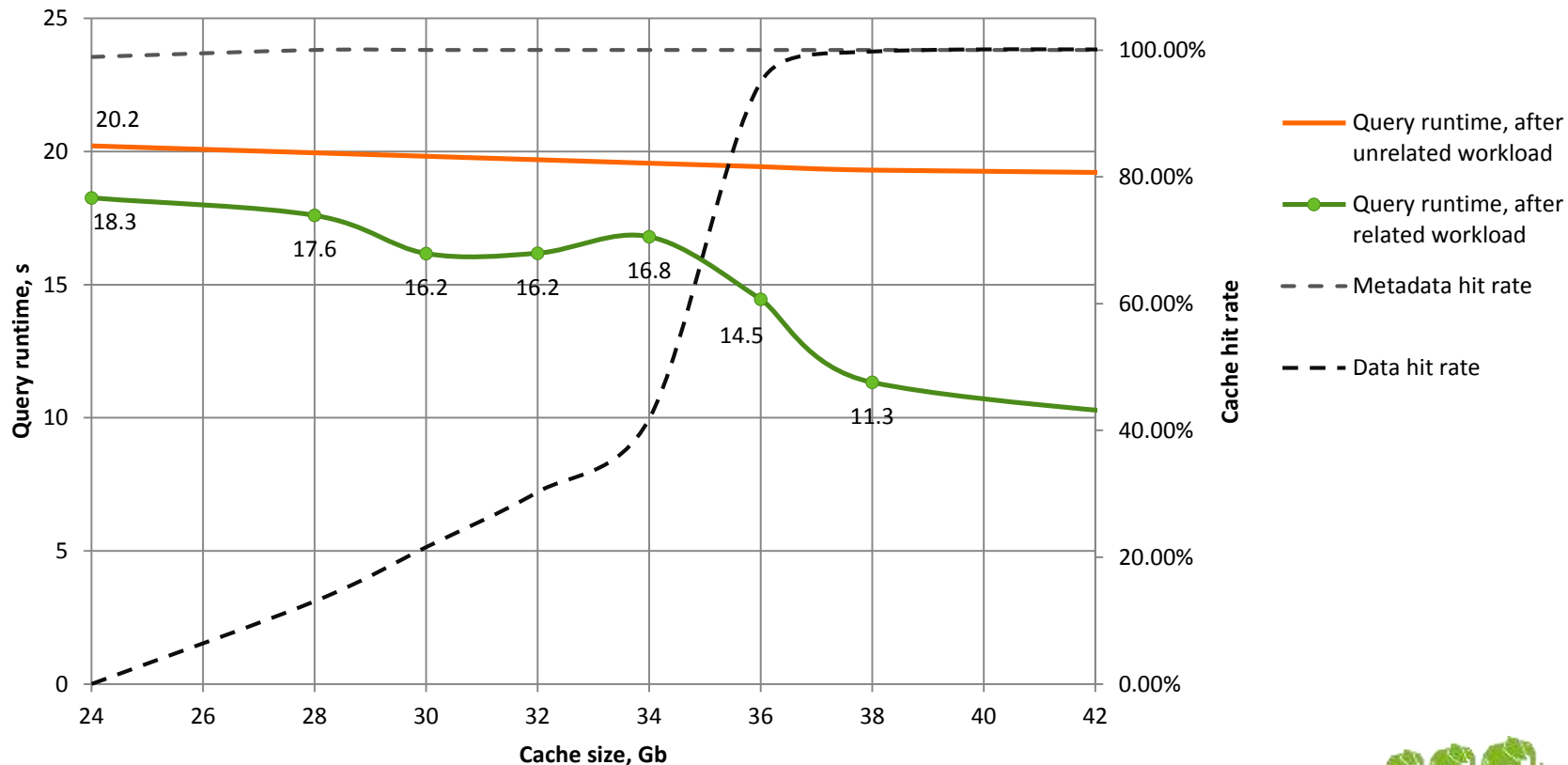
Mark	Concurrent Queries	Average Runtime
<div></div>	5	7.76s
<div></div>	25	36.24s
<div></div>	100	102.89s

Hive with LLAP: Linear Scale



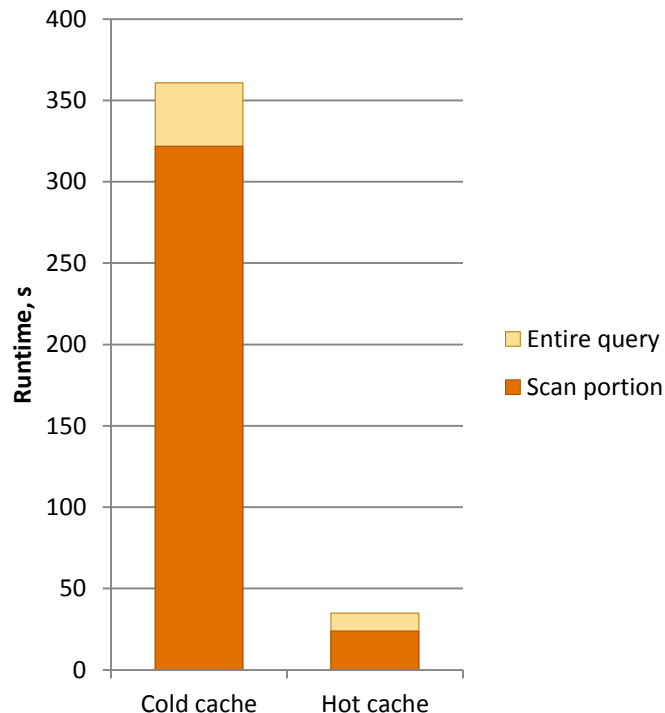
Average and Maximum query times
Mixed SQL workload of simple and complex queries
1TB Dataset Size

Performance – cache on HDFS, 1Tb scale

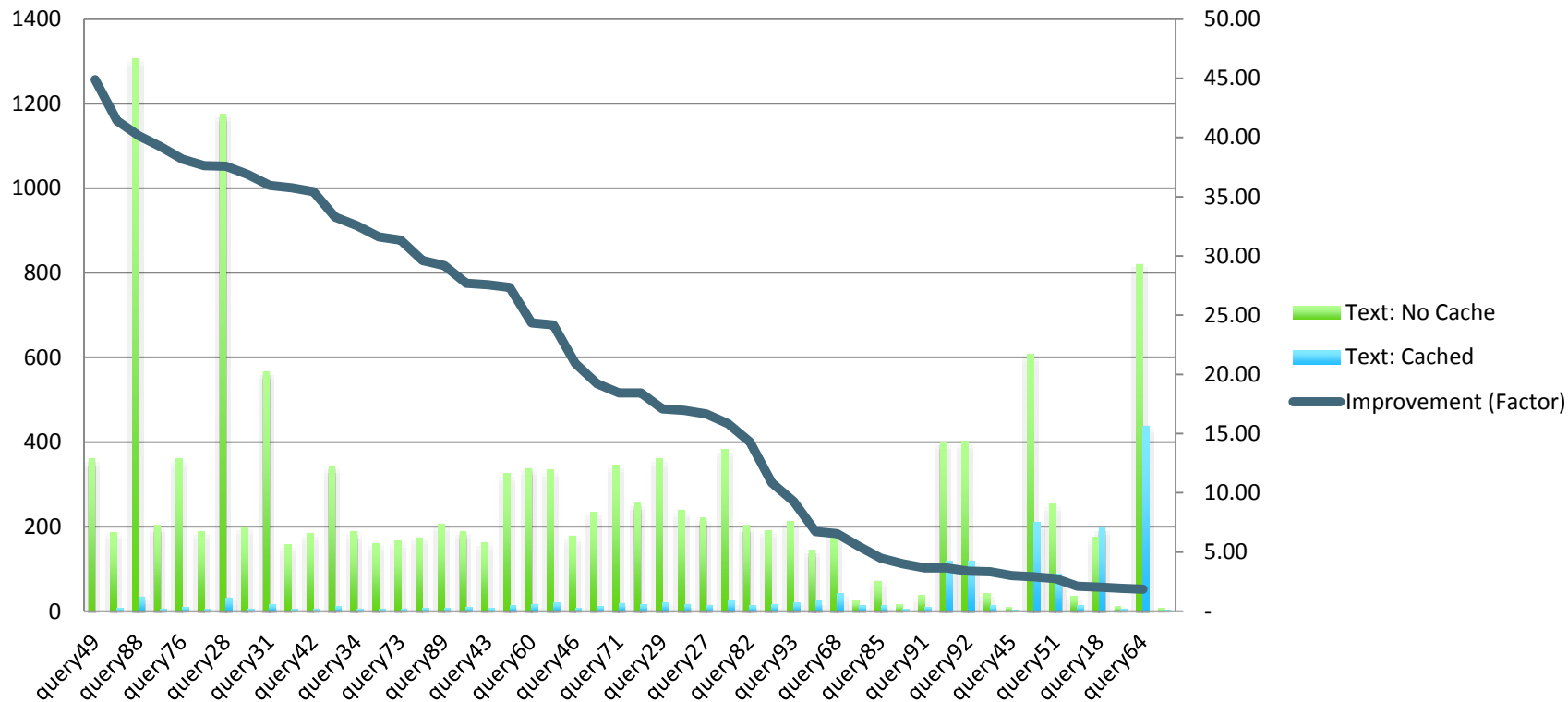


Performance – cache on cloud storage

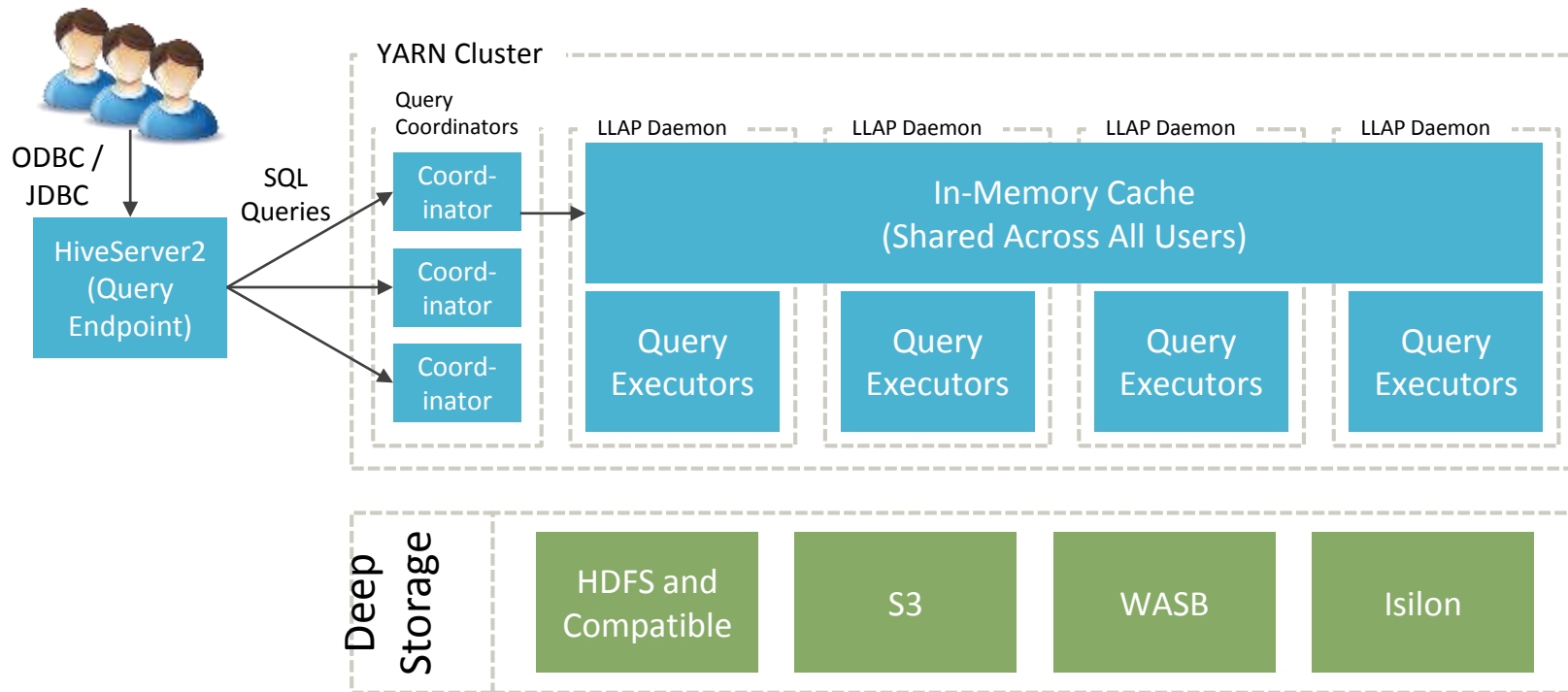
- Cloud storage, slower than on-prem HDFS
- Large scan on 1Tb scale
- Local HD/SDD cache
 - Massive improvement on slow storage with little memory cost



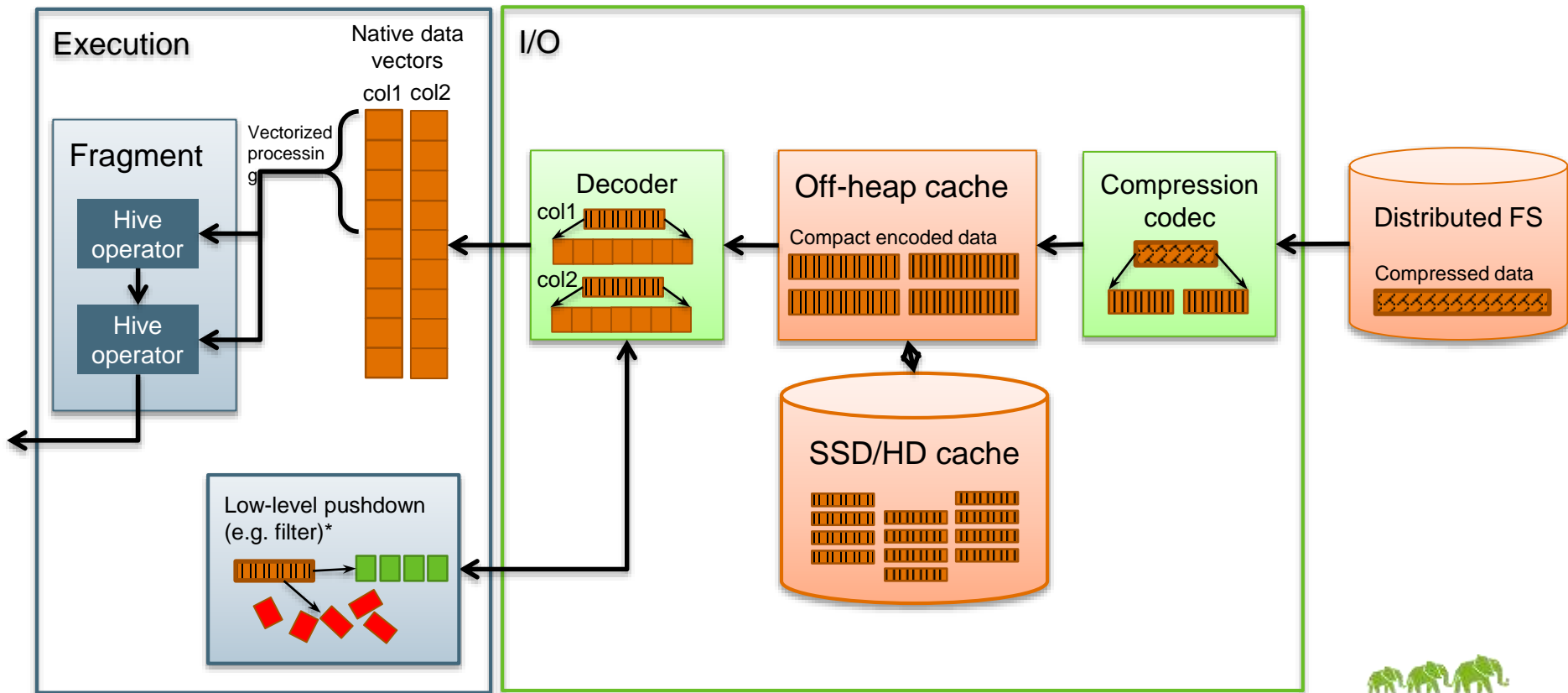
Hive with LLAP on text data (CSV, logs, etc)



Hive with LLAP: Architecture Overview

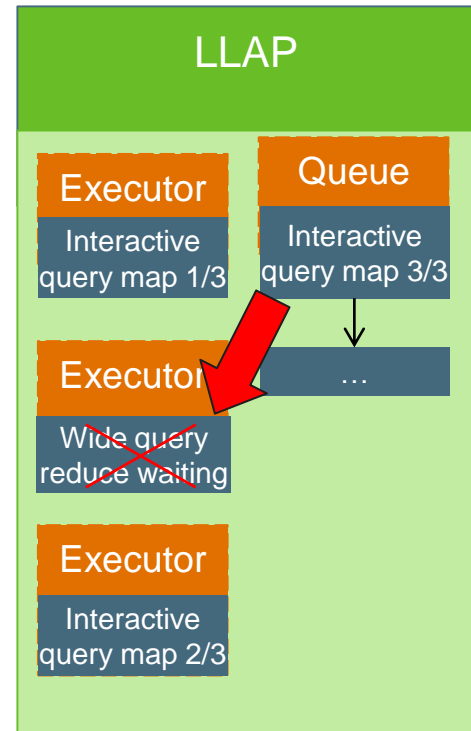
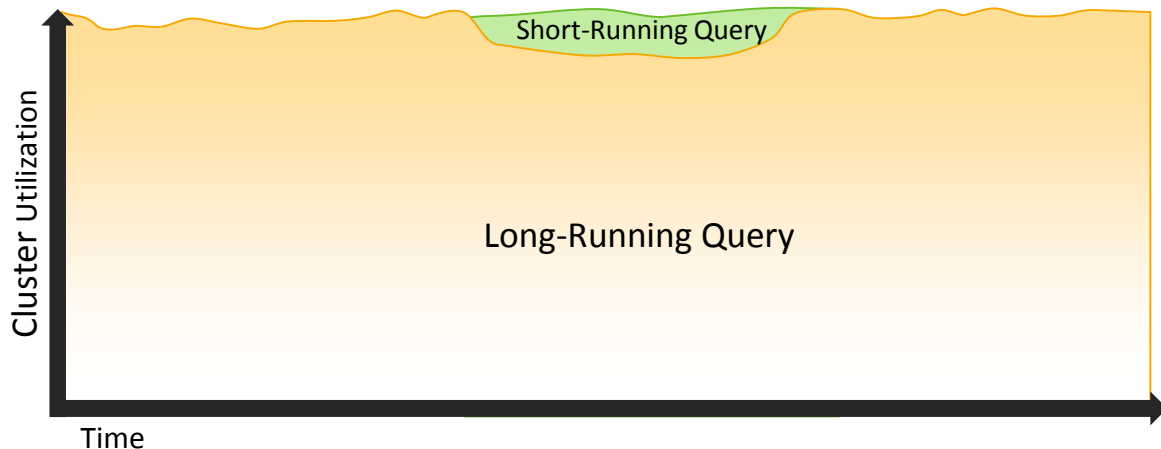


LLAP: In-memory processing



Parallel queries – priorities, preemption


- Lower-priority fragments can be preempted
 - For example, a fragment can start running before its inputs are ready, for better pipelining; such fragments may be preempted
- LLAP work queue examines the DAG parameters to give preference to interactive (BI) queries



Sidebar: Relational data node

- LLAP can provide a "relational datanode" view of the data
- Standard interface to read tables directly from LLAP nodes
 - Supports column-level security
 - LLAP handles ACID, schema-evolution
- Optimized for fast access/scans
 - Read horizontal table partitions in parallel on compute nodes
 - Push projections, filters, aggregates into LLAP
 - Utilizes cache and LLAP IO subsystem

Example - SparkSQL integration

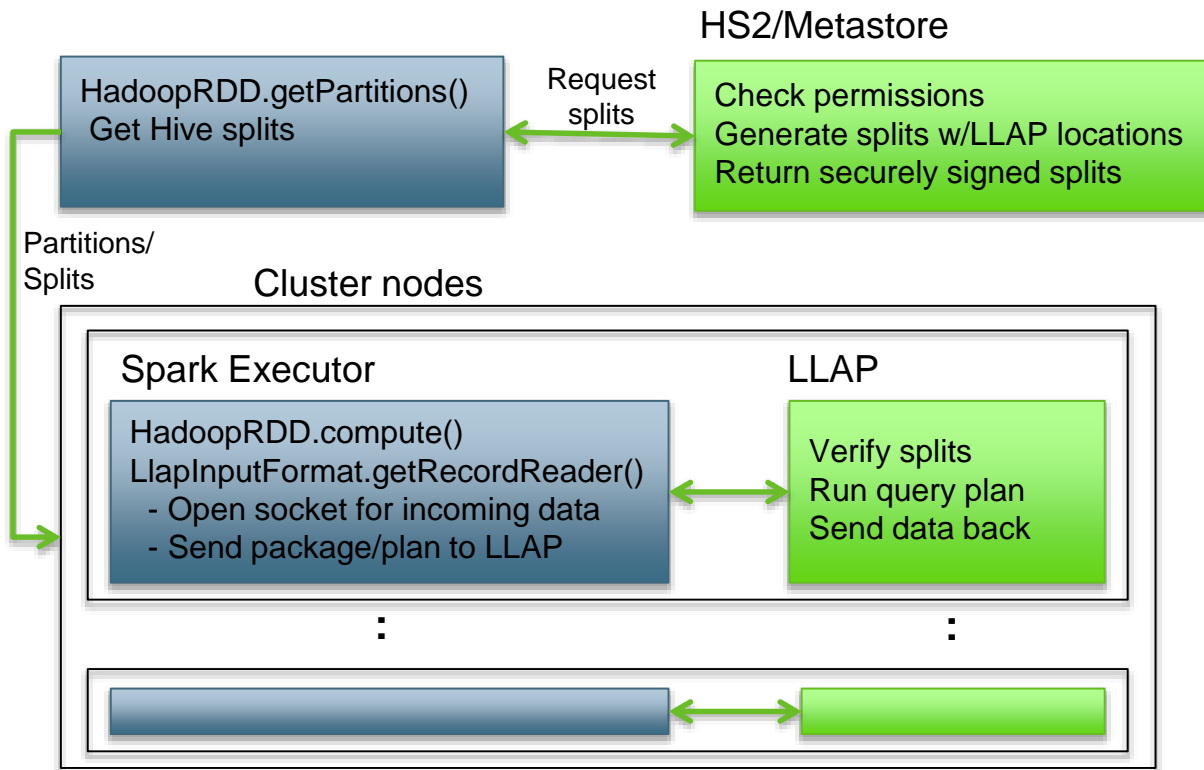
- Allowing direct access to Hive table data from Spark breaks security model for warehouses secured using Ranger or SQL Standard Authorization
 - Other features may not work correctly unless support added in Spark
 - Row/Cell level security (when implemented), ACID, Schema Evolution, ...
- 
- SparkSQL has interfaces for external sources; they can be implemented to access Hive data via HS2/LLAP
 - SparkSQL Catalyst optimizer hooks can be used to push processing into LLAP (e.g. filters on the tables)
 - Some optimizations, like dynamic partition pruning, can be used by Hive even if SparkSQL doesn't support them; if execution pushdown is advanced enough

Example - SparkSQL integration – execution flow

```
var llapContext =  
LlapContext.newInstance(  
sparkContext, jdbcUrl)
```

```
var df: DataFrame =  
llapContext.sql("select *  
from tpch_text_5.region")
```

DataFrame for Hive/LLAP data



LLAP Availability

- First version shipped in Apache Hive 2.0
 - Hive 2.0.1 contains important bug fixes to make it production ready
 - Hive 2.1 has new features and further perf improvements
- HDP 2.5 includes beta version of LLAP
- HDP 2.6 includes GA version of Apache Hive 2.1 w/ LLAP

Setup: Ambari

- Simple setup
- Integrated into Hive configuration page
- Launches HS2 instance and LLAP cluster
- Handles configuration, slider, security, alerts and monitoring, ...
- Requires Ambari 2.5.x
- Requires HDP 2.6.x stack (includes Apache Hive 2.1 w/ LLAP)

Interactive Query

Enable Interactive Query (requires YARN pre-emption)

Yes

Interactive Query Queue

llap

Number of nodes used by Hive's LLAP



Maximum Total Concurrent Queries



Setup: Cloud

- **Spin up LLAP clusters in minutes**
- **Exposes HS2 endpoint**
 - JDBC/ODBC
 - Zeppelin/Hive view also options
- **RDS + S3 friendly**
 - Use HD cache for fast S3 access
 - RDS for shared metastore instance
- **HDP 2.6 available now!**

The screenshot shows the 'CREATE CLUSTER' page in the Hortonworks Data Cloud console. The page is divided into two main sections: 'GENERAL CONFIGURATION' and 'HARDWARE & STORAGE'. Both sections have a 'SHOW ADVANCED OPTIONS' link. In the 'GENERAL CONFIGURATION' section, the 'Cluster Name' is 'llap-cluster', 'HDP Version' is 'HDP 2.6 (Cloud)', and 'Cluster Type' is 'EDW-Analytics: Apache Hive 2 LLAP, Apache Zeppelin 0.7.0'. In the 'HARDWARE & STORAGE' section, the 'Master Instance Type' is 'm4.xlarge (16vCPU, 64.0 GB Memory)', the 'Worker Instance Type' is 'm3.xlarge (4vCPU, 15.0 GB Memory, 2x40 GB Local Storage)', and the 'Instance Count' is '3'. There is a 'node' dropdown next to the instance count. An 'Auto repair' toggle is set to 'ON'. At the bottom, the 'Compute Instance Type' is also 'm3.xlarge (4vCPU, 15.0 GB Memory, 2x40 GB Local Storage)'.

HORTONWORKS DATA CLOUD Settings

CLUSTERS

CREATE CLUSTER

GENERAL CONFIGURATION (* ARE REQUIRED) SHOW ADVANCED OPTIONS

Cluster Name* llap-cluster

HDP Version HDP 2.6 (Cloud)

Cluster Type EDW-Analytics: Apache Hive 2 LLAP, Apache Zeppelin 0.7.0

HARDWARE & STORAGE SHOW ADVANCED OPTIONS

Master Instance Type m4.xlarge (16vCPU, 64.0 GB Memory)

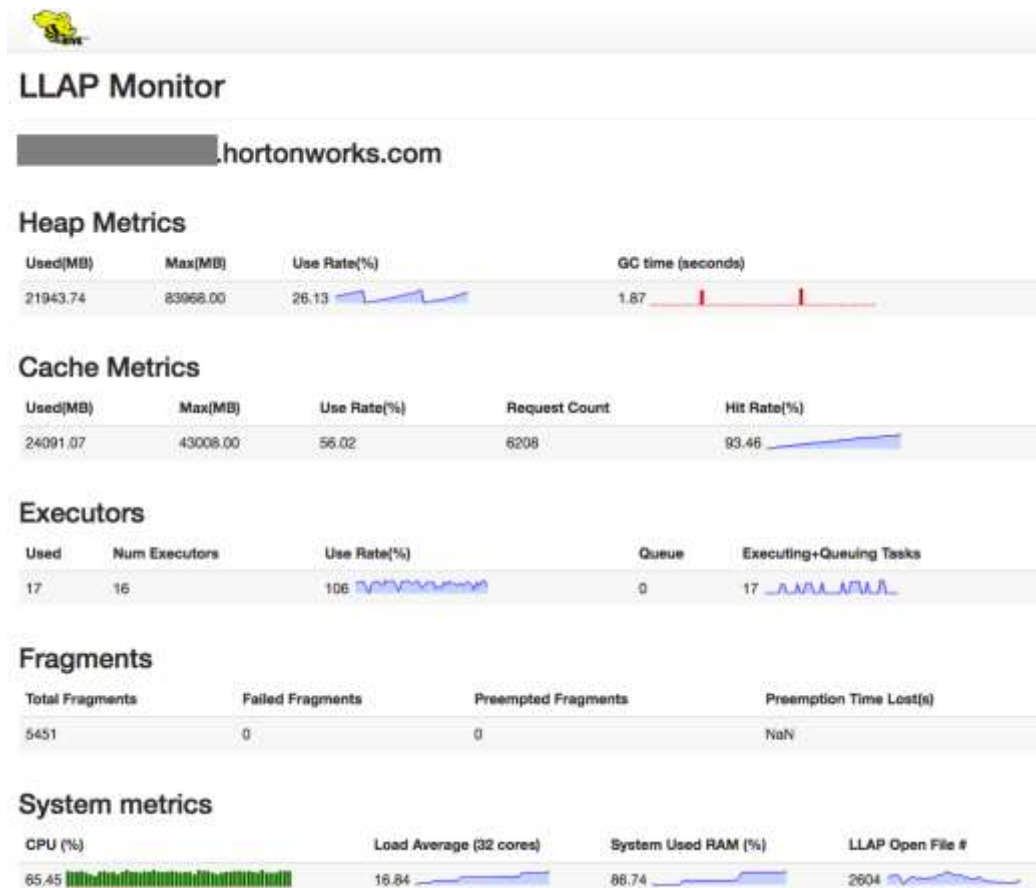
Worker Instance Type m3.xlarge (4vCPU, 15.0 GB Memory, 2x40 GB Local Storage)

Instance Count 3 node **ON** Auto repair

Compute Instance Type m3.xlarge (4vCPU, 15.0 GB Memory, 2x40 GB Local Storage)

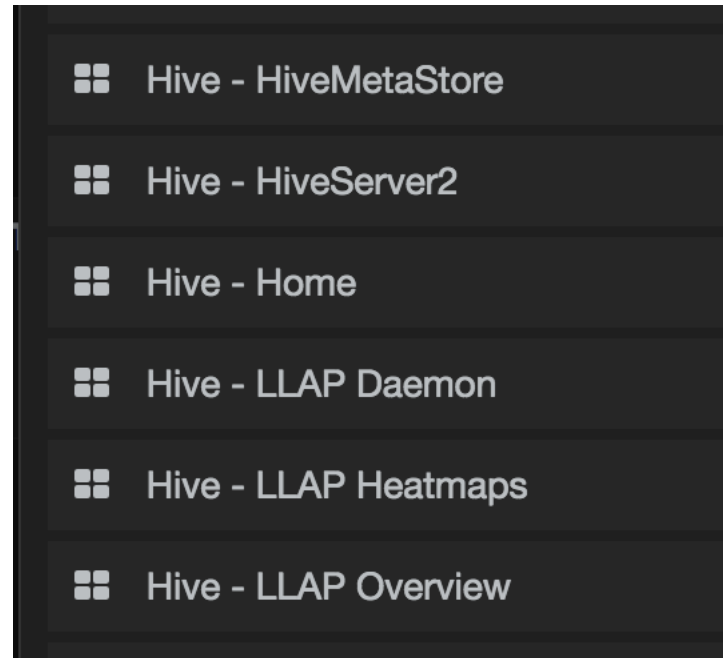
Monitoring

- Simple UI for monitoring
- JMX endpoint with more data
- Logs and jstack endpoints



Monitoring

- HDP ships with number of default Grafana dashboards for LLAP
- Show wealth of metrics for cache, compute and scheduling
- Helps to identify bottlenecks and issues
- Helps with capacity planning
- Easy to extend or integrate with other systems



Watching queries – Tez UI integration

TEZ All DAGs / DAG [select i_brand_id brand_id, i... / Graphical View Version 0.8.3-SNAPSHOT ?

DAG Details DAG Counters Graphical View All Vertices All Tasks All Task Attempts ☒ Auto Refresh Last refreshed at 28 Mar 2016 15:56:32 Refresh

The DAG shows the following flow:

- Inputs: **item** (green oval) and **date_dim** (green oval).
- Map 5 (SUCCEEDED) receives input from **item**.
- Map 4 (SUCCEEDED) receives input from **date_dim**.
- Map 1 (SUCCEEDED) receives input from both Map 5 and Map 4.
- Reducer 2 (SUCCEEDED) receives input from Map 1.
- Reducer 3 (undefined) receives input from Reducer 2.
- Output: **out.Reducer...** (red oval) receives input from Reducer 3.

Map 1 Task Details:

Vertex Name	Map 1
Vertex Id	vertex_1455662455106_10401_318_02
Status	SUCCEEDED
Duration	1210
First Task Start Time	1459203764104
Tasks	16
LlapIOs - CACHE_HIT_BYTES	365787310
LlapIOs - CACHE_MISS_BYTES	174196594
LlapIOs - METADATA_CACHE_HIT	84
LlapIOs - METADATA_CACHE_MISS	40
LlapIOs - NUM_VECTOR_BATCHES	93144
LlapIOs - ROWS_EMITTED	93131406
LlapIOs - HDFS_TIME_NS	1148123937

Future work

- **Fine-grained workload management**
 - Reservations & priorities
- **Configurable guardrails**
 - Maximum permissible scan range, runtime, cost
 - Integrated with CBO's cost model
- **Admin tools**
 - Identify hotspots & data layout issues
 - Easy reporting on health, performance and utilization

Summary

- LLAP is a new execution substrate for fast, concurrent analytical workloads, harnessing Hive vectorized SQL engine and efficient in-memory caching layer
- Provides secure relational view of the data through a simple API
- Available in Hive 2, integrated with ecosystem (Ambari, Cloud, Grafana, jmx, Hive & Tez views)

Questions?



Interested? Stop by the Hortonworks booth to learn more