Hortonworks

Apache Tez

Hitesh Shah
hitesh@hortonworks.com
@hitesh1892

Agenda

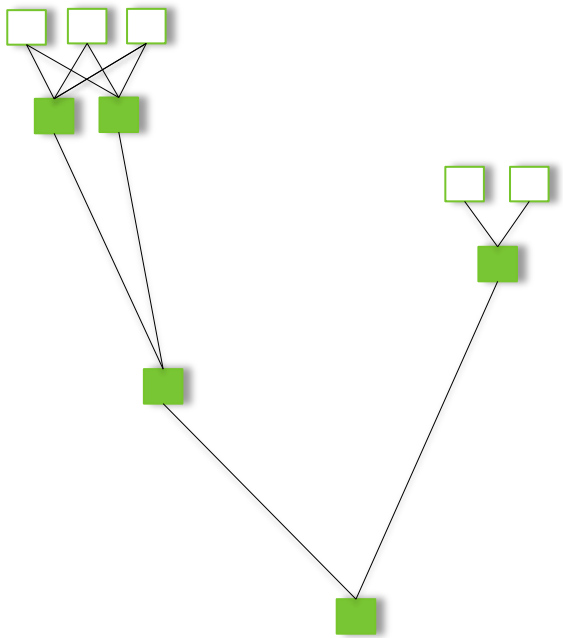
- **Introduction**
- **Overview**
- **Tez Internals**
- **Theory to Practice**

Apache Tez at The ASF

- **Entered Apache Incubator in Feb 2013**
- **Committers from various companies:**
  - Hortonworks
  - Microsoft
  - Yahoo
  - LinkedIn
  - NASA JPL
  - Cloudera, Twitter, …
- **Graduated to a TLP in July 2014.**

Tez – Introduction

- **Distributed execution framework targeted towards data-processing applications.**
- **Based on expressing a computation as a dataflow graph.**
- **Highly customizable to meet a broad spectrum of use cases.**
- **Built on top of YARN – the resource management framework for Hadoop.**



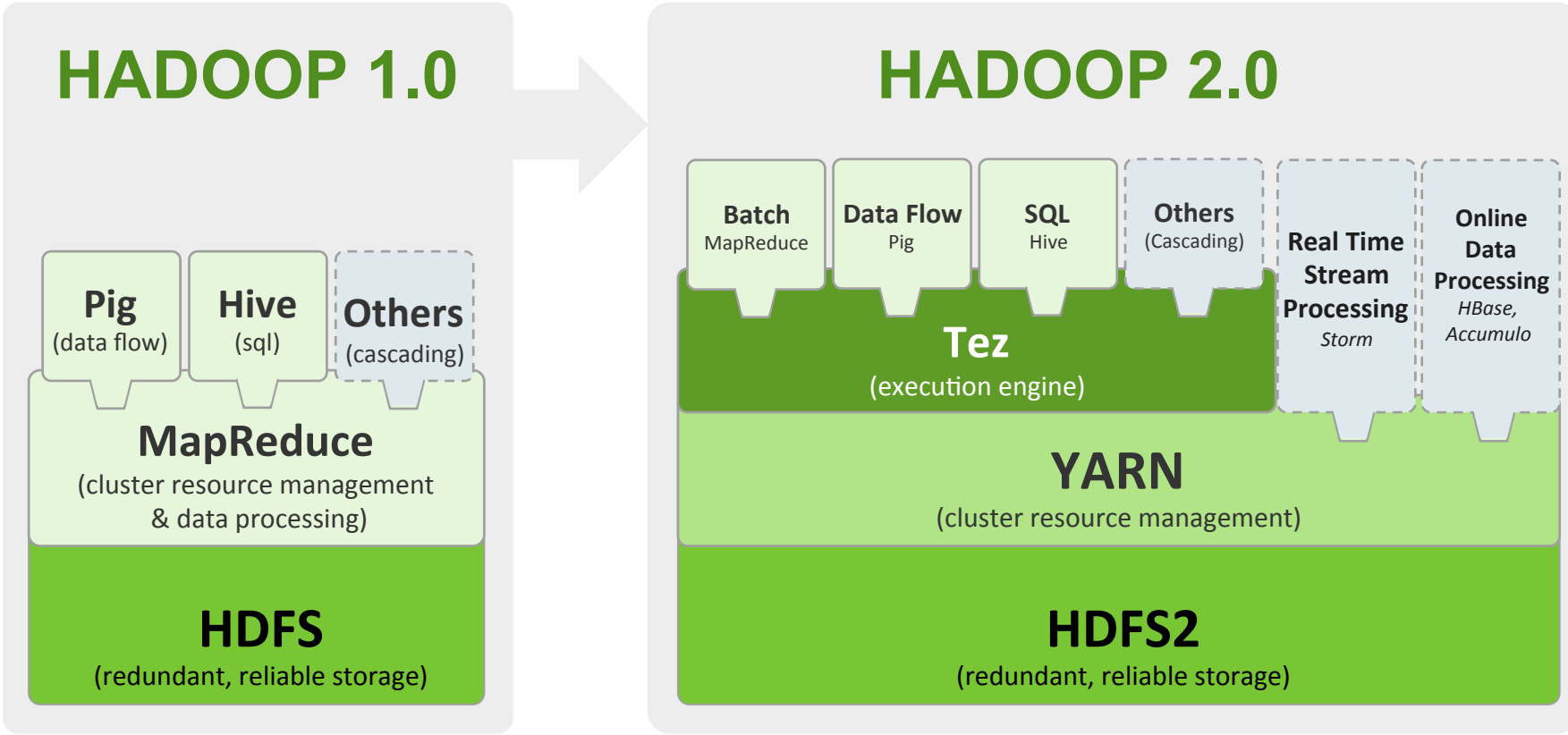
Hadoop 1 -> Hadoop 2

Monolithic

- **Resource Management**
- **Execution Engine**
- **User API**

Layered

- **Resource Management – YARN**
- **Execution Engine – Tez**
- **User API – Hive, Pig, Cascading, Your App!**



Tez – Empowering Applications

- **Tez solves hard problems of running on a distributed Hadoop environment**
- **Apps can focus on solving their domain specific problems**
- **This design is important to be a platform for a variety of applications**

App	• Custom application logic • Custom data format • Custom data transfer technology
Tez	• Distributed parallel execution • Negotiating resources from the Hadoop framework • Fault tolerance and recovery • Horizontal scalability • Resource elasticity • Shared library of ready-to-use components • Built-in performance optimizations • Security

Tez – Design considerations

Look to the Future with an eye on the Past

Don’t solve problems that have already been solved. Or else you will have to solve them again!

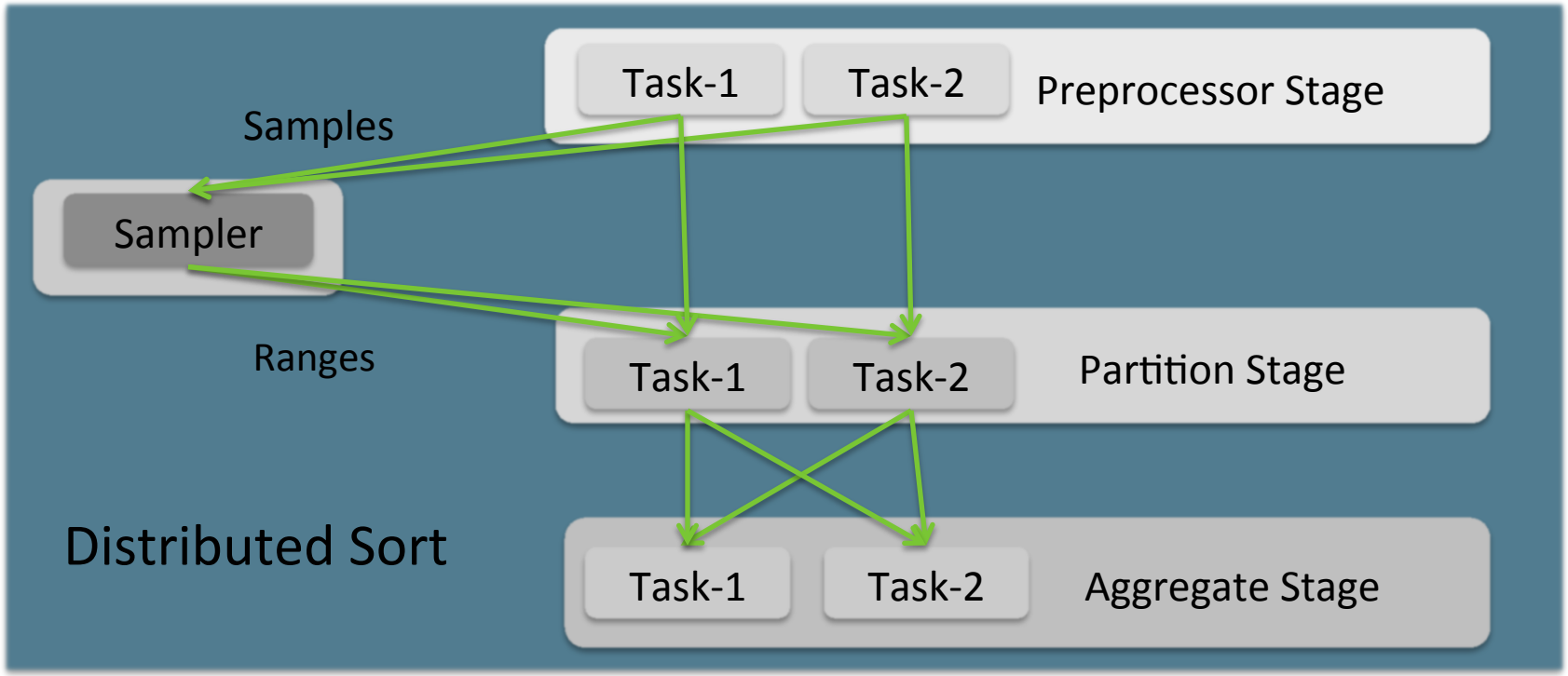
- Leverage discrete task based compute model for elasticity, scalability and fault tolerance
- Leverage several man years of work in Hadoop Map-Reduce data shuffling operations
- Leverage proven resource sharing and multi-tenancy model for Hadoop and YARN
- Leverage built-in security mechanisms in Hadoop for privacy and isolation

Tez – Problems that it addresses

- **Expressing the computation**
  - Direct and elegant representation of the data processing flow
  - Interfacing with application code and new technologies
- **Performance**
  - Late Binding : Make decisions as late as possible using real data from at runtime
  - Leverage the resources of the cluster efficiently
  - Just work out of the box!
  - Customizable engine to let applications tailor the job to meet their specific requirements
- **Operation simplicity**
  - Painless to operate, experiment and upgrade

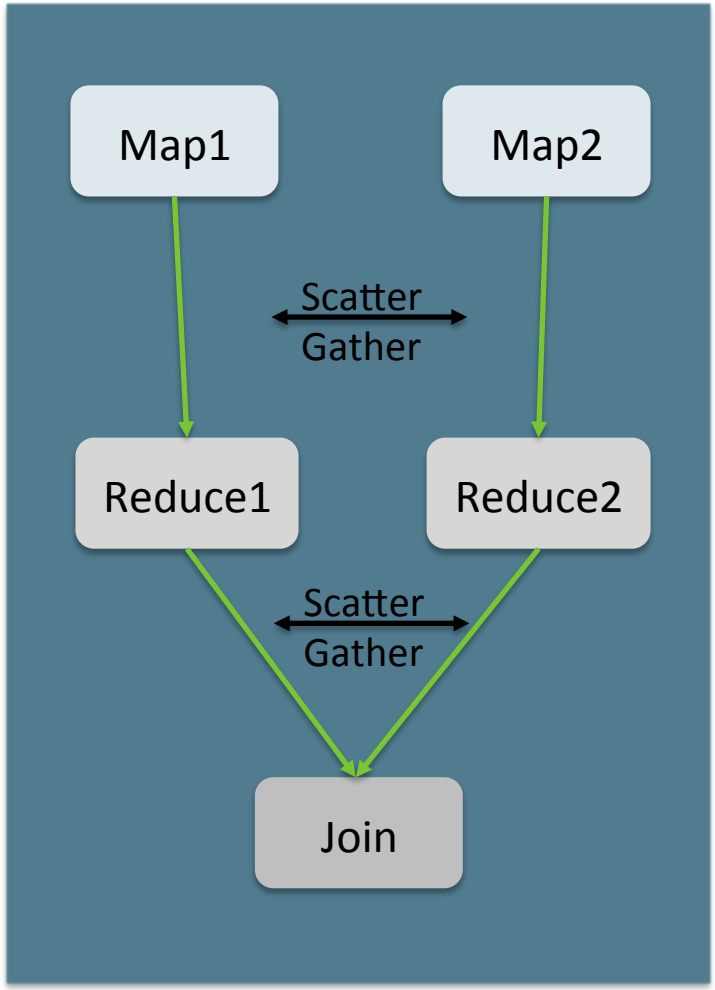
Tez – Expressing the computation

- **Distributed data processing jobs typically look like DAGs (Directed Acyclic Graph).**
- **Vertices in the graph represent data transformations**
- **Edges represent data movement from producers to consumers**



Tez – API

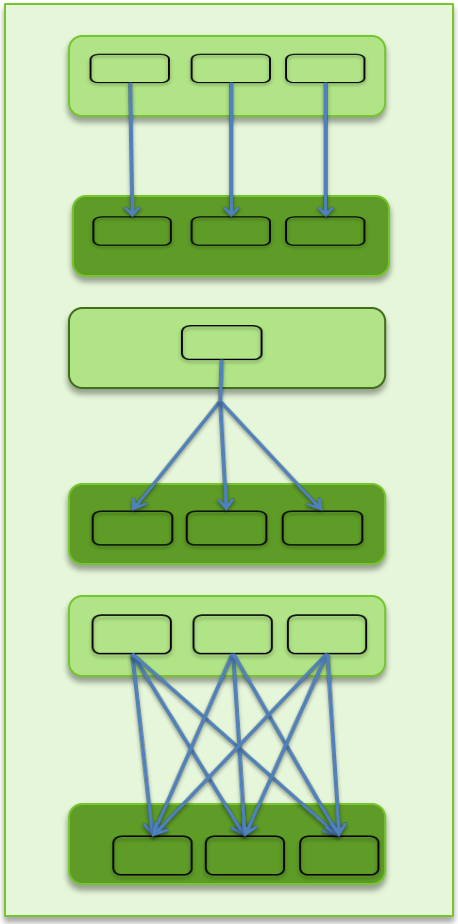
```
// Define DAG
DAG dag = new DAG();

// Define Vertex
Vertex Map1 = new
Vertex(Processor.class);

// Define Edge
Edge edge = Edge(Map1, Reduce2,
SCATTER_GATHER, PERSISTED,
SEQUENTIAL, Output.class, Input.class);

// Connect them
dag.addVertex(Map1).addEdge(edge)…
```



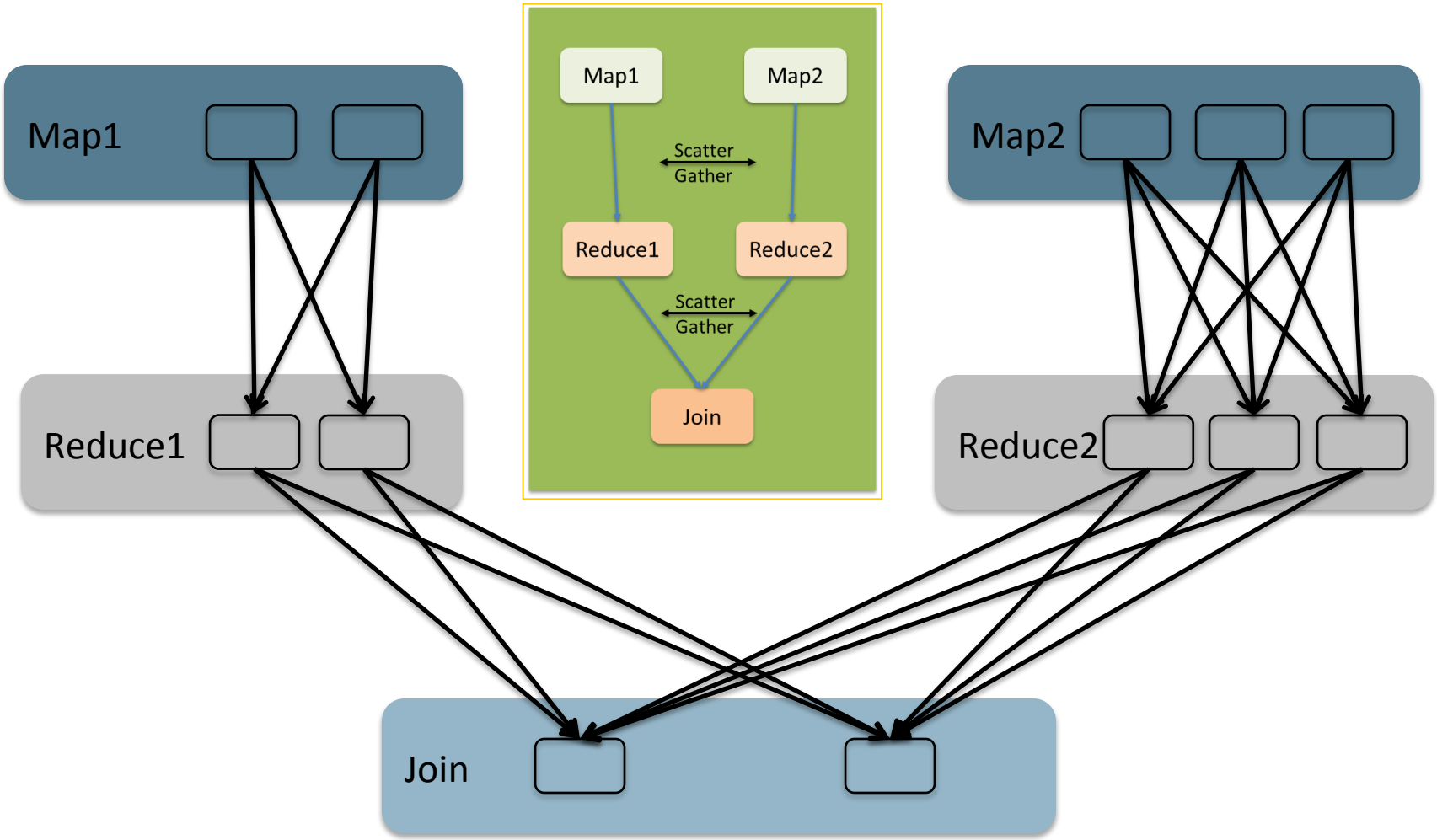
Tez – DAG API

Edge properties define the connection between producer and consumer tasks in the DAG

- **Data movement – Defines routing of data between tasks**
  - **One-To-One** : Data from the i^{th} producer task routes to the i^{th} consumer task.
  - **Broadcast** : Data from a producer task routes to all consumer tasks.
  - **Scatter-Gather** : Producer tasks scatter data into shards and consumer tasks gather the data. The i^{th} shard from all producer tasks routes to the i^{th} consumer task.
- **Scheduling – Defines when a consumer task is scheduled**
  - **Sequential** : Consumer task may be scheduled after a producer task completes.
  - **Concurrent** : Consumer task must be co-scheduled with a producer task.
- **Data source – Defines the lifetime/reliability of a task output**
  - **Persisted** : Output will be available after the task exits. Output may be lost later on.
  - **Persisted-Reliable** : Output is reliably stored and will always be available
  - **Ephemeral** : Output is available only while the producer task is running



Tez – Logical DAG expansion at Runtime



Tez – Runtime API

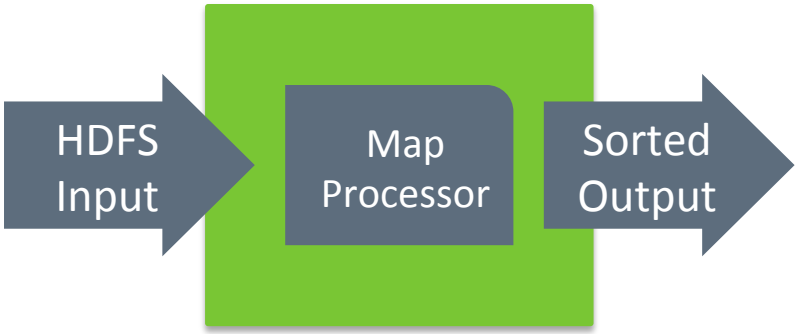
Flexible Inputs-Processor-Outputs Model

- Thin API layer to wrap around arbitrary application code
- Compose inputs, processor and outputs to execute arbitrary processing
- Event routing based control plane architecture
- Applications decide logical data format and data transfer technology
- Customize for performance
- Built-in implementations for Hadoop 2.0 data services – HDFS and YARN ShuffleService. Built on the same API. Your impls are as first class as ours!

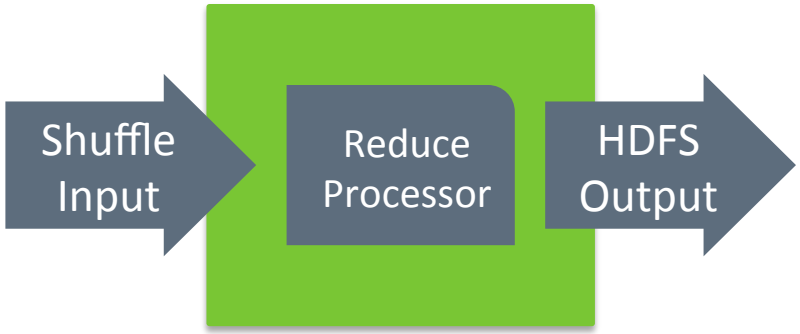
Input	Processor	Output
initialize(TezInputContext ctxt)	initialize(TezProcessorContext ctxt)	initialize(TezOutputContext ctxt)
Reader getReader()	run(List<Input> inputs, List<Output> outputs)	Writer getWriter()
handleEvents(List<Event> evts)	handleEvents(List<Event> evts)	handleEvents(List<Event> evts)
close()	close()	close()

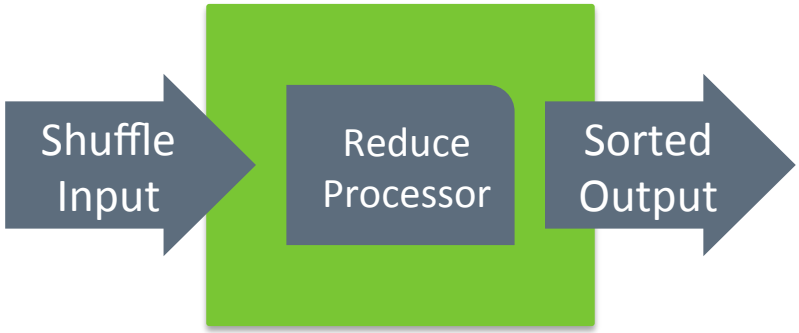
Tez – Library of Inputs and Outputs

Classical 'Map'



Classical 'Reduce'



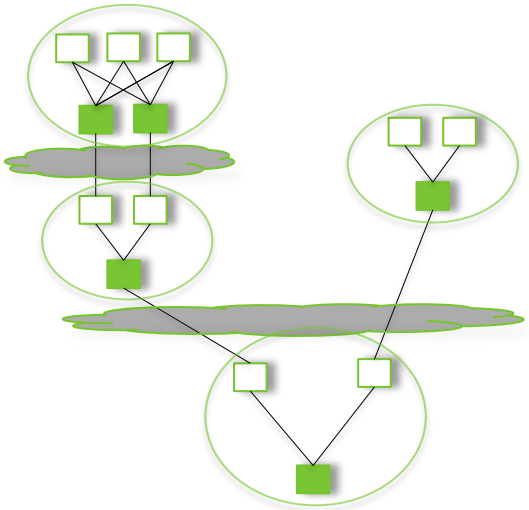


Intermediate 'Reduce' for Map-Reduce-Reduce

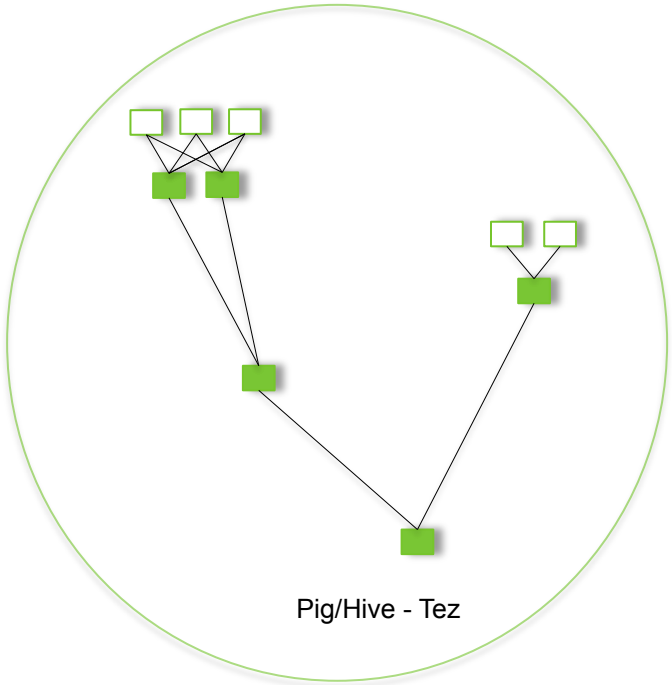
- **What is built in?**
  - Hadoop InputFormat/OutputFormat
  - SortedGroupedPartitioned Key-Value Input/Output
  - UnsortedGroupedPartitioned Key-Value Input/Output
  - Key-Value Input/Output

Tez – Performance

- **Benefits of expressing the data processing as a DAG**
  - Reducing overheads and queuing effects
  - Gives system the global picture for better planning



Pig/Hive - MR



Pig/Hive - Tez

Tez – Performance

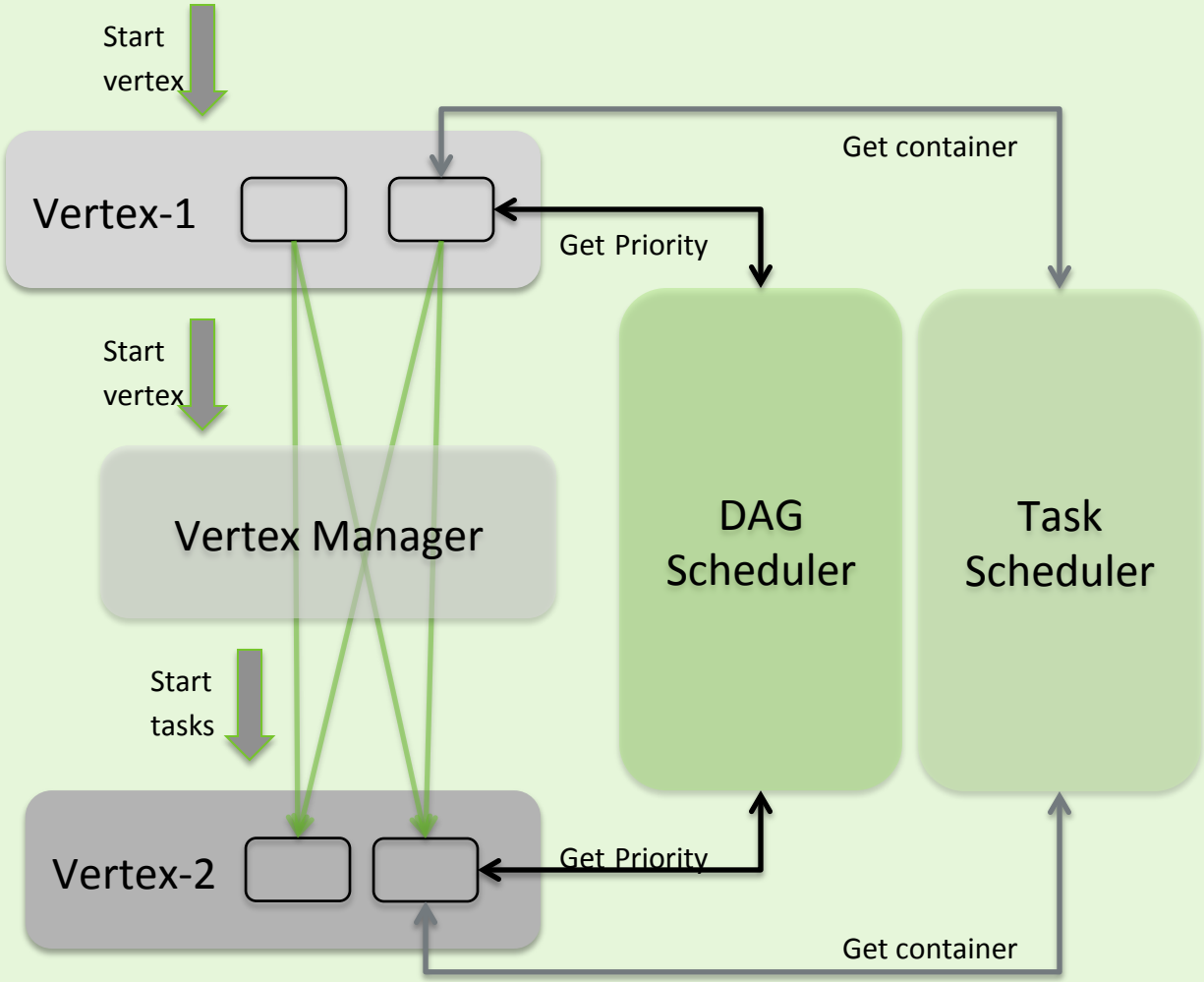
- **Efficient use of resources**
  - Re-use resources to maximize utilization
  - Pre-launch, pre-warm and cache
  - Locality & resource aware scheduling
- **Support for application defined DAG modifications at runtime for optimized execution**
  - Change task concurrency
  - Change task scheduling
  - Change DAG edges
  - Change DAG vertices

Tez – Customizable Core Engine

- Vertex Manager
  - Determines task parallelism
  - Determines when tasks in a vertex can start.
- DAG Scheduler

  Determines priority of task
- Task Scheduler

  Allocates containers from YARN and assigns them to tasks



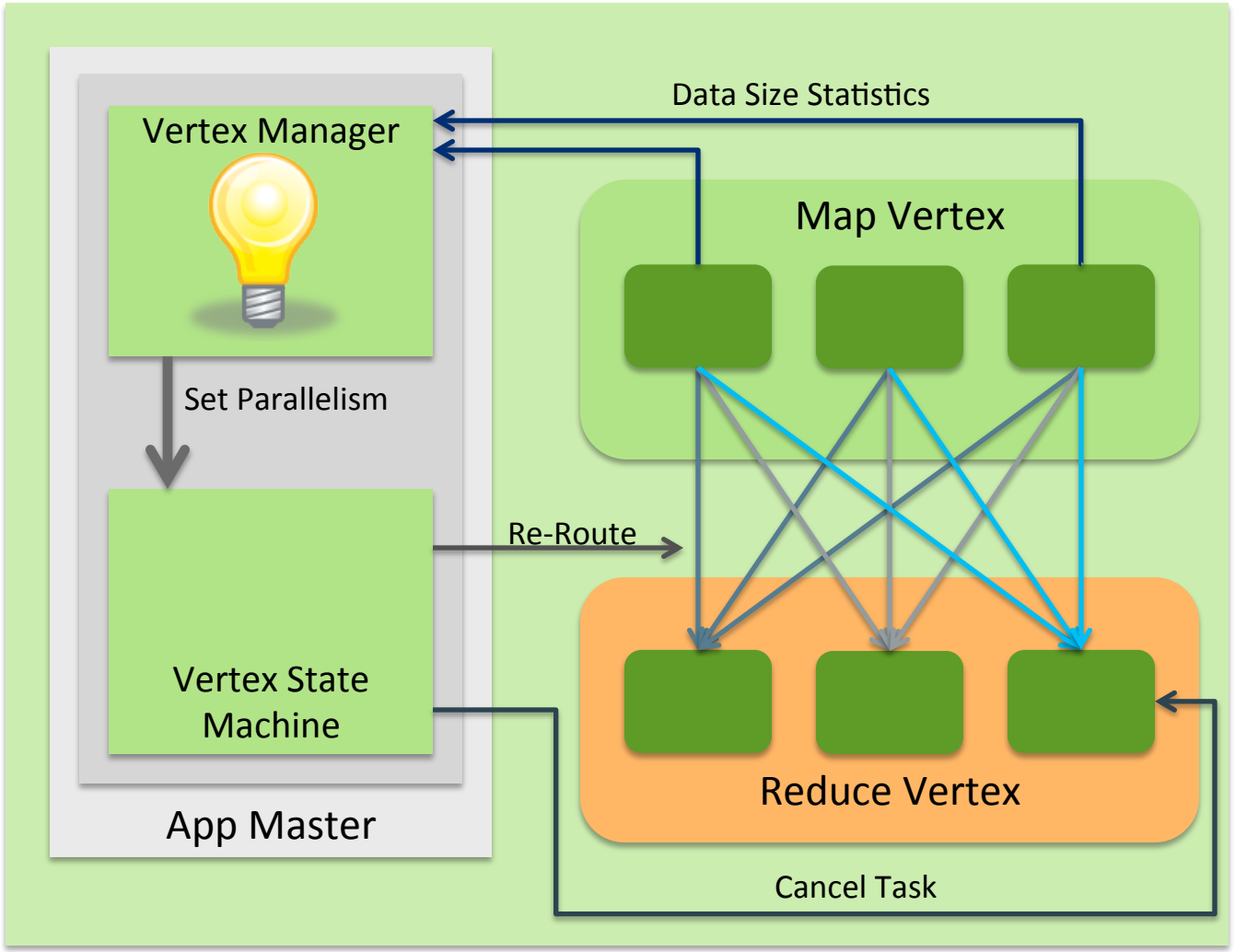
Tez – Automatic Reduce Parallelism

Event Model

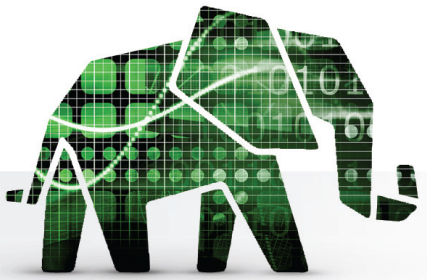
Map tasks send data statistics events to the Reduce Vertex Manager.

Vertex Manager

Pluggable application logic that understands the data statistics and can formulate the correct parallelism. Advises vertex controller on parallelism

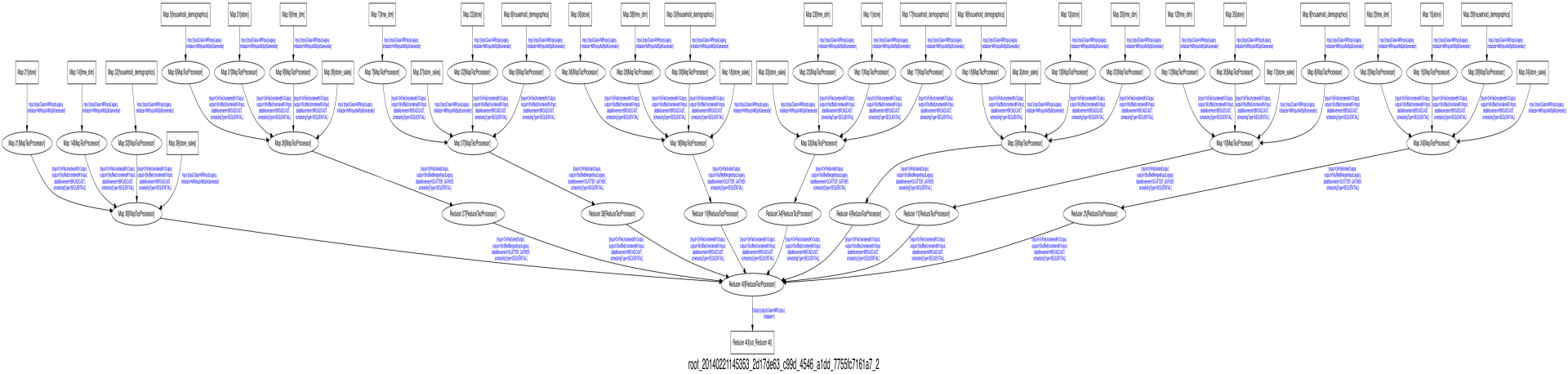


Theory to Practice

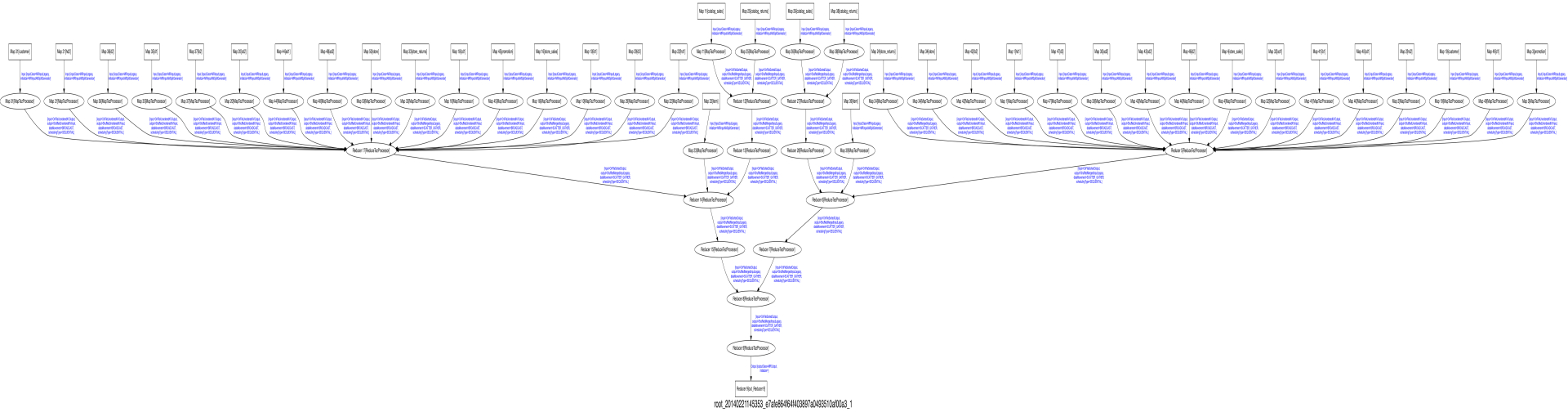


Tez – DAG definition at scale

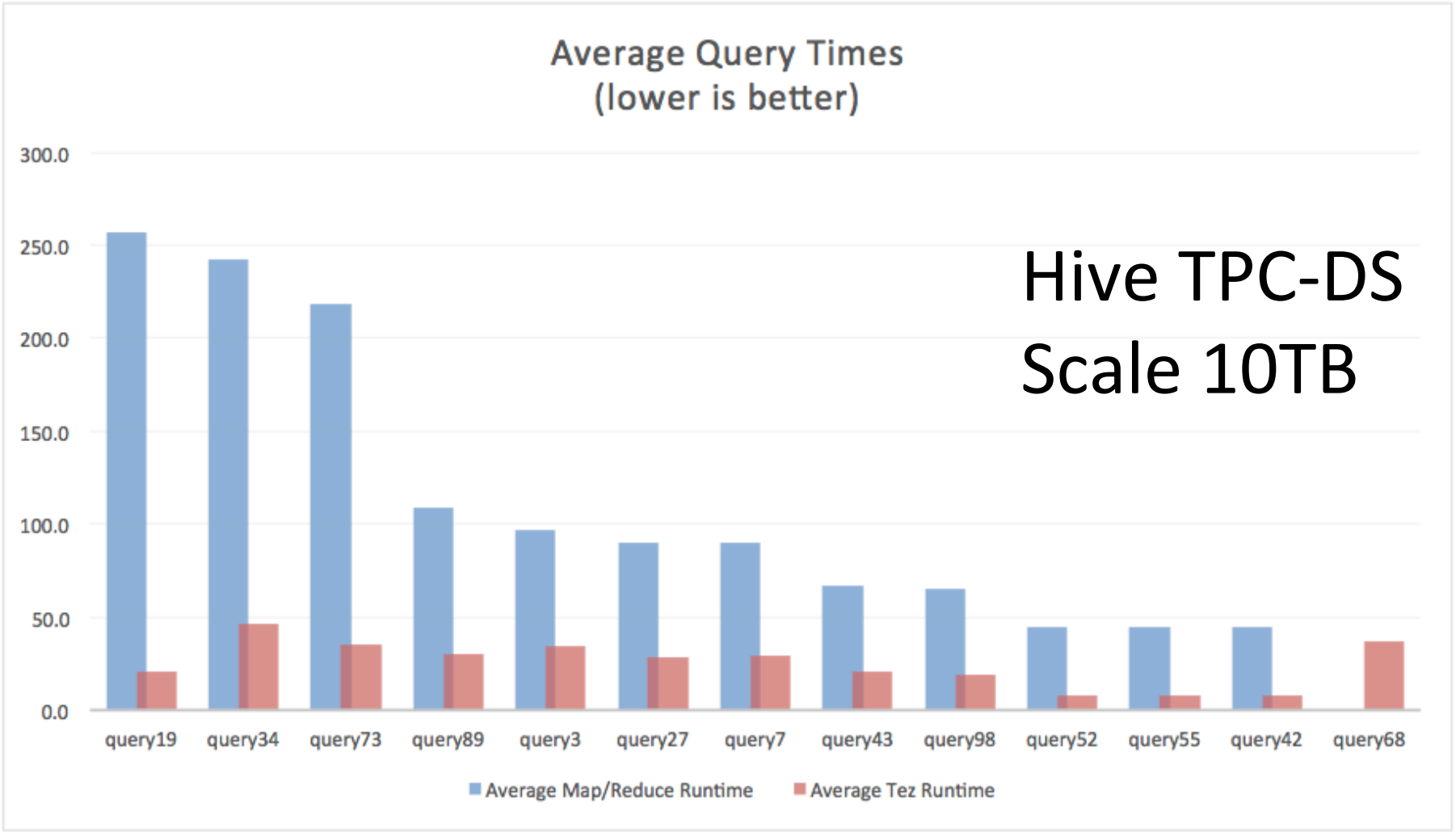
Hive : TPC-DS Query 64 Logical DAG



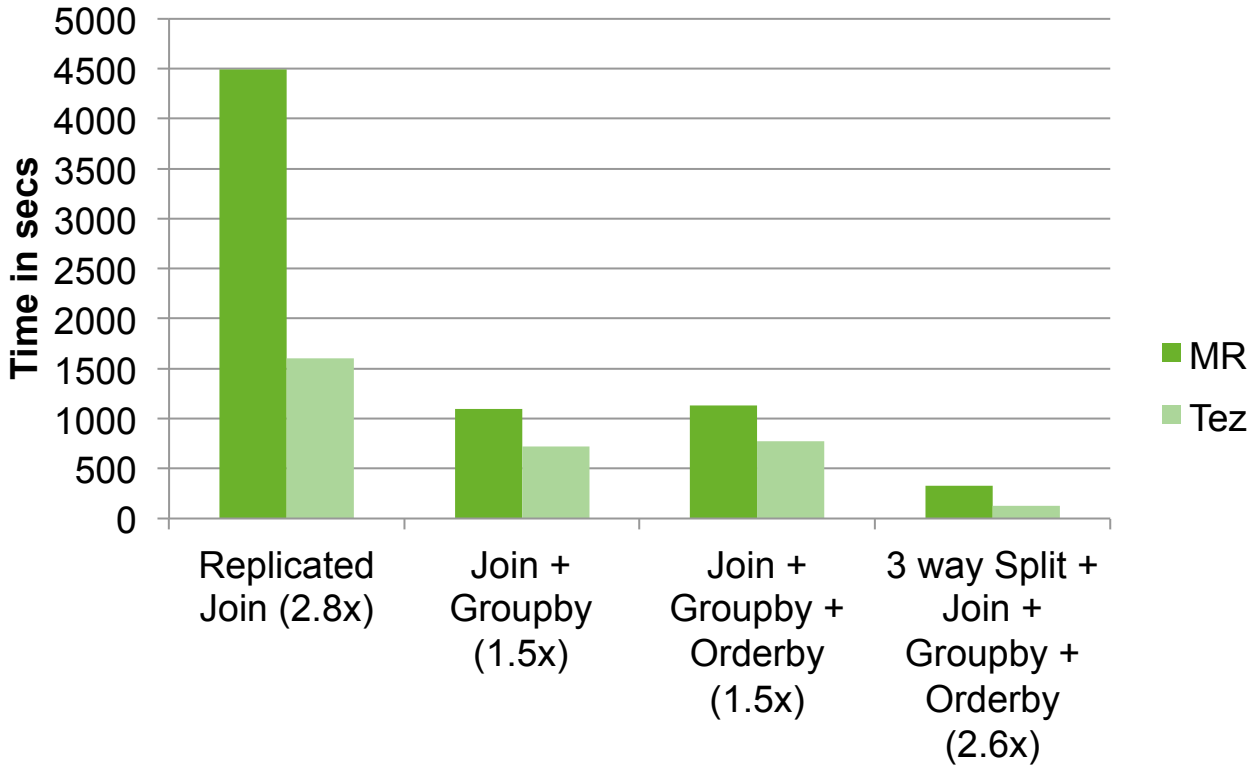
Hive : TPC-DS Query 88 Logical DAG



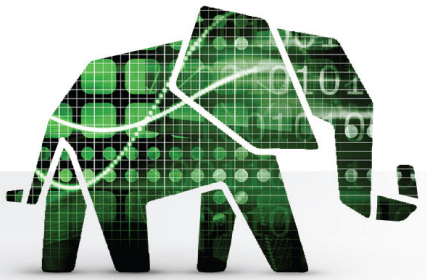
Tez – Data at scale



Tez – Pig performance gains



Real-world Examples



Tez – Broadcast Edge

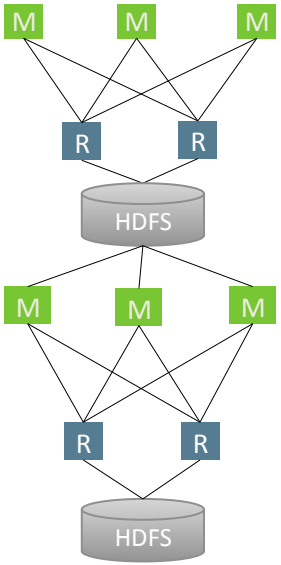
```
SELECT ss.ss_item_sk, ss.ss_quantity, avg_price, inv.inv_quantity_on_hand
FROM (select avg(ss_sold_price) as avg_price, ss_item_sk, ss_quantity_sk from store_sales
group by ss_item_sk) ss
JOIN inventory inv
ON (inv.inv_item_sk = ss.ss_item_sk);
```

Hive : Broadcast Join

Hive – MR

Store Sales scan. Group by and aggregation.

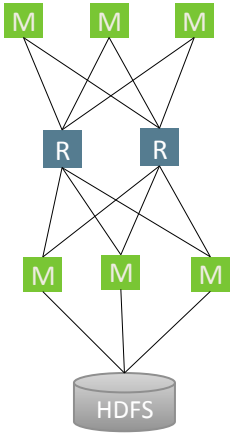
Inventory and Store Sales (aggr.) output scan and shuffle join.



Hive – Tez

Store Sales scan. Group by and aggregation reduce size of this input.

Inventory scan and Join



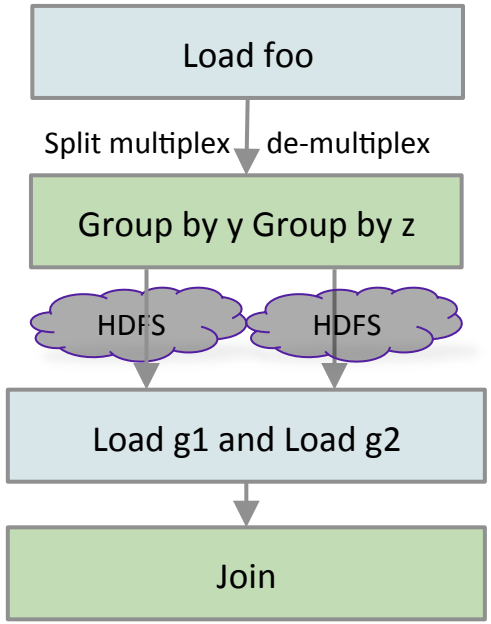
Broadcast edge

Tez – Multiple Outputs

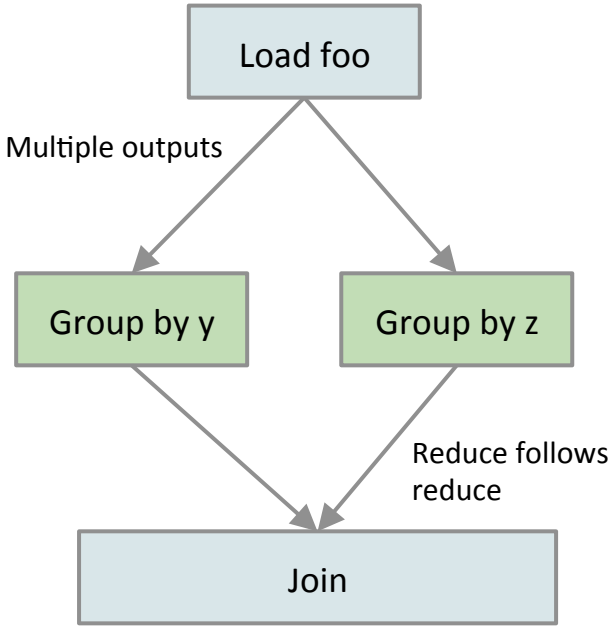
```
f = LOAD 'foo' AS (x, y, z);
g1 = GROUP f BY y;
g2 = GROUP f BY z;
j = JOIN g1 BY group,
        g2 BY group;
```

Pig : Split & Group-by

Pig – MR



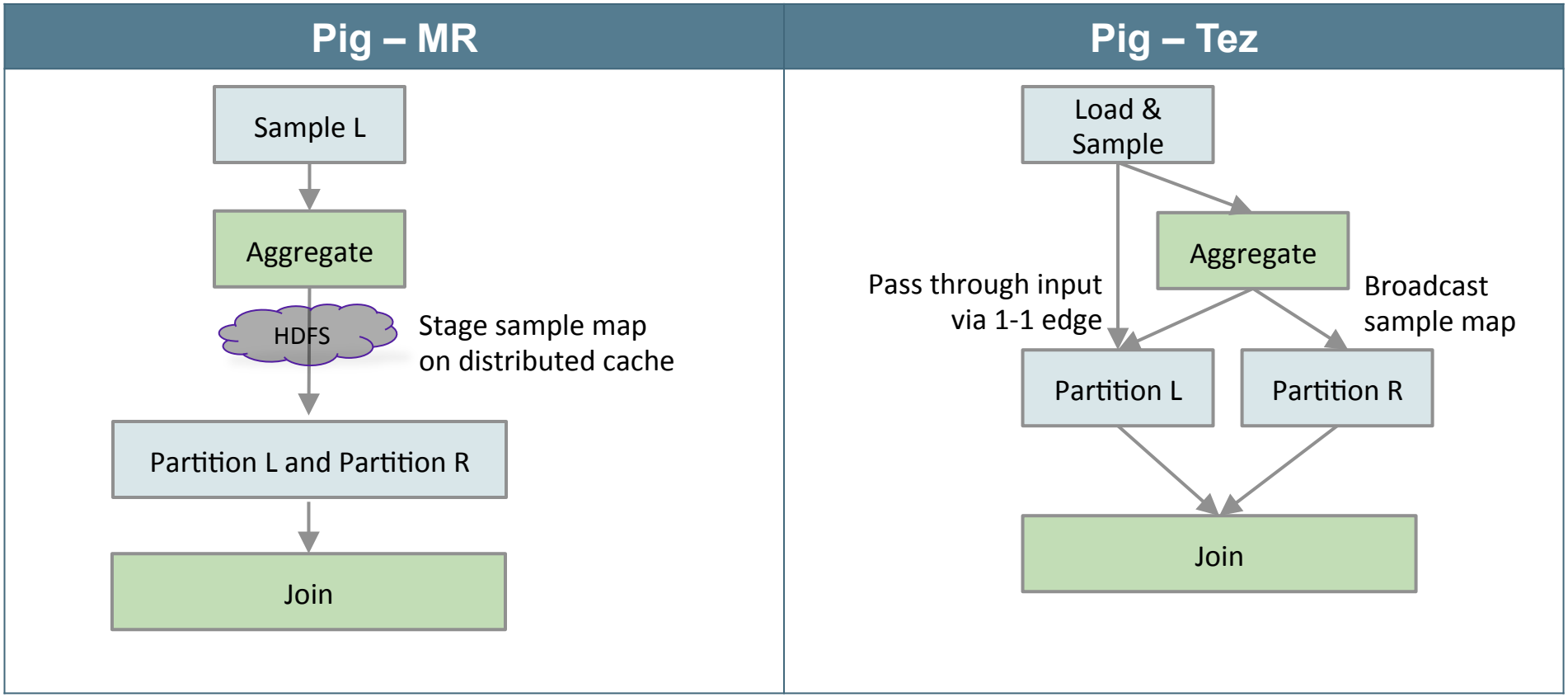
Pig – Tez



Tez – One to One Edge

```
l = LOAD 'left' AS (x, y);
r = LOAD 'right' AS (x, z);
j = JOIN l BY x, r BY x
    USING 'skewed';
```

Pig : Skewed Join



Tez – Current Status

- **Adoption**
  - Apache Hive 0.13+
  - Apache Pig 0.14+
  - Cascading 3.0
  - Apache Flink (Incubating) – Initial prototype
- **Releases:**
  - 0.5.2 recently released
  - 0.6.0 coming soon

Tez – Summary

- **Tez website: http://tez.apache.org**
- **Questions: Drop a mail to users@tez.apache.org**
- **Try running your Hive/Pig/MR jobs in Tez mode**
  - set hive.execution.engine=tez
  - bin/pig –x tez
  - mapred –Dmapreduce.framework.name=yarn-tez

Questions?