Hortonworks

Putting the Sting in Hive

Alan F. Gates
@alanfgates

Stinger Overview

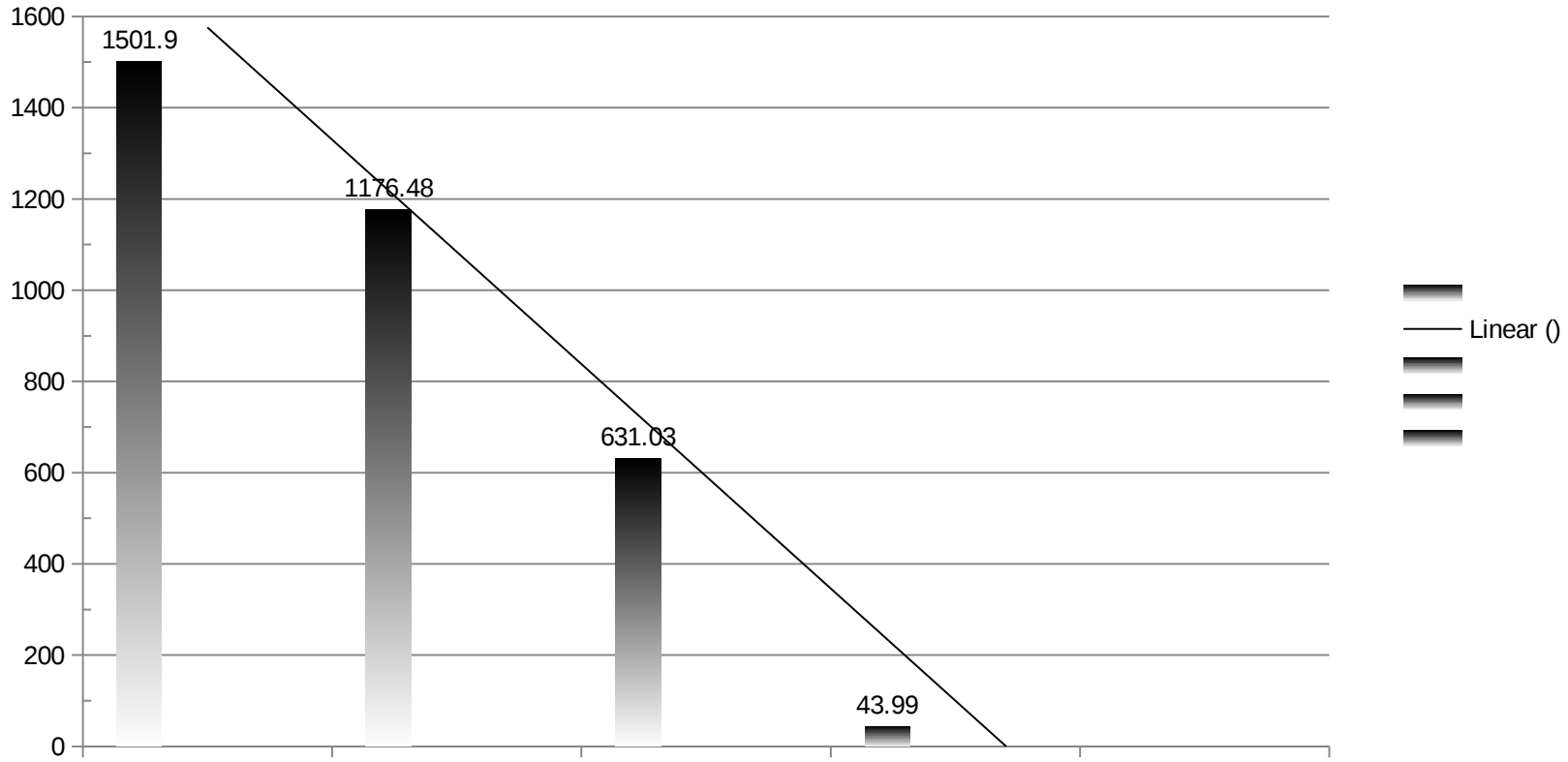
- An initiative, not a project or product
- Includes changes to Hive and a new project Tez
- Two main goals
  - Improve Hive performance 100x over Hive 0.10
  - Extend Hive SQL to include features needed for analytics
- Hive will support:
  - BI tools connecting to Hadoop
  - Analysts performing ad-hoc, interactive queries
  - Still excellent at the large batch jobs it is used for today

Hive Performance Gains in 0.11

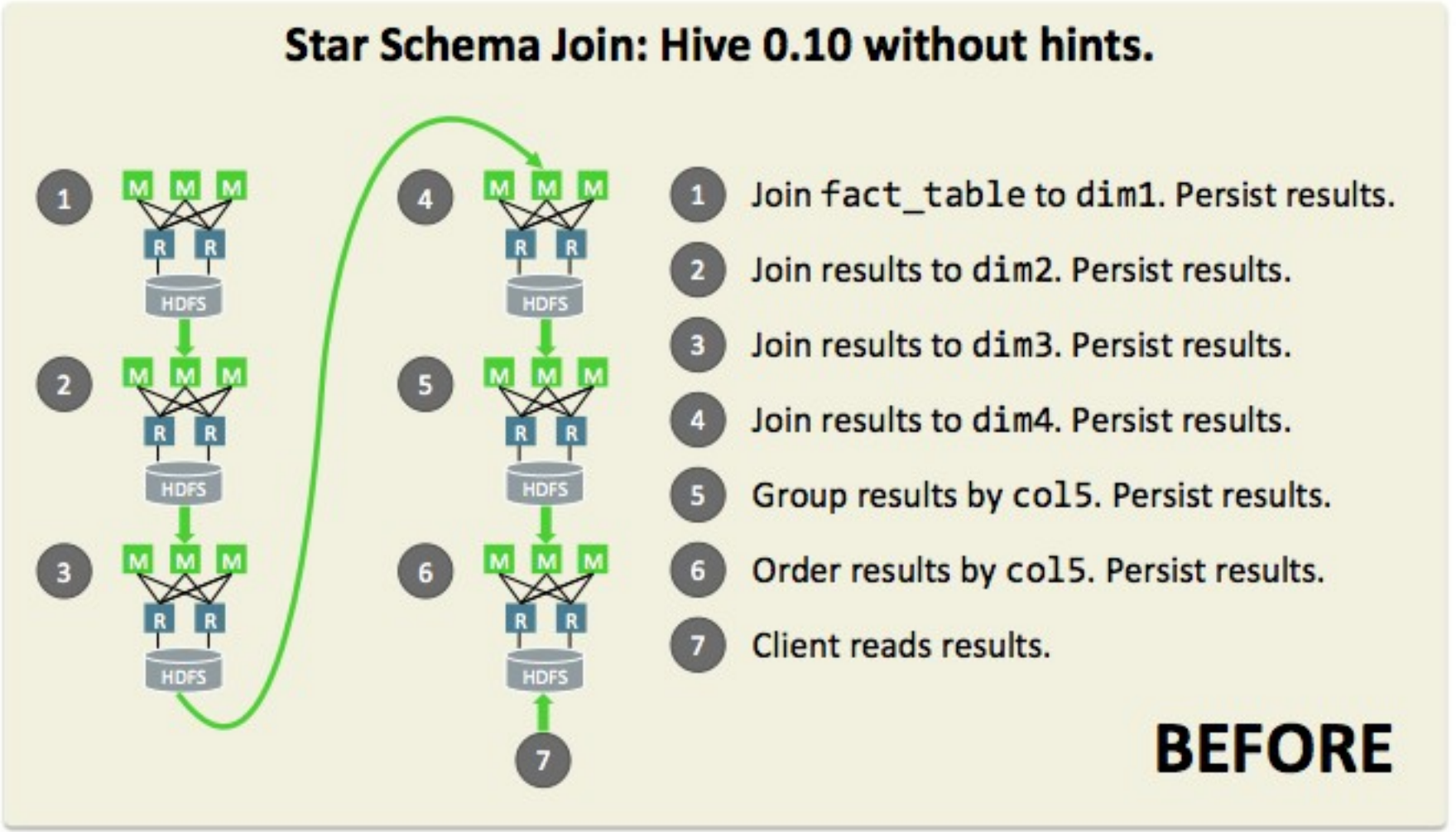
- Enable star joins by improving Hive's map join (aka broadcast join)
  - Where possible do in single map only task
  - When not possible push larger tables to separate tasks
- Collapse adjacent jobs where possible
  - Hive has lots of M->MR type plans, collapse these to MR
  - Collapse adjacent jobs on sufficiently similar keys when feasible
  - join followed by group
  - join followed by order
  - group followed by order
- Improvements in Sort Merge Bucket (SMB) joins

Improvements in Map Joins

- **TPC-DS Query 27, Scale=200, 10 EC2 nodes (40 disks)**

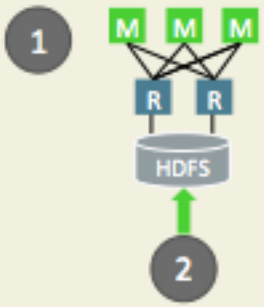


Before



After

Star Schema Join: Hive 0.11 without hints.

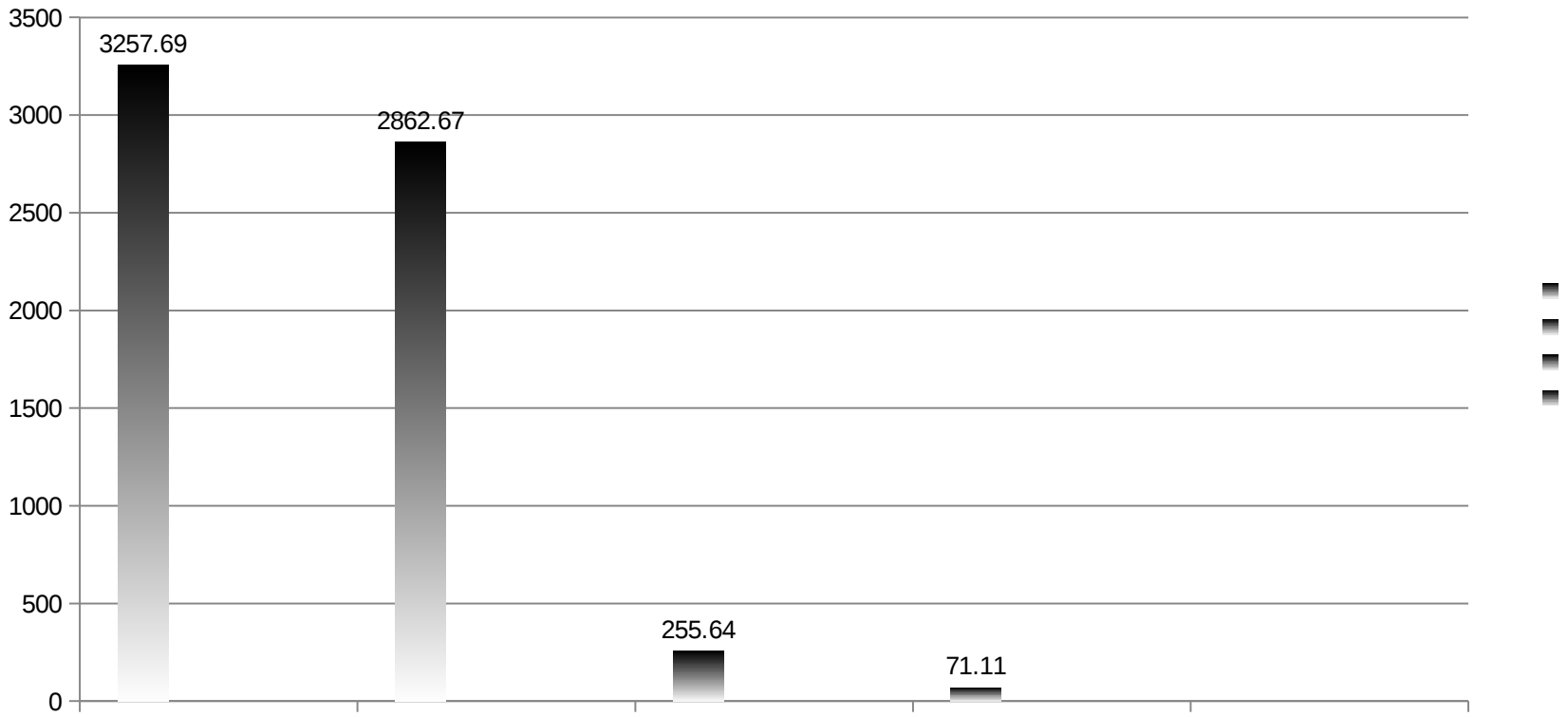


1. Load `dim1` – `dim4` into memory on all nodes. (HIVE-3784)
   Perform map-side joins. (HIVE-3952)
   Collapse ORDER BY and GROUP BY into single reducer. (HIVE-2340)
   Persist results.
2. Client reads results.

AFTER

Improvements in SMB Joins

- **TPC-DS Query 82, Scale=200, 10 EC2 nodes (40 disks)**



Extending Hive SQL in 0.11 - OVER

- OVER clause
  - PARTITION BY, ORDER BY, ROWS BETWEEN/FOLLOWING/PRECEDING
  - Works with existing aggregate functions
  - New analytic and window functions added
  - ROW_NUMBER, RANK, DENSE_RANK, LEAD, LAG, LEAD, FIRST_VALUE, LAST_VALUE, NTILE, CUME_DIST, PERCENT_RANK

```
SELECT salesperson, AVG(salesprice) OVER
    (PARTITION BY region ORDER BY date
     ROWS BETWEEN 10 PRECEEDING AND 10 FOLLOWING)
FROM sales;
```

Extending Hive SQL Continued

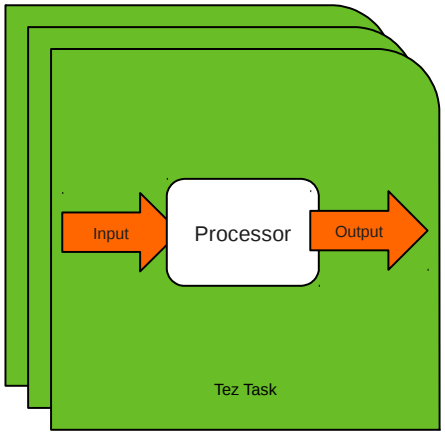
- CUBE BY, ROLLUP in Hive 0.10
- Subqueries in WHERE
  - Non-correlated first
  - [NOT] IN first, then extend to (in)equalities and EXISTS
- Datatype conformance – Hive has Java type model, add support for SQL types:
  - date, timestamp beyond unix long time
  - char() and varchar()
  - add precision and scale to decimal and float
  - aliases for standard SQL types (BLOB = binary, CLOB = string, integer = int, real/number = decimal)
- Security
  - Add security checks to views
  - Secure operations such as GRANT, REVOKE

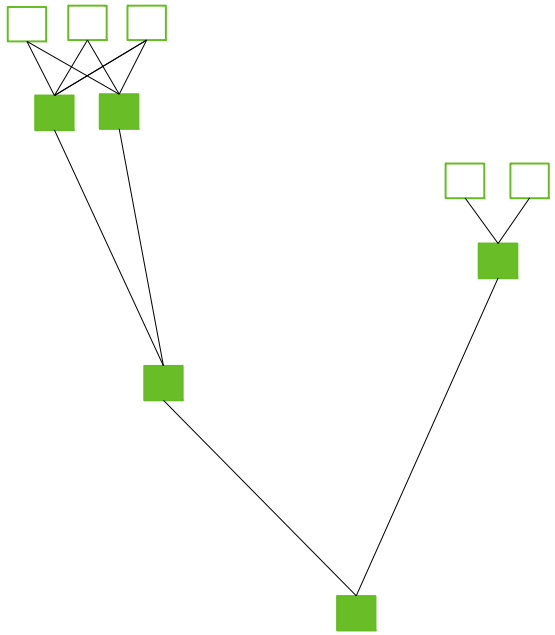
Tez

- Low level data-processing execution engine
- Use it for the base of MapReduce, Hive, and Pig
- Enables pipelining of jobs
- Removes task and job launch times
- Hive and Pig jobs no longer need to move to the end of the queue between steps in the pipeline
- Does not write intermediate output to HDFS
  - Much lighter disk and network usage
- Built on YARN

Tez- Core Idea

Task with pluggable *Input*, *Processor* & *Output*

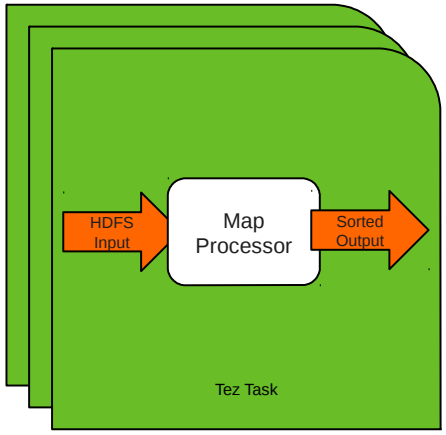




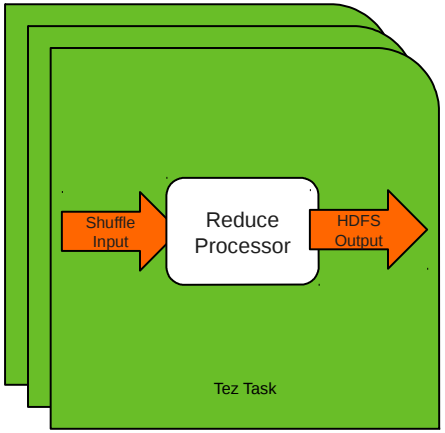
YARN ApplicationMaster to run DAG of Tez Tasks

Tez – Blocks for building tasks

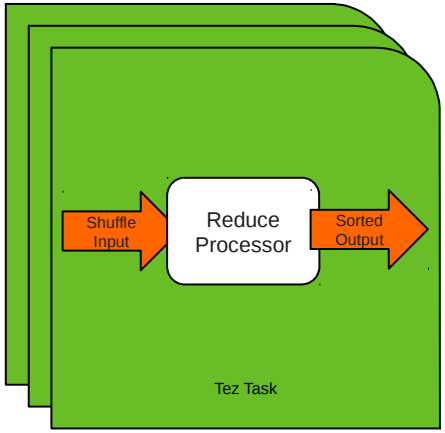
MapReduce 'Map'



MapReduce 'Reduce'

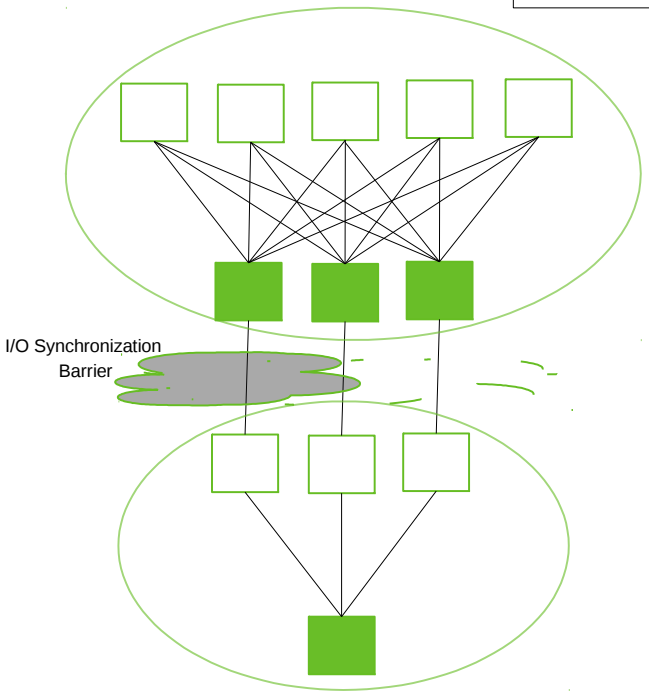


Intermediate 'Reduce' for Map-Reduce-Reduce

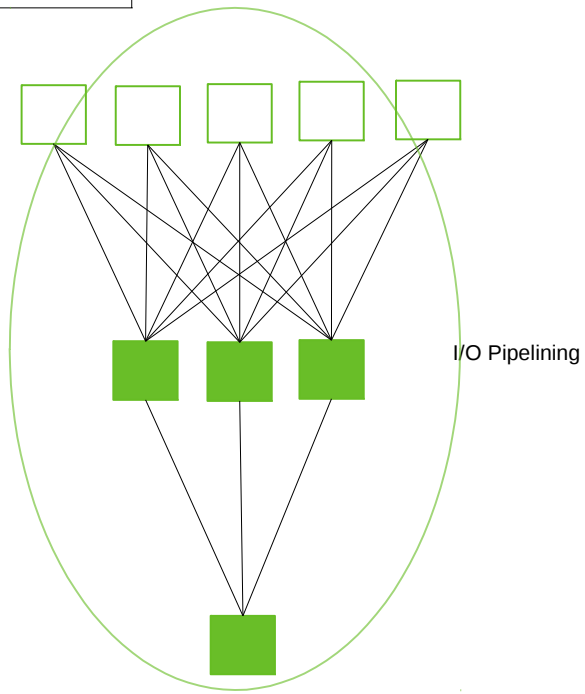


Hive/MR versus Hive/Tez

```
SELECT a.state, COUNT(*)
FROM a JOIN b ON (a.id = b.id)
GROUP BY a.state
```



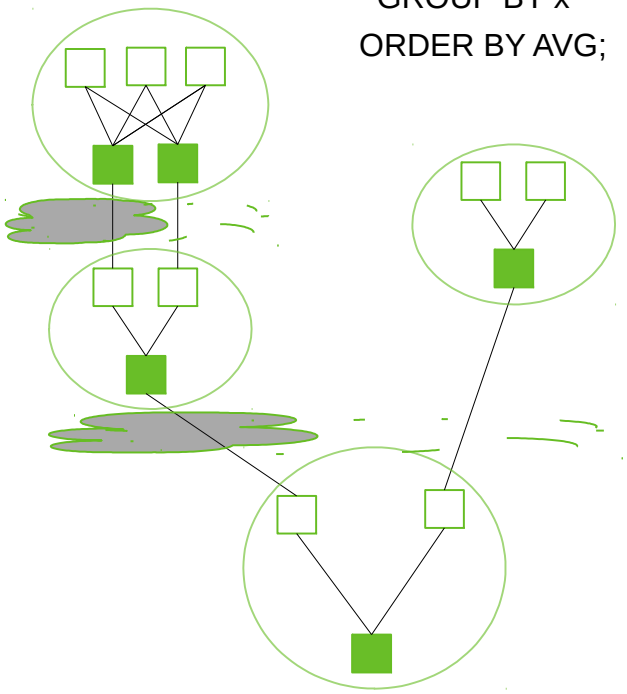
Hive - MR



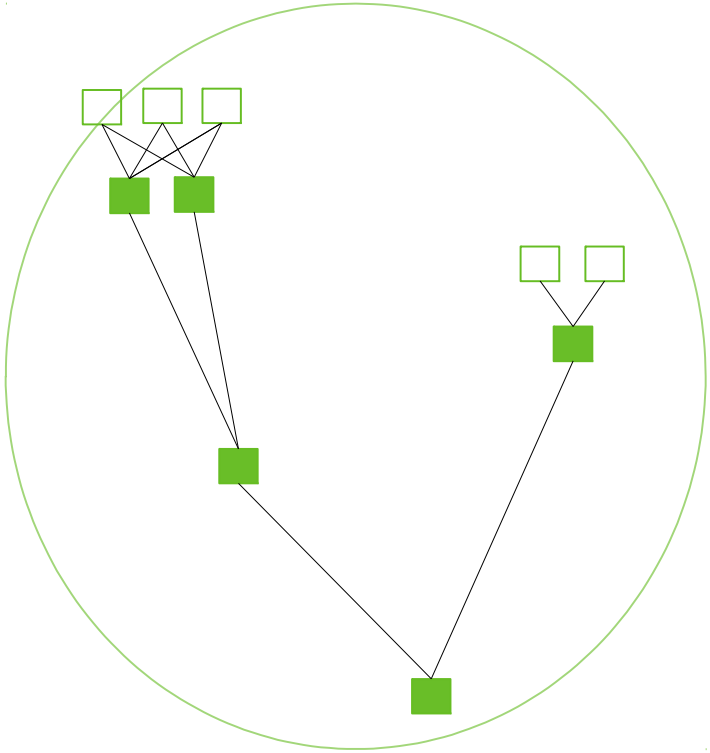
Hive - Tez

Hive/MR versus Hive/Tez

```
SELECT a.x, AVERAGE(b.y) AS avg
  FROM a JOIN b ON (a.id = b.id)
  GROUP BY a
UNION SELECT x, AVERAGE(y) AS AVG
  FROM c
  GROUP BY x
ORDER BY AVG;
```



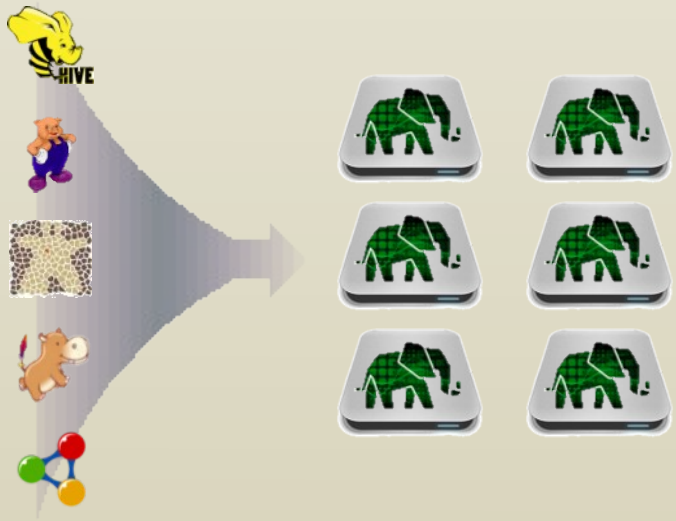
Hive - MR



Hive - Tez

FastQuery: Beyond Batch with YARN

- Hive Query startup expensive
  - Job launch & task-launch latencies are fatal for short queries
- Solution -Tez Service
  - Removes job and task launch overhead
  - Hive submits query-plan to Tez Service
- ***Native Hadoop service, not ad-hoc***



Always-On Tez Service

Low latency processing for all Hadoop data processing

Hive Performance Longer Term

- ORC file – new columnar format optimized for performance, see Owen O'Malley's talk
- Keep working on the optimizer
  - Y Smart work from Ohio State University
  - Start using statistics to make intelligent decisions about how many mappers and reducers to spawn
  - Start using statistics to choose between competing plan options
- Column oriented operators
  - Motivated by MonetDB paper
  - Rewrite operators to work on arrays of Java scalars
  - Operates on blocks of 1K or more records
  - Size the block to fit in L1 cache, avoid cache misses
  - Generate code for operators to remove function calls and branches from inner loop, maximize use modern processers' deep pipelines
  - Want to write this in a way it can be shared by Pig, Cascading, MR programmers

Questions