Re-optimizing Data Parallel Computing

Sameer Agarwal
Srikanth Kandula, Nicolas Bruno, Ming-Chuan Wu, Ion Stoica, Jingren Zhou

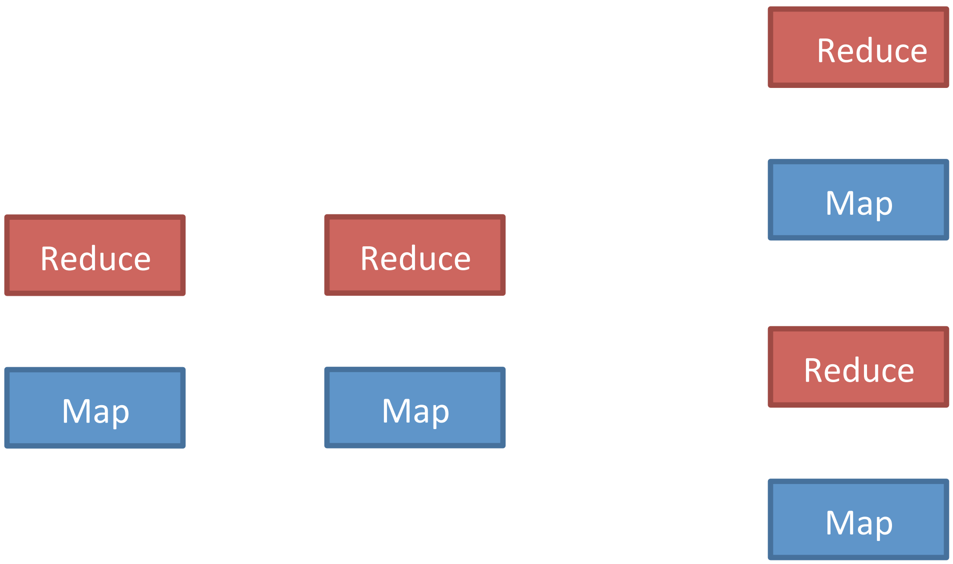
Microsoft Research
bing
amplab UC Berkeley

A Data Parallel Job…

can be a collection of maps,

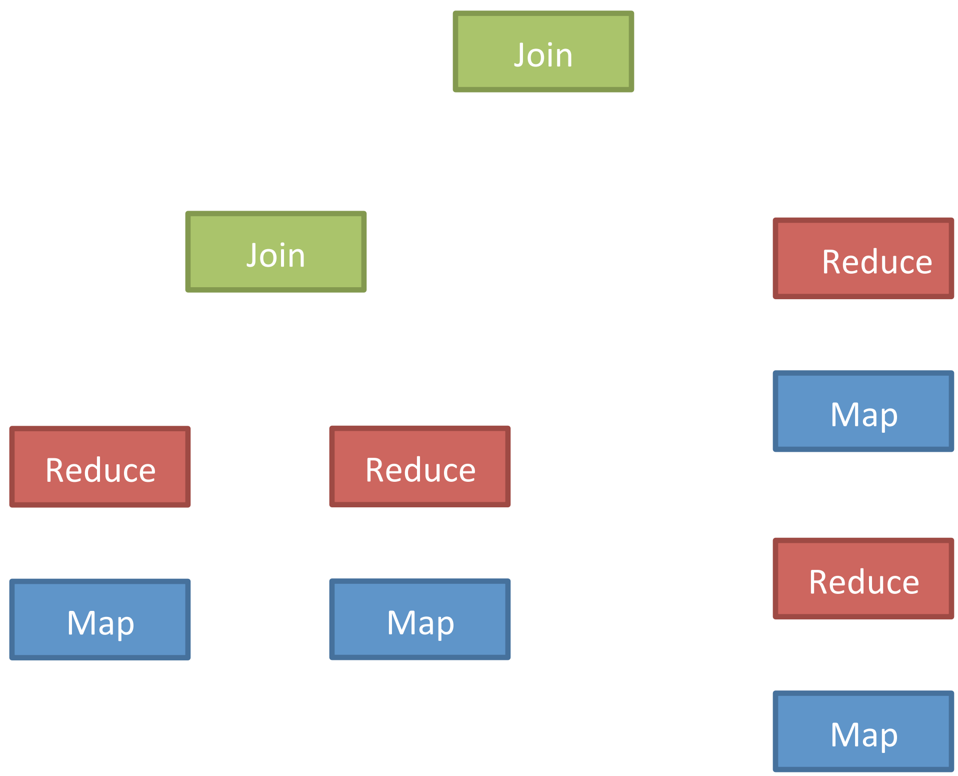
A Data Parallel Job…

can be a collection of maps, reduces



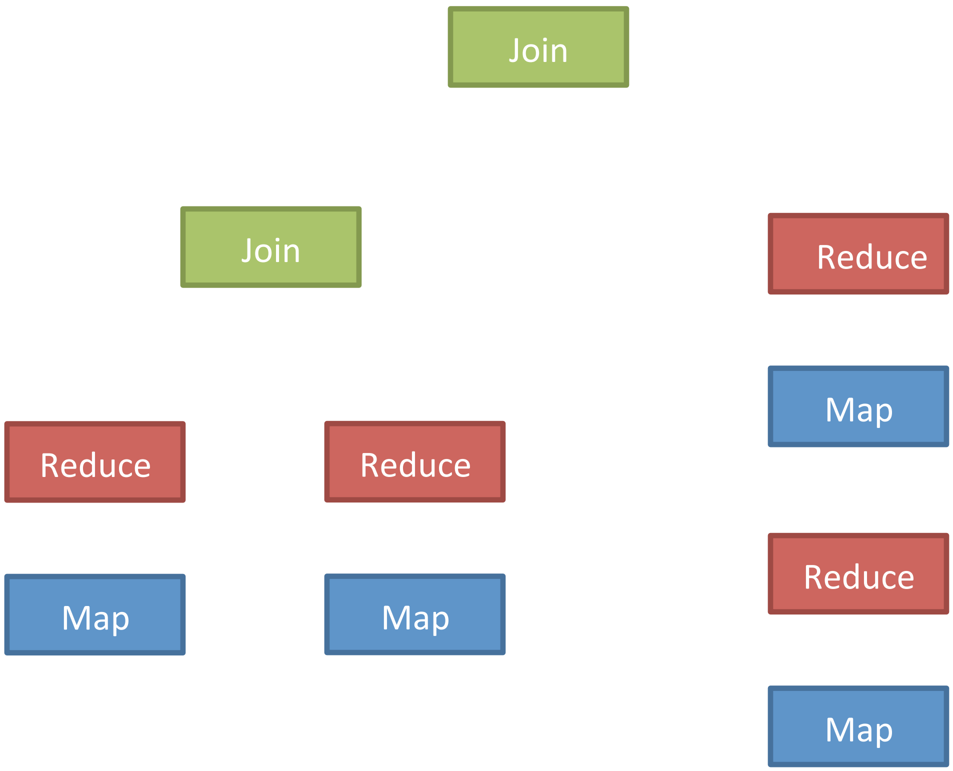
A Data Parallel Job…

can be a collection of maps, reduces, joins

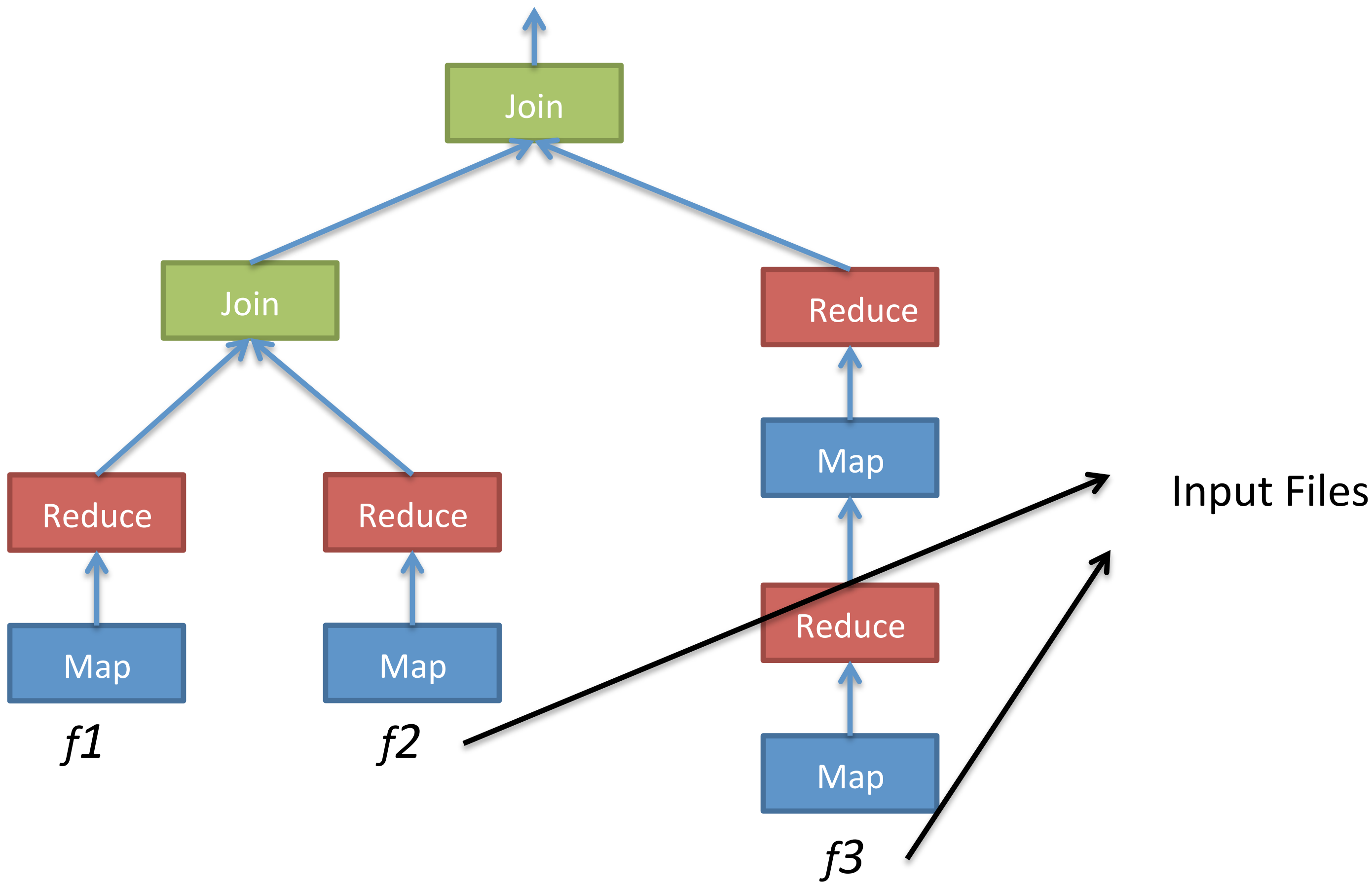


A Data Parallel Job…

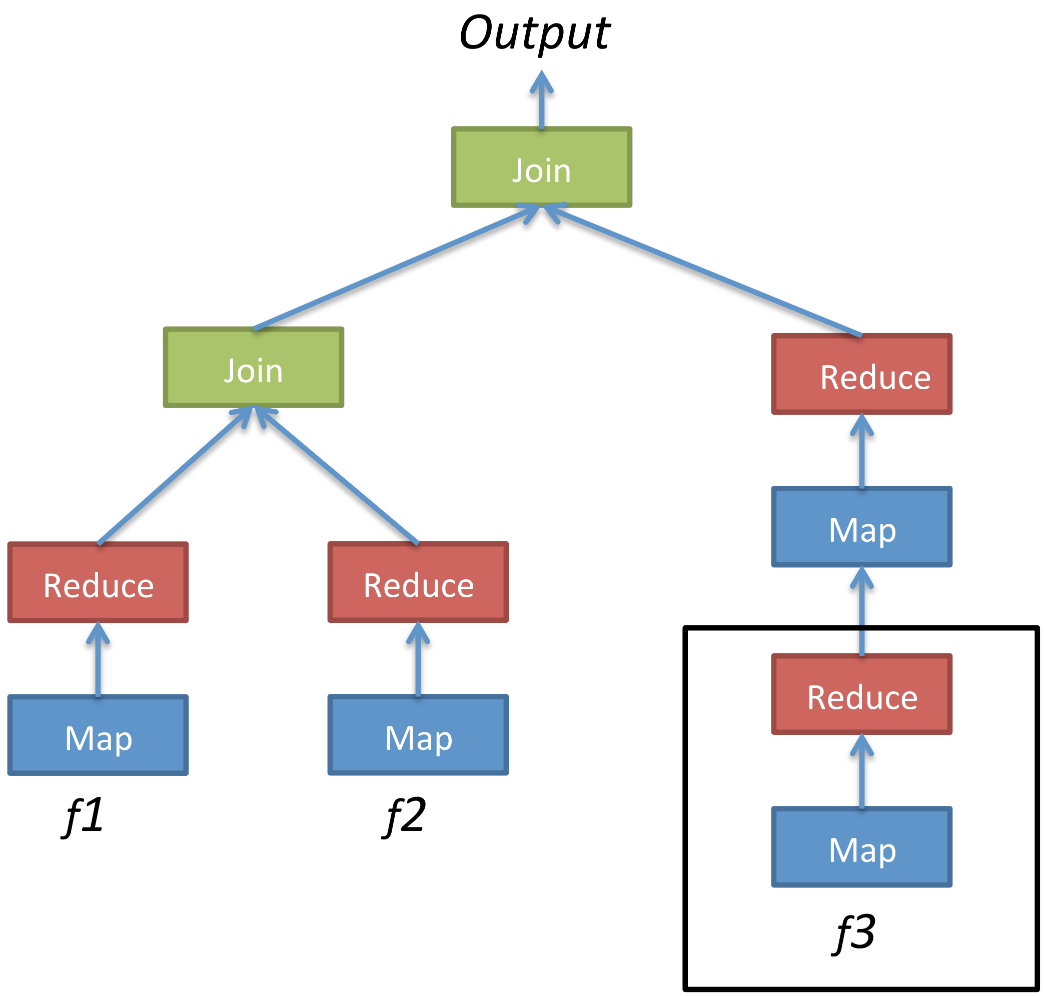
… and other framework dependent operations



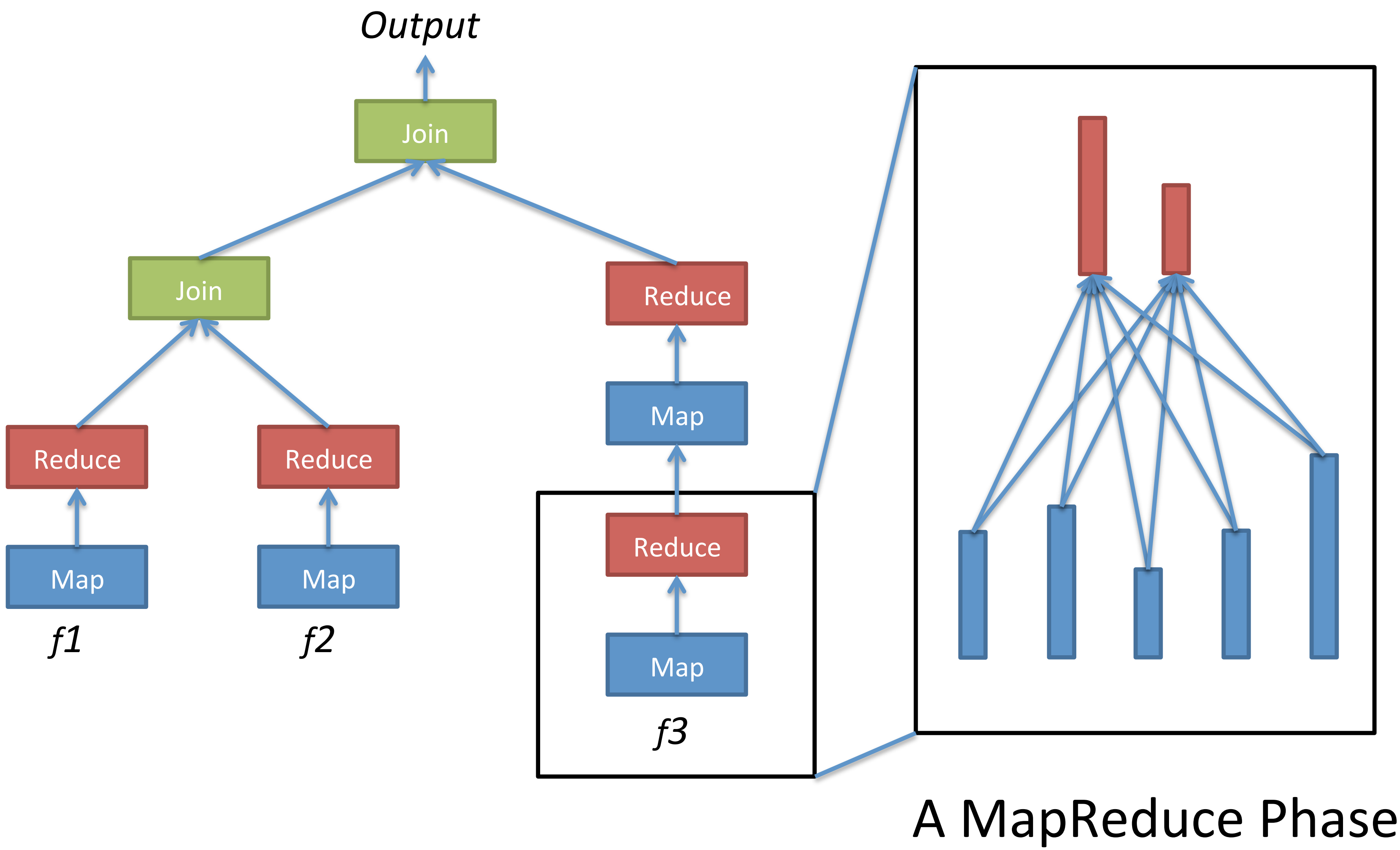
A Data Parallel Job…



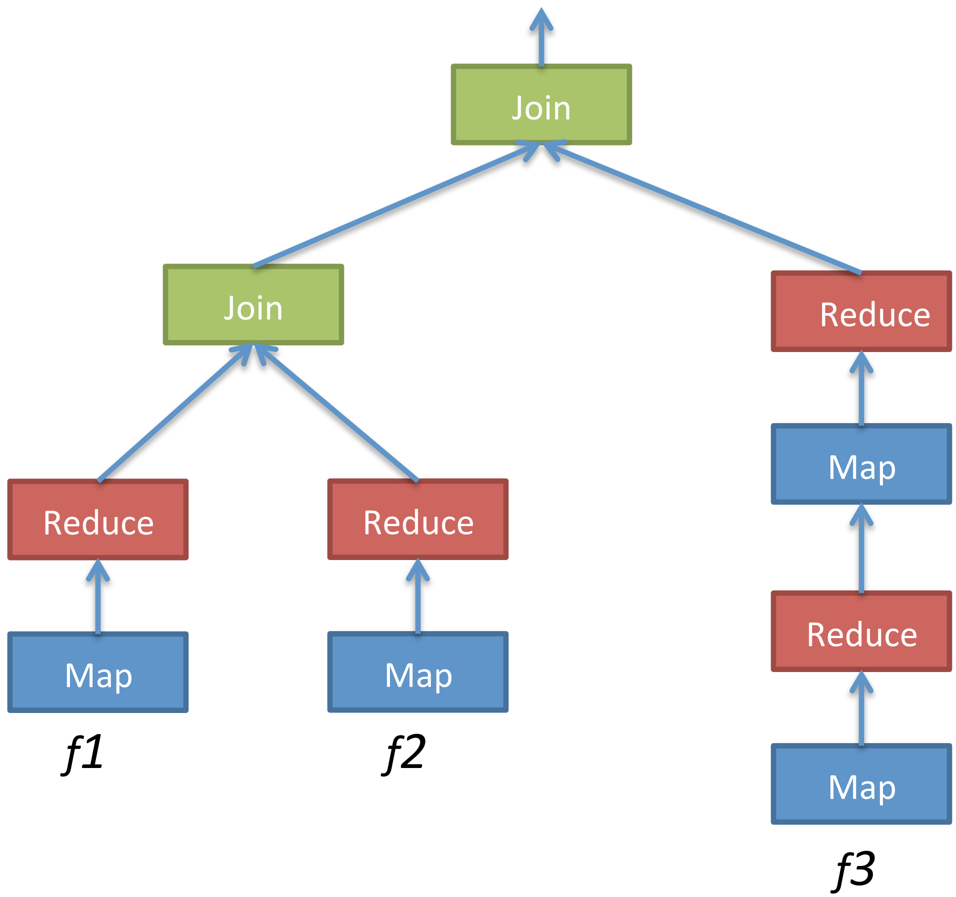
A Data Parallel Job…



A Data Parallel Job…



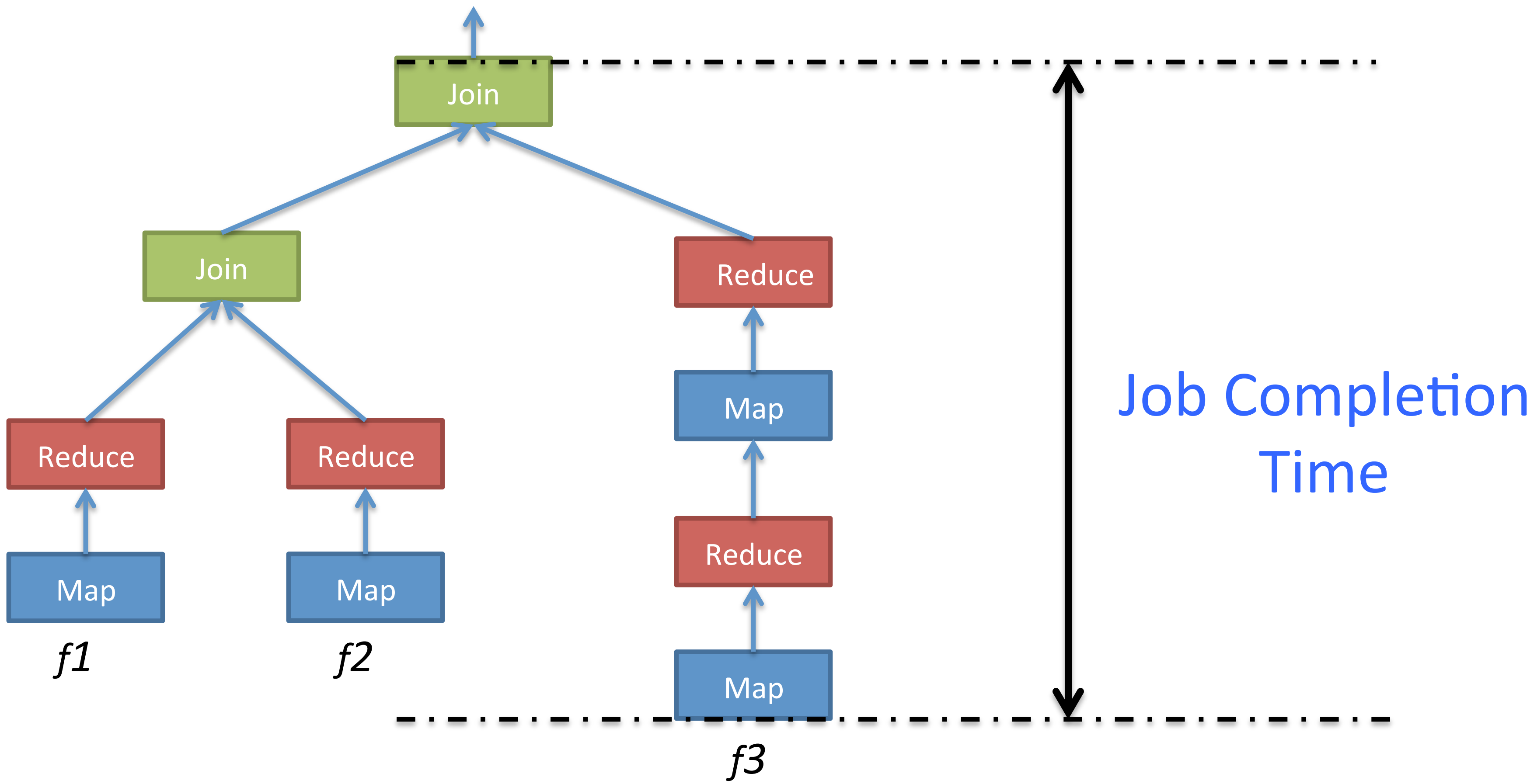
A Data Parallel Job…



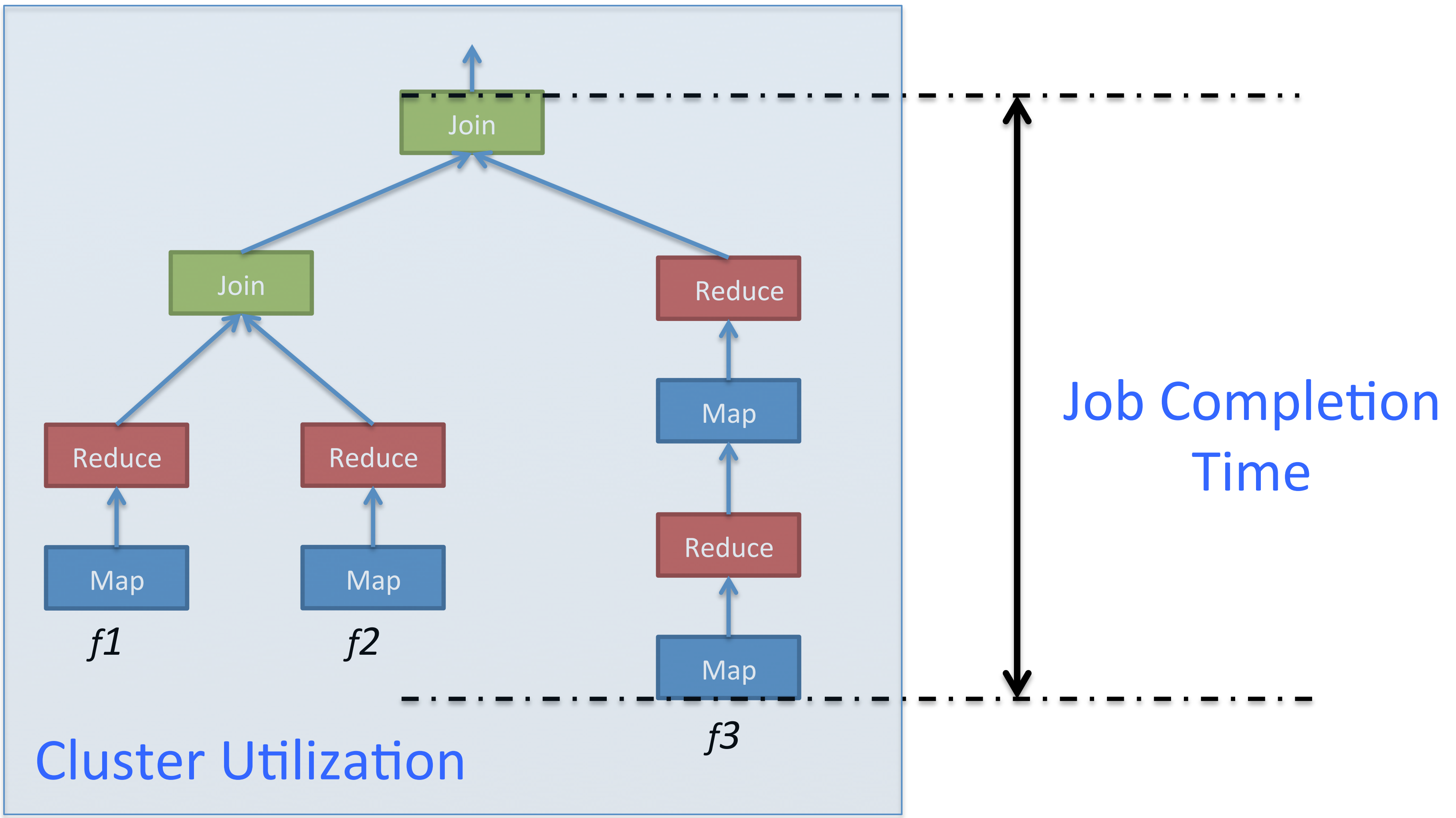
Runs on Large Clusters

Batched Jobs (Dryad/Hadoop)

Key Metrics

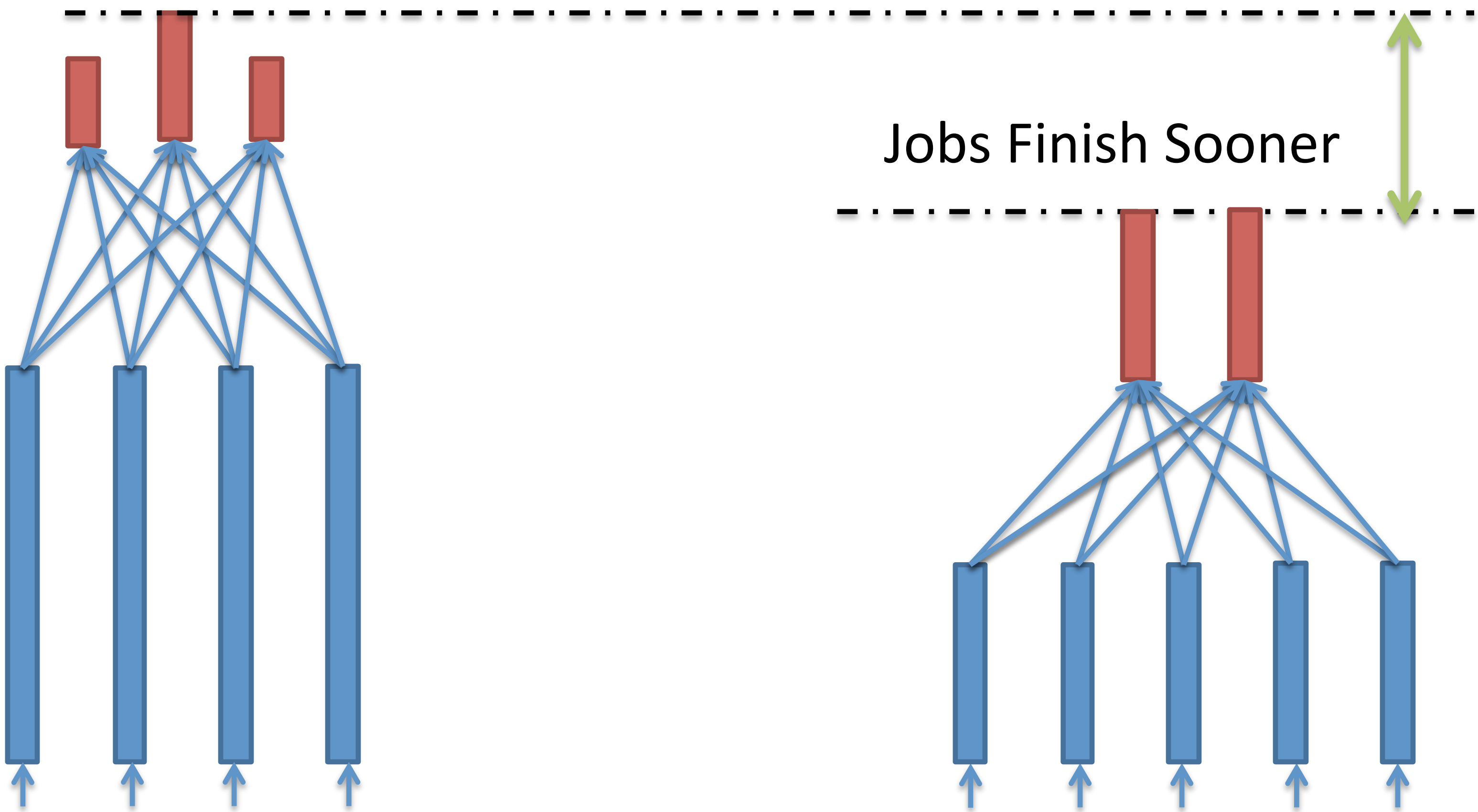

Join
Join
Reduce
Reduce
Reduce
Map
Map
Map
Reduce
Map
f1
f2
f3
Job Completion
Time

Key Metrics



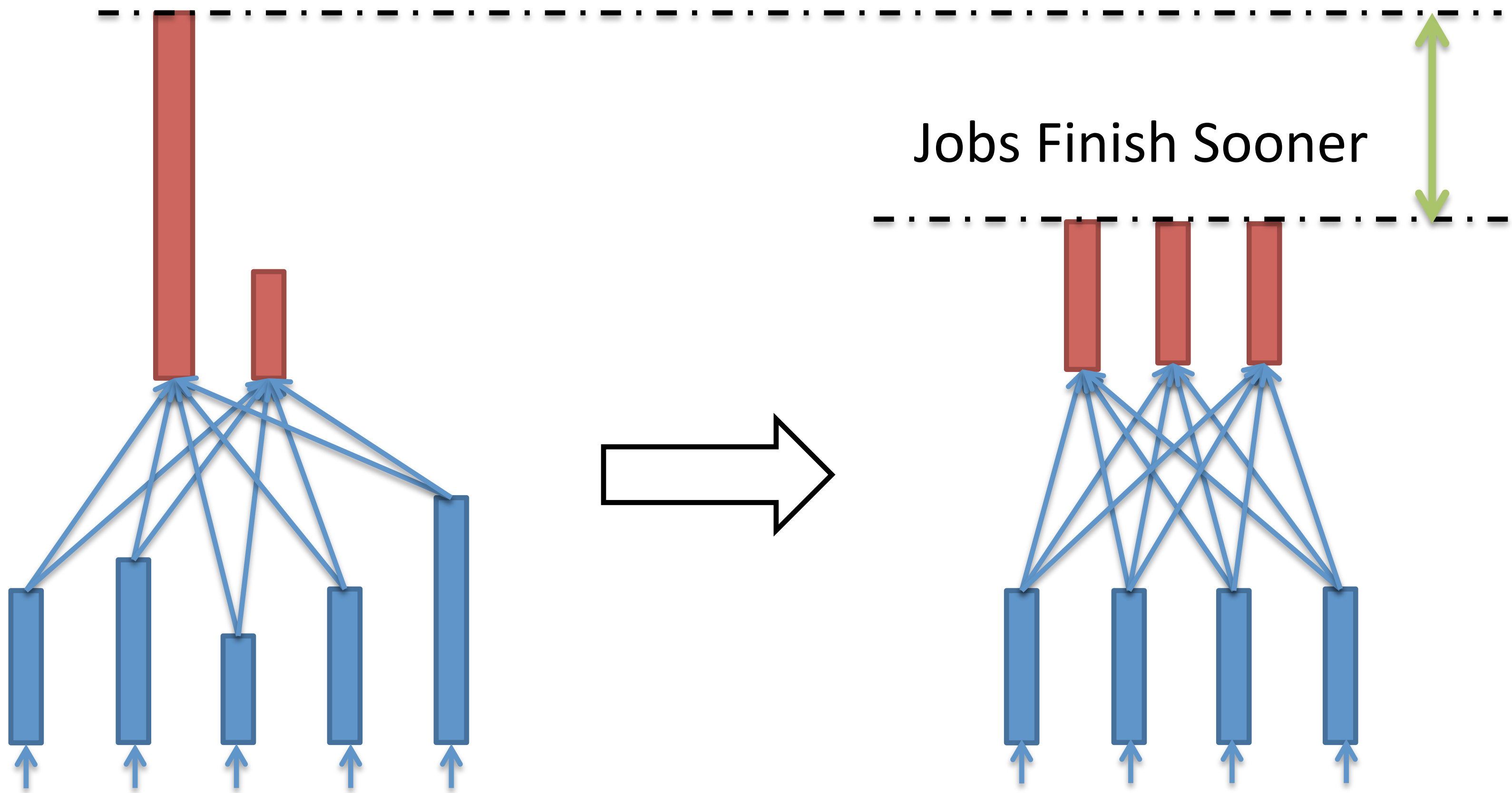
Join
Join
Reduce
Map
Reduce
Reduce
Map
Map
Reduce
Map
f1
f2
f3
Cluster Utilization
Job Completion Time

Imbalance



Different Work is being done at Every Stage (Data + Complexity)

Data Skews



Data is not independent and uniformly distributed

These problems are real!

Imbalance ~ Avg. Task time per Stage

Most stages take < 10 seconds

Top 4% (1%) take > 100 (1000) seconds

These problems are real!

$$\text{Skew} = \frac{\text{Maximum Partition Size}}{\text{Average Partition Size}}$$

Most stages have skew < 2

But, 5% have > 10

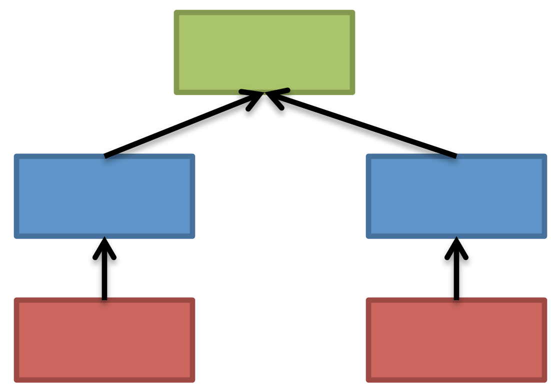
These problems are real!

Others in paper…

Optimizing the Job Execution

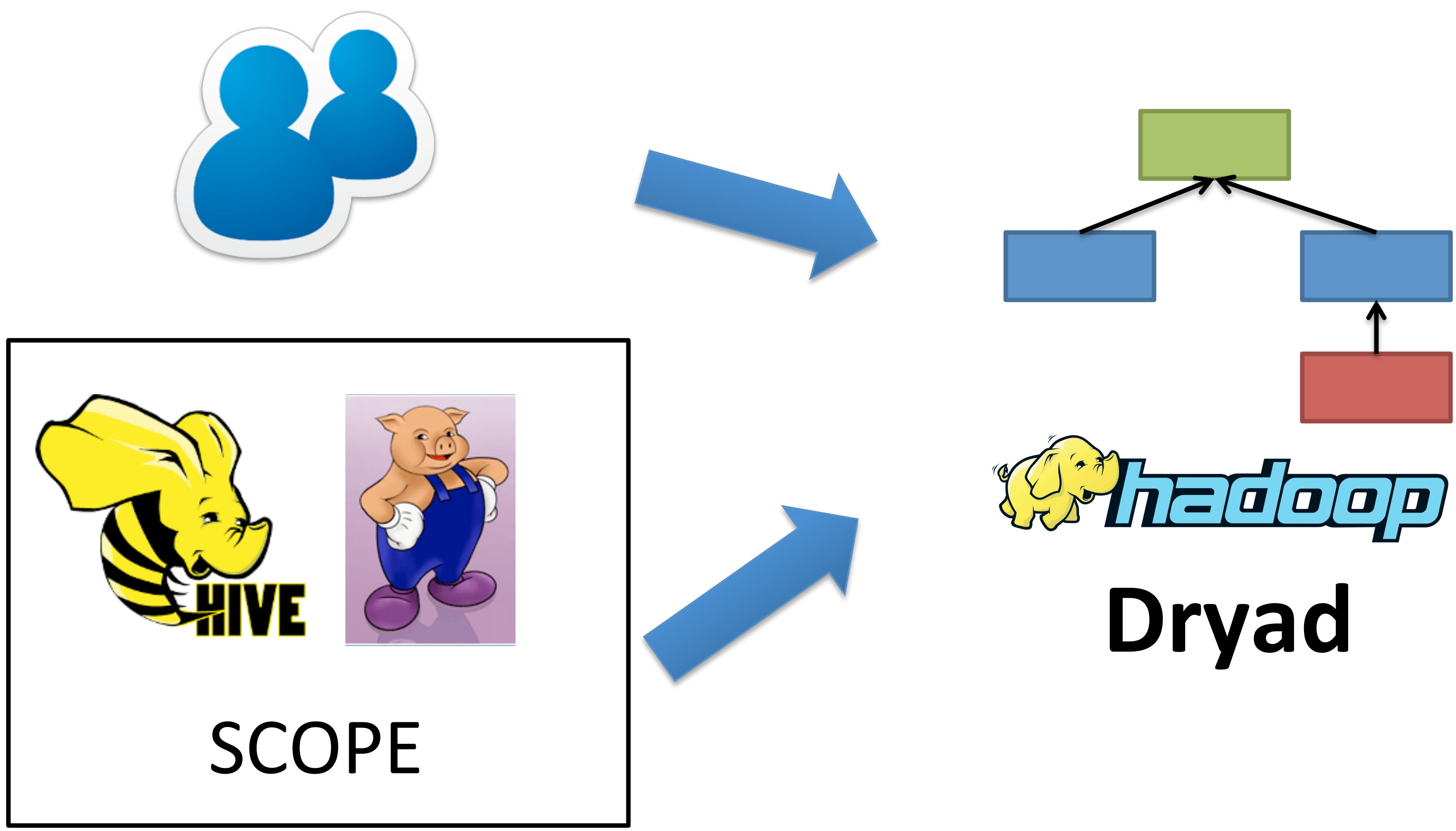
To optimize the completion time and cluster utilization, ideally you need to control:

- Parallelism
- Partition Sizes
- Operator Sequence
- Operator Implementation



The *Execution Plan* lets you control these knobs

Who generates execution plans?



High Level Abstractions

Hadoop	Hive	Pig/Scope	Database

High Level Abstractions

	Hadoop	Hive	Pig/Scope	Database
Parallelism	Statically Configurable	Statically Configurable	Statically Configurable	Static

High Level Abstractions

	Hadoop	Hive	Pig/Scope	Database
Parallelism	Statically Configurable	Statically Configurable	Statically Configurable	Static
Data Partition	Statically Configurable	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*

High Level Abstractions

	Hadoop	Hive	Pig/Scope	Database
Parallelism	Statically Configurable	Statically Configurable	Statically Configurable	Static
Data Partition	Statically Configurable	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Implementation	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*

High Level Abstractions

	Hadoop	Hive	Pig/Scope	Database
Parallelism	Statically Configurable	Statically Configurable	Statically Configurable	Static
Data Partition	Statically Configurable	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Implementation	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Sequence	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*

High Level Abstractions

	Hadoop	Hive	Pig/Scope	Database
Parallelism	Statically Configurable	Statically Configurable	Statically Configurable	Static
Data Partition	Statically Configurable	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Implementation	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Sequence	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*

High Level Abstractions

	Hadoop	Hive	Pig/Scope	Database
Parallelism	Statically Configurable	Statically Configurable	Statically Configurable	Static
Data Partition	Statically Configurable	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Implementation	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*
Operator Sequence	N/A	Rule Based (Data Size)	Cost Based (Fixed Cost)	Cost Based*

Want automatic control of all four knobs

- Based on the data and computation
- Robust for User Defined Functions (UDFs)

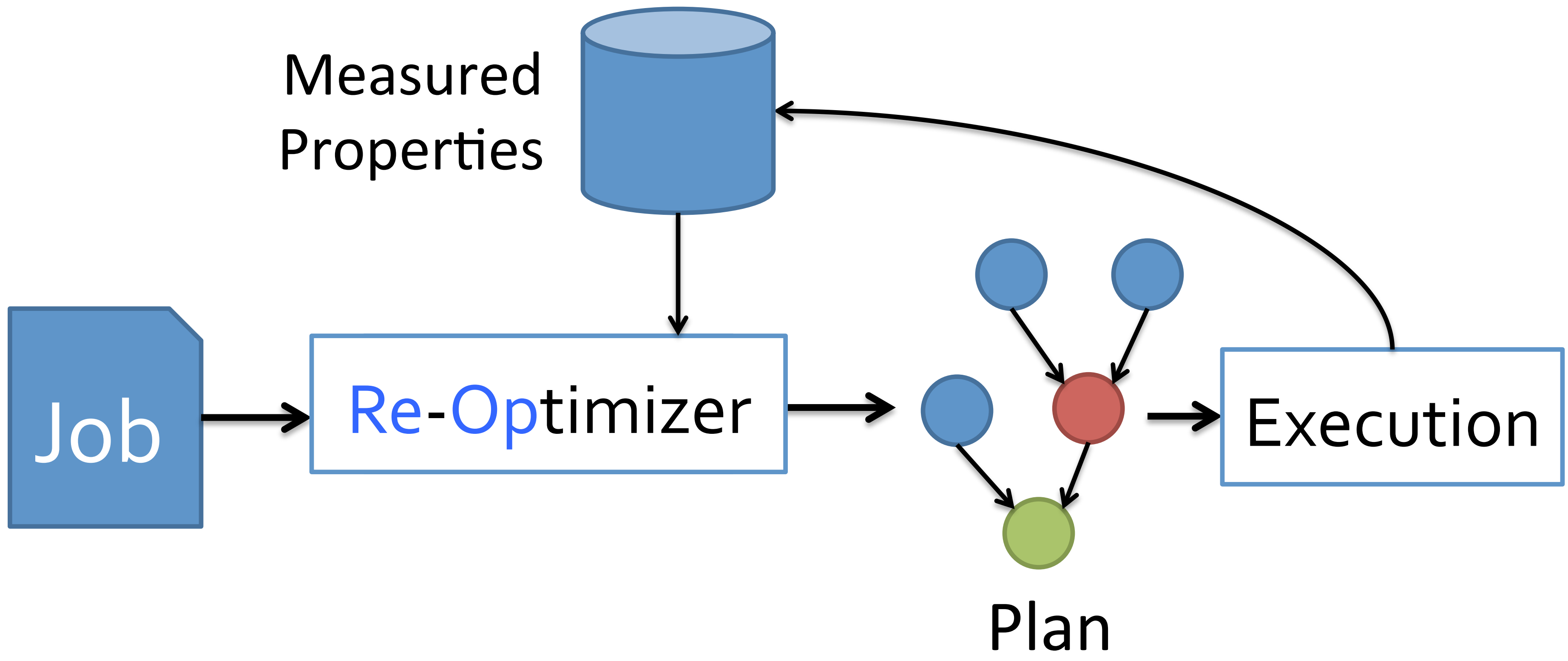
RoPE: Re-optimizer for Parallel Executions

Data & Computation Information

- Degree of Parallelism at every operation
- Data Partitions for each operation
- Implementations for each operation
- Sequence of operations

Cost-based query optimizer

+ information about code and data

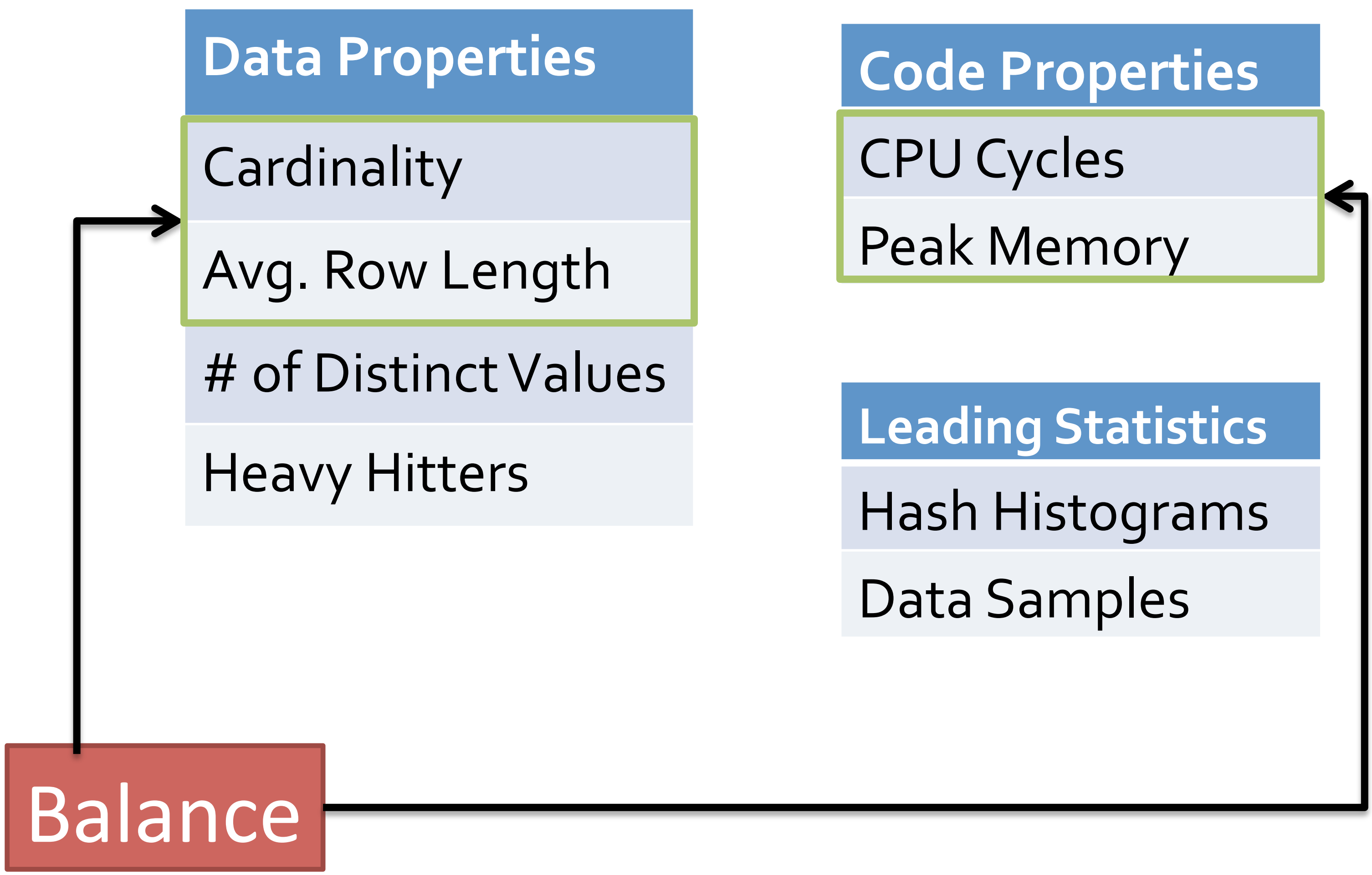


Usage Scenarios:

Better Execution Plans For

1. Recurring Jobs
   - Same “code” runs hourly on new data
   - Data properties are stable
2. “Similar” Jobs
   - e.g., jobs with identical parts
3. Future parts of this Job
   - e.g., after a barrier

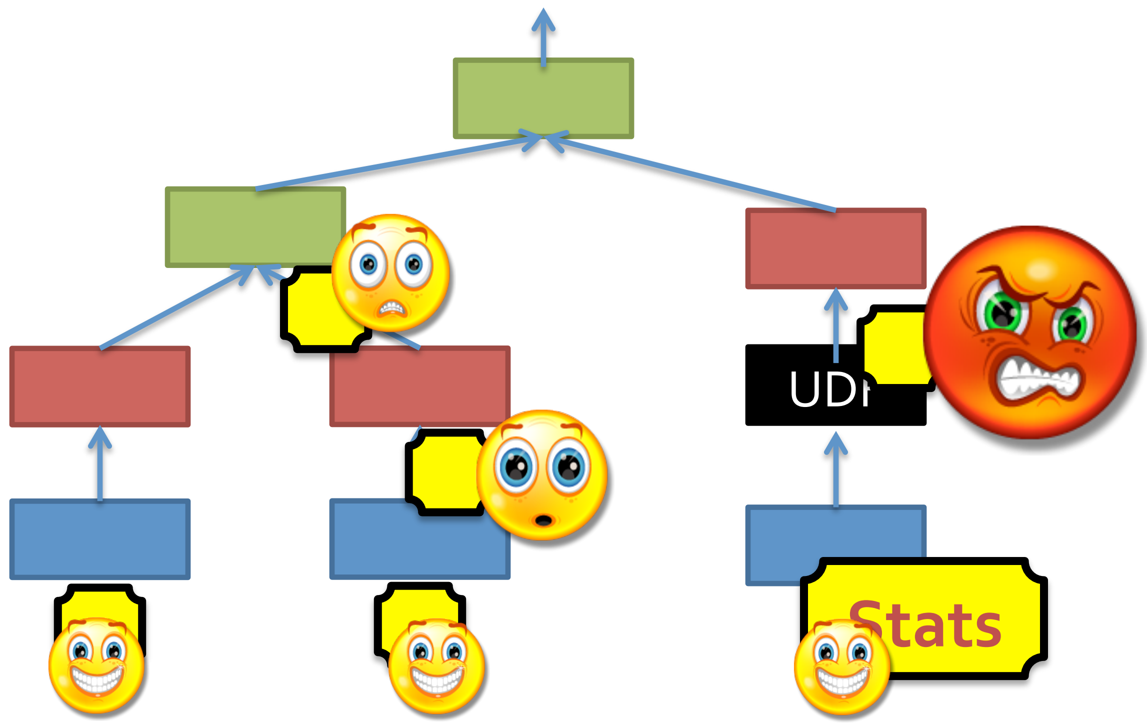
Information collected



Efficiently collecting this information is Challenging

Collecting Information

Option 1: Measure Input Properties, Propagate over operations



Collecting Information

Option 1: Measure Input Properties, Propagate over operations

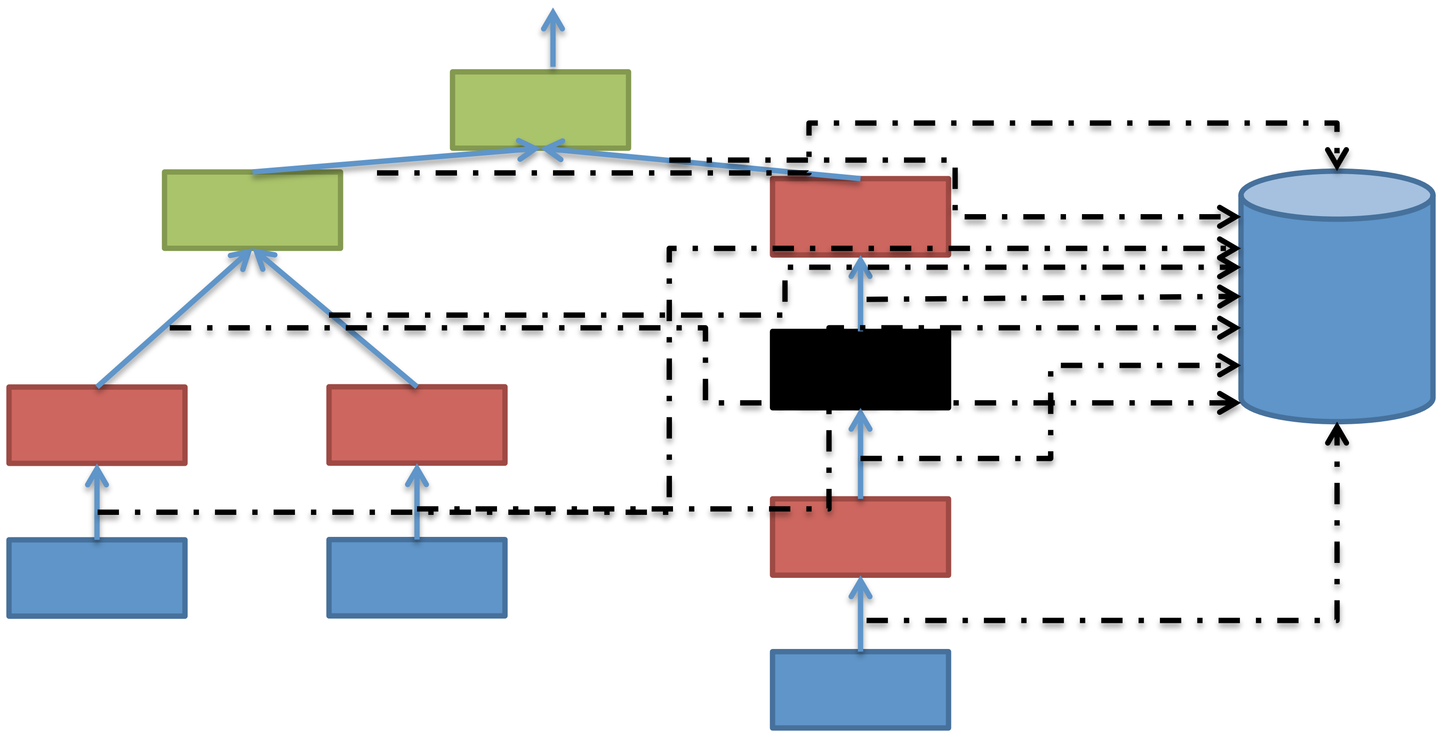
Low Overhead

Bad Accuracy

- Estimation error increases exponentially with #operations [Rio]
- Hard to reason about user defined functions

Collecting Information

Option 2: Store all intermediate data, analyze offline



Collecting Information

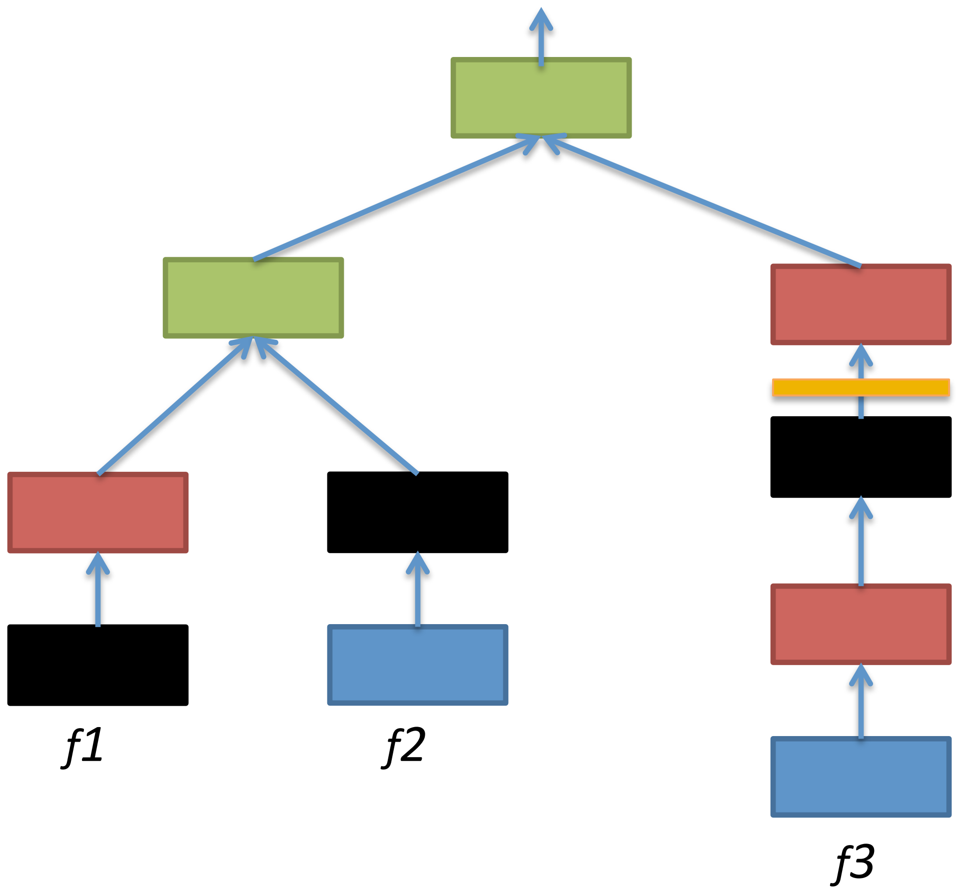
Option 2: Store all intermediate data, analyze offline

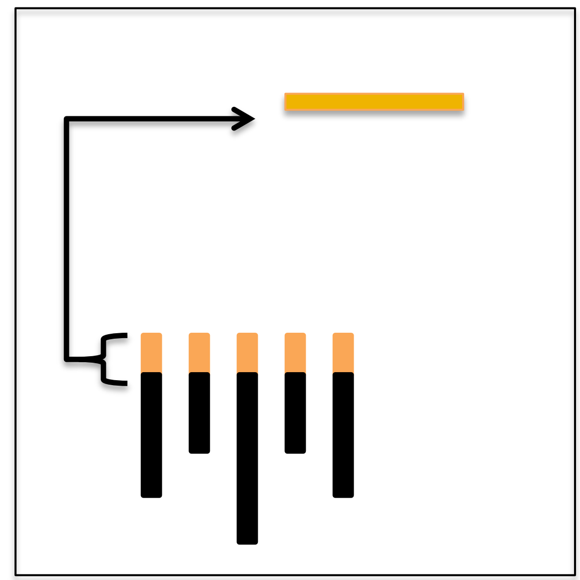
High Overhead

Good Accuracy

- Lots of Intermediate Data
- Job latencies and resource requirements went up ~10x

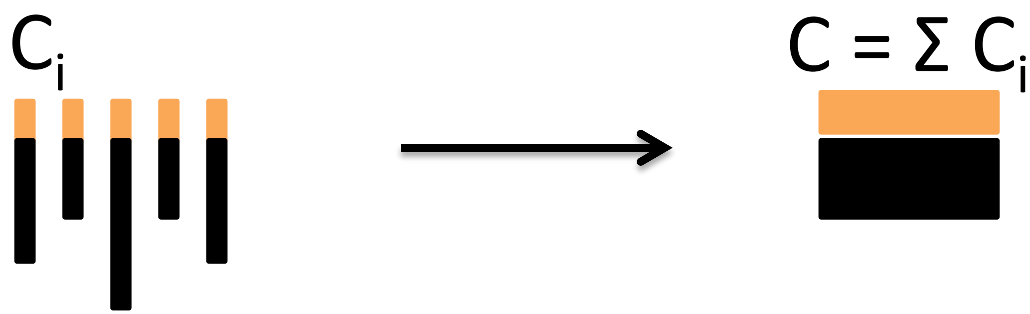
RoPE's Distributed Stats Collection





- Composable
- Light (single pass, sub-linear state)

Composing statistics with Resource constraints



- Trivial for some … e.g., cardinality
- Doable for some … e.g., heavy hitters, #distinct values
- Open problem for others … e.g., equi-width histograms

Related Work

Implementing Execution Plans Well

- Placing Tasks to maximize locality (Quincy, Delay Scheduling)
- Dealing with Outliers (LATE, Mantri)
- Orchestrating Network Transfers (Orchestra, Camdoop, SUDO)

RoPE finds better execution plans

Related Work

Re-optimization in Databases

- Single Machine/Short Queries (Kabra/Dewitt)
- Creating Robust plans given uncertainty in properties (Rio)
- Parametric Query optimization

RoPE reasons about Parallel Plans and deals with User-Defined Functions

Does it work?

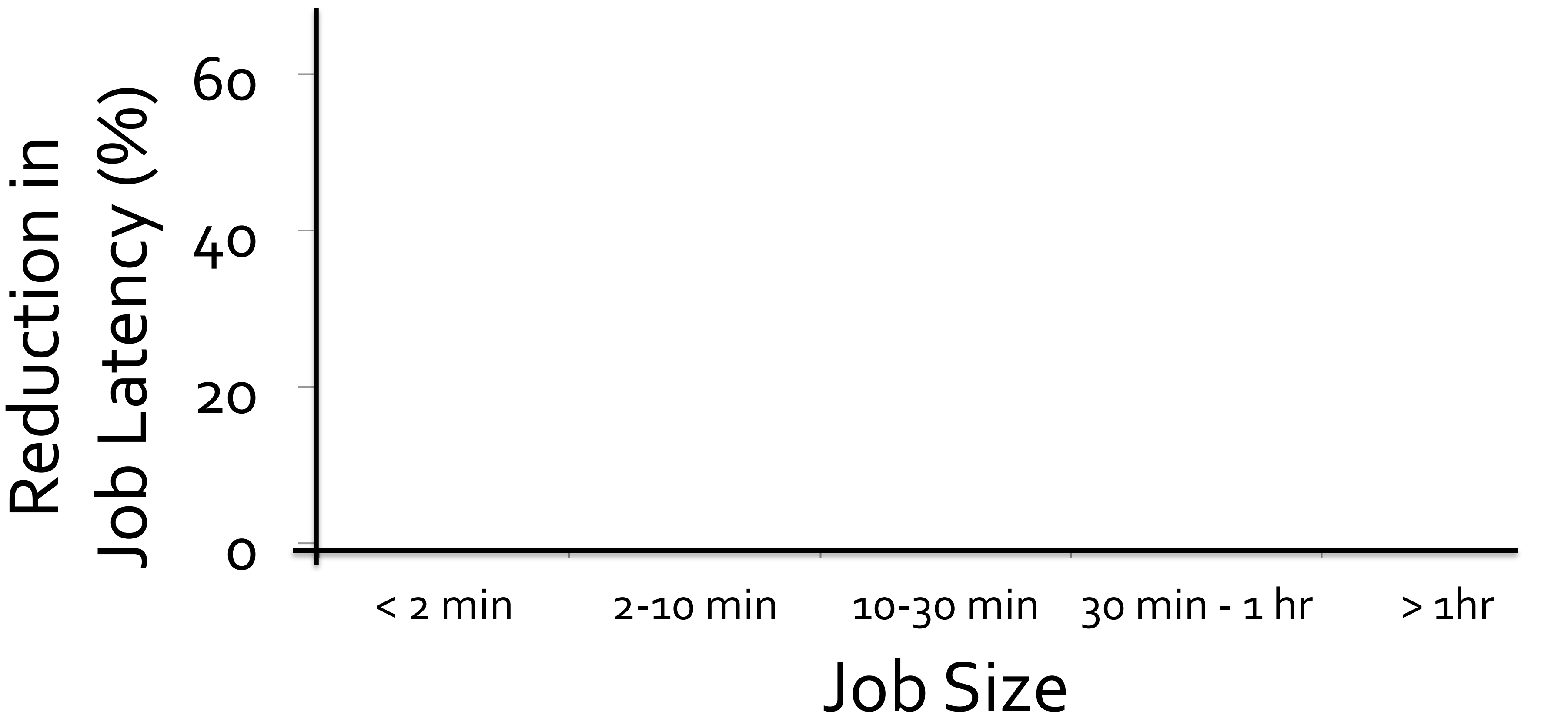
Evaluation

- We evaluated RoPE on a large production cluster
- Our workload suite consisted of hundreds of production jobs from a wide range of users during March 2012.
- **Baseline**: Production Scope (State-of-Art-QO)
- **Metrics**: Completion Time & Cluster Utilization

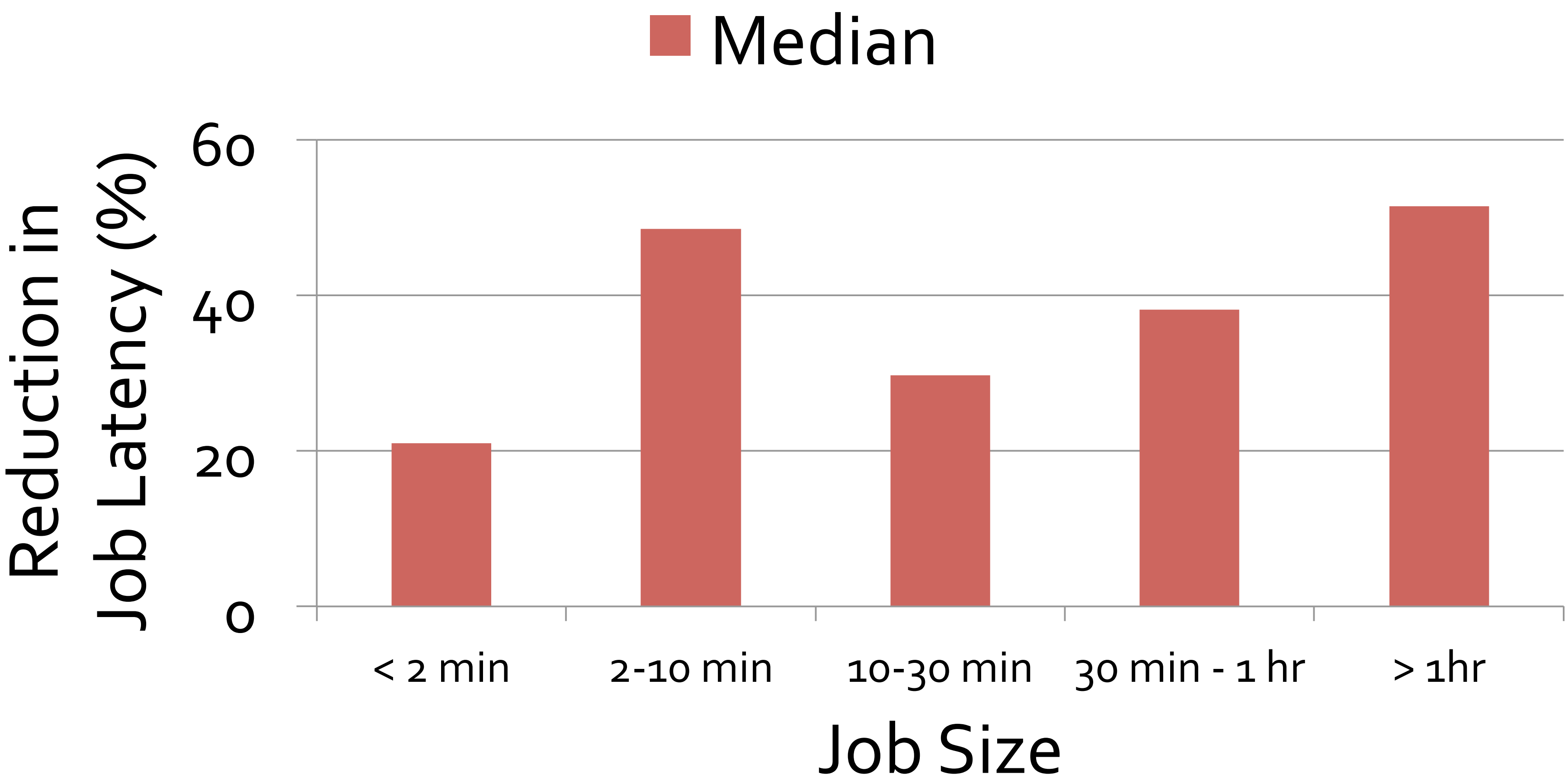
Highlights

- **2x** improvements across job latency for production jobs at **75th** percentile while using **1.5x** fewer resources
  - Better Execution Plans
  - Reduction in terabytes of Intermediate data and cross-rack shuffles
  - Better Parallelism
- 2-5% overhead incurred by our distributed statistics collection framework

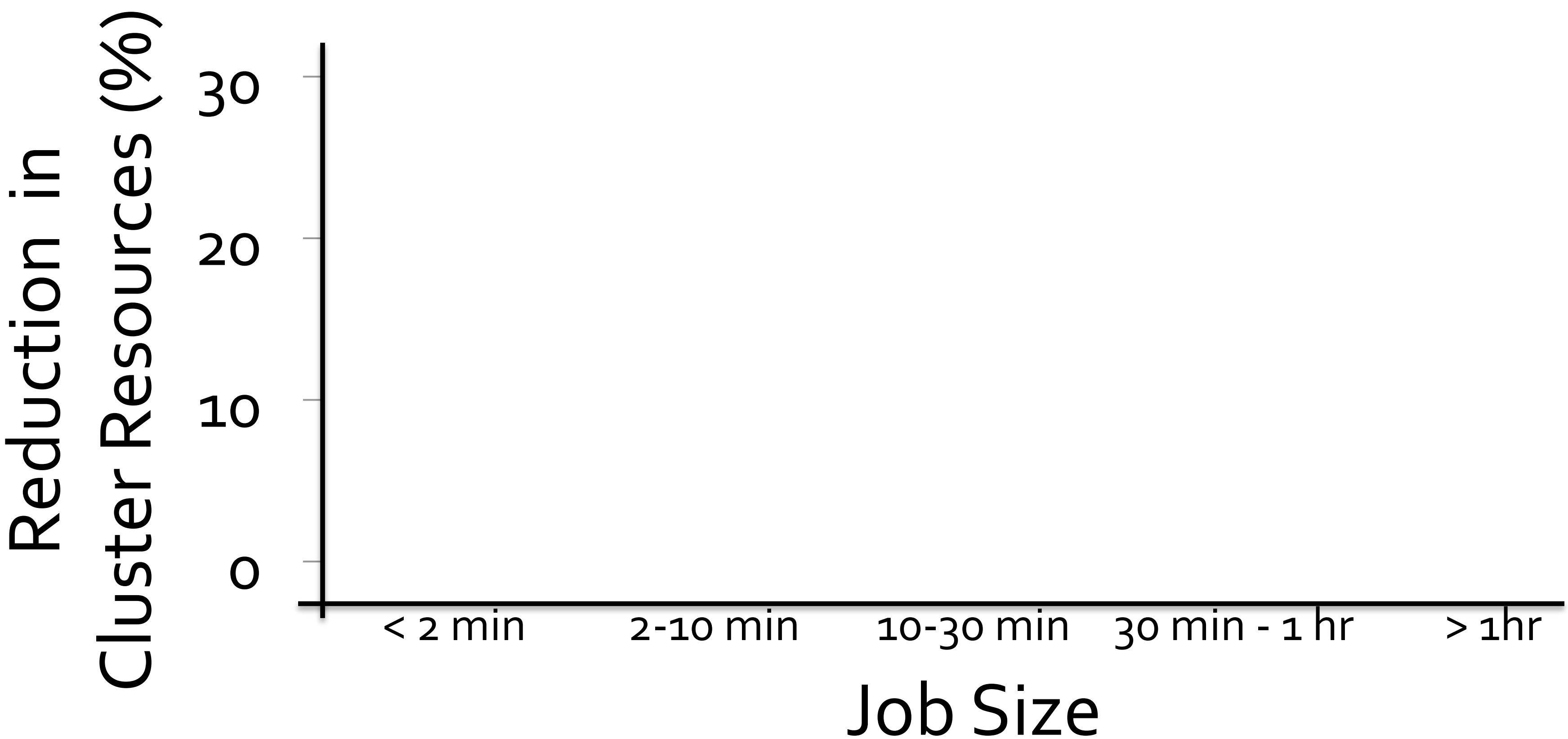
Job Completion Time



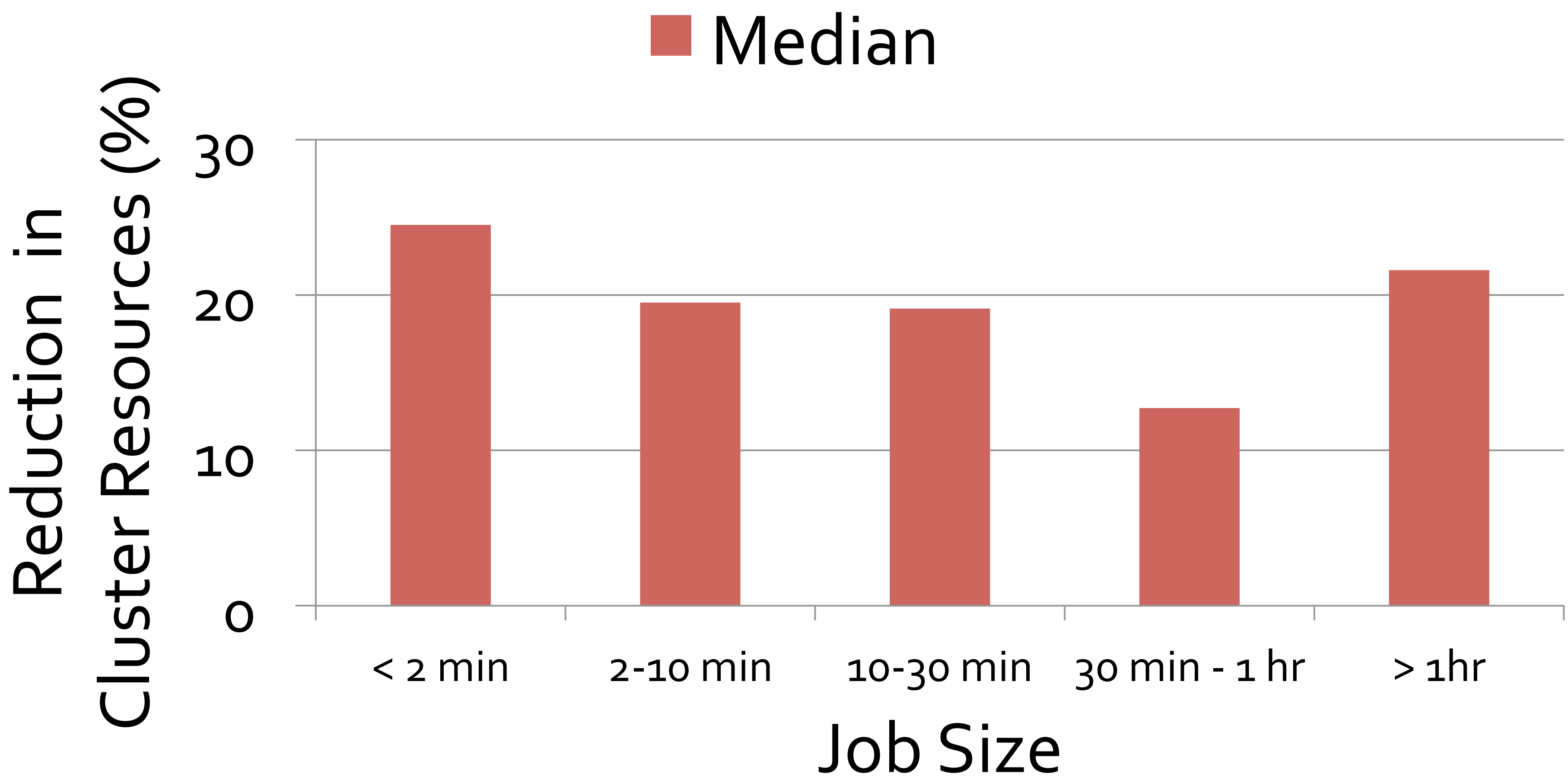
Job Completion Time



Cluster Utilization



Cluster Utilization



RoPE improves execution plans for data-parallel jobs

- Measurements identify novel problems and opportunities
- To leverage, built RoPE, a re-optimizer, that learns/ uses code- and data- properties
- Running live in Bing Production Clusters since December 2011

Thank you