LLAP: Sub-Second Analytical Queries in Hive

Sergey Shelukhin, Siddharth Seth

LLAP: long-lived execution in Hive

- What is LLAP?
- LLAP in Hive; performance primer
- How does LLAP make queries faster?
- LLAP as a unified data access layer
- Deploying and managing LLAP
- Future work

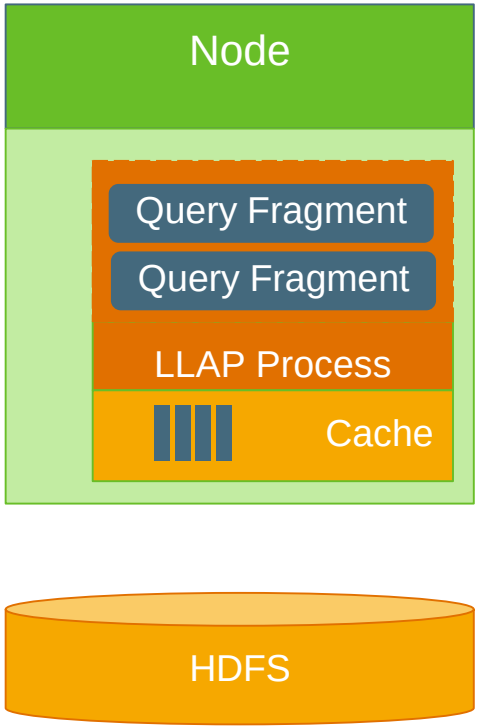
Hortonworks

Hortonworks

What is LLAP?

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

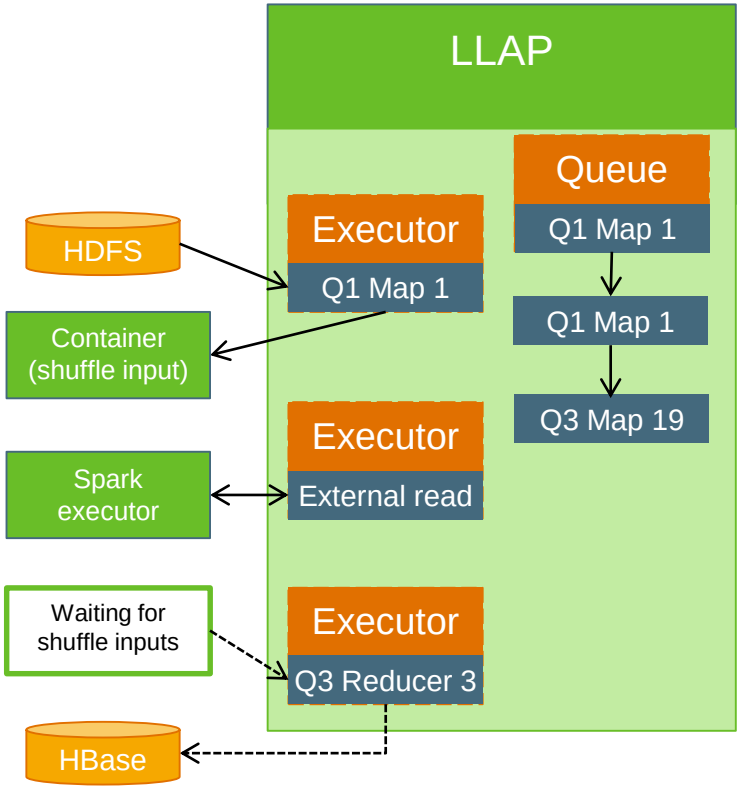


What LLAP isn't

- Not another "execution engine" (like Tez, MR, Spark…)
  - Tez, Spark (work in progress) can work on top of LLAP
    - Engines provide coordination and scheduling
    - some work (e.g. large shuffles) can be scheduled in containers
- Not a storage substrate
  - Daemons are stateless and read (and cache) data from HDFS/S3/Azure/..

Technical overview – execution

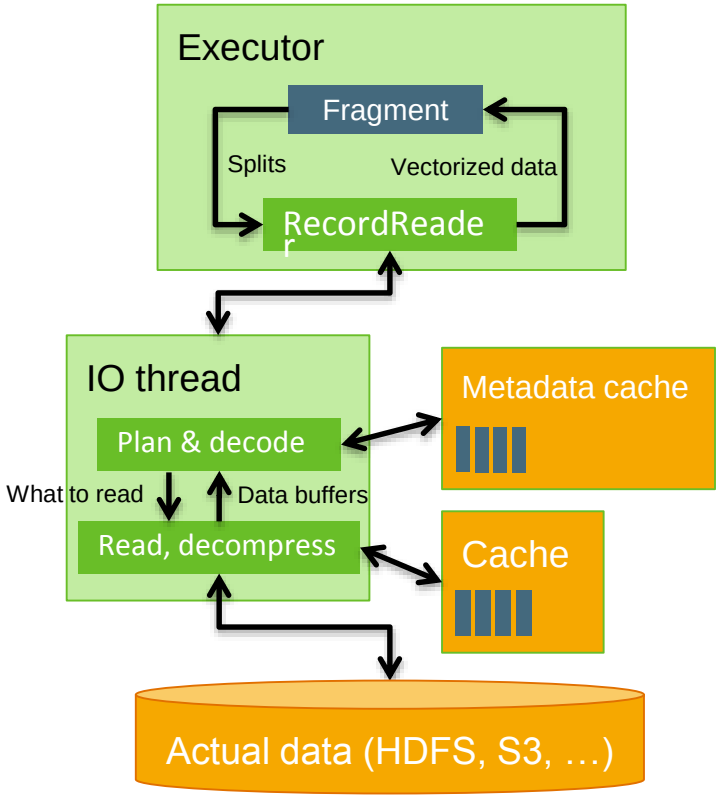
- LLAP daemon has a number of executors (think containers) that execute work "fragments"
- Fragments are parts of one, or multiple parallel workloads (e.g. Hive SQL queries)
- Work queue with pluggable priority
  - Geared towards low latency queries over long-running queries (by default)
- I/O is similar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



Hortonworks

Technical overview – IO layer

- Optional when executing inside LLAP
- Wraps existing InputFormat
  - Currently only ORC is supported
- Asynchronous IO for Hive
- Transparent, compressed in-memory cache
  - Format-specific, extensible
  - SSD cache is work in progress

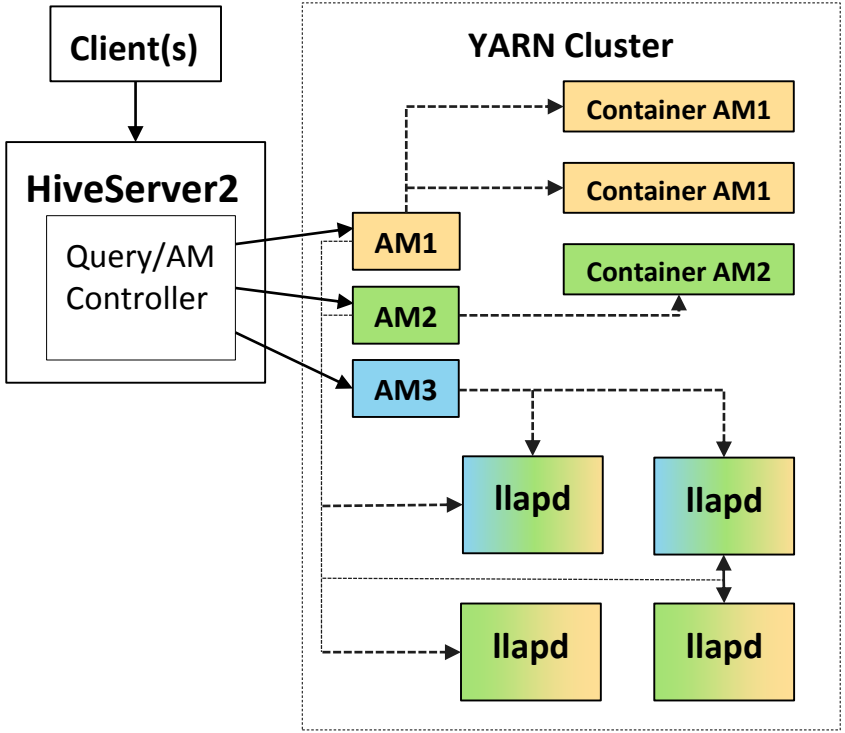


Hortonworks

LLAP in Hive; performance primer

Overview - Hive + LLAP

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



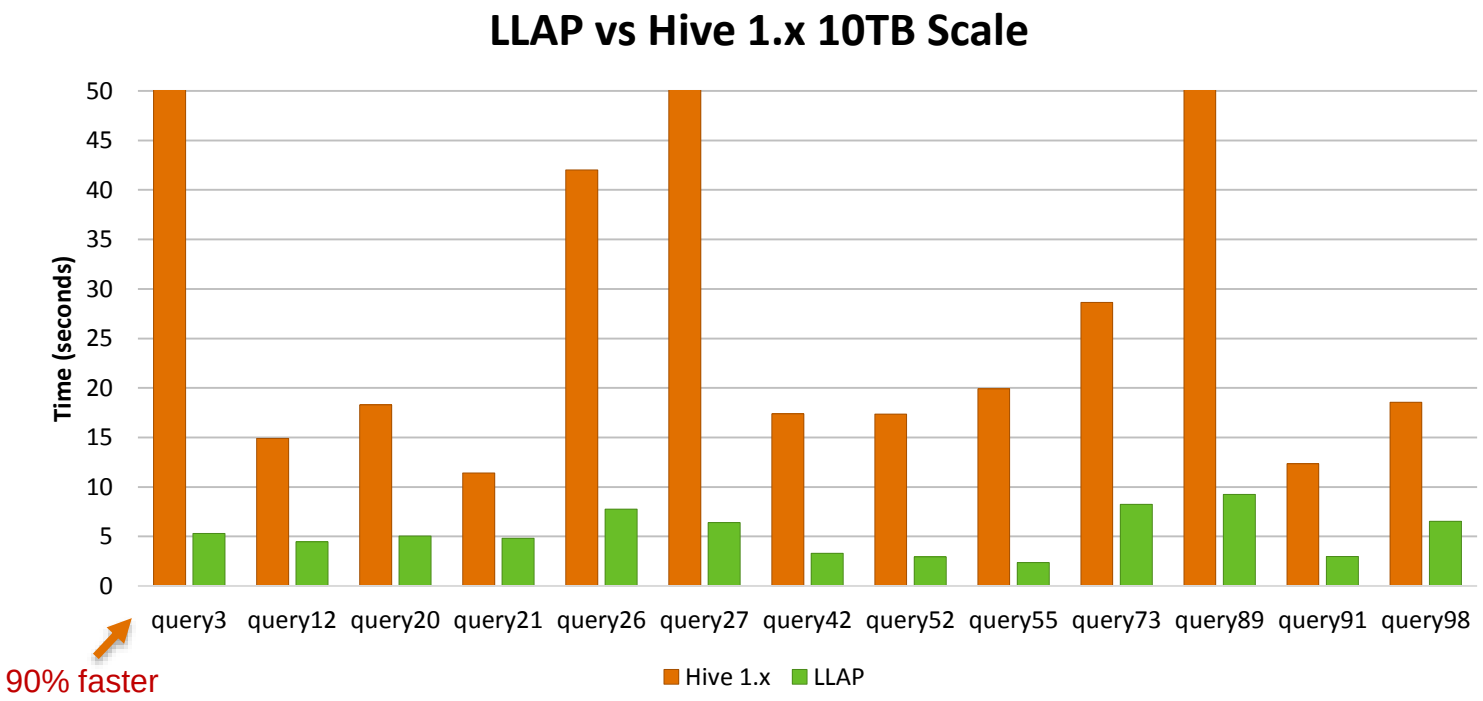
Hortonworks

How does LLAP speed up the queries?

- Eliminates container startup costs
- JIT optimizer has a chance to work
  - Especially important for vectorized processing
- Asynchronous IO
- Caching
- Data sharing (hash join tables, etc.)
- Optimizations for parallel queries

Hortonworks

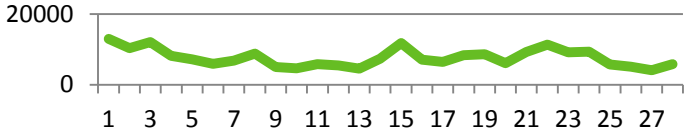
Industry benchmark – 10Tb scale

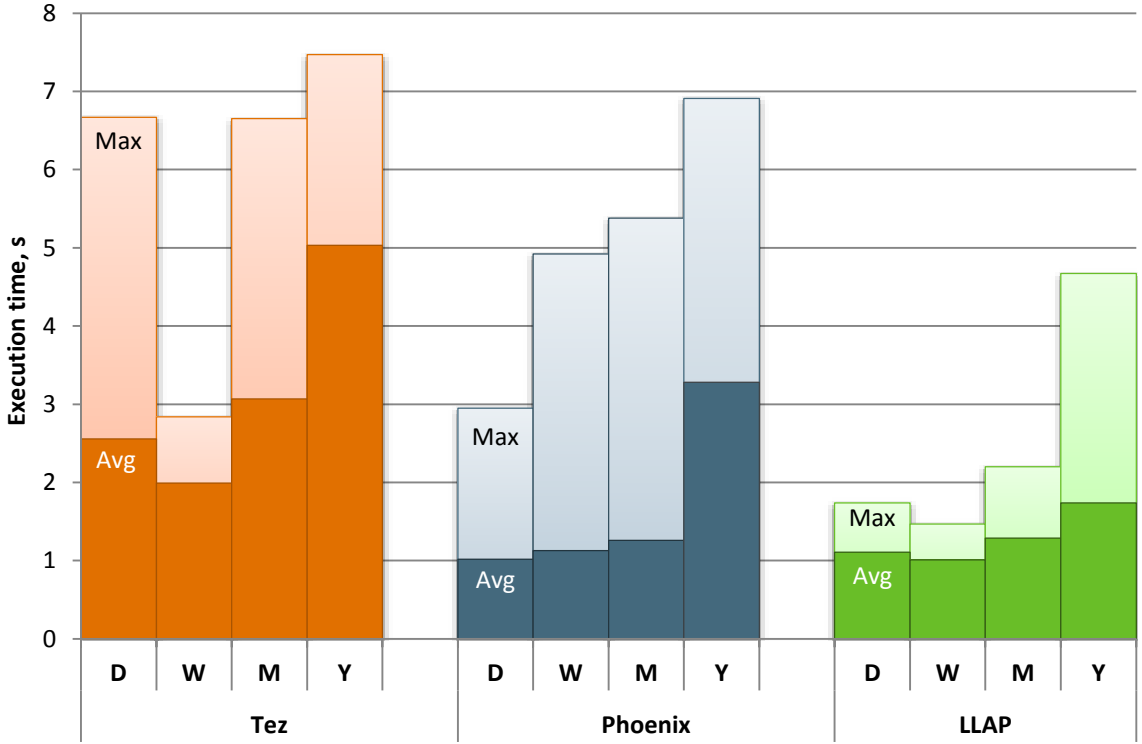


Hortonworks

Evaluation for a real use case

- Aggregate daily statistics for a time interval:



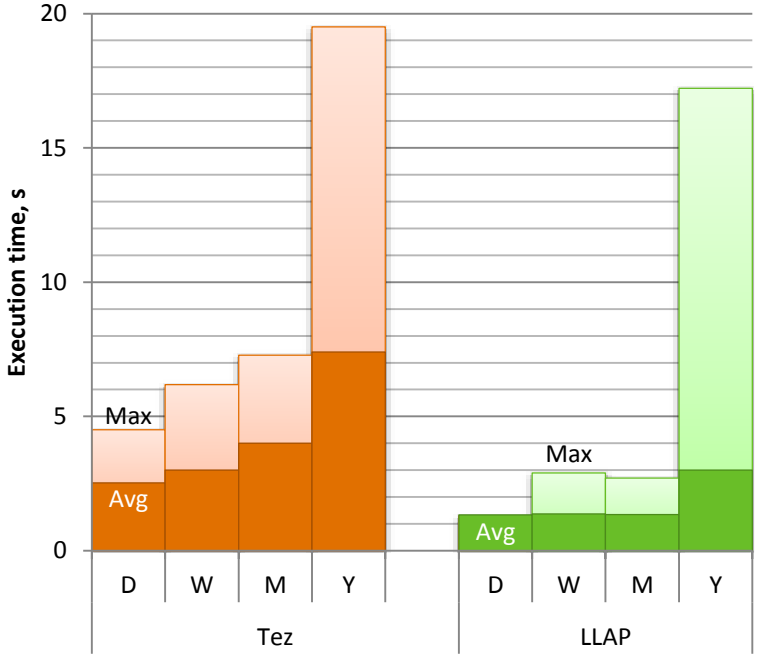



```
SELECT yyyymmdd,
  sum(total_1),
  sum(total_2),
            ...
  from table
  where yyyymmdd >= xxx
            and yyyymmdd < xxx
   and userid = xxx
  group by userid, yyyymmdd;
```


Hortonworks

Evaluation for a real use case

- Display a large report
- Phoenix runtimes not displayed (30-250s on large ranges)



Execution Time, s.

Hortonworks

Hortonworks

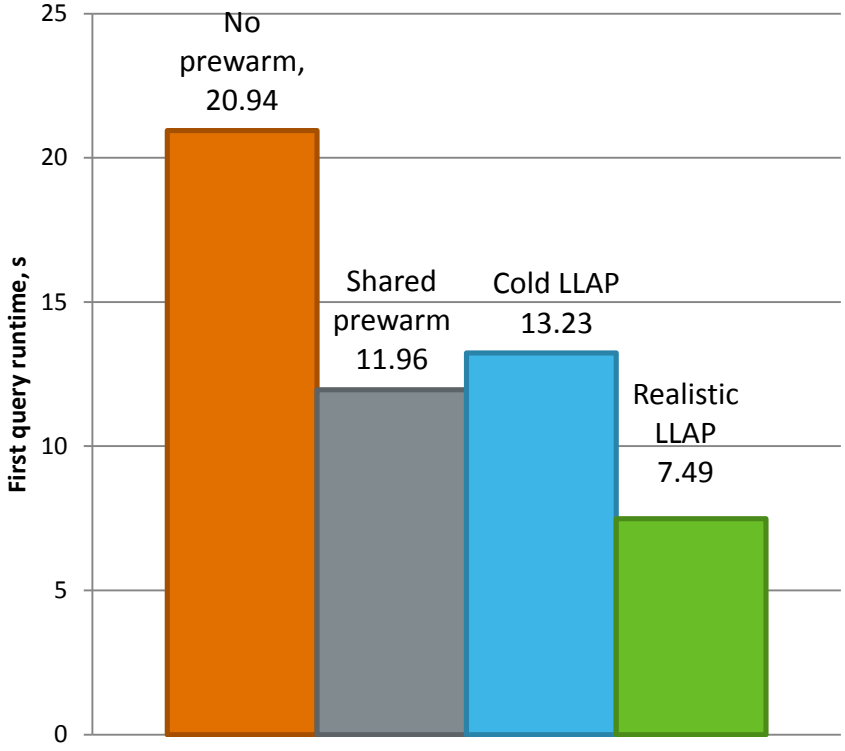
How does LLAP make queries faster?

Persistent daemons (recap)

- Reduced startup time
- Better use of JIT optimizer
- Hash join hash tables, fragment plans are shared
  - Multiple tasks do not all generate the table or deserialize the plans

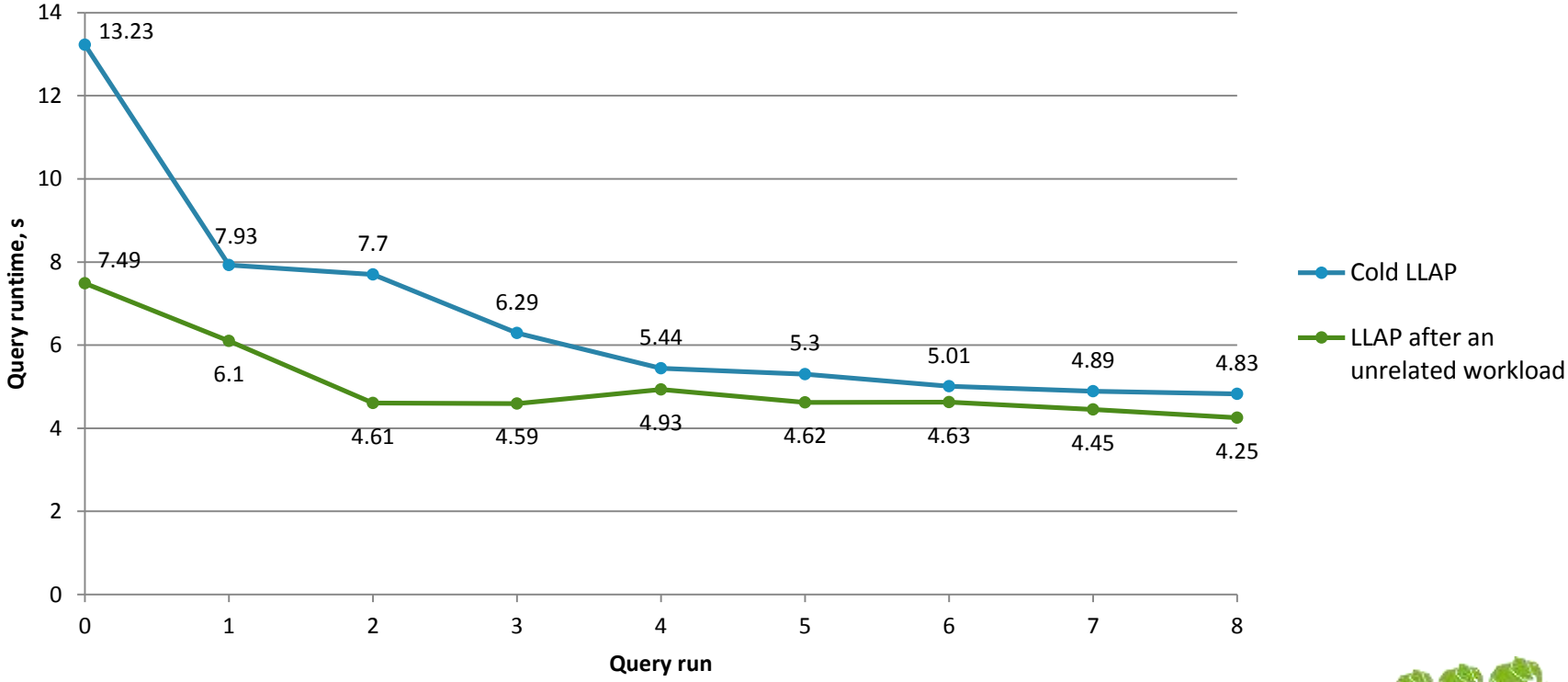
Performance – first query

- 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



Performance – repeated query

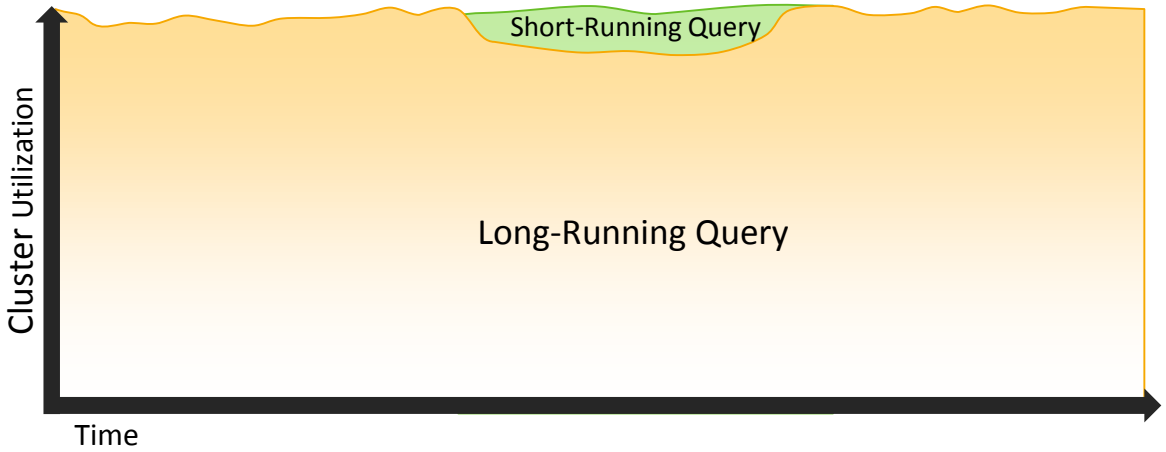
- Cache disabled!

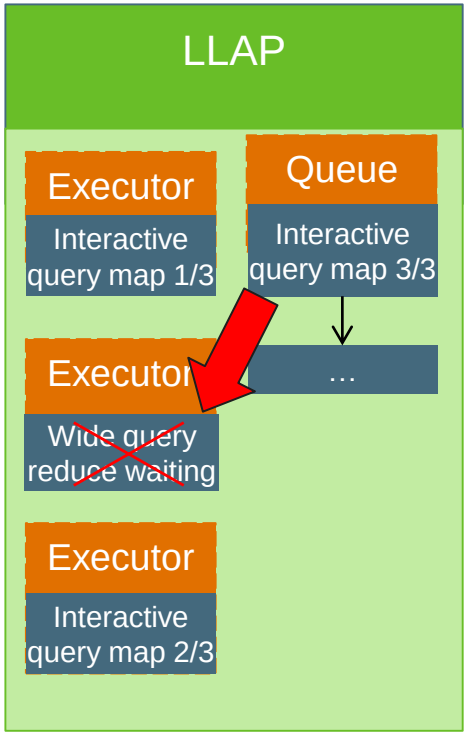


Hortonworks

Parallel queries – priorities, preemption

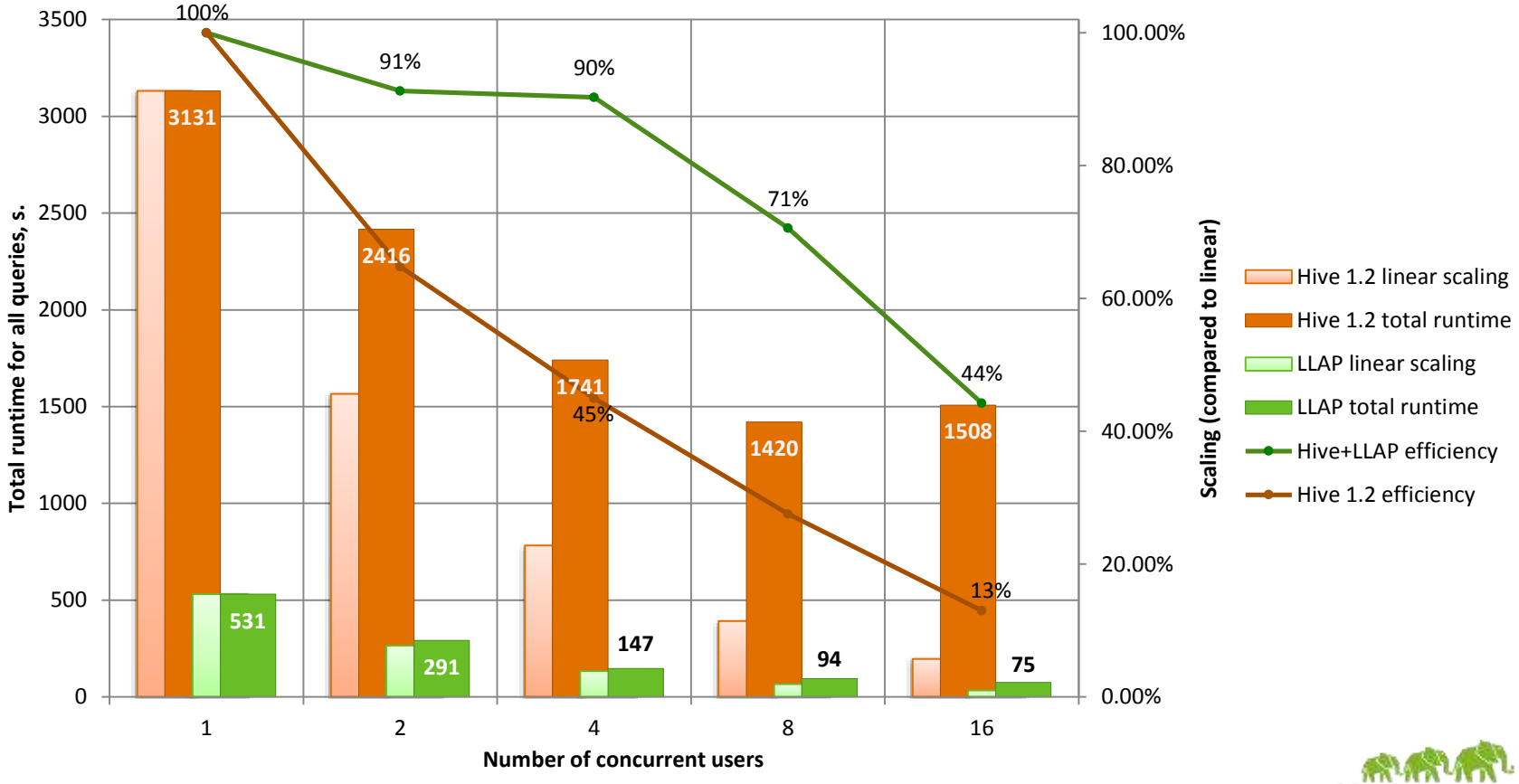
- Lower-priority fragments can be preempted
  - For example, a fragment can start running before its inputs are ready, for better pipelining; such fragments may be preempted
- LLAP work queue examines the DAG parameters to give preference to interactive (BI) queries





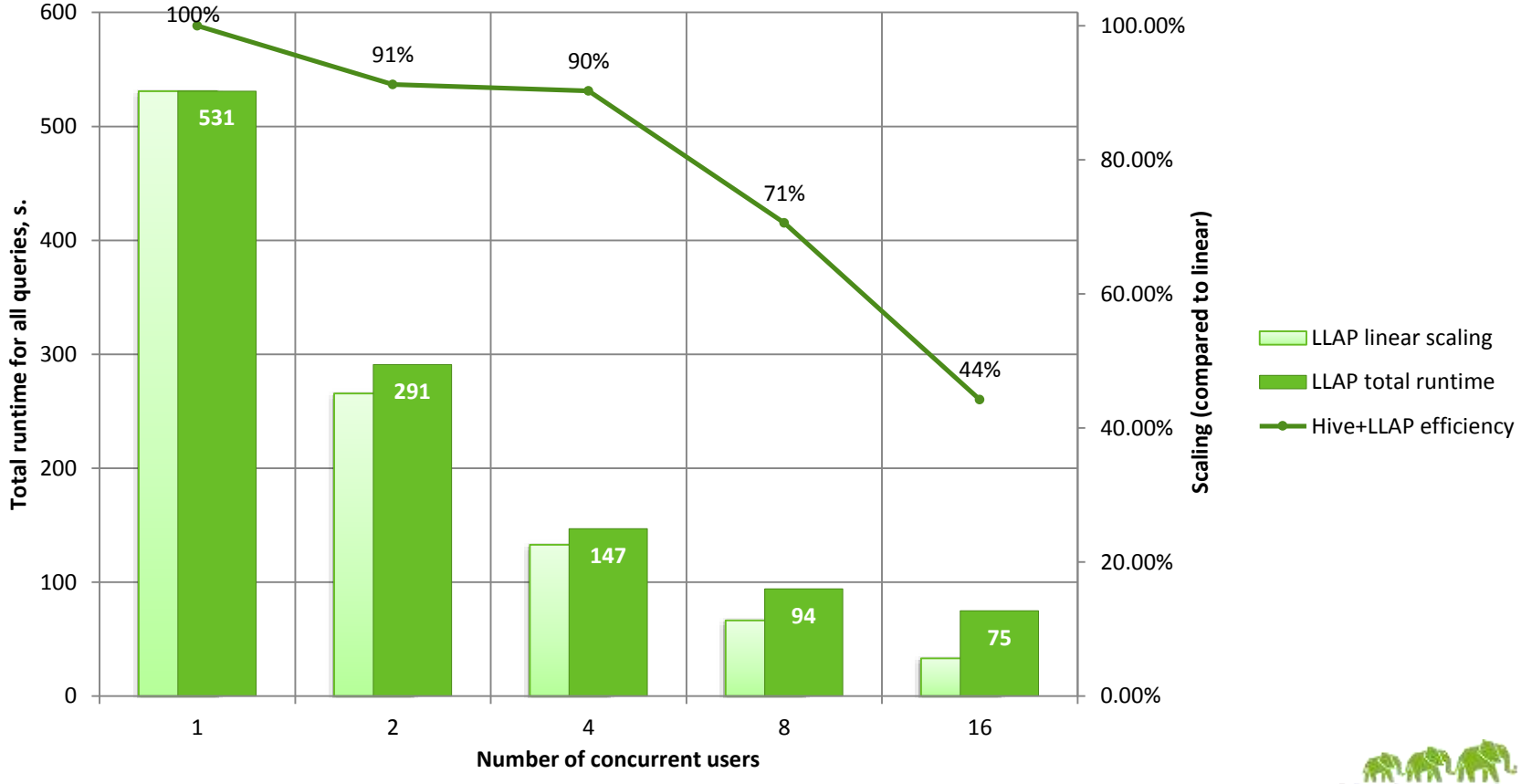
Hortonworks

Parallel query execution – LLAP vs Hive 1.2



Hortonworks

Parallel query execution – 10Tb scale

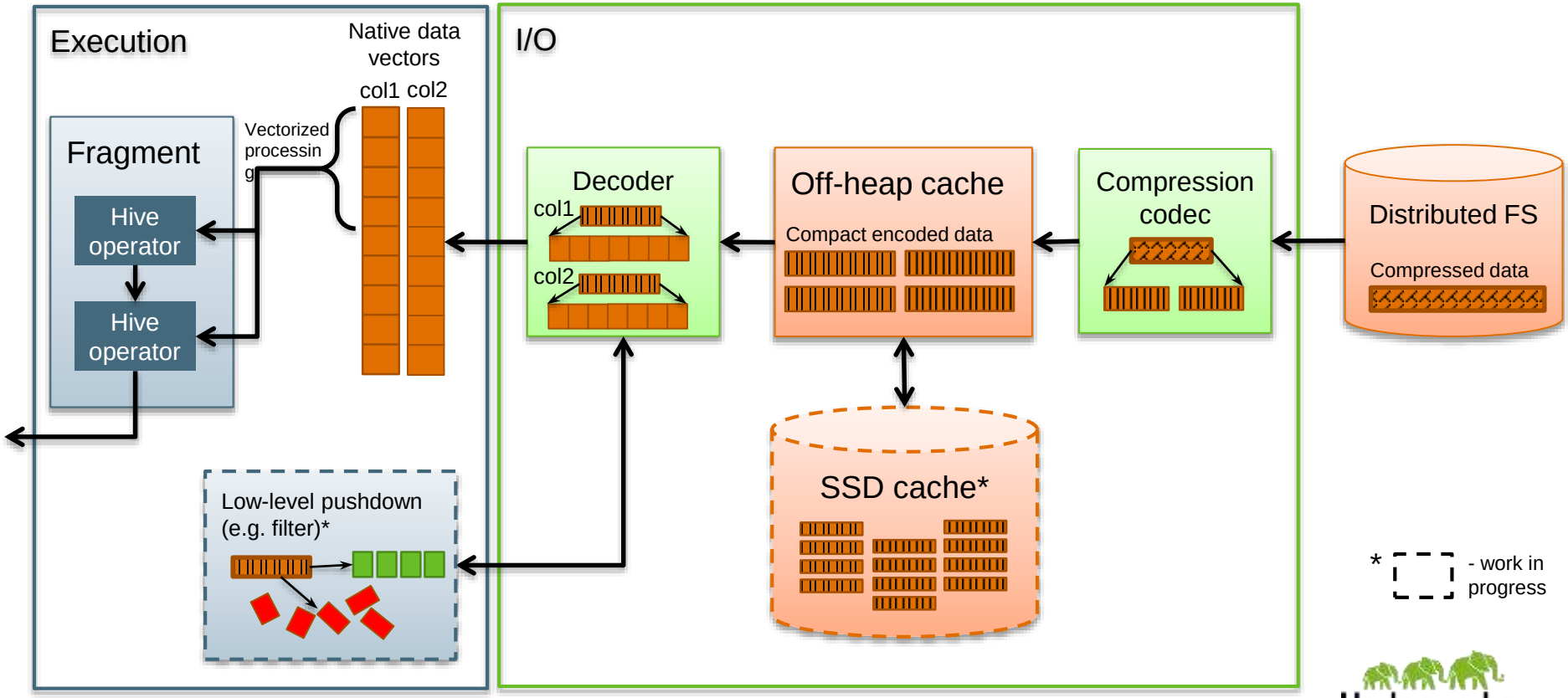


Hortonworks

IO elevator and caching

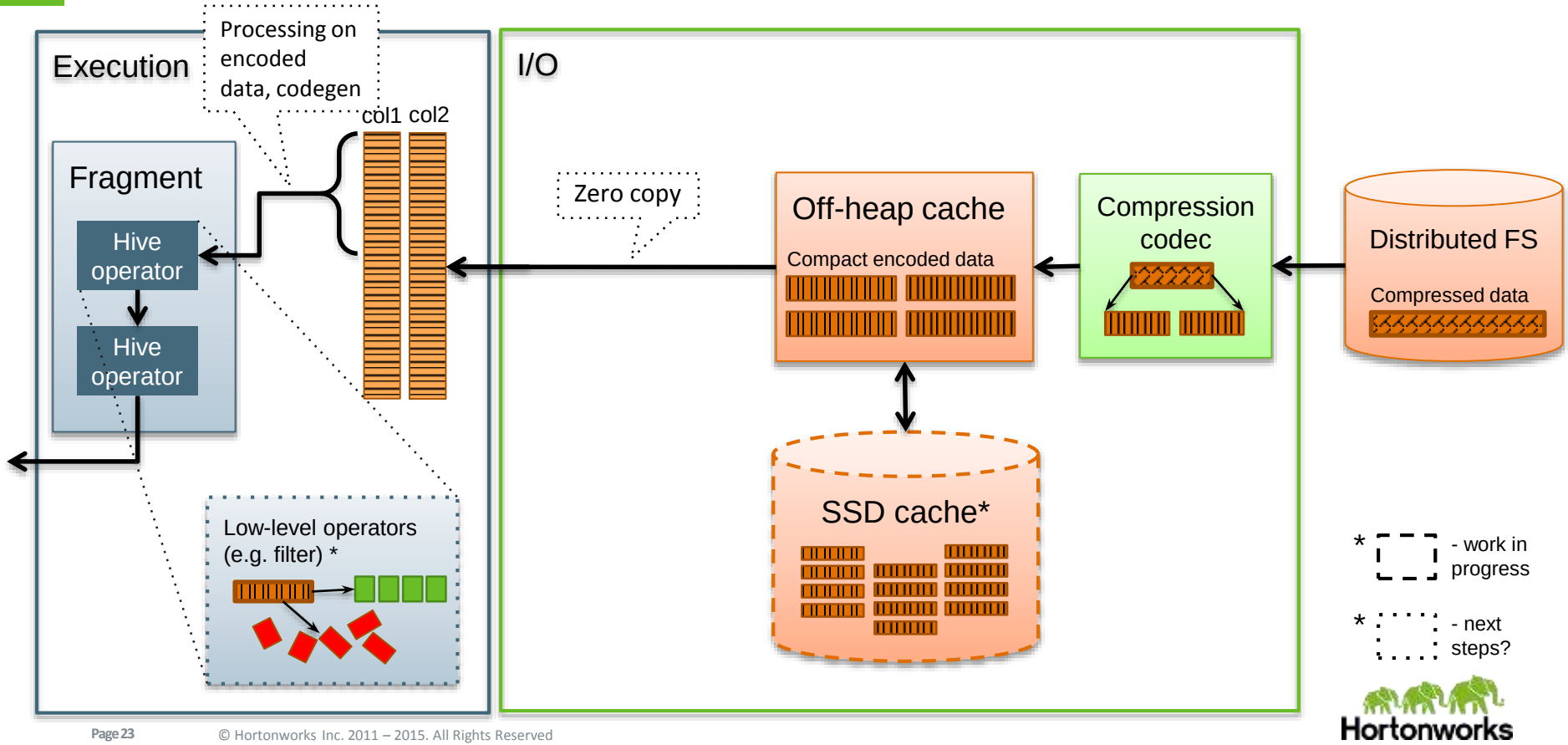
- Reading, decoding and processing are parallel
  - Unlike in regular Hive, where processing can wait for e.g. an S3 read
- Logically-compresses (e.g. ORC-encoded) data is cached off-heap
  - Metadata is cached (currently on-heap) with high priority
  - Replacement policy is pluggable (LRFU is the default)
- Tez schedules work to preferred location to improve locality
- Tradeoff between HDFS reads and operator speed
  - Cache takes memory away from operators (sort buffers, hash join tables, etc.)
  - Depends on how much memory you have, workflow, dataset size, etc.
  - Ex. 4Gb per executor x 16 CPUs – 64Gb for executors, rest for cache

In-memory processing – present and future



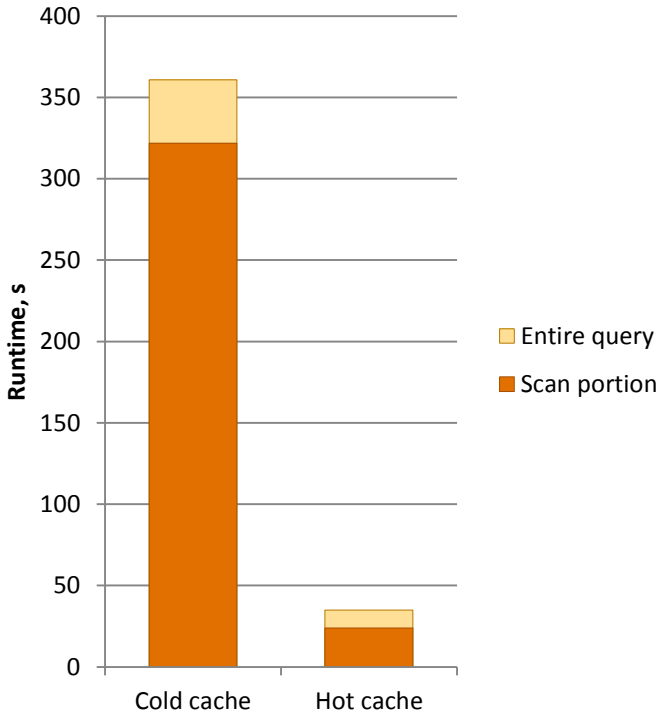
Hortonworks

In-memory processing – future?



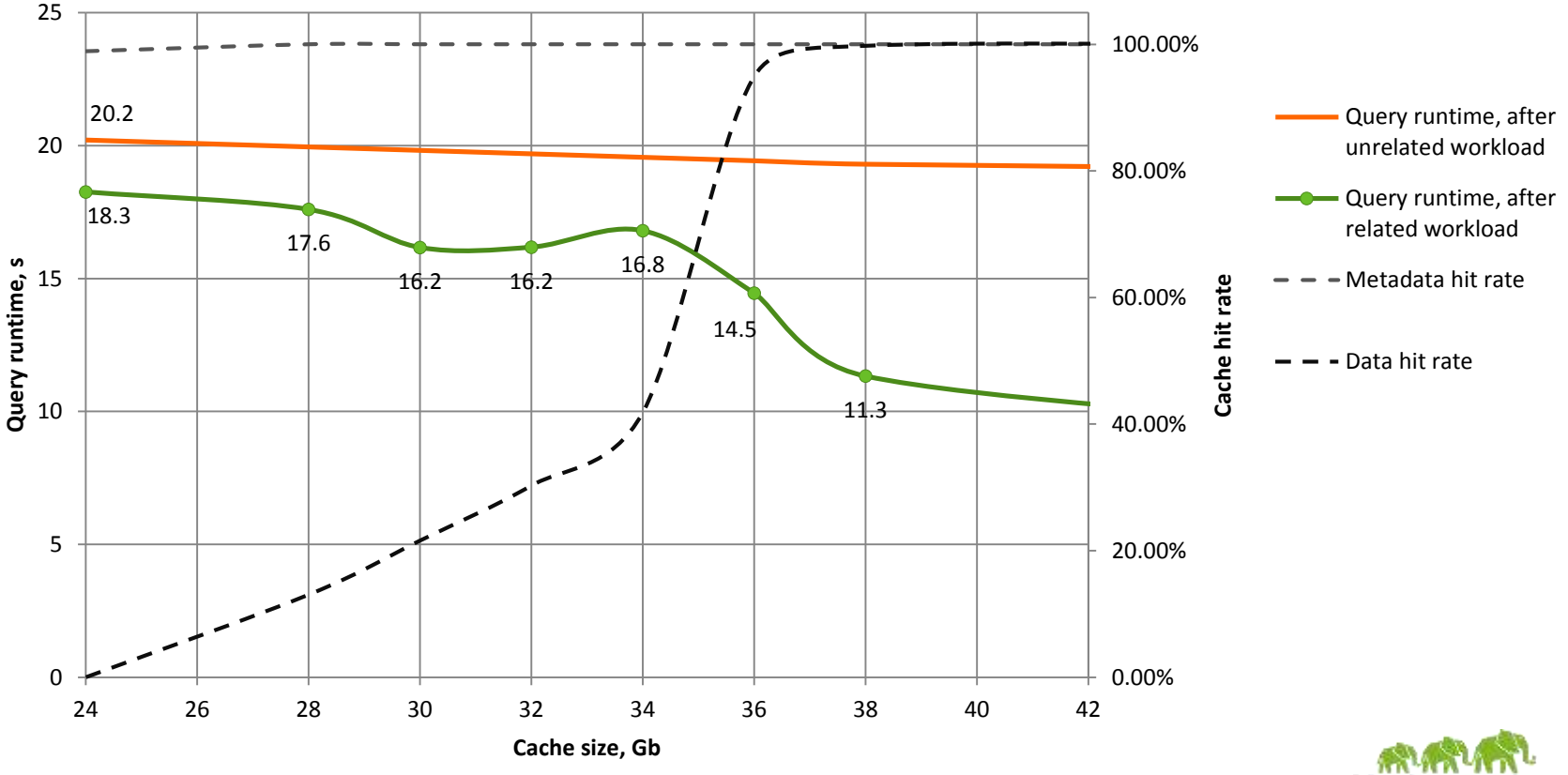
Performance – cache on slow FS

- Industry-standard cloud FS, much slower than on-prem HDFS
- Large scan on 1Tb scale
- Local HD/SDD cache (WIP)
  - Massive improvement on slow storage with little memory cost



Hortonworks

Performance – cache on HDFS, 1Tb scale



Hortonworks

Perf overview – a demo in a gif

Hortonworks

Hortonworks

LLAP as the unified data access layer

Overview

- LLAP can provide a "relational datanode" view of the data
- Security via endpoint ACLs, or fragment signing (for external engines)
  - Granular, e.g. column-level, security is possible
- Anyone (with access) can push the (approved) code in, from complex query fragments to simple data reads
  - SparkSQL/LLAP integration is based on SparkSQL data source APIs
  - DataFrame can be created with LlapInputFormat
- Hive execution engines and other DAG executors can use LLAP directly
  - like Tez does now

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

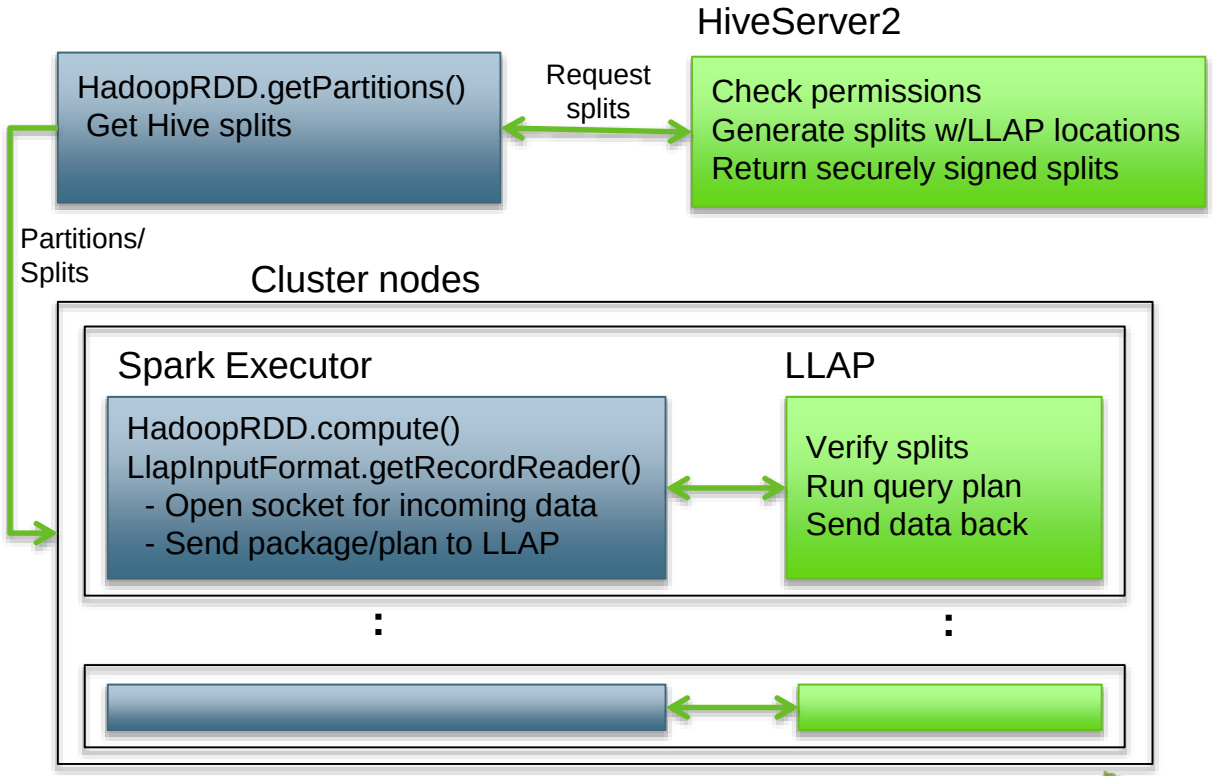
Hortonworks

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



Example - Tez/LLAP integration

- Tez DAG coordination remains mostly unchanged
- Tez plugins (TEZ-2003)
  - Scheduling – allows scheduling parts of a DAG as fragments on LLAP
  - Task communication – custom execution specifications, protocols – allows talking to LLAP, manages security tokens, etc.
- Hive operators used for processing
  - Few or no changes necessary to support any Hive query complexity
  - All new Hive features automatically supported

Hortonworks

Deploying and managing LLAP

Availability

- First version shipped in Apache Hive 2.0
  - Hive 2.0.1 contains important bugfixes to make it production ready
  - Hive 2.1 would have new features and further perf improvements
- A solid platform for the basic scenario – run LLAP-only queries on a secure cluster
  - or a fraction thereof
- Work in progress to support additional features (ACID, UDFs, hybrid deployments, etc.)
- Unified data access layer is the next step

Hortonworks

General overview

- LLAP daemons run on YARN; deployed via Apache Slider
  - Easy to bring up, tear down, flex clusters via slider commands
- Hive-on-Tez should be set up with a compatible version (0.8.2+)
- Zookeeper for service discovery and coordination
- On HS2, using doAs is not recommended; use SQL auth or Ranger
  - However, you can keep using storage-based auth for other queries, as usual
- Java 8 strongly recommended
  - April 2015 called and they want their Java 7 End-of-Lived

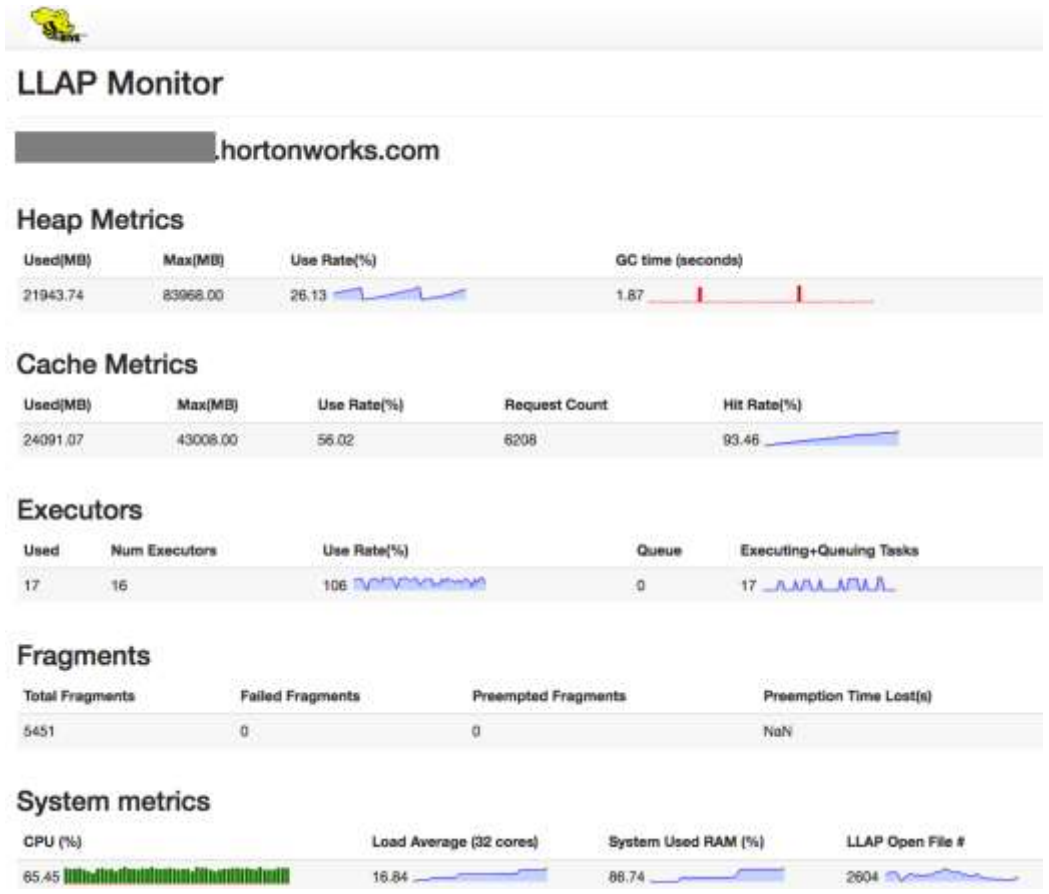
Starting the cluster

- Apache Ambari integration – WIP (will do all of this for you)
- Hive service (`hive --service llap`)
  - Generates a slider package, and a script to deploy it and start the cluster
  - Run as the correct user – slider paths are user-specific!
  - HADOOP_HOME, JAVA_HOME should be set
  - kinit on secure cluster
  - Specify a name, e.g. `--name llap0`; number of instances (e.g. one per machine); memory, cache size, number of executors (e.g. one per CPU); see `--help`
  - G1 GC recommended (e.g. `--args " -XX:+UseG1GC -XX:+ResizeTLAB -XX:-ResizePLAB"`)

Hortonworks

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



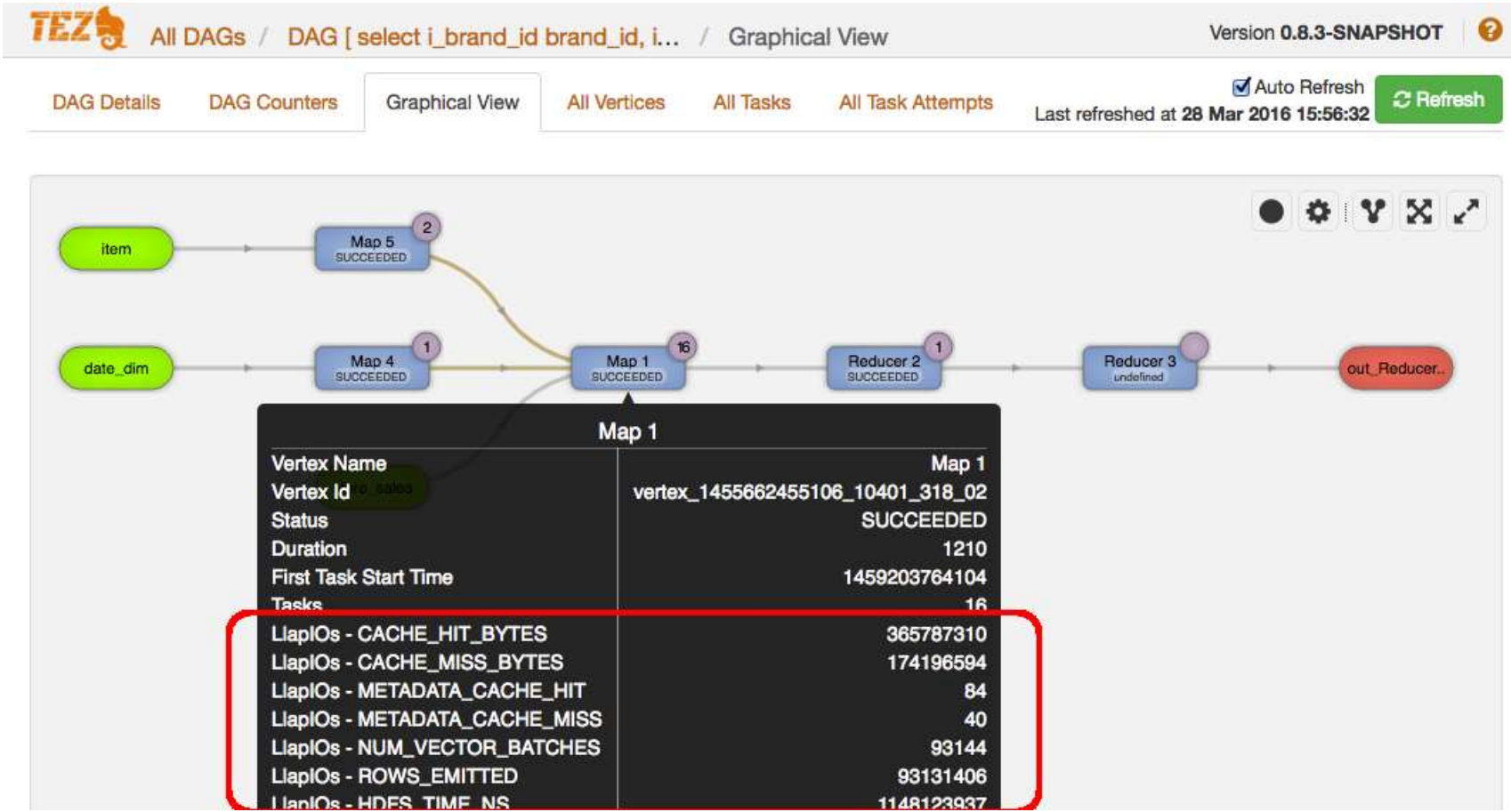
Hortonworks

Running queries

- Once the cluster is started, queries can be run from HS2 and CLI
  - `--hiveconf hive.execution.mode=llap --hiveconf hive.llap.execution.mode=all`
    `--hiveconf hive.llap.io.enabled=true`
    `--hiveconf hive.llap.daemon.service.hosts=@<cluster name>`
- Set the usual perf settings (recommended, if not already set)
  - Enable vectorization, PPD, map join; use ORC
  - LLAP-specific: `set hive.tez.input.generate.consistent.splits=true;`
  - For interactive queries, disable CBO
  - Parallel compilation on HS2 – otherwise your parallel queries might bottleneck there!
- Good to go!

Hortonworks

Watching queries – Tez UI integration



Security

- Again, Ambari will do this for you soon
- SQL or Ranger auth recommended, with hive user running queries
- LLAP uses keytabs and ACLs, similar to datanode, to secure endpoints
  - Keytabs need to be set up; certs may need to be set up for Web UI
- HS2 (or client) obtains a token to access a particular LLAP cluster, and gives it to Tez AM
- LLAP-s share tokens using secure ZK, similar to HA token sharing
- Signing fragments on HS2 and granular security checks are work in progress

Hortonworks

Summary and future work

Future work

- **Data view for external services**
  - Column-level security
- **Better integration and deployment**
  - Ambari, monitoring UI, fine-grained log aggregation
- Hybrid deployment
  - Resizing LLAP to accommodate containers, YARN resource delegation, etc.
- Feature work
  - ACID, text cache, better cache policies, UDFs, SSD cache, etc.
- More performance work

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 thru a standard InputFormat API
- Available in Hive 2, with better ecosystem integration in the works

Questions?

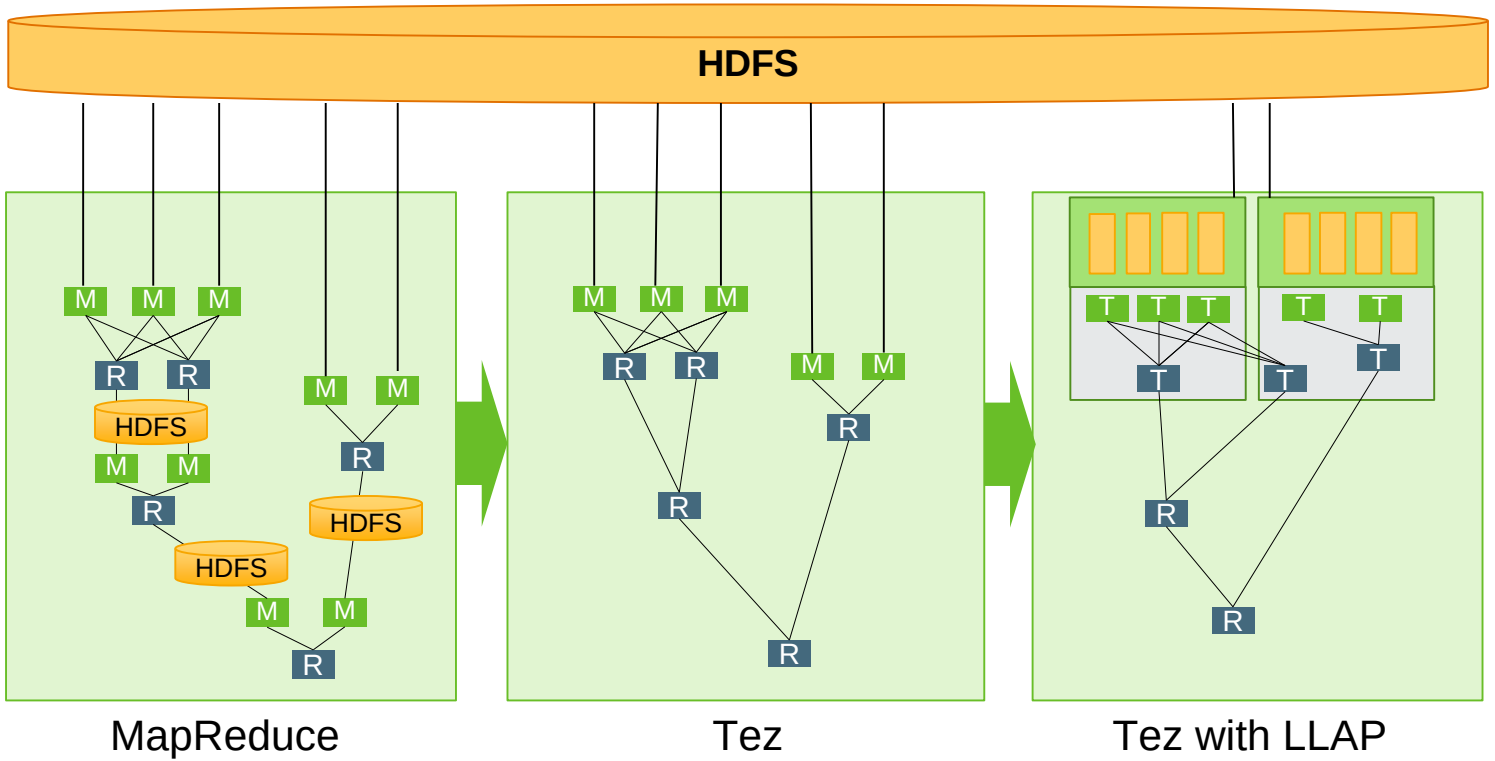
Interested? Stop by the Hortonworks booth to learn more

Hortonworks

Hortonworks

Backup slides

Example execution: MR vs Tez vs Tez+LLAP



Hortonworks

IO Elevator

- Reading, decoding and processing are parallel
  - Unlike in regular Hive
- HDFS reads are expensive (more so on S3, Azure)
- Data decompression and decoding is expensive