cloudera
Ask Bigger Questions

Resource Management with YARN and Impala

Henry Robinson | @HenryR

Strata + Hadoop World 2013, 2013-10-29

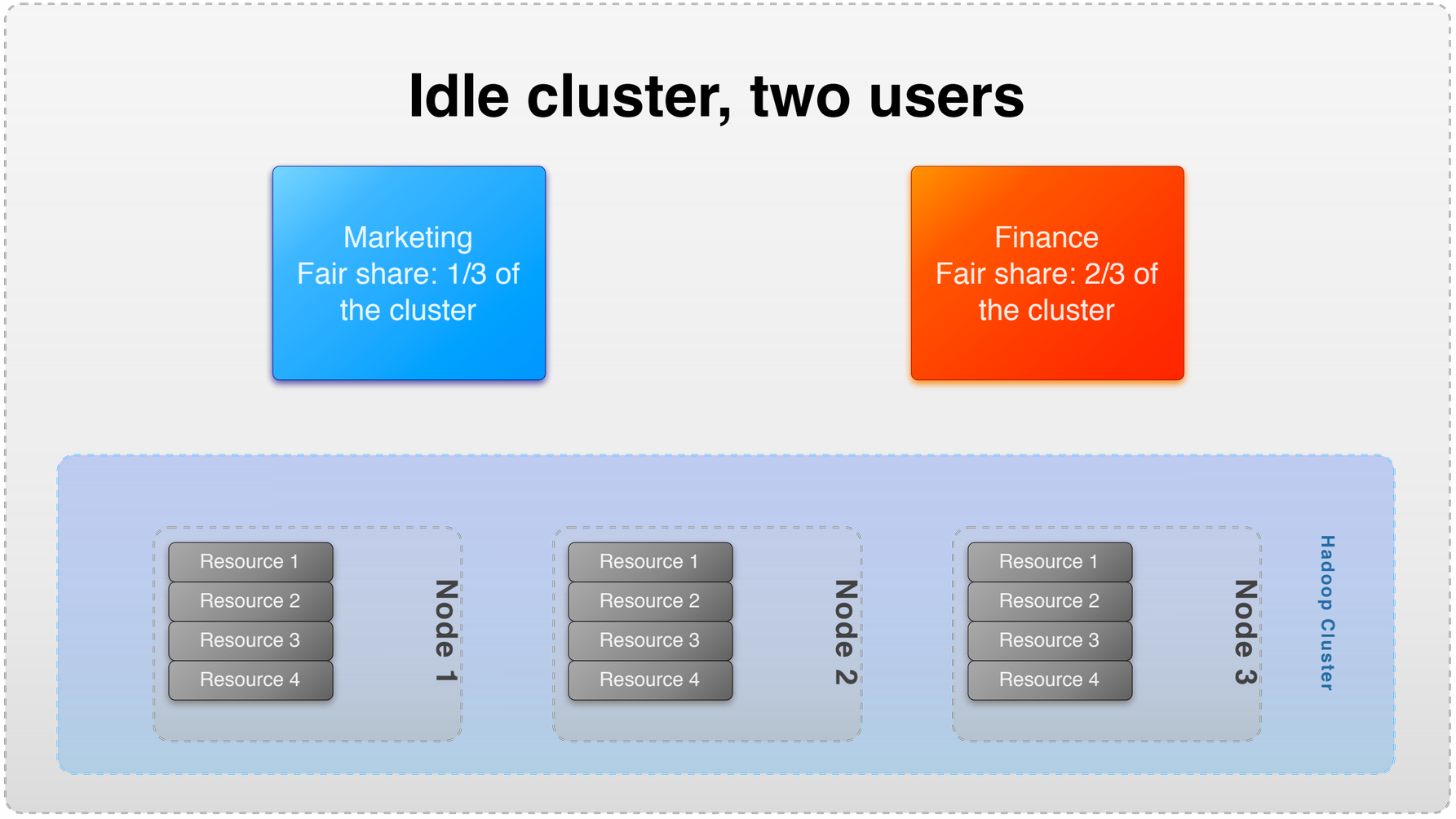
Resource management?

Can’t we all just get along?

Real-world resource management

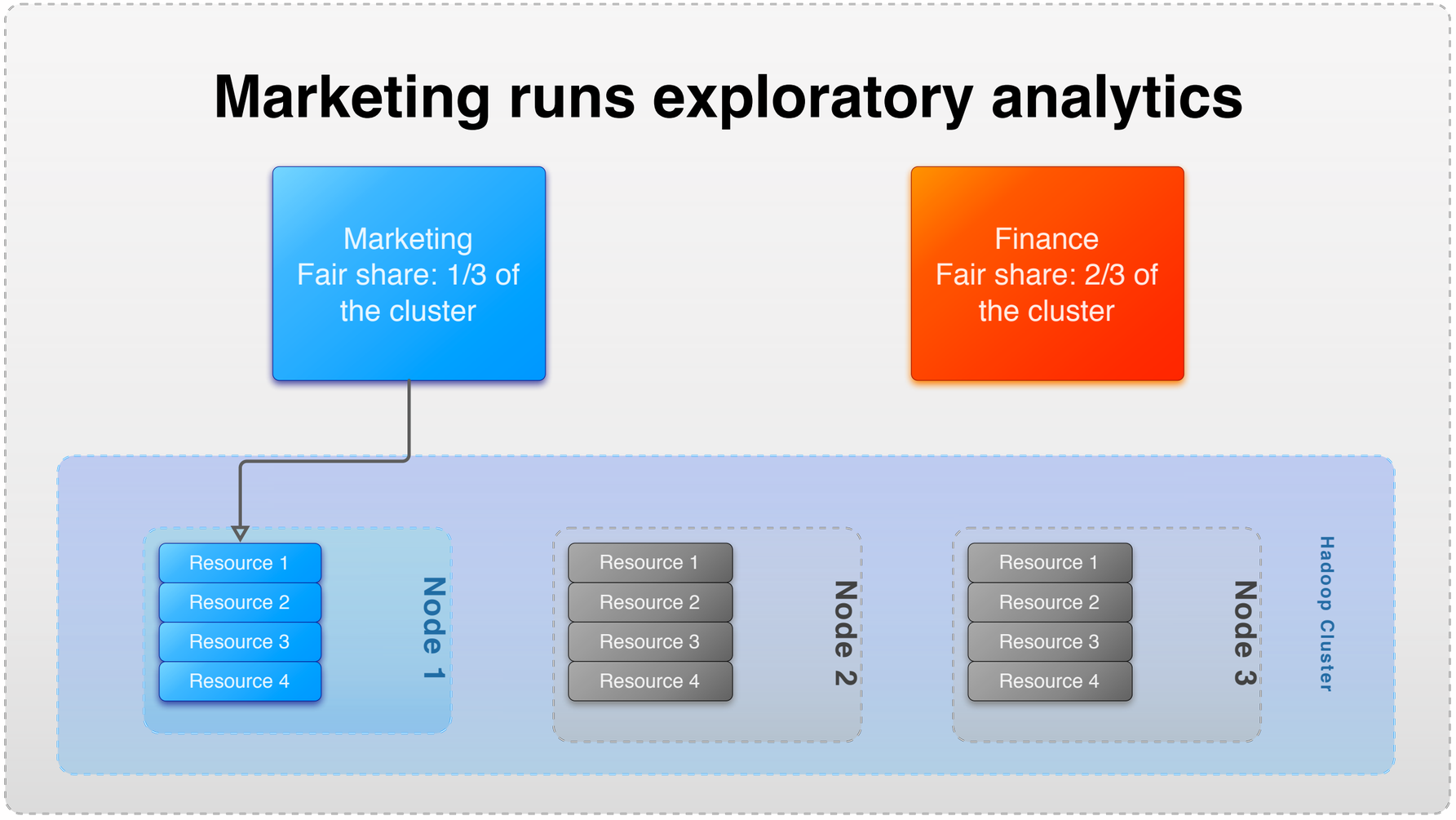
- Hadoop brings generalised computation to big data
- Workloads are becoming more and more diverse
  - From frameworks including MR, **Impala**, Spark, Search
- Some workloads are more important than others
- A cluster is only a finite resource
  - Limited CPU, memory, disk and network bandwidth
- **How do we make sure each workload gets the resources it deserves?**

Example: Finance vs Marketing

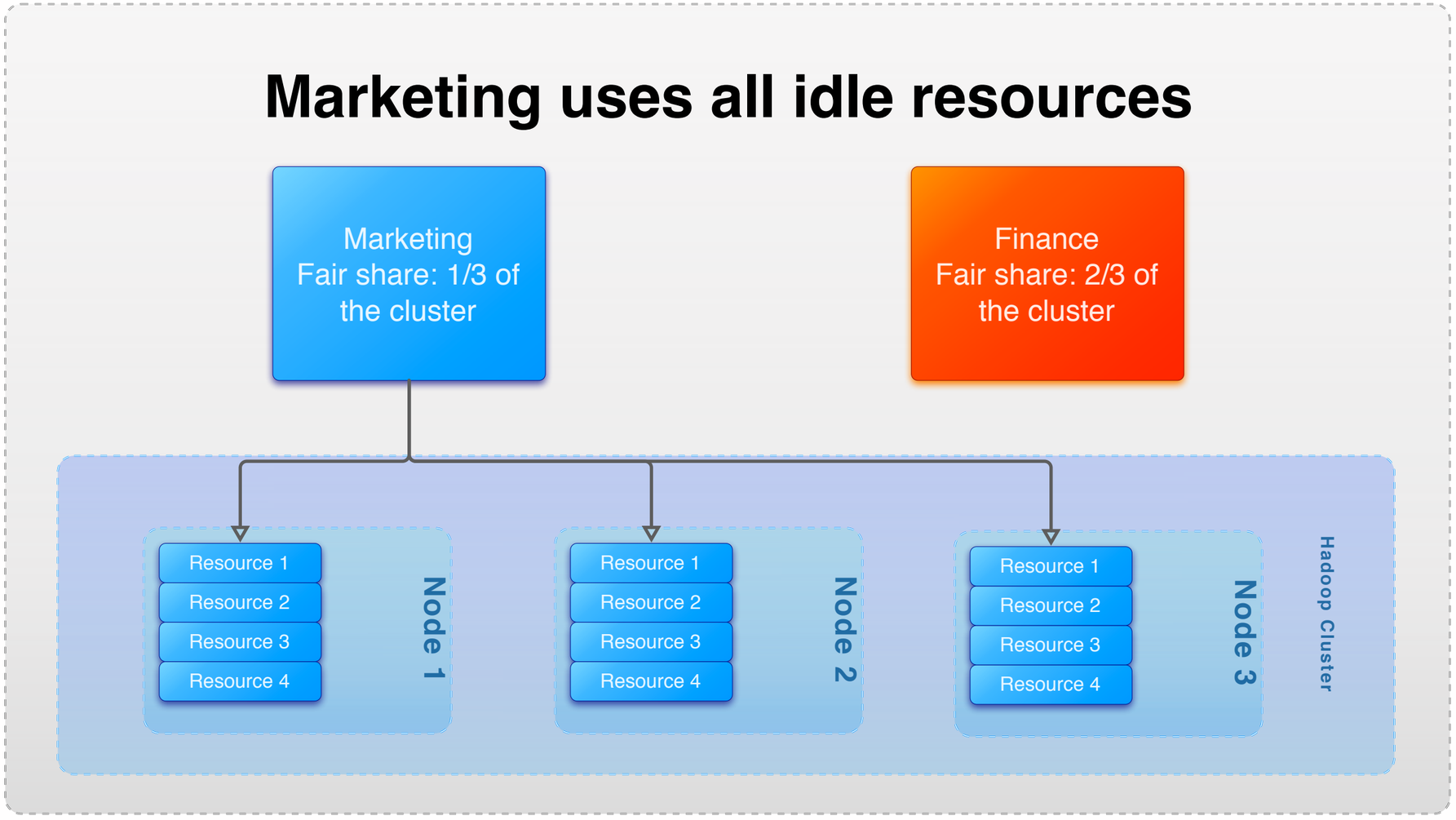


cloudera
Ask Bigger Questions

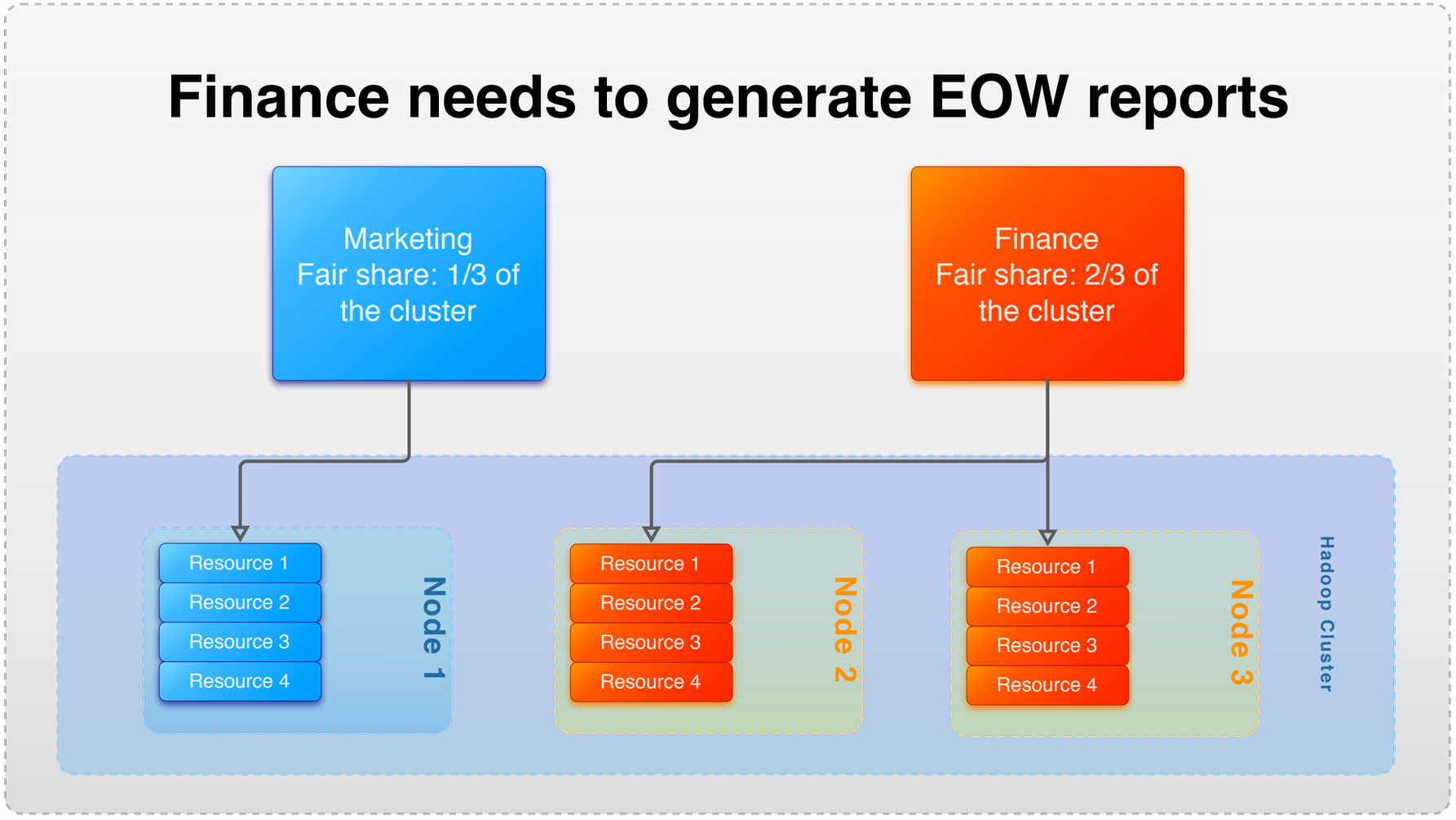
Example: Finance vs Marketing



Example: Finance vs Marketing



Example: Finance vs Marketing



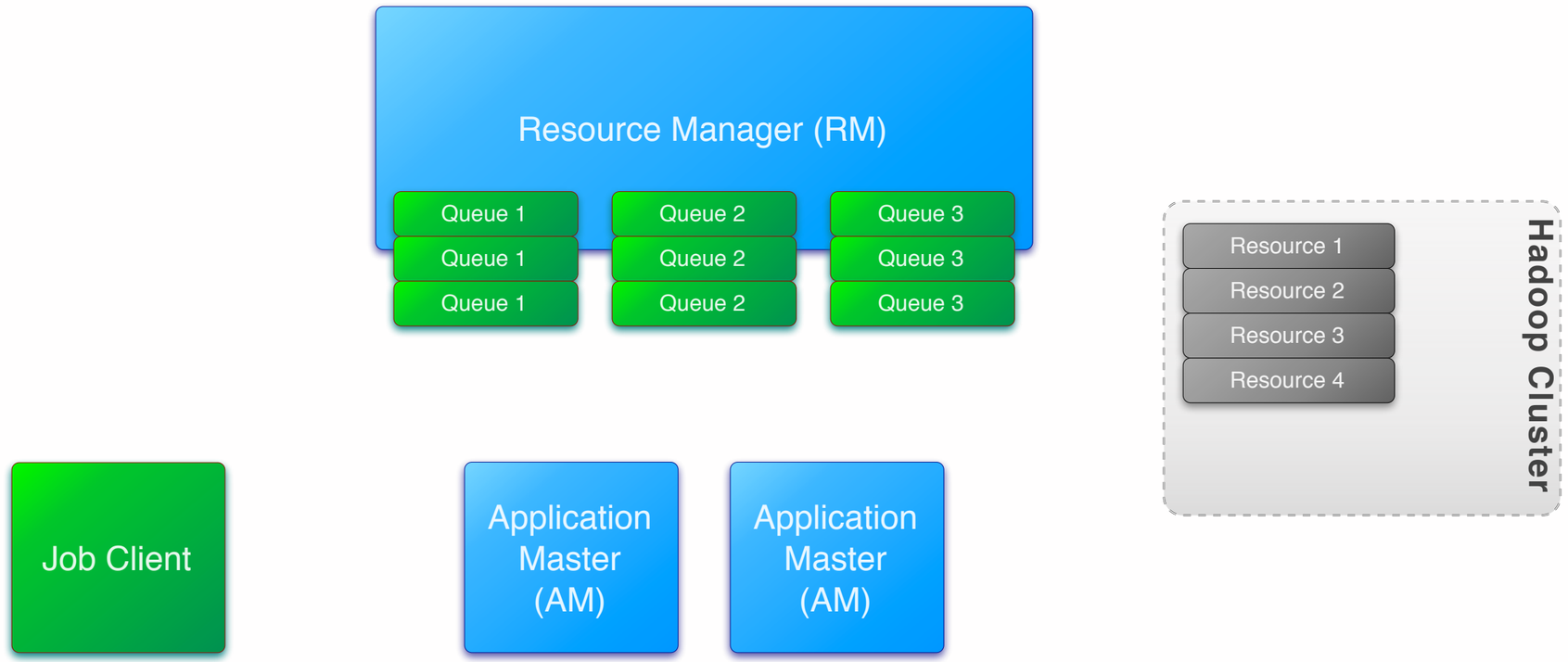
cloudera
Ask Bigger Questions

State-of-the-art: Apache Yarn

- YARN brings resource management to Hadoop
  - Via a centralised **resource manager** which satisfies resource allocation requests in turn
- Resources are shared between **queues**, each of which is entitled to a share of the cluster
  - e.g. Finance gets 2/3, marketing gets 1/3
- Each **job** registers as an **application master** with YARN
  - YARN manages the resource lifecycle on behalf of that job

Jobs and resources

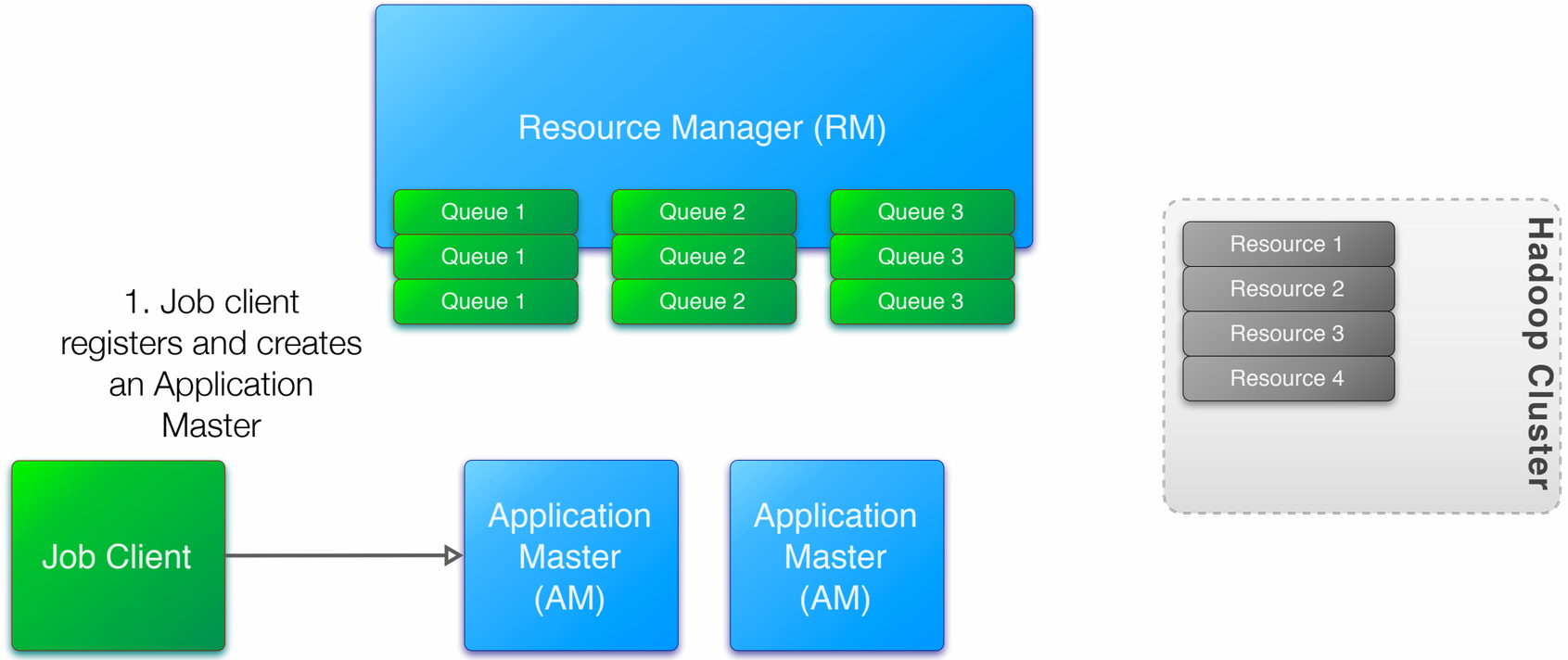
Job creation and resource acquisition



cloudera
Ask Bigger Questions

Jobs and resources

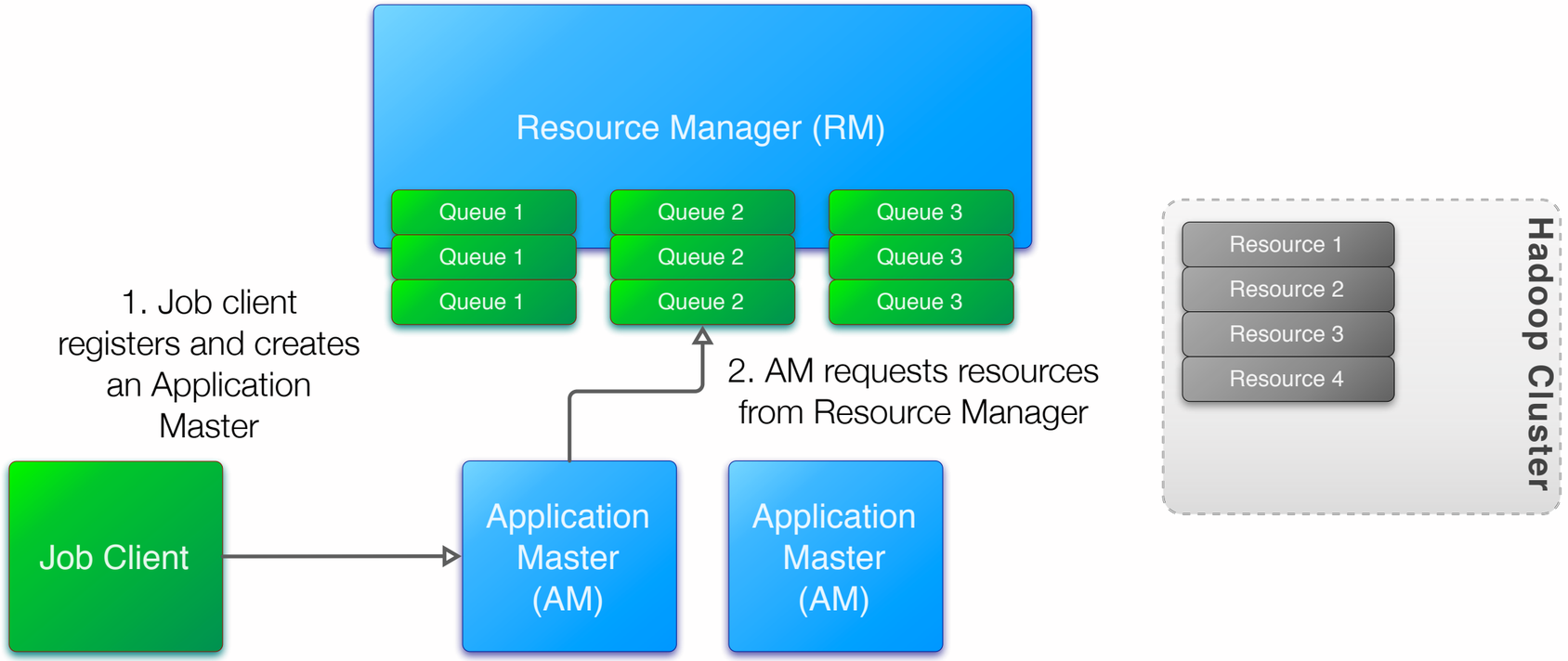
Job creation and resource acquisition



cloudera
Ask Bigger Questions

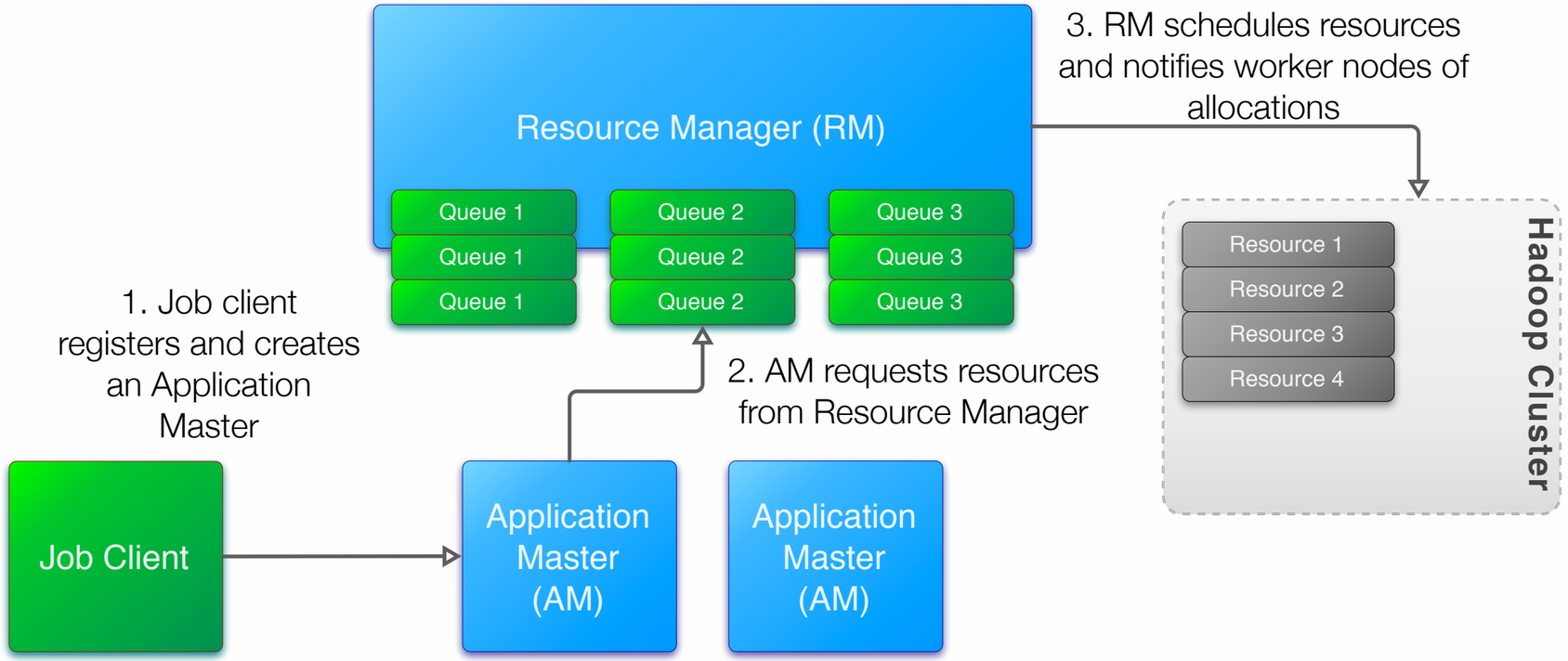
Jobs and resources

Job creation and resource acquisition



Jobs and resources

Job creation and resource acquisition



cloudera
Ask Bigger Questions

More on the Application Master

- Creating an Application Master involves two round trips to the central Resource Manager
- An Application Master only lasts as long as the job does
- AM creation latency is much higher than resource acquisition

YARN is focused on batch processing

- Hadoop comes from batch-processing via MapReduce
- Batch job runtimes are usually dominated by the time to actually process the data
- So MapReduce can afford the cost of AM registration protocol, and of allocating resources centrally
- Number of concurrent jobs is usually in the hundreds
- Median job time usually > 10s

Hadoop is no longer only about batch

- Cloudera Impala, announced a year ago, brings **interactive, low-latency SQL queries** to Hadoop
- Serves a number of critical use cases as more and more data migrates to Hadoop
  - Such as Business Intelligence, exploratory analytics, general SQL processing
- Query times routinely sub-second. Concurrent query volumes can be closer to 1000.
- **The overhead of YARN is much more significant**

How do we bring the two worlds together?

- Impala wants to participate fully in resource management to allows users to get the resources their workloads deserve
- YARN is not yet prepared for the latency and throughput requirements that Impala, Spark and other similar frameworks will bring.
- **How do we overcome the impedance mismatch?**

Resource management and Impala

- First part (most of the rest of this talk): integrating Impala with YARN, and overcoming the most significant hurdles

- Second part: Future plans to improve performance even further

1. Fixing the Impedance Mismatch

Fitting a square peg into a round hole

Too many Application Masters

- As we saw, YARN's model is one AM-per-job
- We quickly discovered this doesn't scale well to very high query volumes
- Impala submits thousands of queries-per-minute
- Something's got to give

Long-Lived Application Masters

- We created a new component called the **Long-Lived Application Master**
- The idea is to register only one AM per-queue-per-framework
- Takes the load off YARN (far fewer AMs)
- And takes the load off queries (AMs already established)

Long-Lived Application Masters

- We created a new component called the **Long-Lived Application MAster**

Long-Lived Application Masters

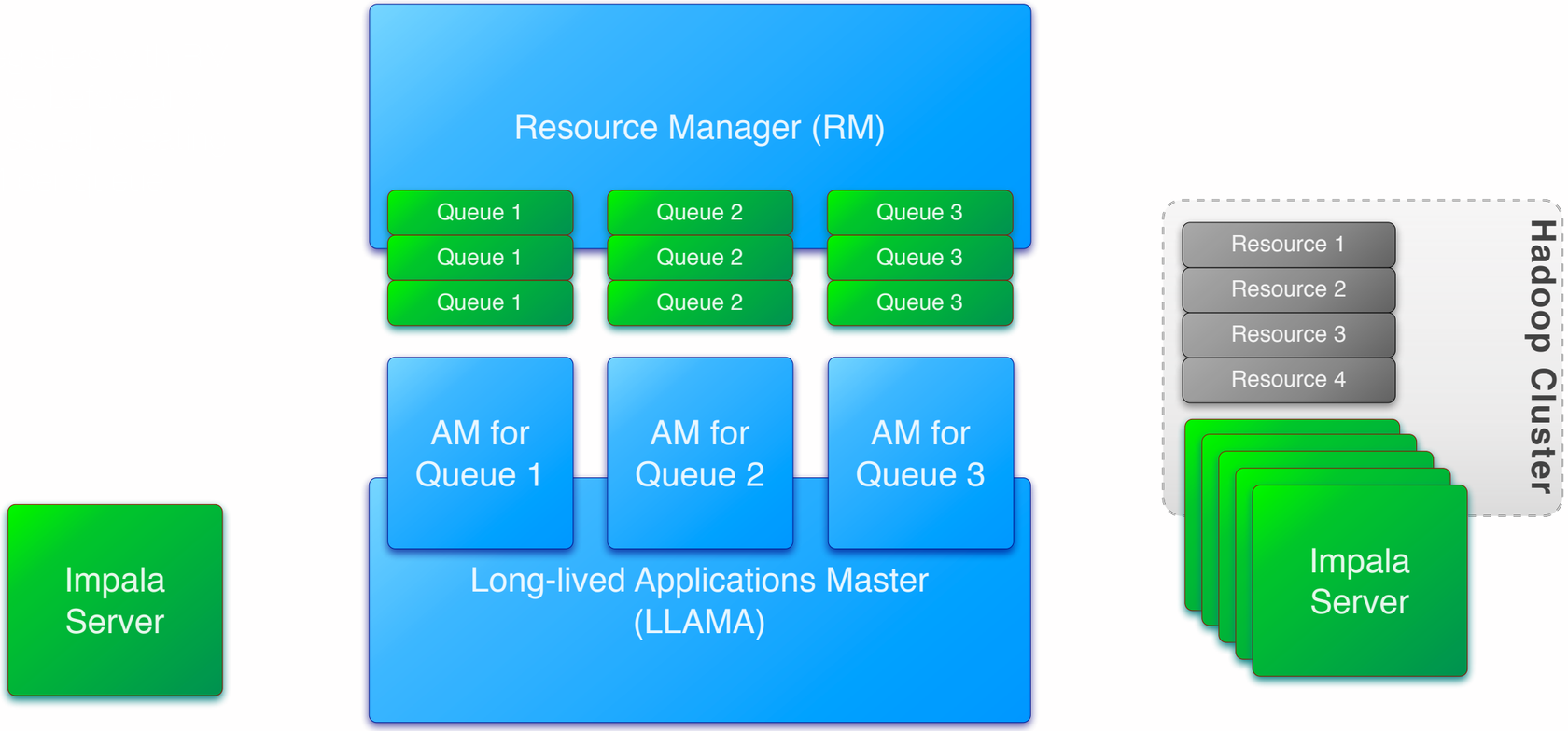
- We created a new component called the **Long-Lived Application MAster: LLAMA**

Photo credit: Wikipedia

cloudera
Ask Bigger Questions

How Llama fits in

Llama's role in Impala resource acquisition

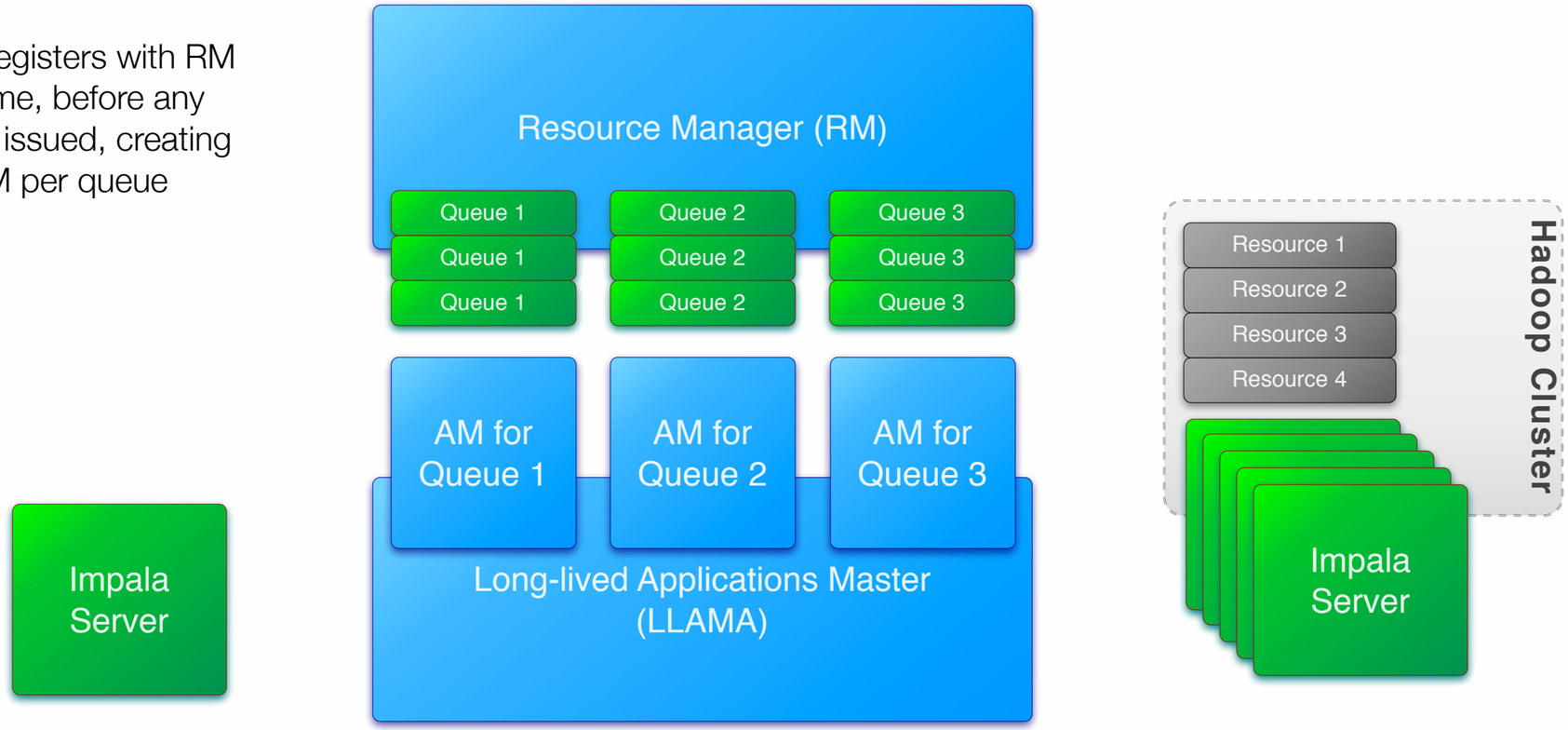


cloudera
Ask Bigger Questions

How Llama fits in

Llama's role in Impala resource acquisition

1. LLAMA registers with RM at start time, before any queries are issued, creating one AM per queue



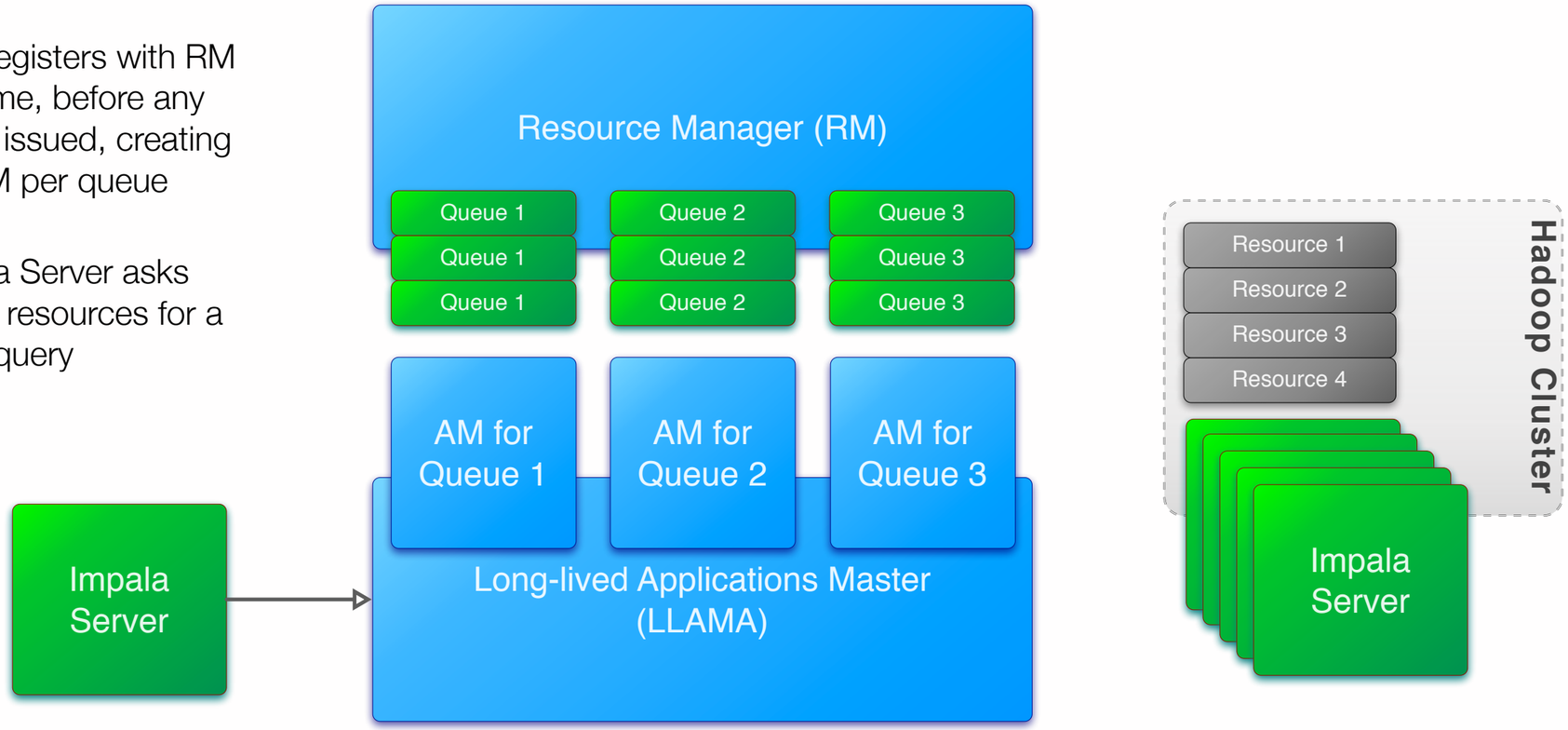
cloudera
Ask Bigger Questions

How Llama fits in

Llama's role in Impala resource acquisition

1. LLAMA registers with RM at start time, before any queries are issued, creating one AM per queue

2. Impala Server asks LLAMA for resources for a query



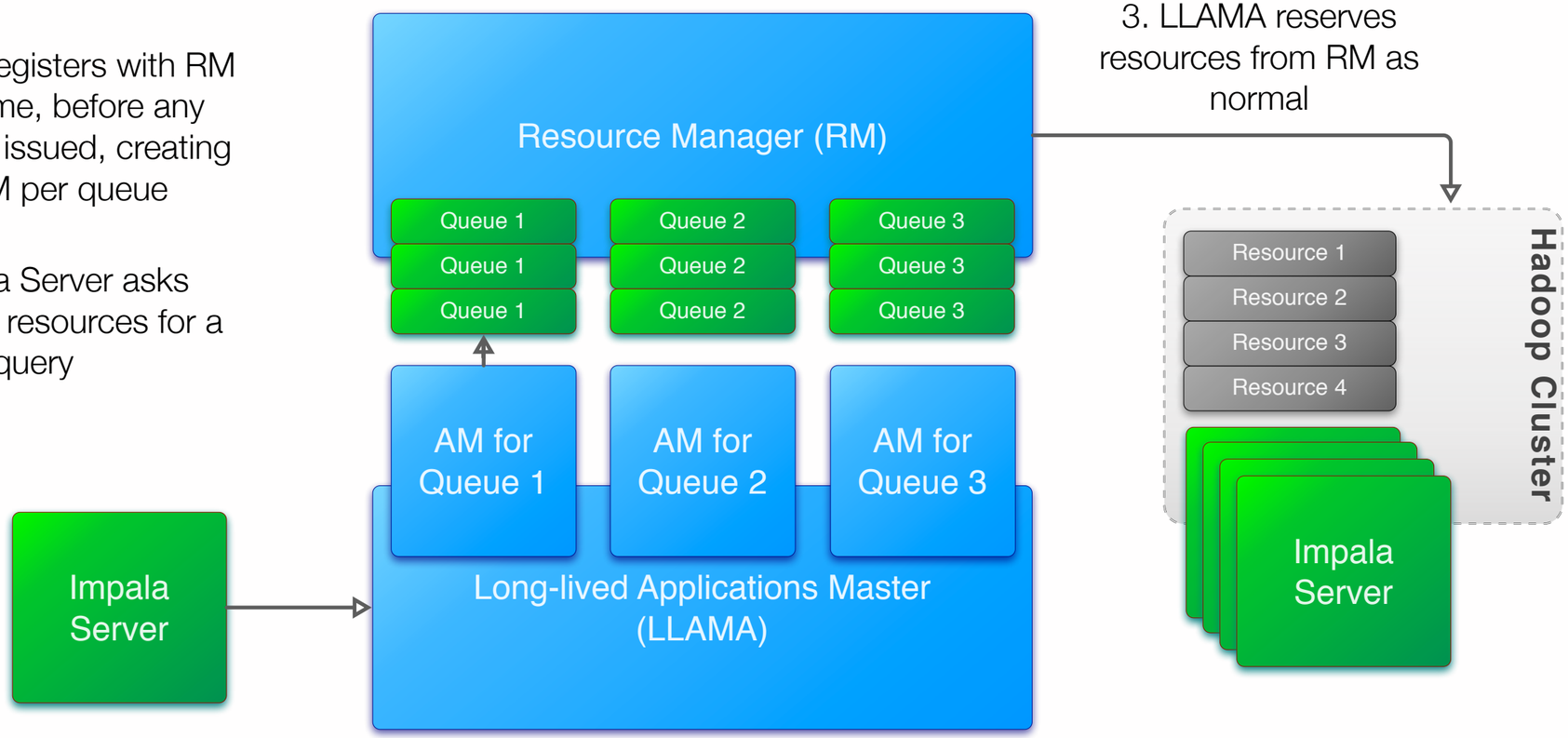
cloudera
Ask Bigger Questions

How Llama fits in

Llama's role in Impala resource acquisition

1. LLAMA registers with RM at start time, before any queries are issued, creating one AM per queue

2. Impala Server asks LLAMA for resources for a query



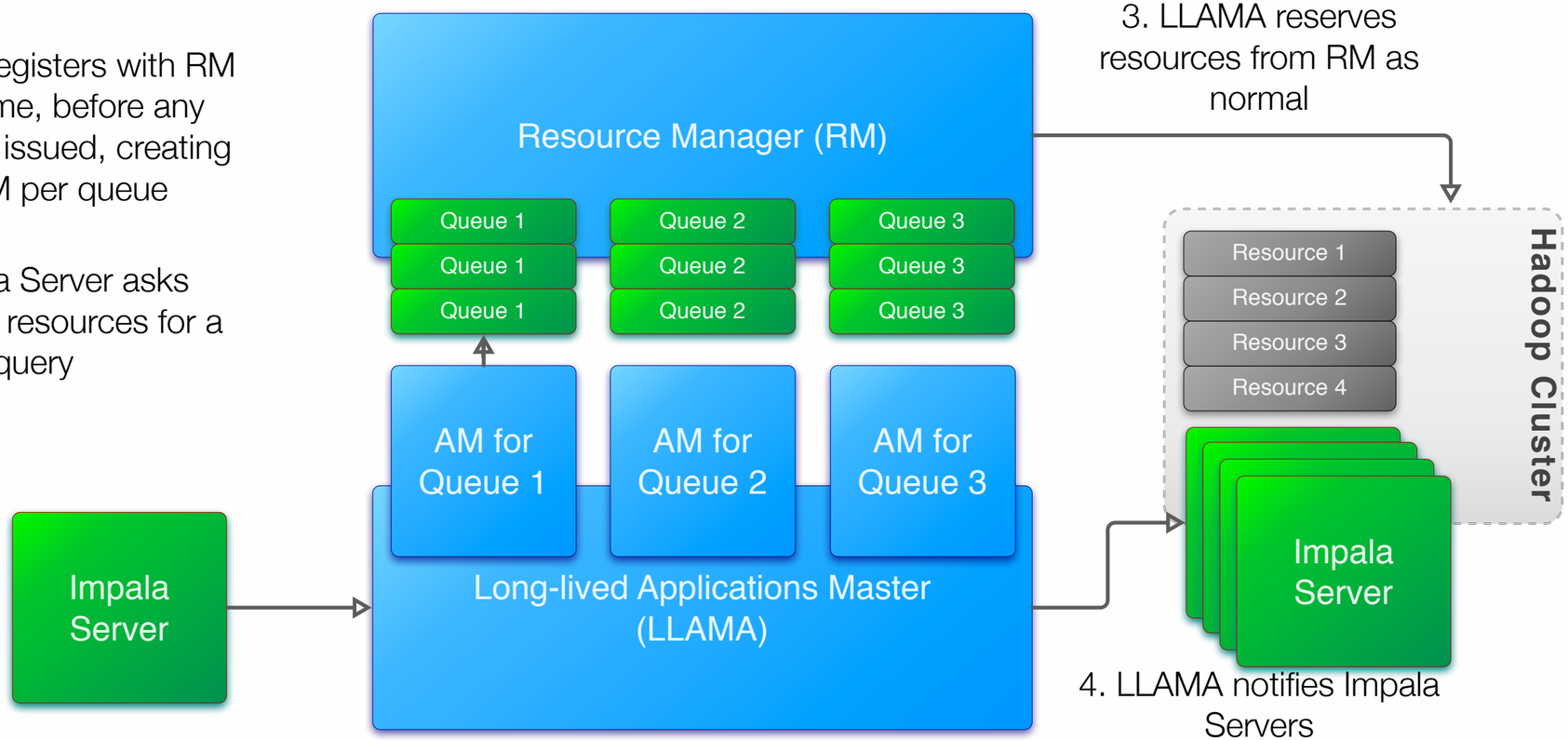
cloudera
Ask Bigger Questions

How Llama fits in

Llama's role in Impala resource acquisition

1. LLAMA registers with RM at start time, before any queries are issued, creating one AM per queue

2. Impala Server asks LLAMA for resources for a query



cloudera
Ask Bigger Questions

Gang scheduling

- YARN returns resources in a trickle, as they become available
- For MR this is perfect, as tasks are mostly independent (and checkpoint to disk)
- For low-latency queries, we require all resources to be available **at once** so that query tasks can stream results to one another
- Llama buffers resources between YARN and Impala to make resource requests appear **atomic** and **indivisible**

Framework-based resource enforcement

- In MR, YARN is responsible for enforcing memory and CPU resource constraints
- The only reasonable way to do this is to manage tasks at the **process** level
- YARN uses **cgroups** to control per-process resource usage
- One process <-> one cgroup
- YARN starts the process, and puts it in its cgroup

Single-process enforcement

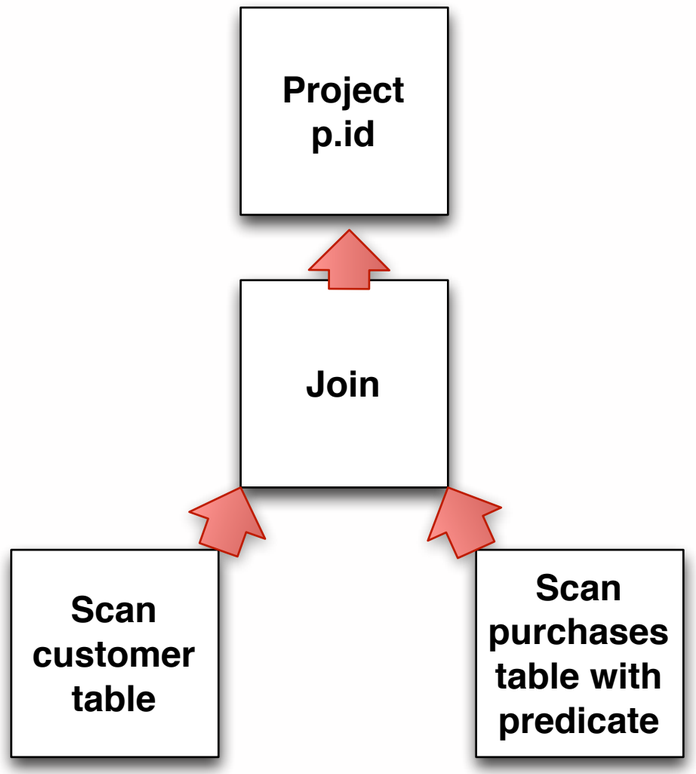
- **Problem:** that presumes that one process belongs to exactly one job
- This does not hold for Impala, which runs all queries in the same process, reusing threads for multiple queries
- YARN still wants a process to manage
- So we trick it! We tell YARN to run a process that does nothing, and Impala uses the resources granted to it

Resource Estimation

- Parts of Impala were extensively re-tooled to support integration with YARN via LLAMA
- Most interesting change: **resource estimation**
- Impala has to compute how much CPU and memory each query is likely to require
  - Too little: query can't run
  - Too large: resources are wasted

Resource estimation

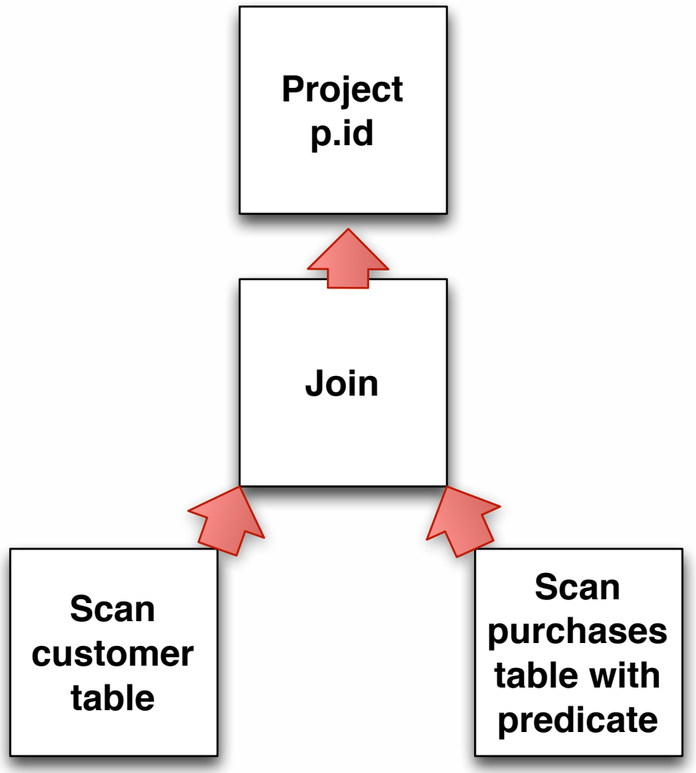
```
select p.id from customer c join purchases p on
    c.id = p.cid AND p.date > "01/01/2013"
```



cloudera
Ask Bigger Questions

Resource estimation

```
select p.id from customer c join purchases p on
    c.id = p.cid AND p.date > "01/01/2013"
```

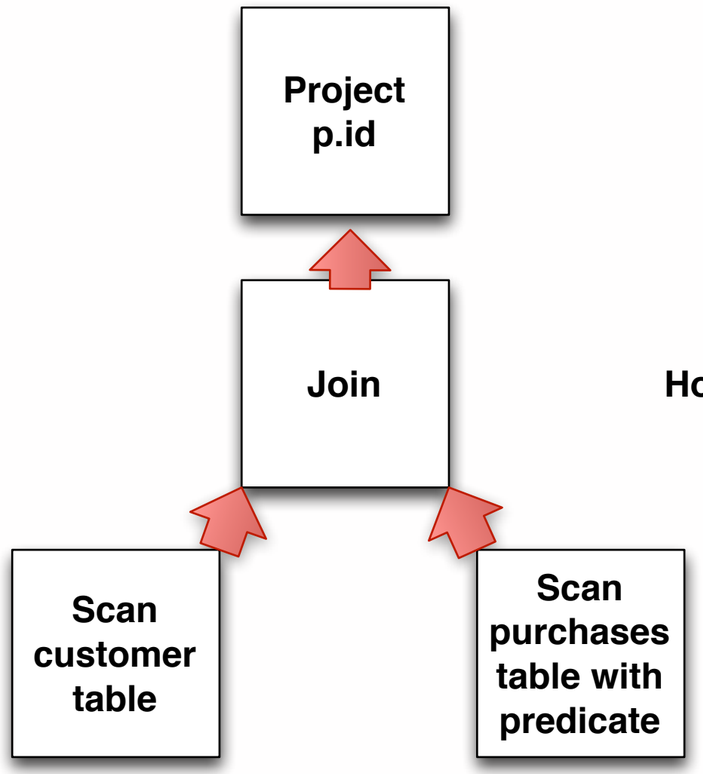


How many rows are expected to match the predicate?

cloudera
Ask Bigger Questions

Resource estimation

```
select p.id from customer c join purchases p on
    c.id = p.cid AND p.date > "01/01/2013"
```



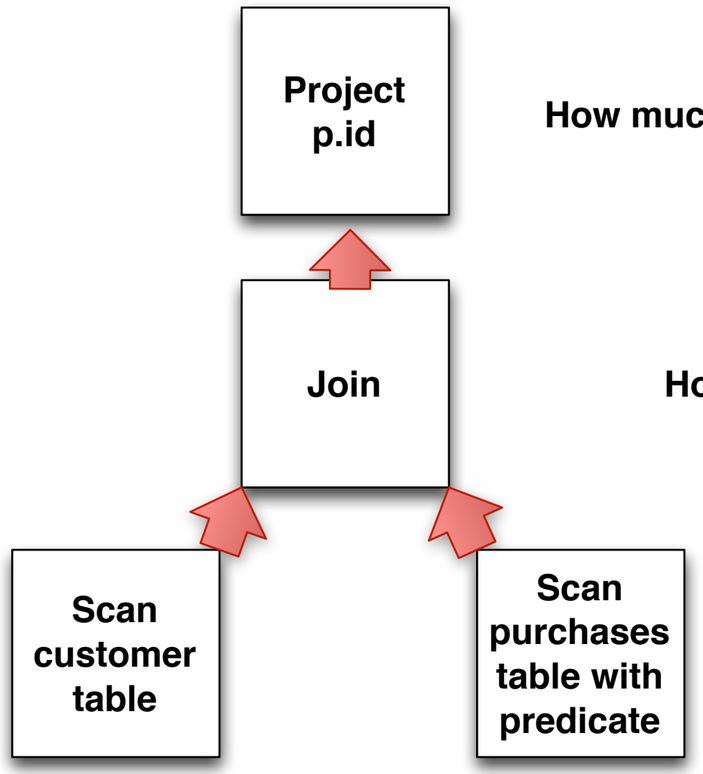
How big does the hash-table need to be?

How many rows are expected to match the predicate?

cloudera
Ask Bigger Questions

Resource estimation

```
select p.id from customer c join purchases p on
    c.id = p.cid AND p.date > "01/01/2013"
```



How much memory is required for the ID column?

How big does the hash-table need to be?

How many rows are expected to match the predicate?

cloudera
Ask Bigger Questions

Available Today!

- Cloudera's **CDH5 Beta 1** contains preview support for everything we've seen up until now:
  - LLAMA demon
  - YARN support inside Impala
  - Resource estimation algorithms
- Make sure you have computed **statistics** for your tables, otherwise resource estimation will be wrong
  - Instructions on the Cloudera website

2. Future Work

Better, Bigger, Faster, More!

Cutting the cost of resource management

- Long-lived application masters cut down a lot of the overhead of integrating with YARN
- Resource **acquisition** still requires a round-trip to a central server, and is unlikely to scale to our most aggressive latency and throughput requirements
- For CDH5 GA our focus is on performance, and supporting our customers with the most demanding workloads

Long-lived Containers

- **Idea**: amortise the cost of obtaining a container across many queries.
- Instead of yielding a container after the query finishes, hold on to it for a short time.
- If another query can use it, we already have saved 50% of the resource acquisition cost.
- YARN still enforces the fairness of this approach under load

Routing queries to containers

- How do queries know where the containers are, without asking YARN?
- Impala already has a broadcast mechanism that it uses for table metadata, called the **statestore**
- Each Impala server receives a list of containers and their current usage. New queries get routed to the unused queries first.

Growing and shrinking allocations

- The long-lived container approach works for queries that fit exactly into existing containers
  - Which rarely happens without wastage
- Instead, Impala servers should grow or shrink allocations with demand in small container increments
- Requires running one query in several containers at once
- With some **cgroups magic(tm)** this is possible
- See YARN-1197 for some discussion of a similar approach

Speculative Execution

- **Idea:** Do we even need containers?
- Many queries are small, and take up few resources
- Many cluster don’t always run at 100% capacity
- Let’s run queries in the unused space until it’s needed
- Principle: ask **forgiveness** rather than **permission**
- If the queries are running too long, protect them by **asynchronously** acquiring resources

What we've covered

- YARN is the **resource manager for all Hadoop frameworks**
- Cloudera Impala places new requirements on that crucial subsystem
- We have adapted YARN to help support **low-latency**, **interactive** workloads **alongside traditional batch processing**
- Future work focuses on further improving performance

Further reading

- YARN-1284, YARN-1274, YARN-1253, YARN-1010, YARN-1144, YARN-1137, YARN-1049, YARN-910, YARN-1008, YARN-789, YARN-937, YARN-392 YARN-1321, 1343, YARN-624, YARN-1290, YARN-392, YARN-521…
- http://cloudera.github.io/llama/

Don’t miss!

- **Parquet: An Open Columnar Storage Format for Hadoop**
  - October 29th (**Today**) - 1.45pm, Gramercy Suite
- **Practical Performance Analysis and Tuning for Cloudera Impala**
  - October 30th (**Tomorrow**) - 2:35pm, Murray Hill Suite

Thank you! Questions?

Henry Robinson / henry@cloudera.com / @HenryR


cloudera®
Ask Bigger Questions