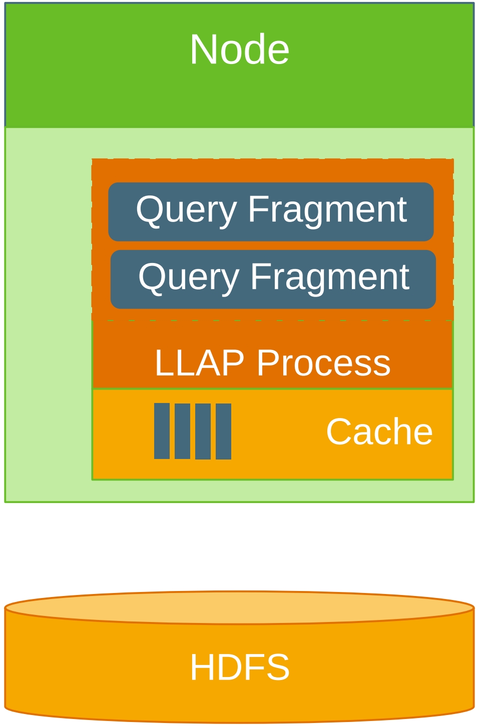
Hortonworks

LLAP: Sub-Second Analytical Queries in Hive

Gopal Vijayaraghavan

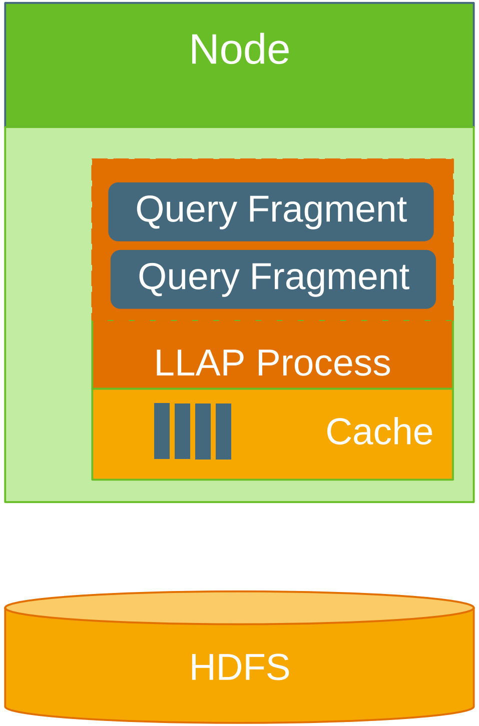
Why LLAP?

- People like Hive
- Disk->Mem is getting further away
  - Cloud Storage isn't co-located
  - Disks are connected to the CPU via network
- Security landscape is changing
  - Cells & Columns are the new security boundary, not files
  - Safely masking columns needs a process boundary
- Concurrency, Performance & Scale are at conflict
  - Concurrency at 100k queries/hour
  - Latencies at 2-5 seconds/query
  - Petabyte scale warehouses (with terabytes of "hot" data)



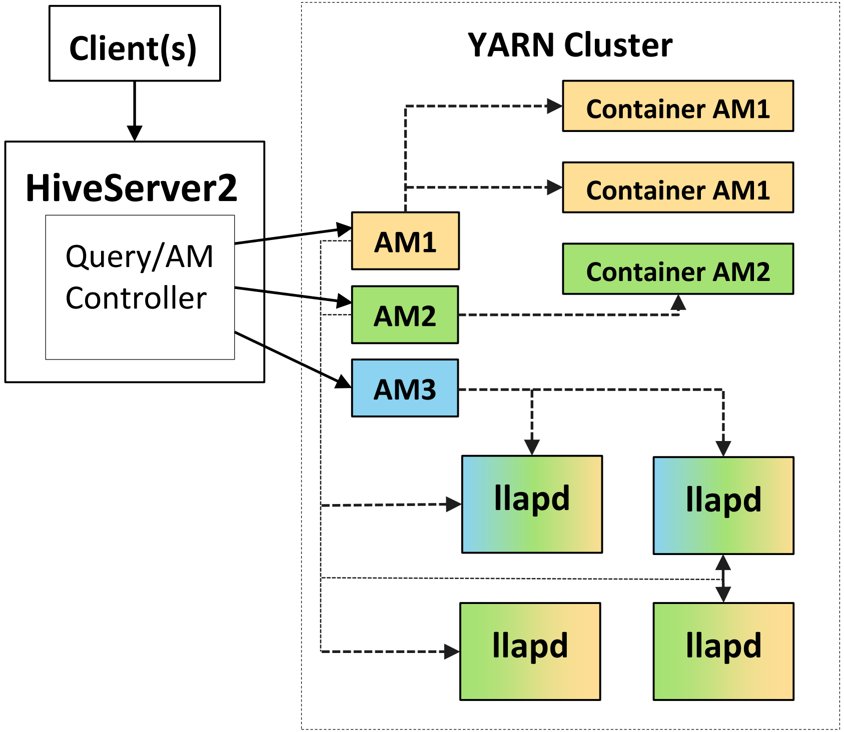
What is LLAP?

- Hybrid model combining daemons and containers for fast, concurrent execution of analytical workloads (e.g. Hive SQL queries)
  - Concurrent queries without specialized YARN queue setup
  - Multi-threaded execution of vectorized operator pipelines
- Asynchronous IO and efficient in-memory caching
- Relational view of the data available thru the API
  - High performance scans, execution code pushdown
  - Centralized data security

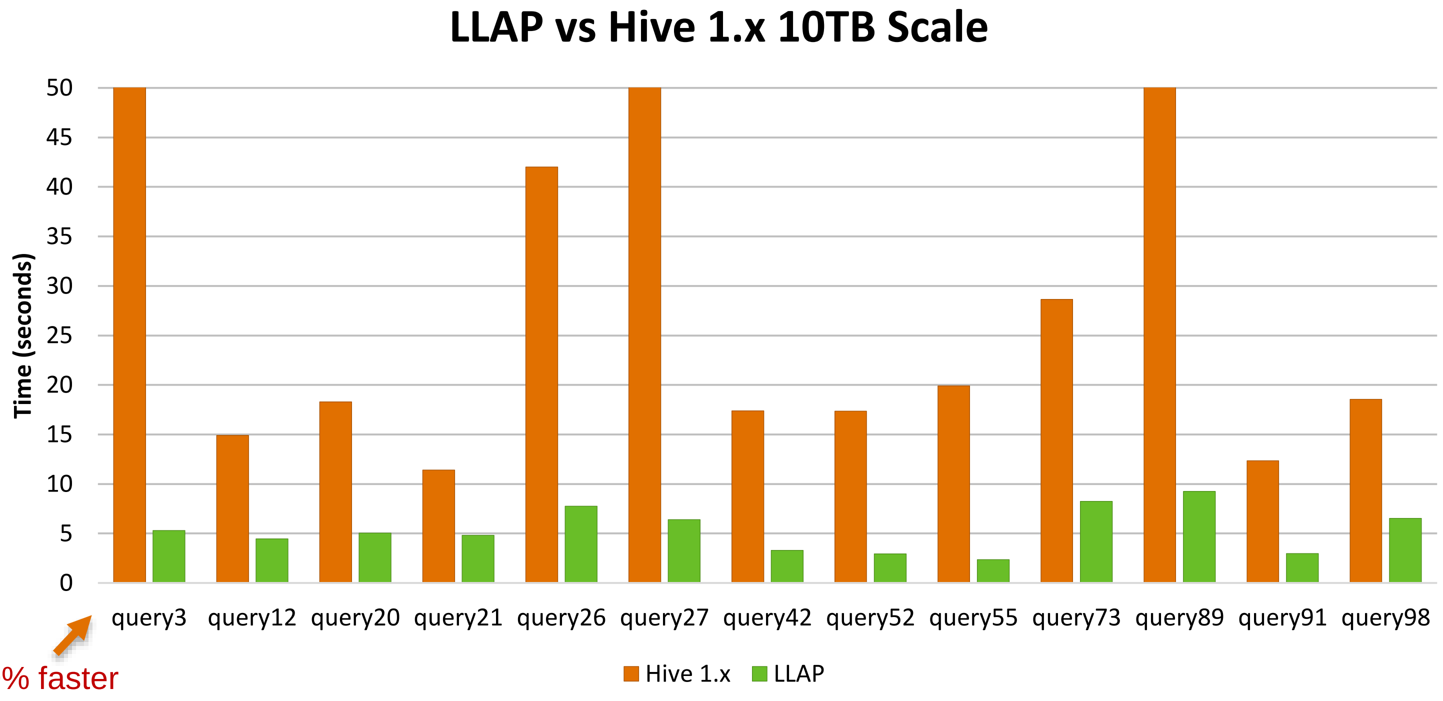


Hive 2.0 (+ LLAP)

- Transparent to Hive users, BI tools, etc.
- Hive decides where query fragments run (LLAP, Container, AM) based on configuration, data size, format, etc.
- Each Query coordinated independently by a Tez AM
- Number of concurrent queries throttled by number of active AMs
- Hive Operators used for processing
- Tez Runtime used for data transfer



Industry benchmark – 10Tb scale

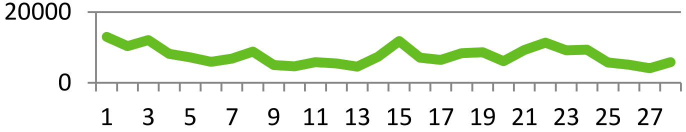


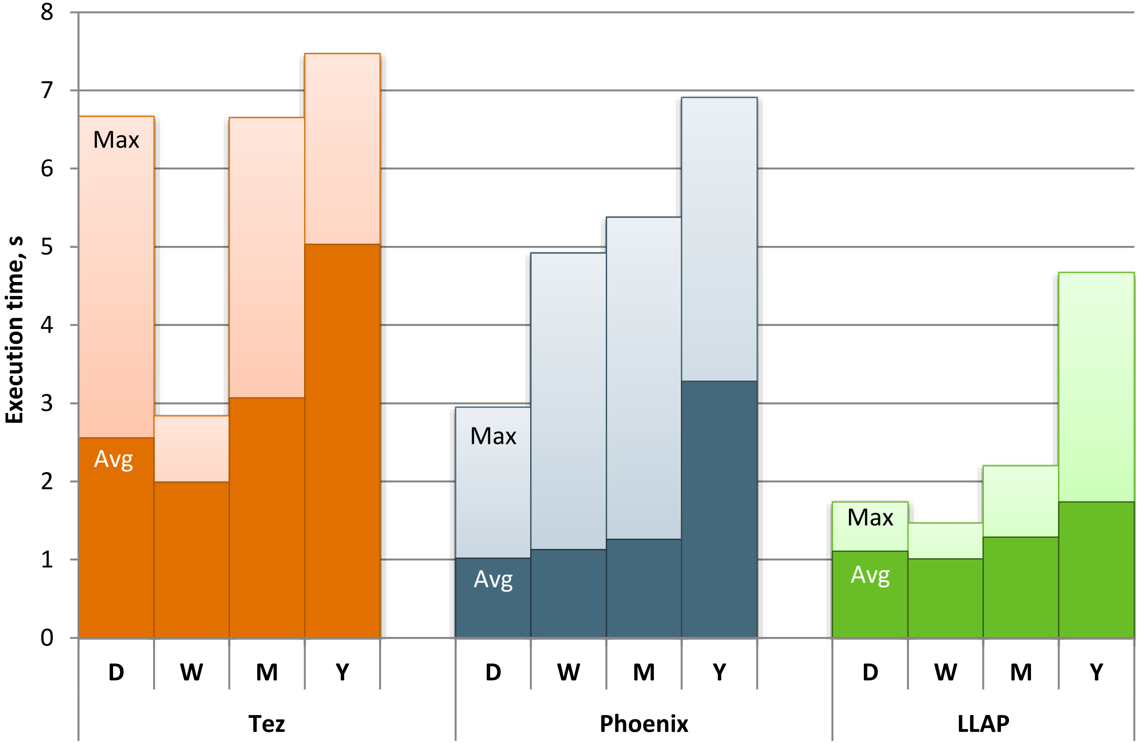
90% faster

Hortonworks

Evaluation from a customer case study

- Aggregate daily statistics for a time interval:



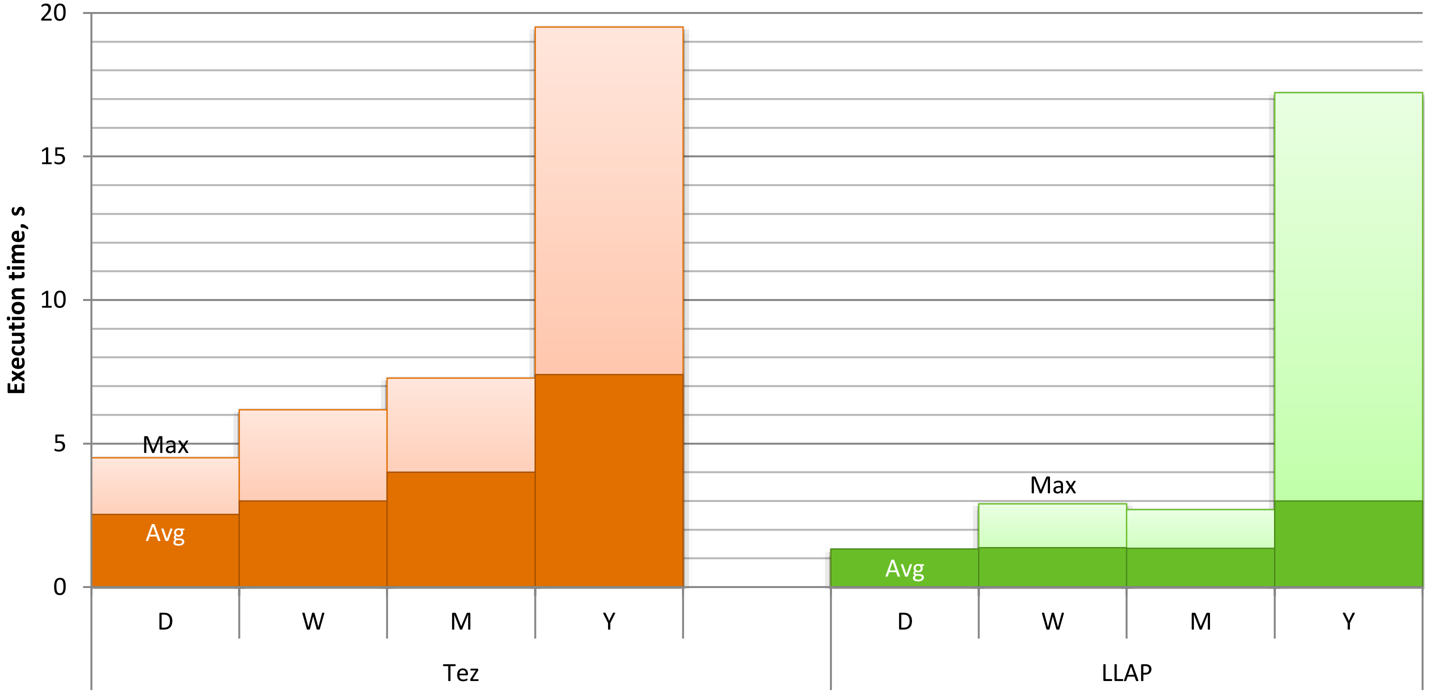


```
SELECT yyyymmdd,
  sum(total_1),
  sum(total_2),
             ...
  from table
  where yyyymmdd >= xxx
           and yyyymmdd < xxx
   and userid = xxx
  group by userid, yyyymmdd;
```

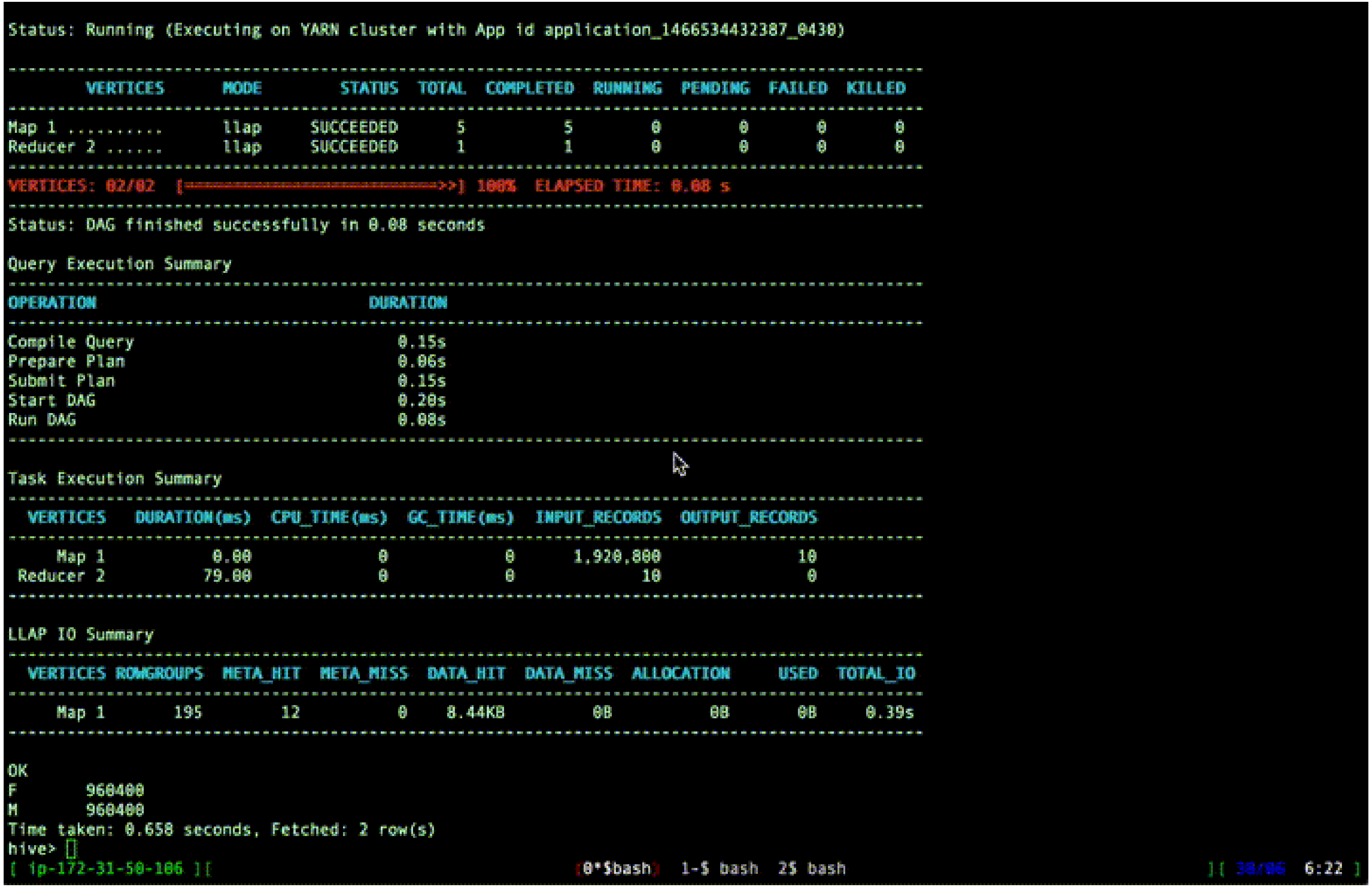
Hortonworks

Evaluation from a customer case study

- Display a large report



Cut-away to demo (GIF)

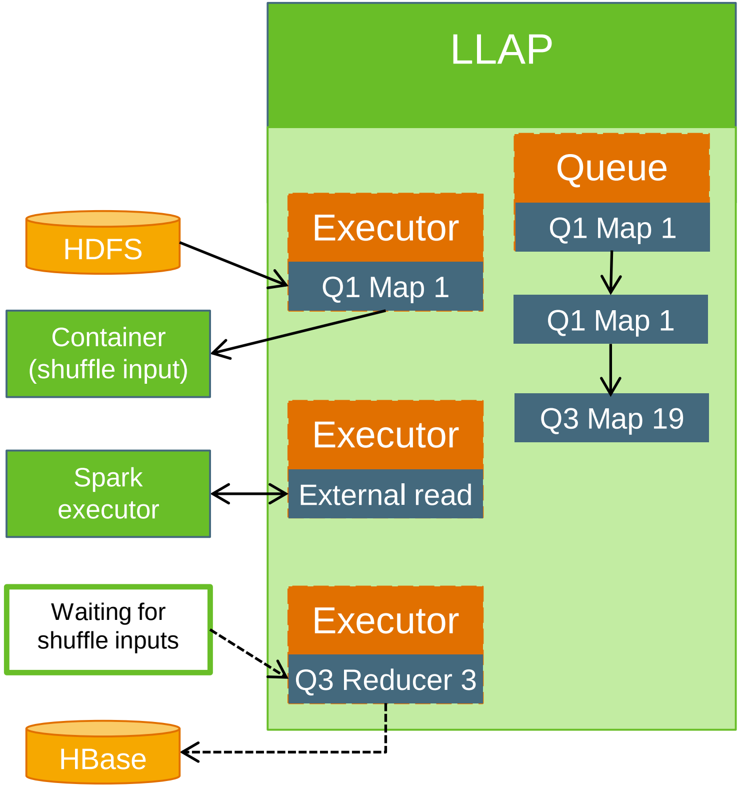


```
Status: Running (Executing on YARN cluster with App id application_1466534432387_0430)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........      llap     SUCCEEDED      5          5        0        0       0       0
Reducer 2 ......      llap     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 0.08 s
----------------------------------------------------------------------------------------------
Status: DAG finished successfully in 0.08 seconds

Query Execution Summary
----------------------------------------------------------------------------------------------
OPERATION                            DURATION
----------------------------------------------------------------------------------------------
Compile Query                           0.15s
Prepare Plan                            0.06s
Submit Plan                             0.15s
Start DAG                               0.20s
Run DAG                                 0.08s
----------------------------------------------------------------------------------------------

Task Execution Summary
----------------------------------------------------------------------------------------------
  VERTICES   DURATION(ms)  CPU_TIME(ms)  GC_TIME(ms)  INPUT_RECORDS  OUTPUT_RECORDS
----------------------------------------------------------------------------------------------
     Map 1           0.00             0            0      1,920,800              10
 Reducer 2          79.00             0            0             10               0
----------------------------------------------------------------------------------------------

LLAP IO Summary
----------------------------------------------------------------------------------------------
  VERTICES ROWGROUPS  META_HIT  META_MISS  DATA_HIT  DATA_MISS  ALLOCATION     USED  TOTAL_IO
----------------------------------------------------------------------------------------------
     Map 1       195        12          0    8.44KB         0B          0B       0B     0.39s
----------------------------------------------------------------------------------------------

OK
F       960400
M       960400
Time taken: 0.658 seconds, Fetched: 2 row(s)
hive> 
[ ip-172-31-50-106 ][                        (0*$bash)  1-$ bash  2$ bash                          ][ 38/06  6:22 ]
```

How does LLAP make queries faster?

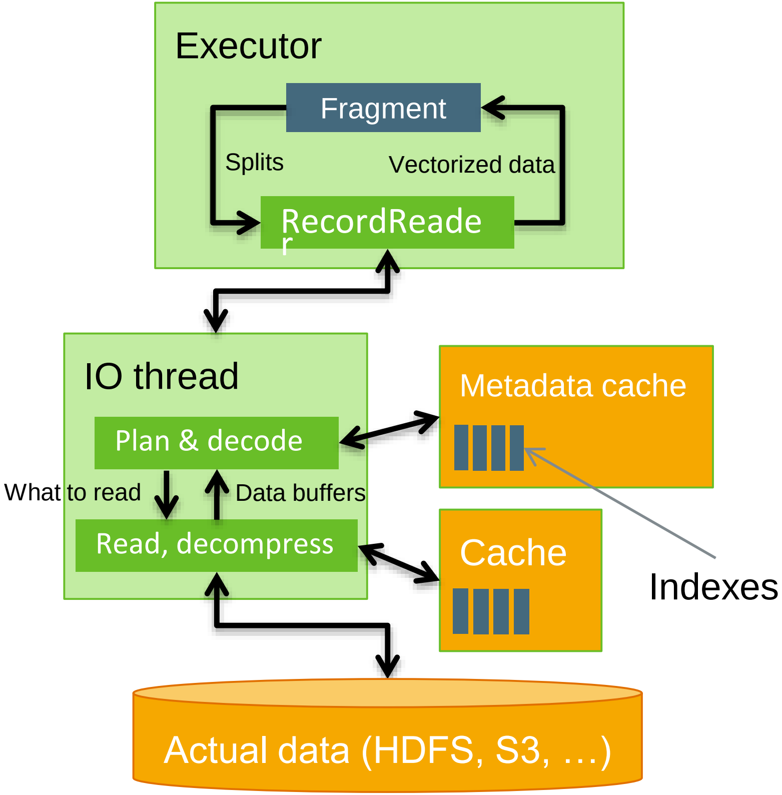
Technical overview – execution

- LLAP daemon has a number of executors (think containers) that execute work "fragments"
- Fragments are parts of one, or multiple parallel workloads (e.g. Hive SQL queries)
- Work queue with pluggable priority
  - Geared towards low latency queries over long-running queries (by default)
- I/O is similar to containers – read/write to HDFS, shuffle, other storages and formats
- Streaming output for data API



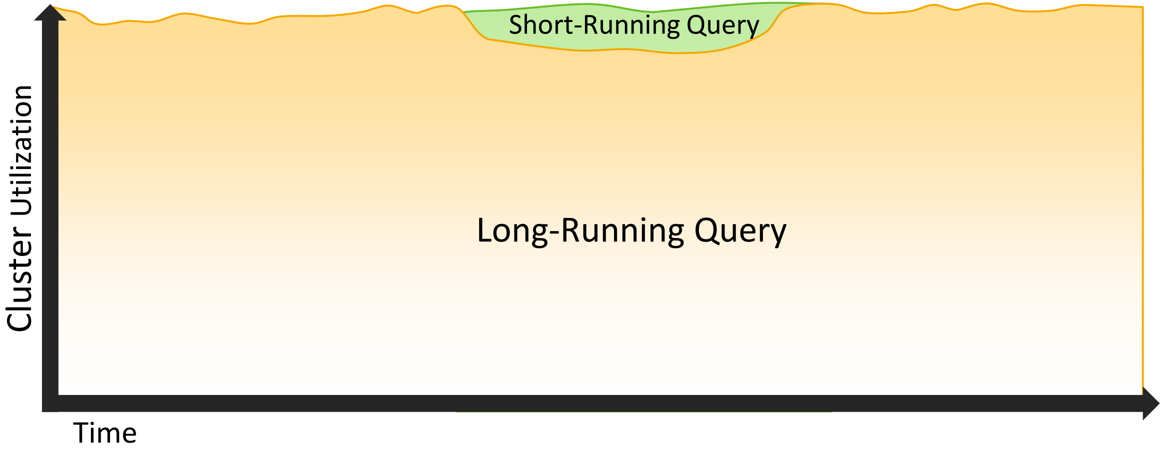
Technical overview – IO layer

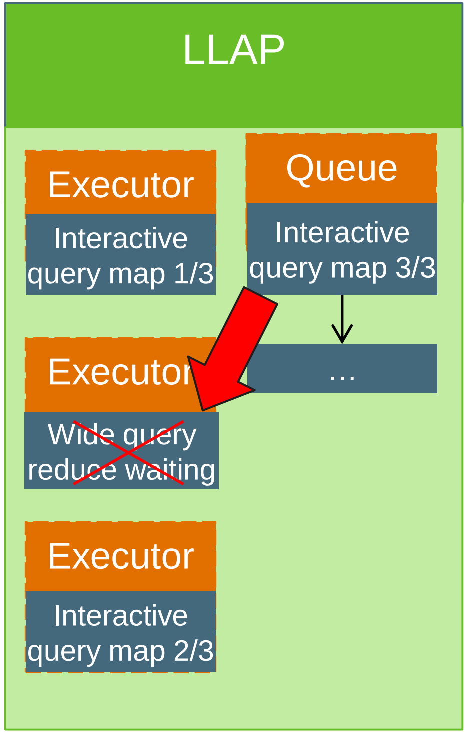
- **Optional:** when executing inside LLAP
  - All other formats use in-sync mode
- Asynchronous IO for Hive
  - Wraps over InputFormat, reads through cache
  - Supported with ORC
- Transparent, compressed in-memory cache
  - Format-specific, extensible
  - NVMe/NVDIMM caches



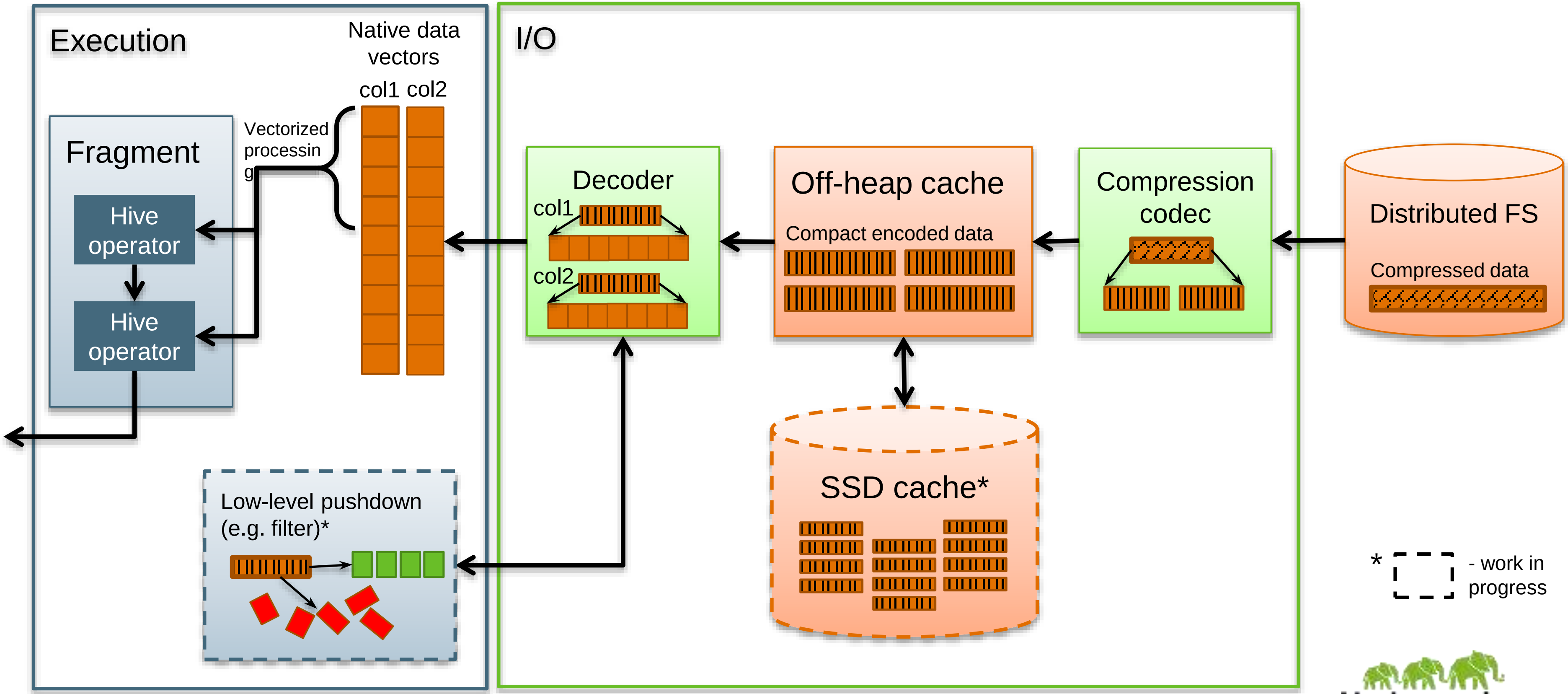
Parallel queries – priorities, preemption

- 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




LLAP
Executor
Interactive query map 1/3
Queue
Interactive query map 3/3
Executor
~~Wide query reduce waiting~~
…
Executor
Interactive query map 2/3

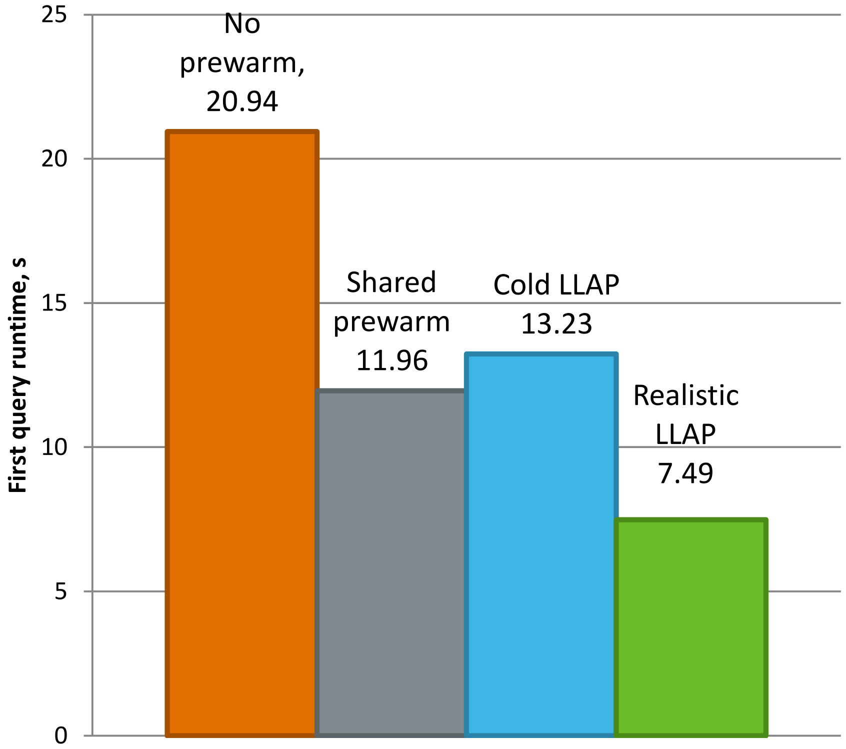
In-memory processing – present and future



Hortonworks

First query erformance

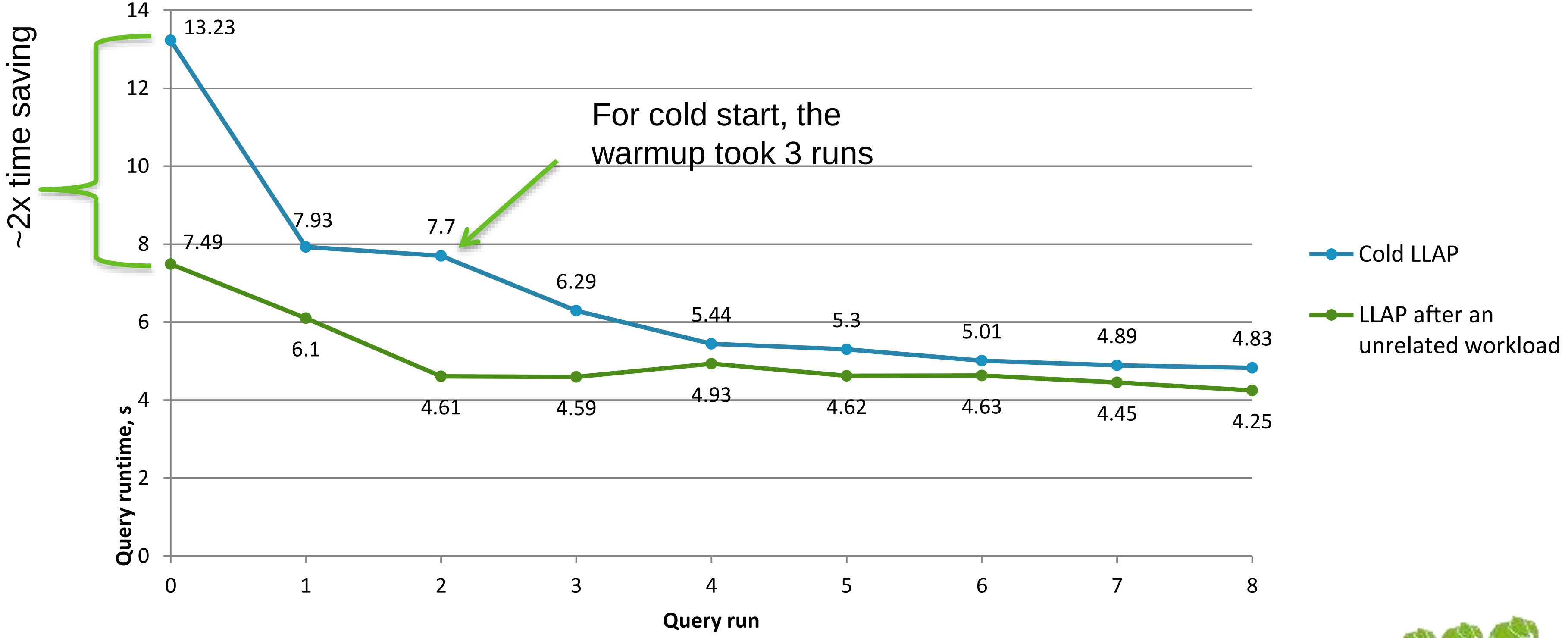
- Cold LLAP is nearly as fast as shared pre-warmed containers (impractical on real clusters)
- Realistic (long-running) LLAP ~3x faster than realistic (no prewarm) Tez



Hortonworks

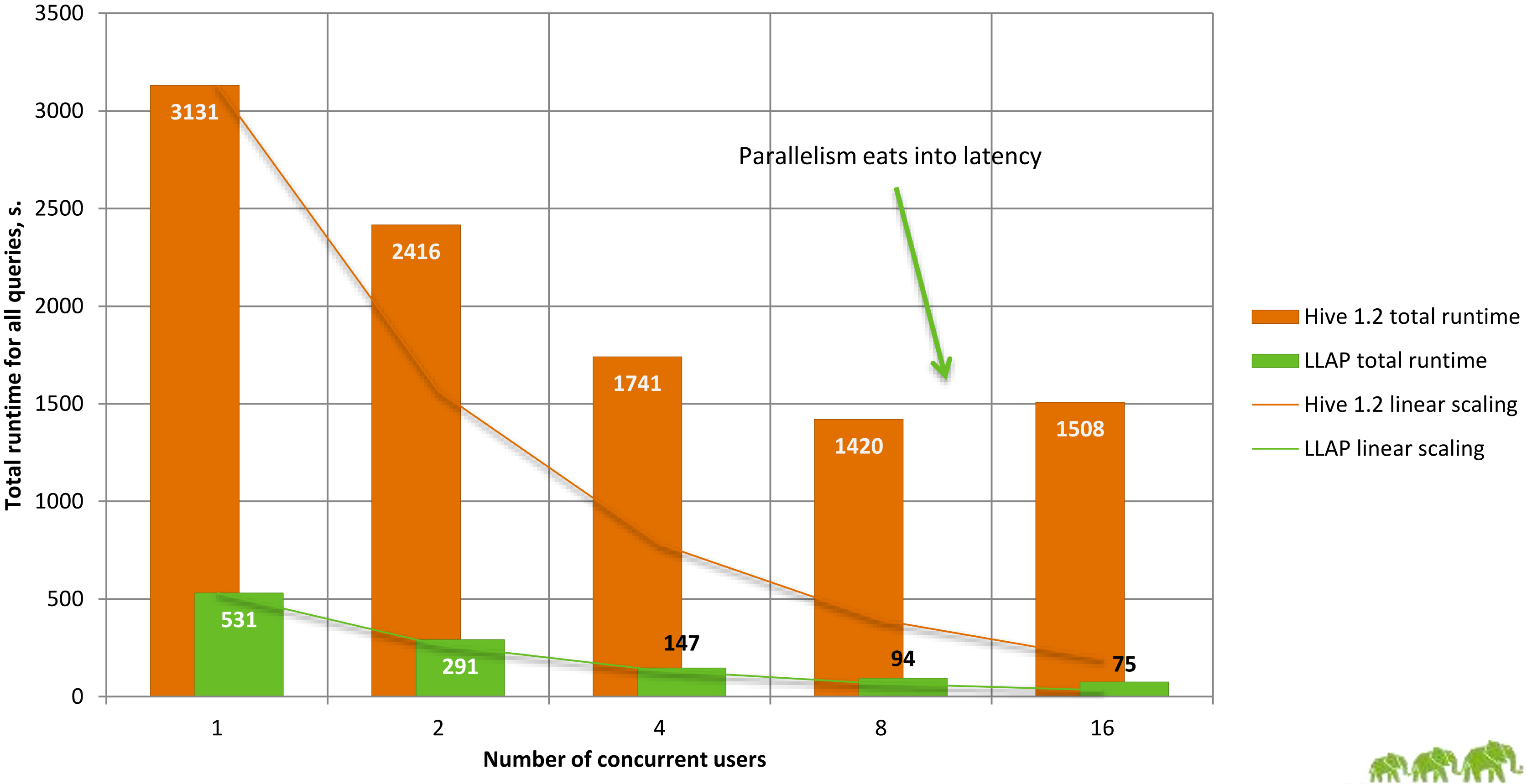
JIT Performance – heavy use

- Cache disabled!



Hortonworks

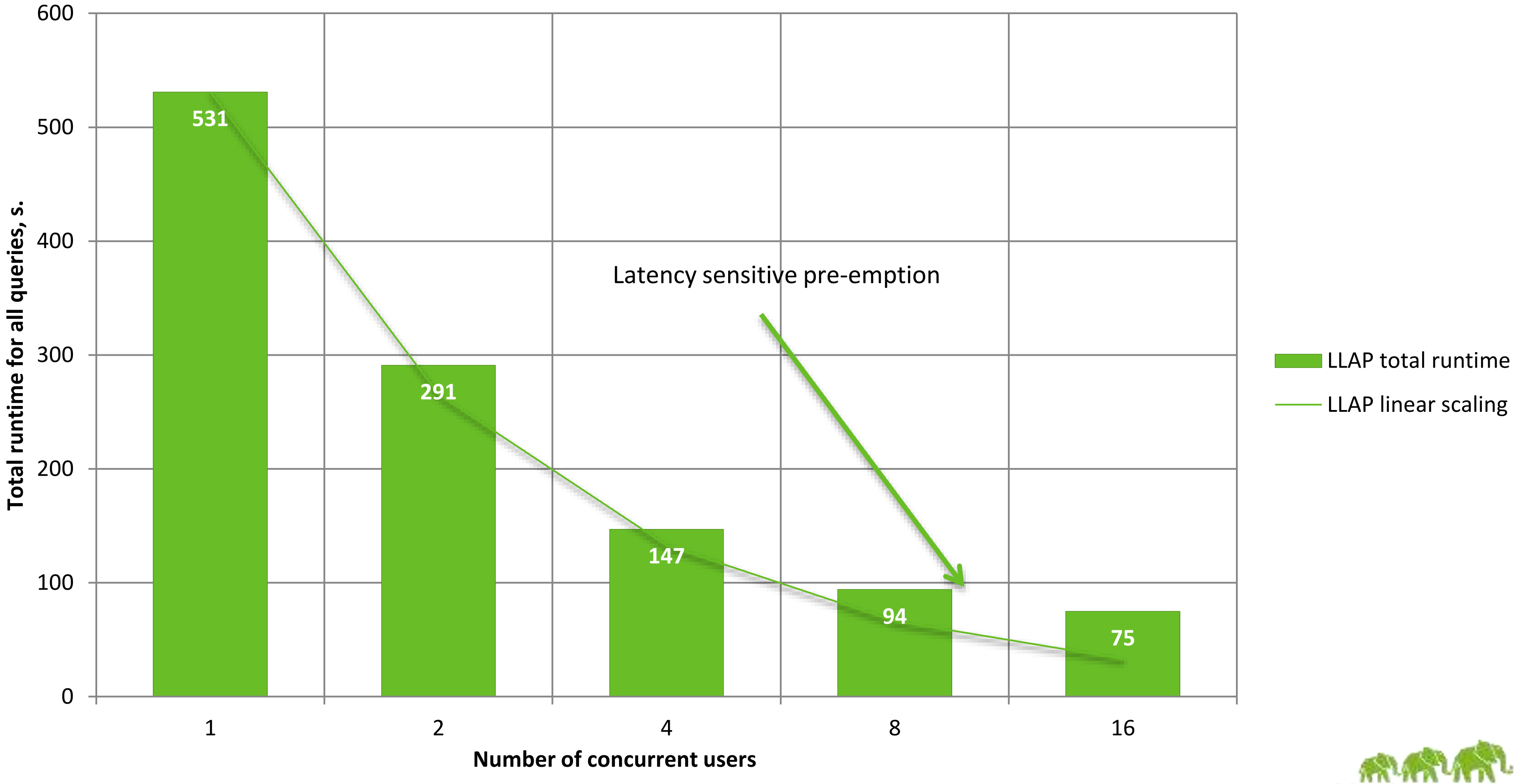
Parallel query execution – LLAP vs Hive 1.2



Parallelism eats into latency

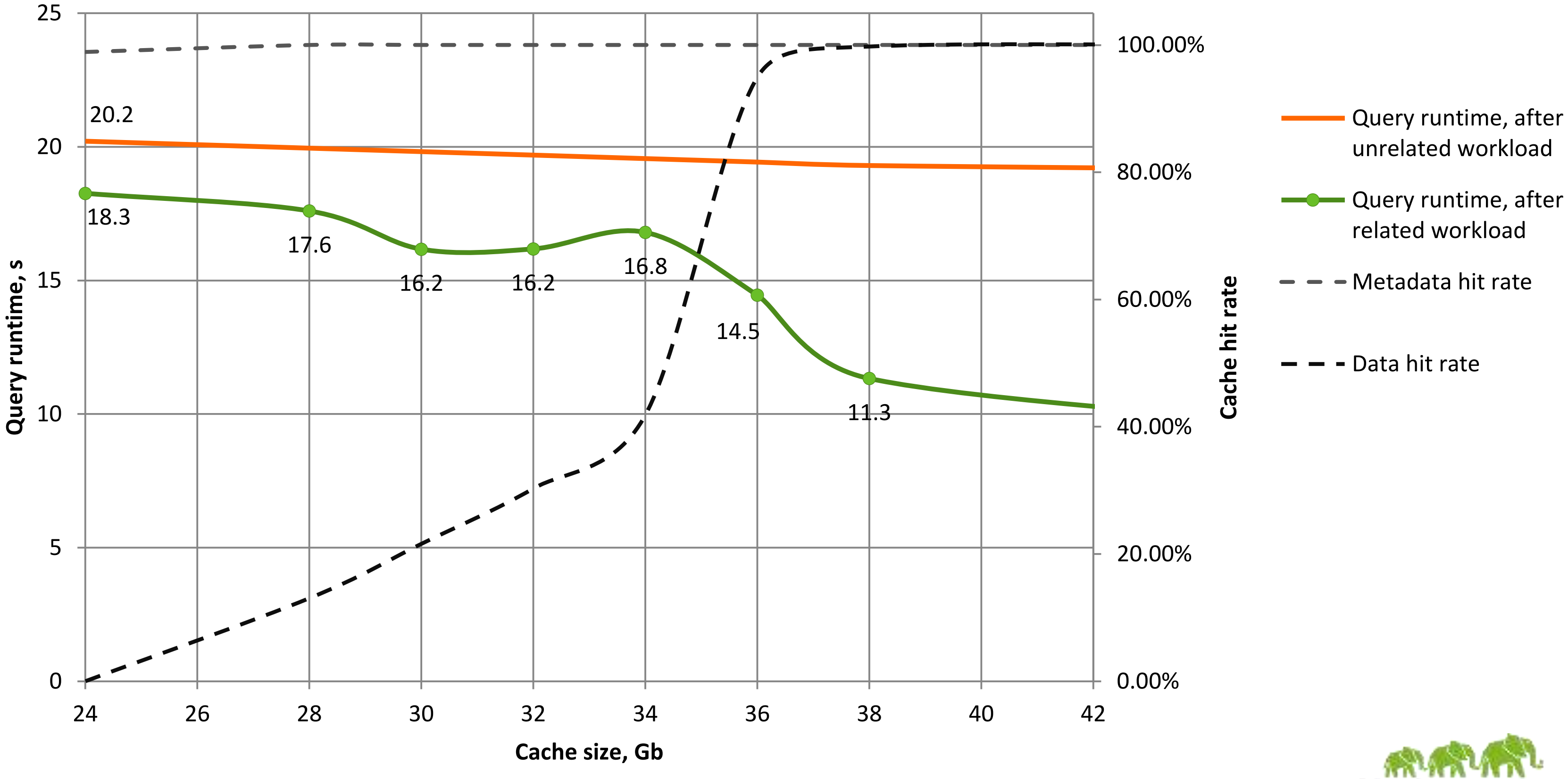
Hortonworks

Parallel query execution – 10Tb scale



Hortonworks

Performance – cache on HDFS, 1Tb scale



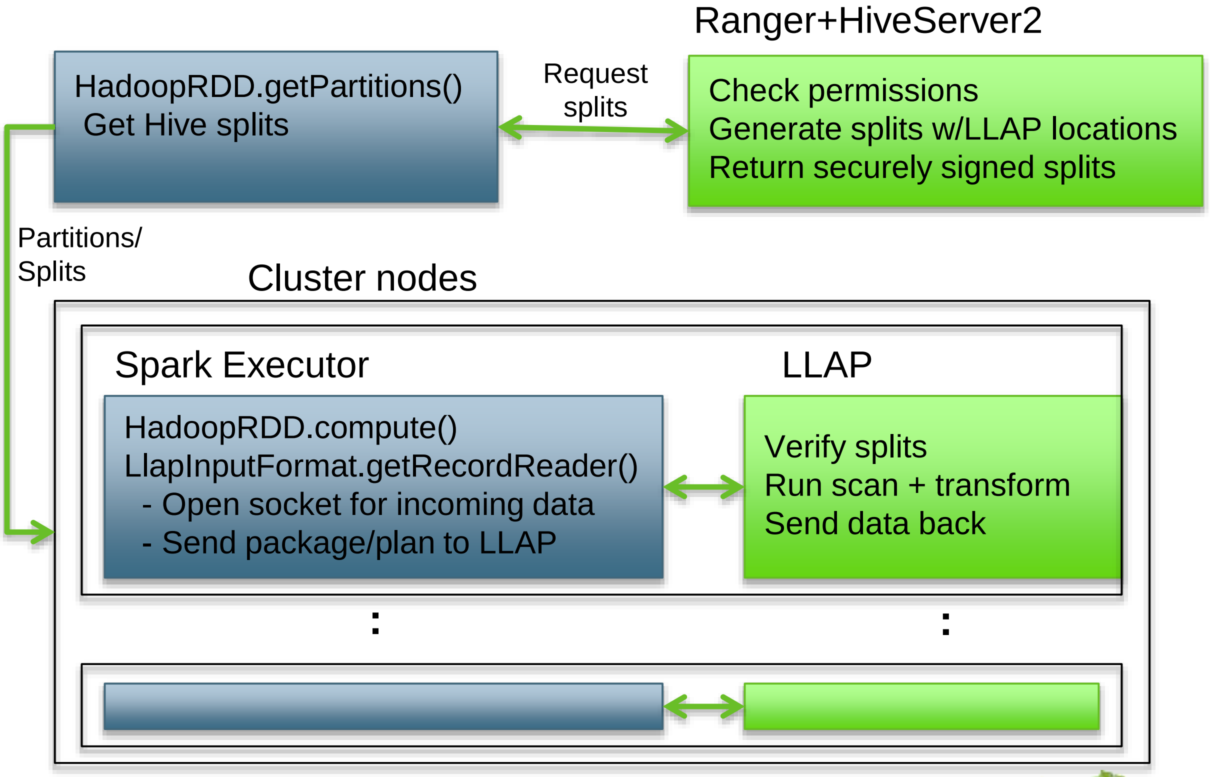
Hortonworks

LLAP as a “relational” datanode

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



Ranger+HiveServer2

HadoopRDD.getPartitions()
Get Hive splits

Request splits

Check permissions
Generate splits w/LLAP locations
Return securely signed splits

Partitions/ Splits

Cluster nodes

Spark Executor

LLAP

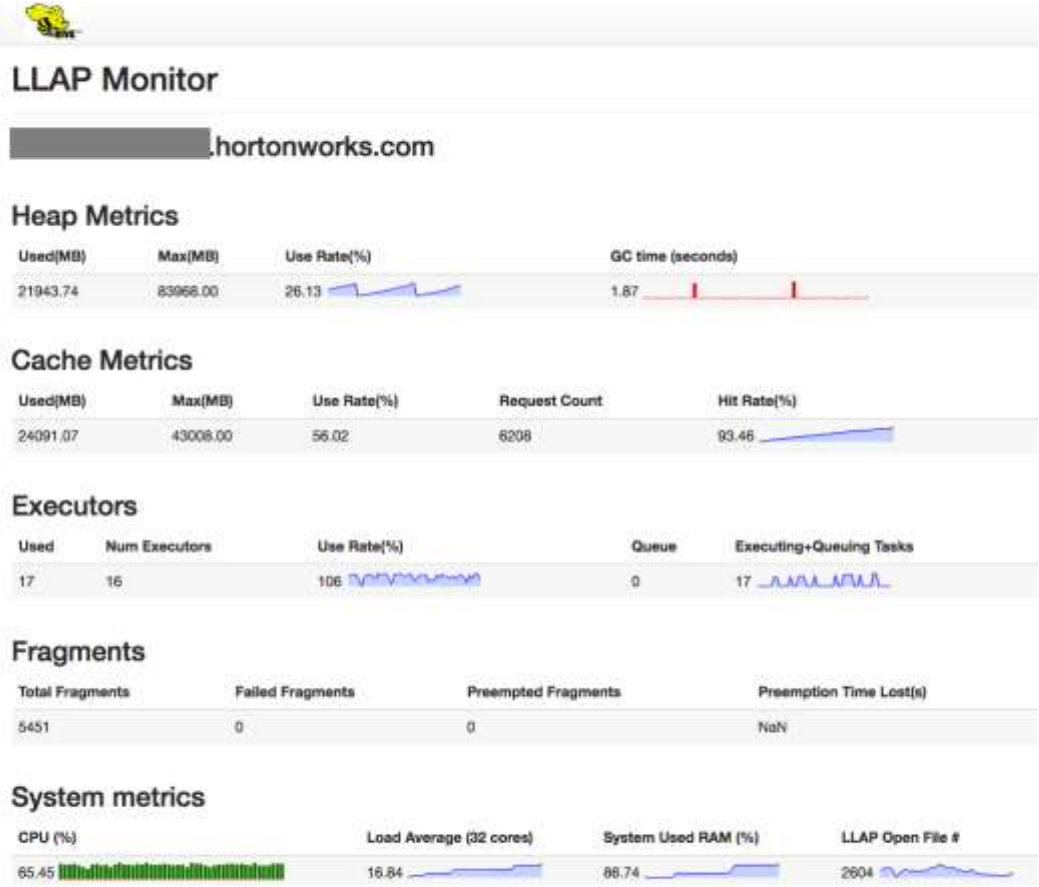
HadoopRDD.compute()
LlapInputFormat.getRecordReader()
- Open socket for incoming data
- Send package/plan to LLAP

Verify splits
Run scan + transform
Send data back

Monitoring LLAP Queries

Monitoring

- LLAP exposes a UI for monitoring
- Also has jmx endpoint with much more data, logs and jstack endpoints as usual
- Aggregate monitoring UI is work in progress



LLAP Monitor

[redacted].hortonworks.com

Heap Metrics

Used(MB)	Max(MB)	Use Rate(%)	GC time (seconds)
21943.74	83968.00	26.13	1.87

Cache Metrics

Used(MB)	Max(MB)	Use Rate(%)	Request Count	Hit Rate(%)
24091.07	43008.00	56.02	6208	93.46

Executors

Used	Num Executors	Use Rate(%)	Queue	Executing+Queuing Tasks
17	16	106	0	17

Fragments

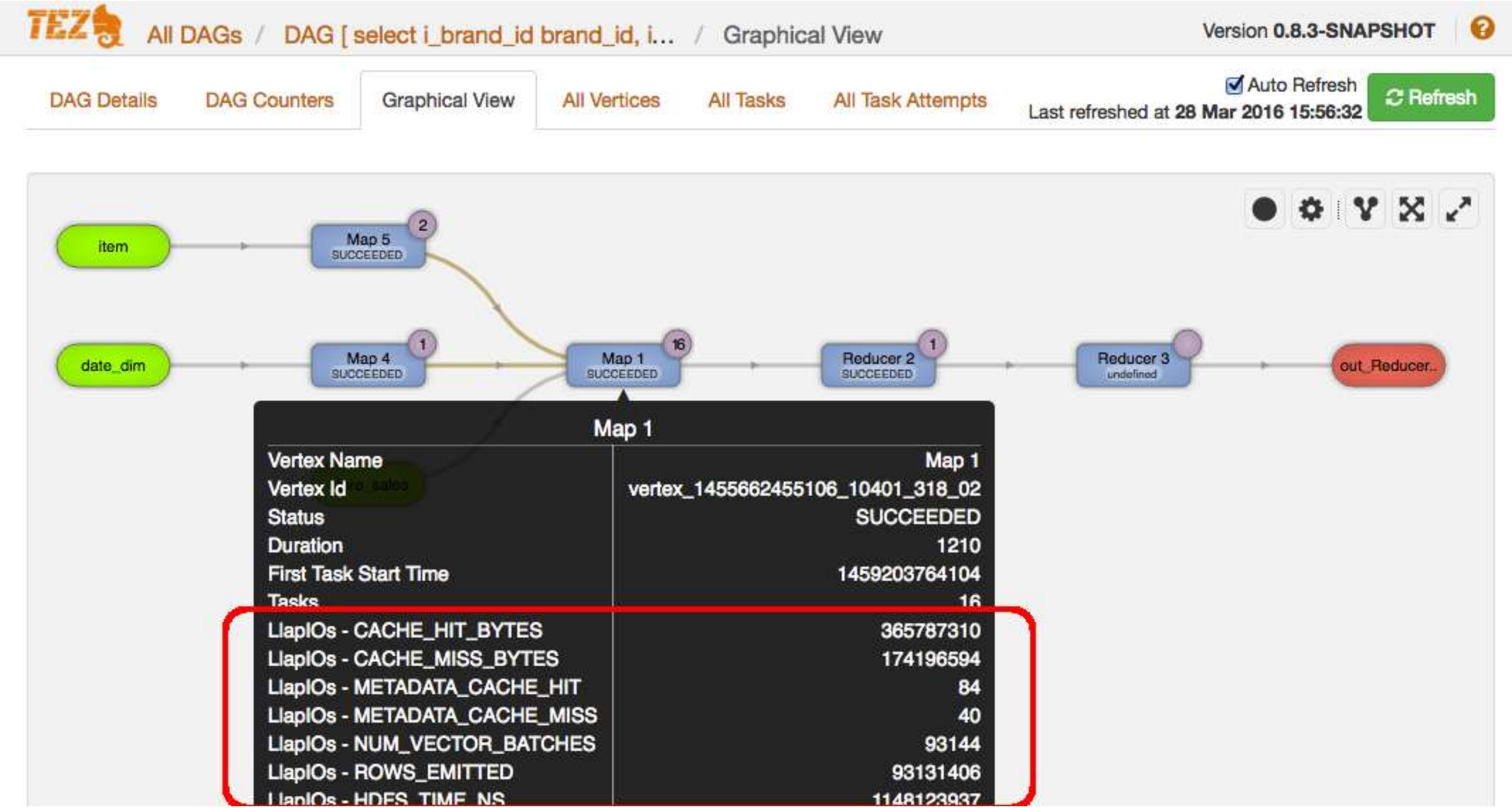
Total Fragments	Failed Fragments	Preempted Fragments	Preemption Time Lost(s)
5451	0	0	NaN

System metrics

CPU (%)	Load Average (32 cores)	System Used RAM (%)	LLAP Open File #
65.45	16.84	88.74	2604

Hortonworks

Watching queries – Tez UI integration



Map 1

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

Questions?

?

Interested? Stop by the Hortonworks booth to learn more