Apache Nemo: A Framework for Building Distributed Dataflow Optimization Policies

Youngseok Yang[1] Jeongyoon Eo[1] Geon-Woo Kim[2] Joo Yeon Kim[3]
Sanha Lee[4] Jangho Seo[1] Won Wook Song[1] Byung-Gon Chun[1]

[1]Seoul National University [2]Viva Republica [3]Samsung Electronics [4]Naver Corp.

Execution of Distributed WordCount

Application

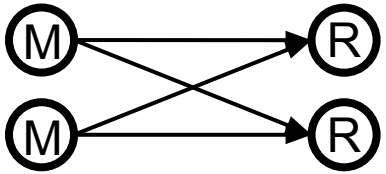
```
dataSet.map(word ⇒ (word, 1))
       .reduceByKey((l, r) ⇒ l + r)
```

Execution of Distributed WordCount

Application

```
dataSet.map(word ⇒ (word, 1))
       .reduceByKey((l, r) ⇒ l + r)
```

Compiler



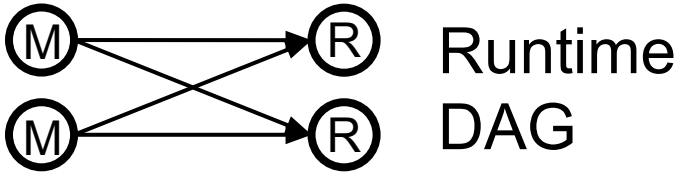
Runtime DAG

Execution of Distributed WordCount

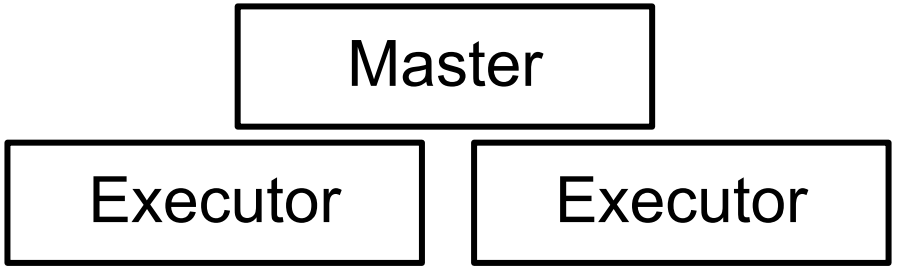
Application

```
dataSet.map(word ⇒ (word, 1))
       .reduceByKey((l, r) ⇒ l + r)
```

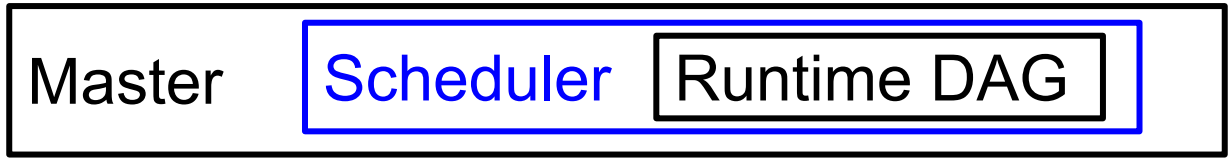
Compiler

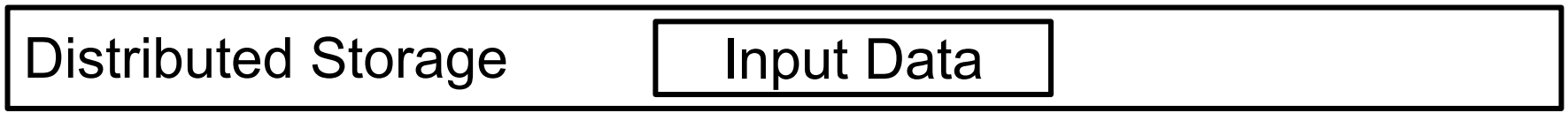


Runtime

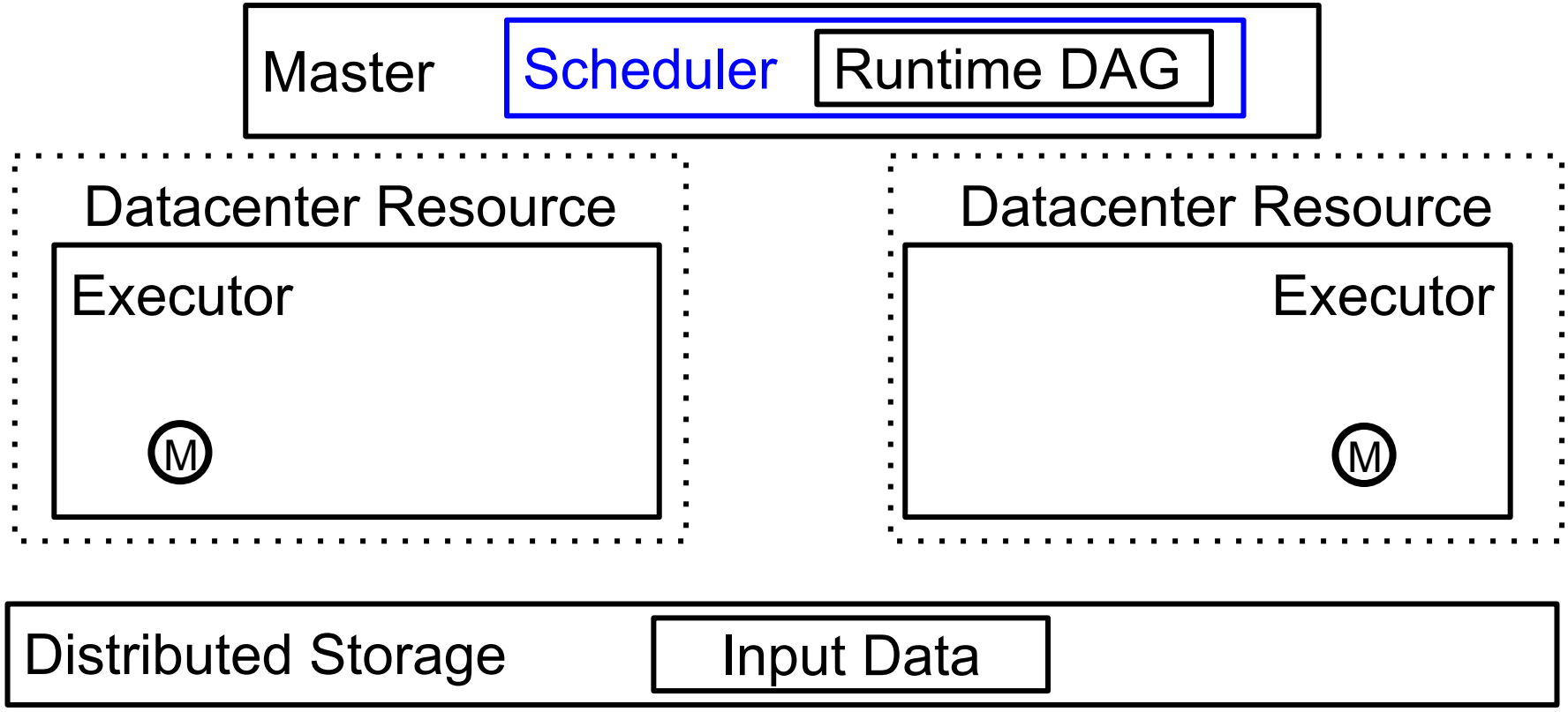


Execution of Distributed WordCount

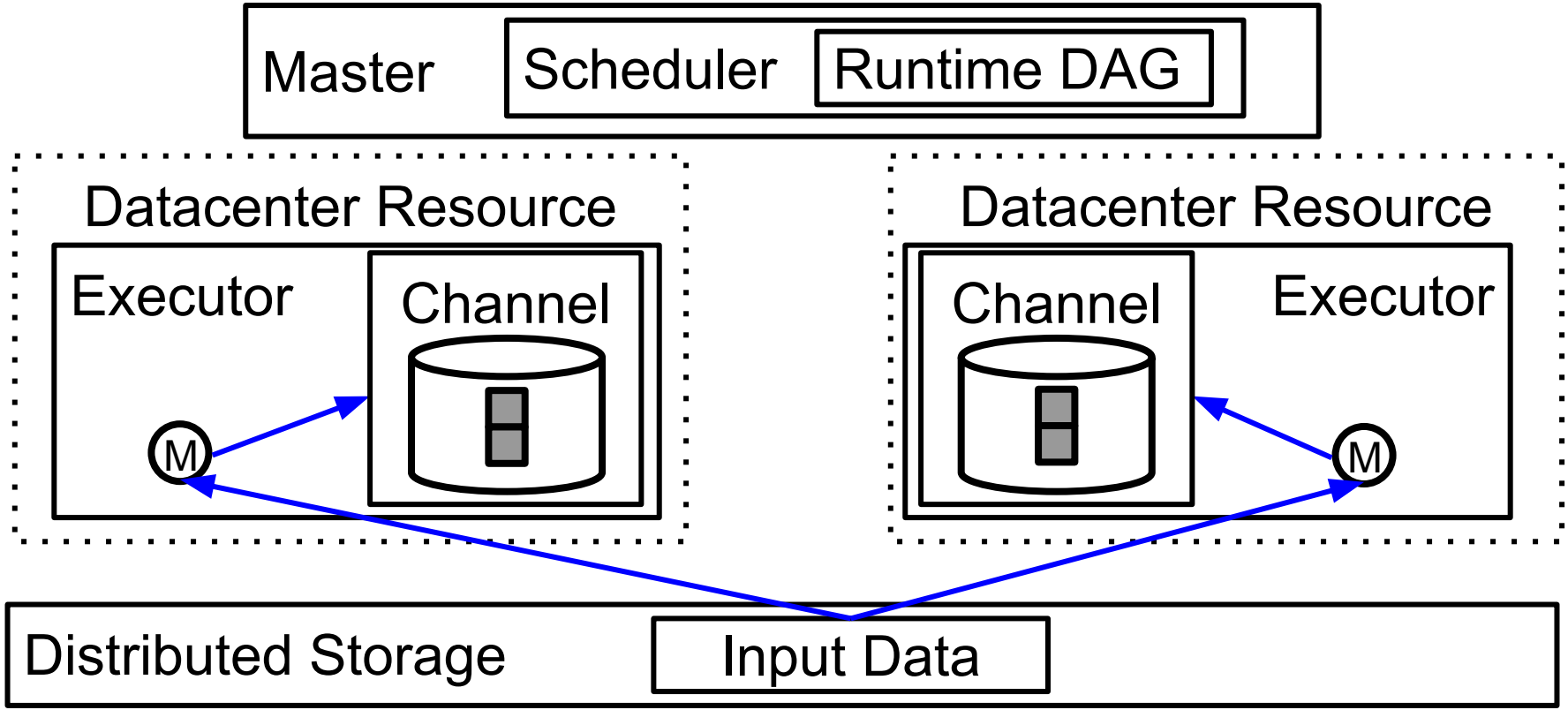




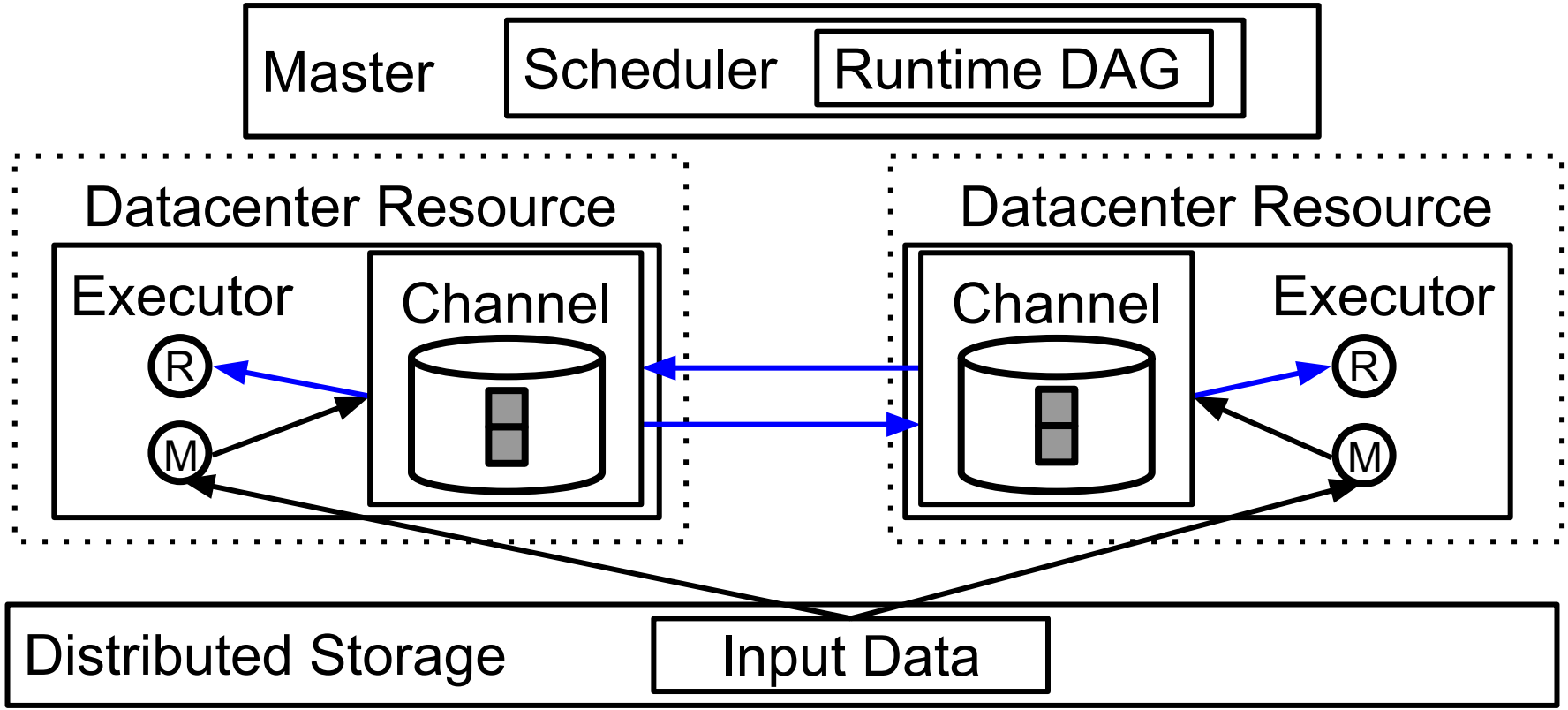
Execution of Distributed WordCount



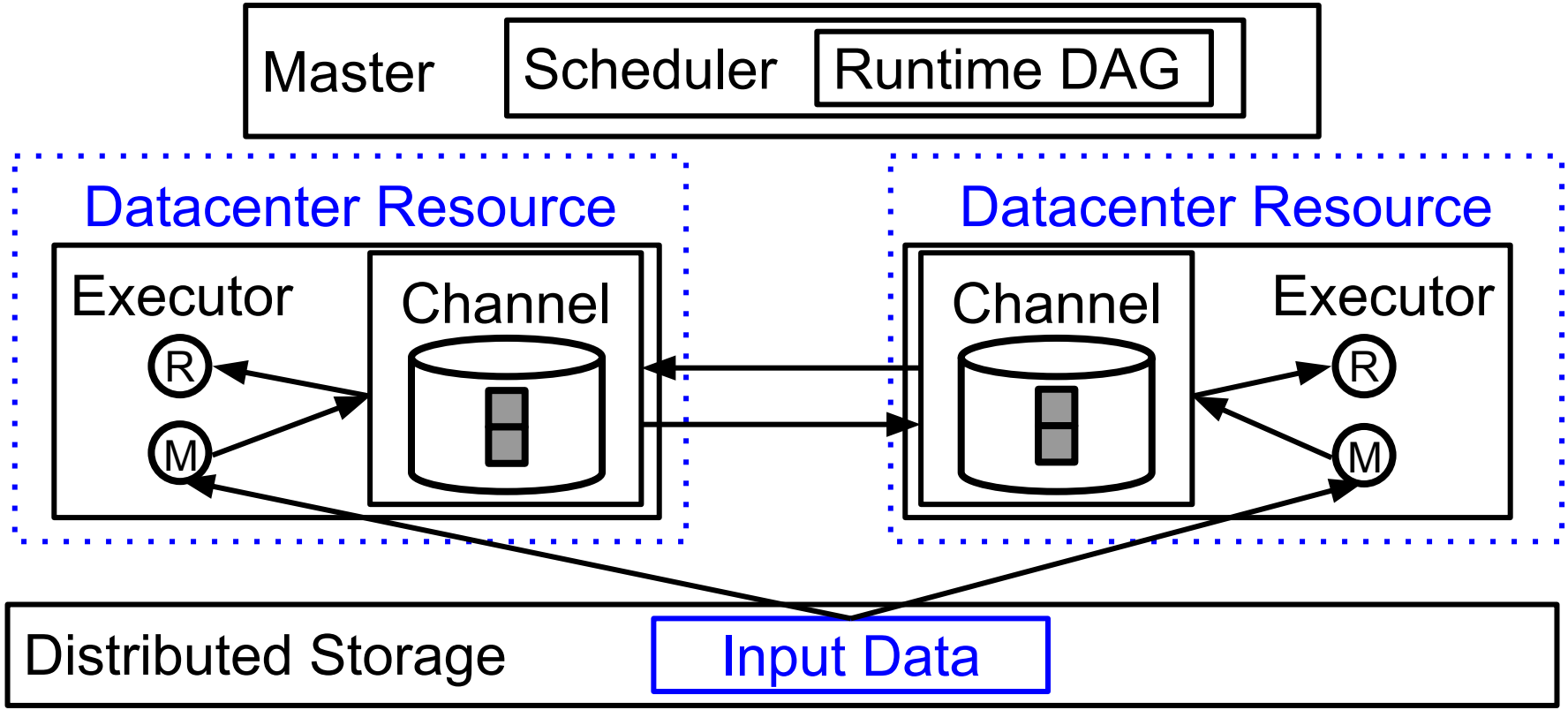
Execution of Distributed WordCount



Execution of Distributed WordCount



Our Observations: Resources & Data



Trend: Diverse Characteristics

Datacenter Resources	**Input Data**
Geographically -distributed	Large-scale
Cheap transient	Skewed

In This Talk (See Paper for Others)

Datacenter Resources

Geographically
-distributed

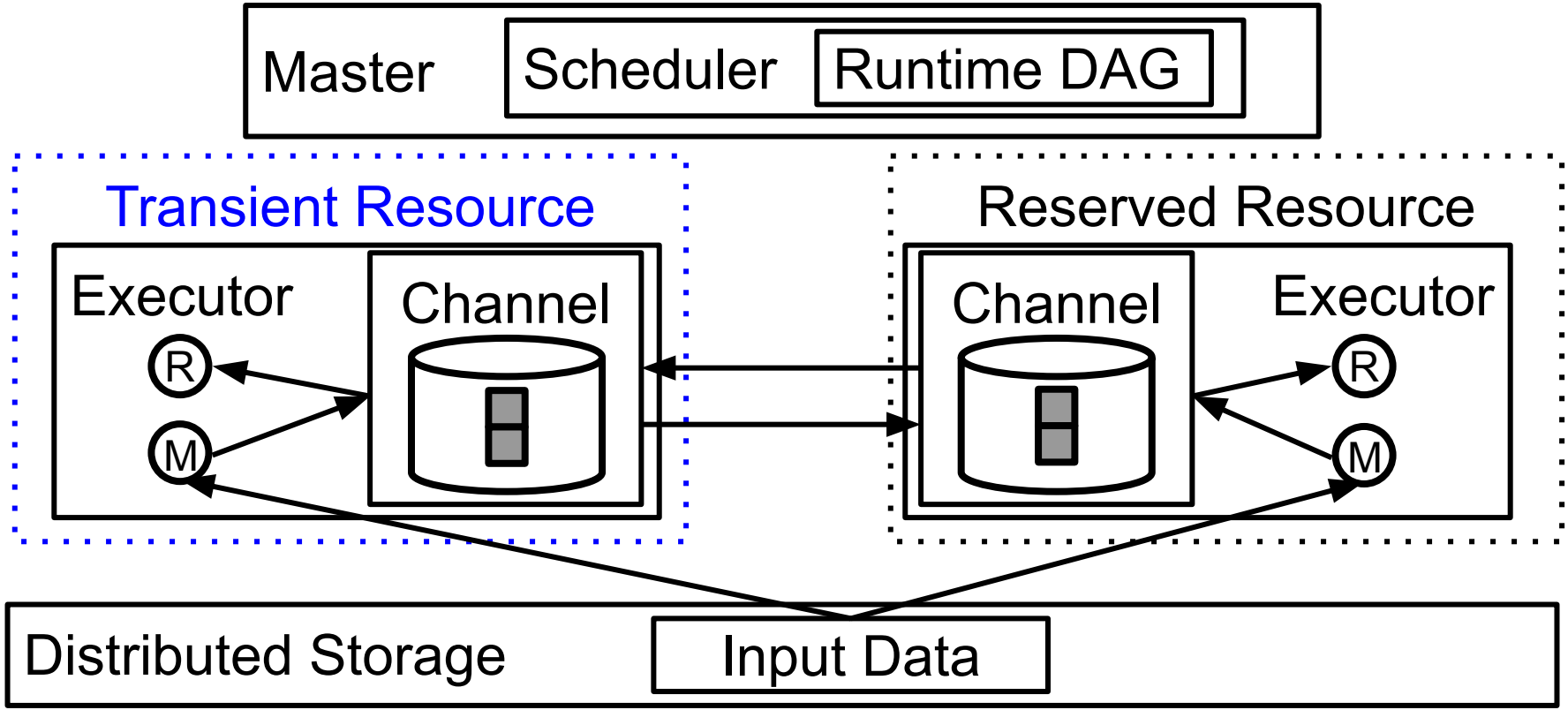
(1) Cheap transient

Input Data

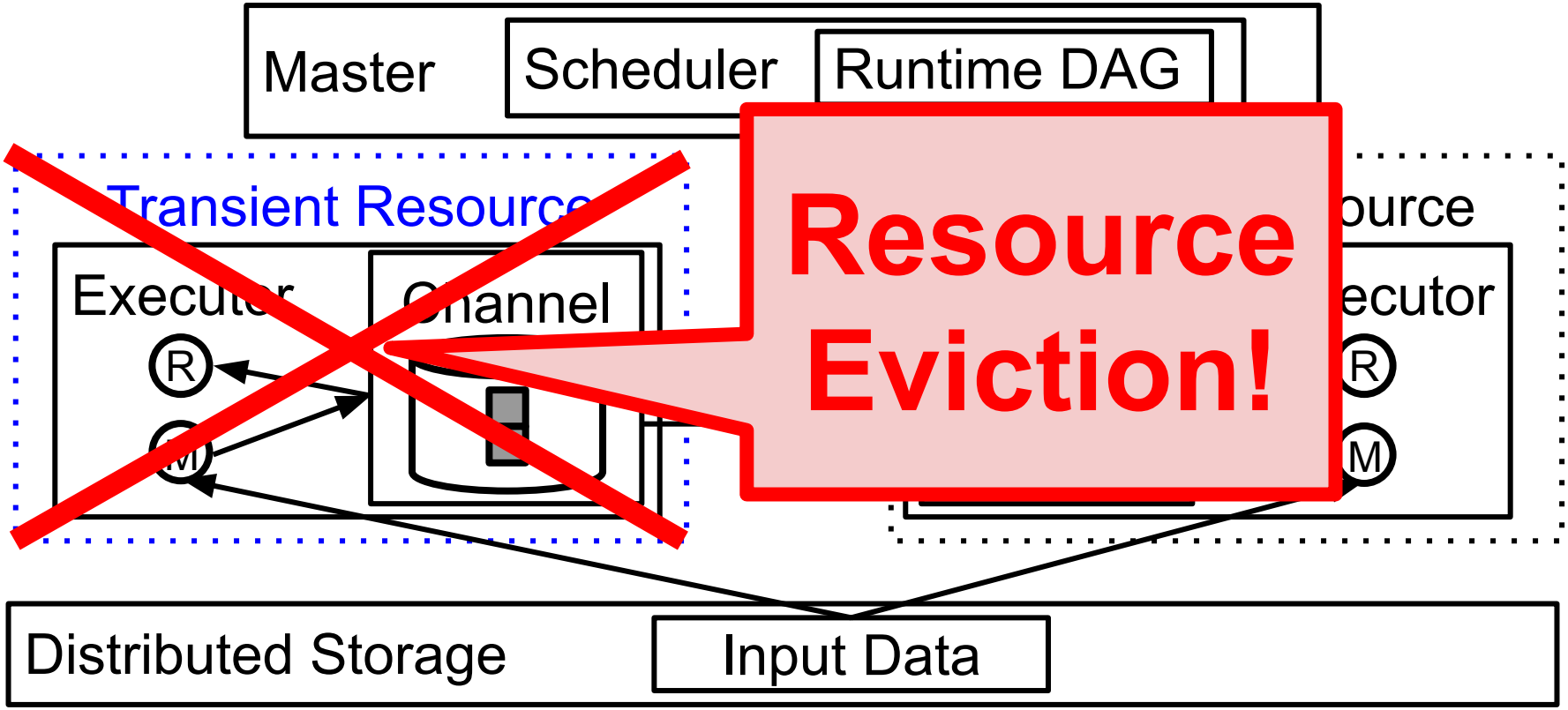
(2) Large-scale

Skewed

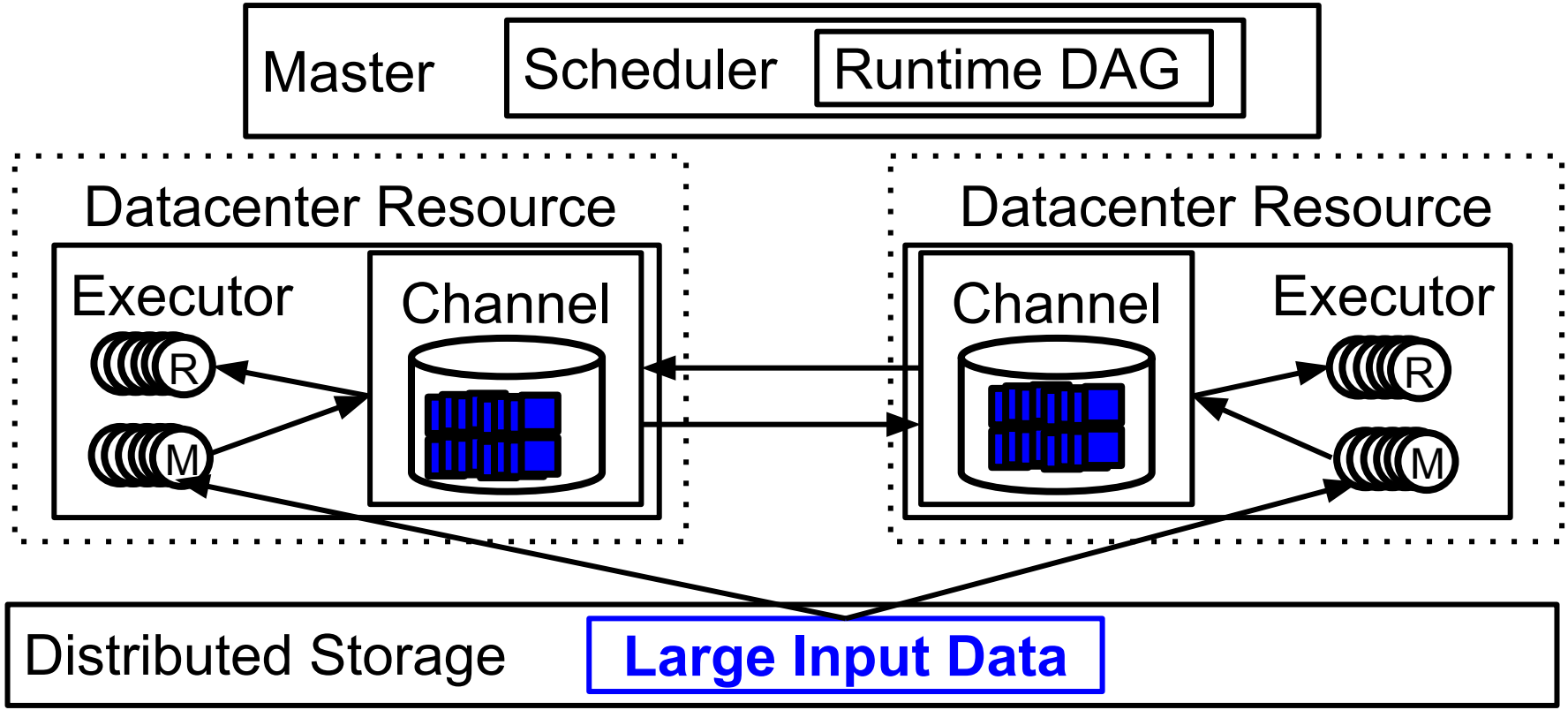
(1) Cheap Transient Resources



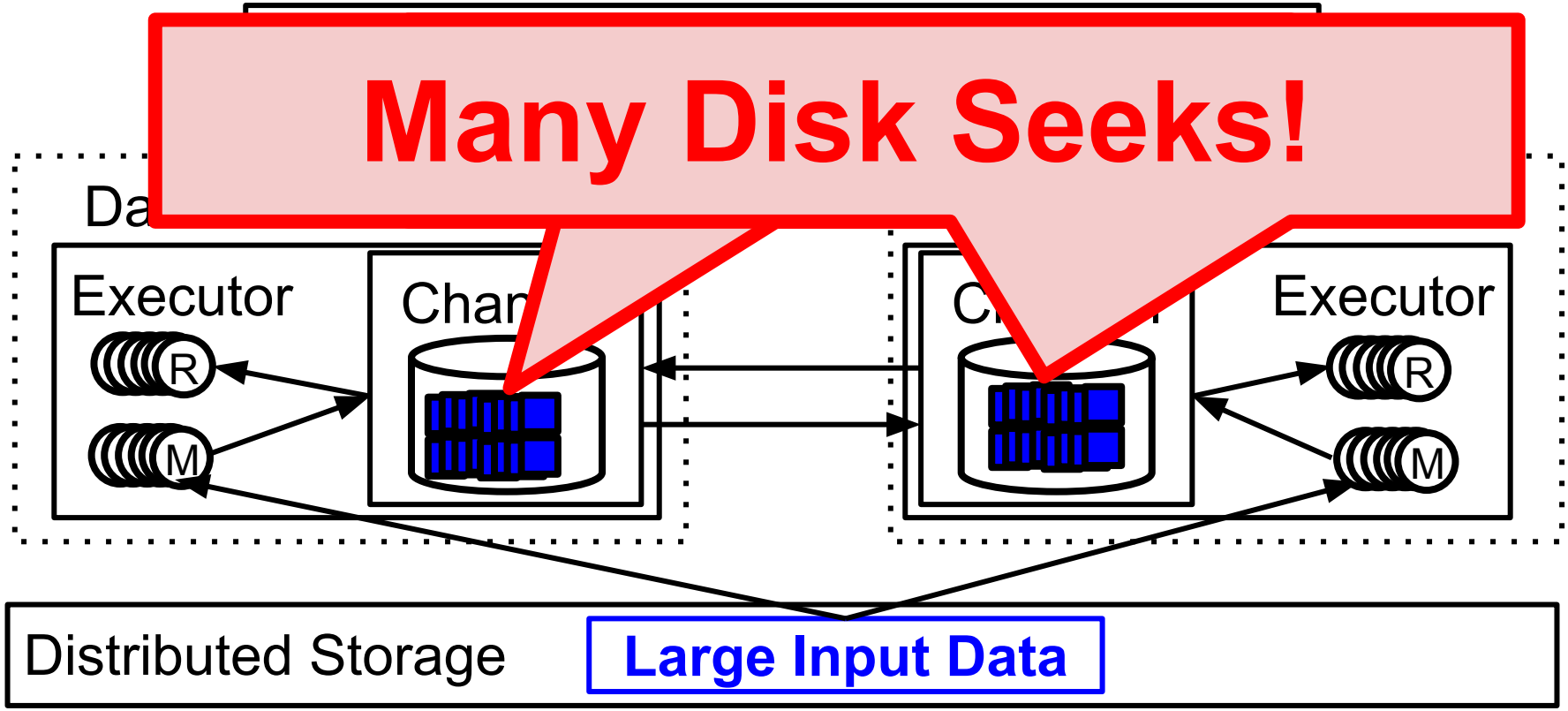
(1) Cheap Transient Resources



(2) Large-scale Data Shuffle

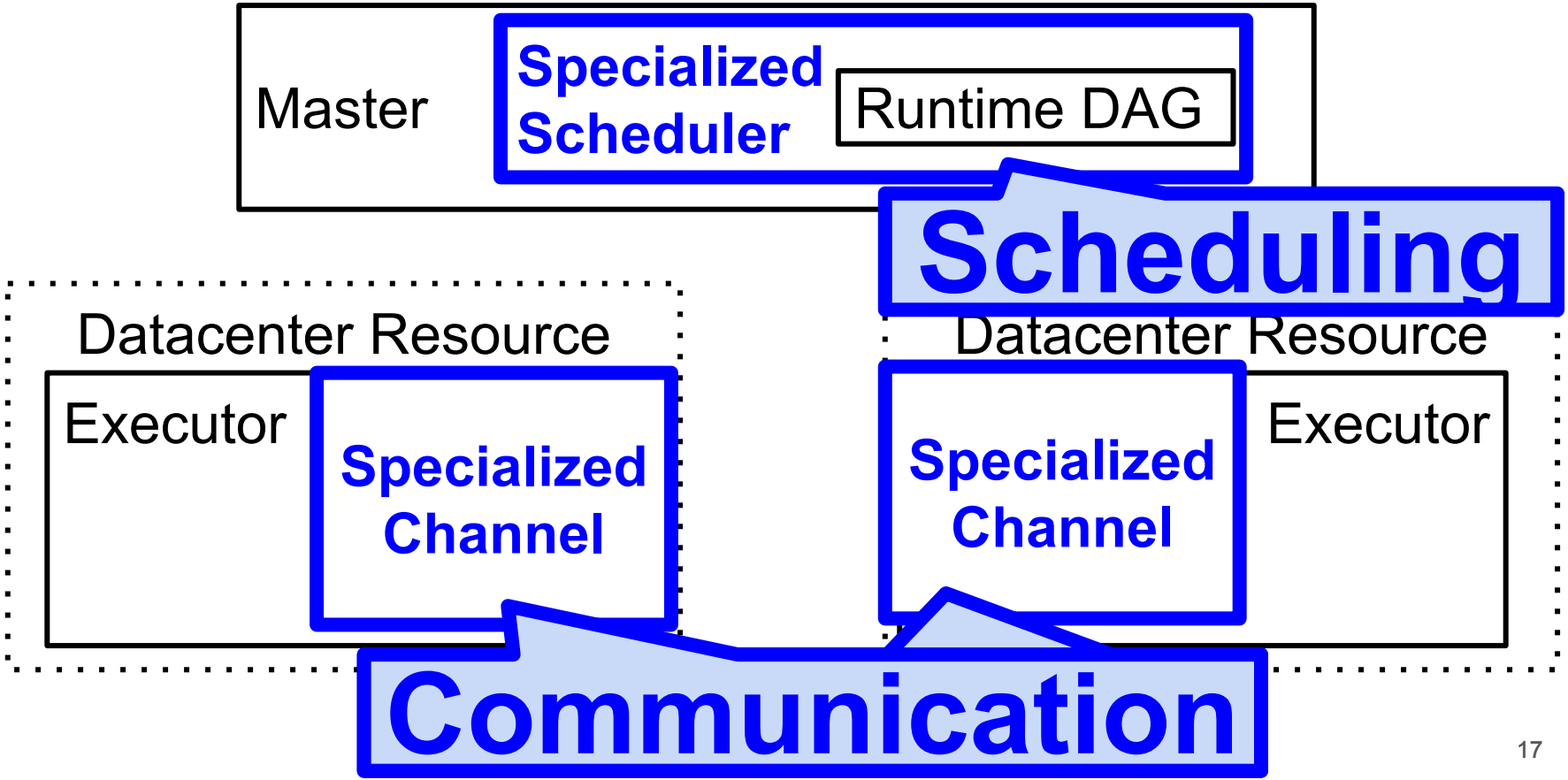


(2) Large-scale Data Shuffle



How to optimize distributed execution?

Existing Approach: Direct Specialization



Direct Specialization: Hard to Ensure...

(1) Correctness

Optimized execution produces the same results

(2) Reusability

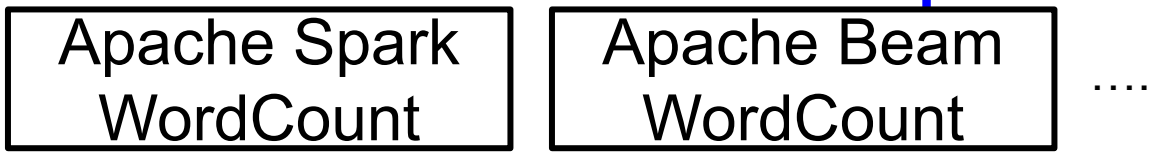
Single specialization across different applications

(3) Composability

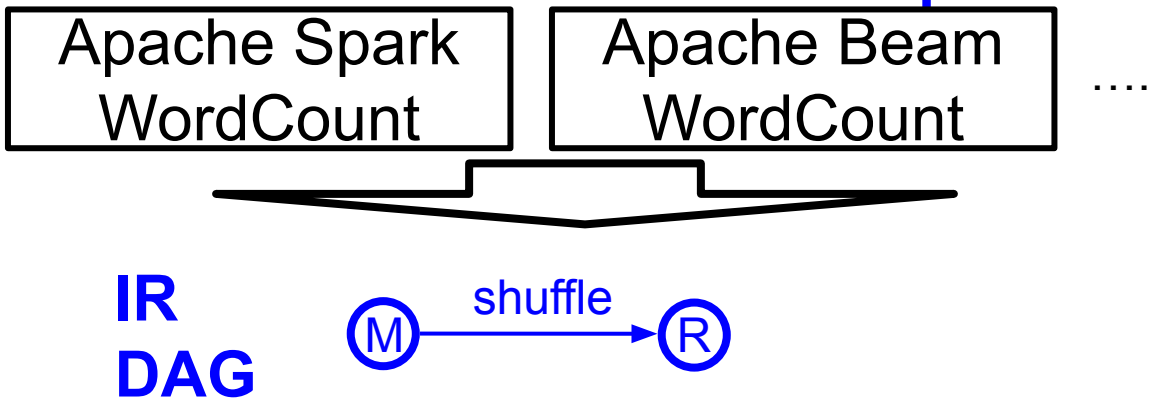
Combine multiple specialized optimizations

Our goal:
Make it easy to optimize distributed execution

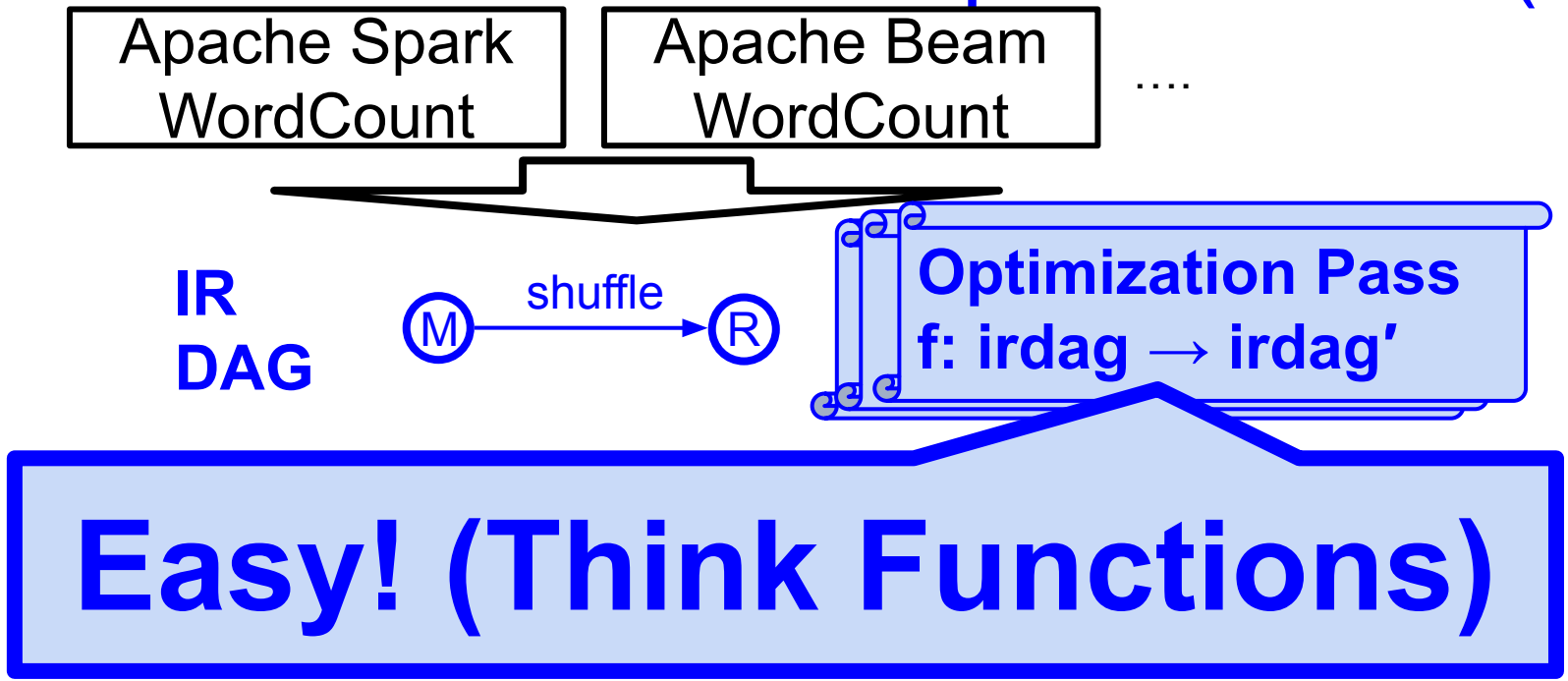
Our Idea: Intermediate Representation (IR)



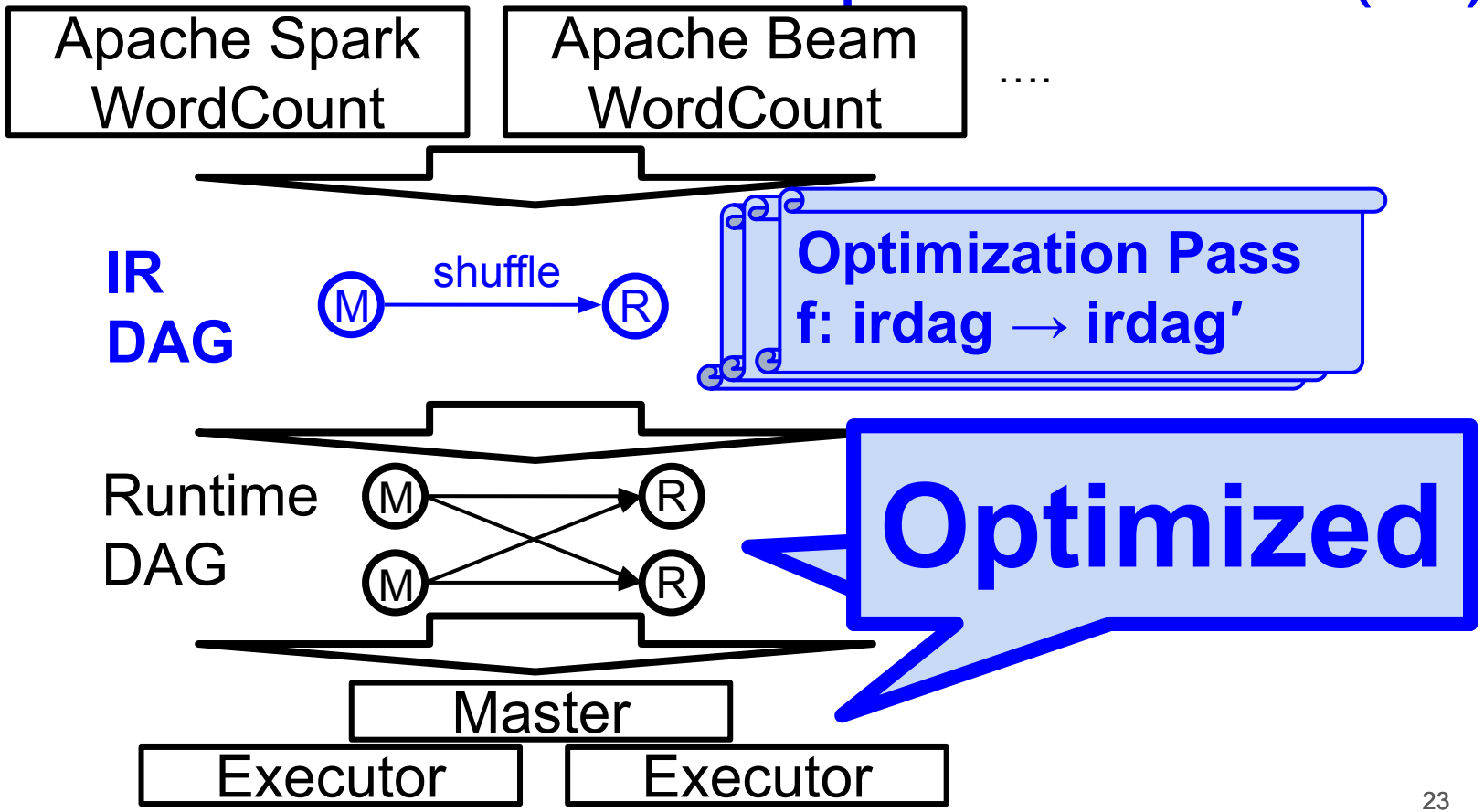
Our Idea: Intermediate Representation (IR)



Our Idea: Intermediate Representation (IR)



Our Idea: Intermediate Representation (IR)



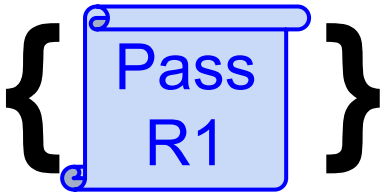
Overall Workflow of Apache Nemo

Nemo User Job Submission (Easy!)

Application

e.g., Spark/Beam Application

Compile-time Passes (List)



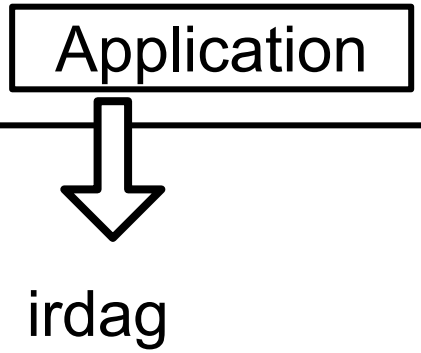
Run-time Passes (Set)

Nemo Applies Compile-time Passes

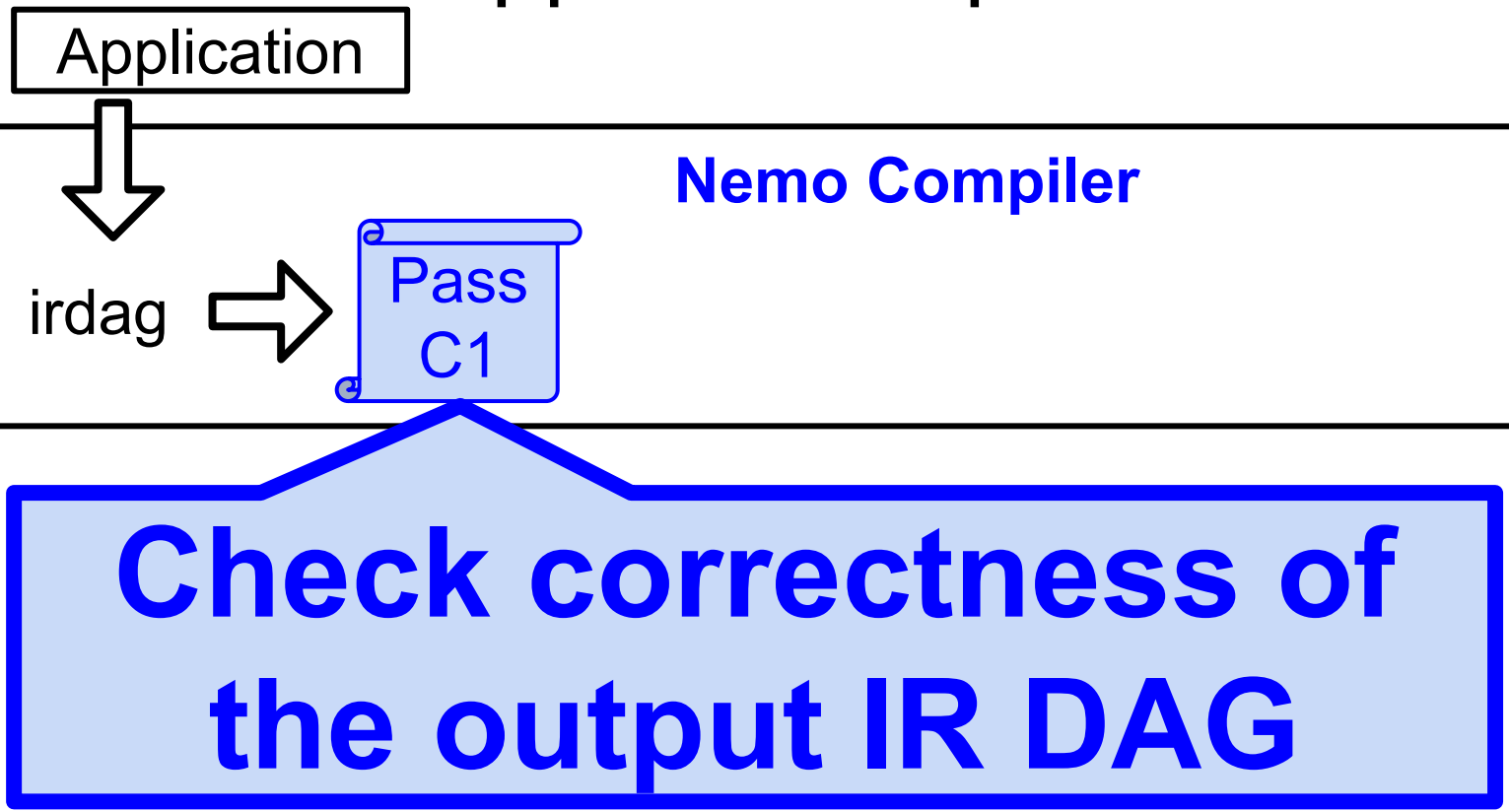
Application

Nemo Compiler

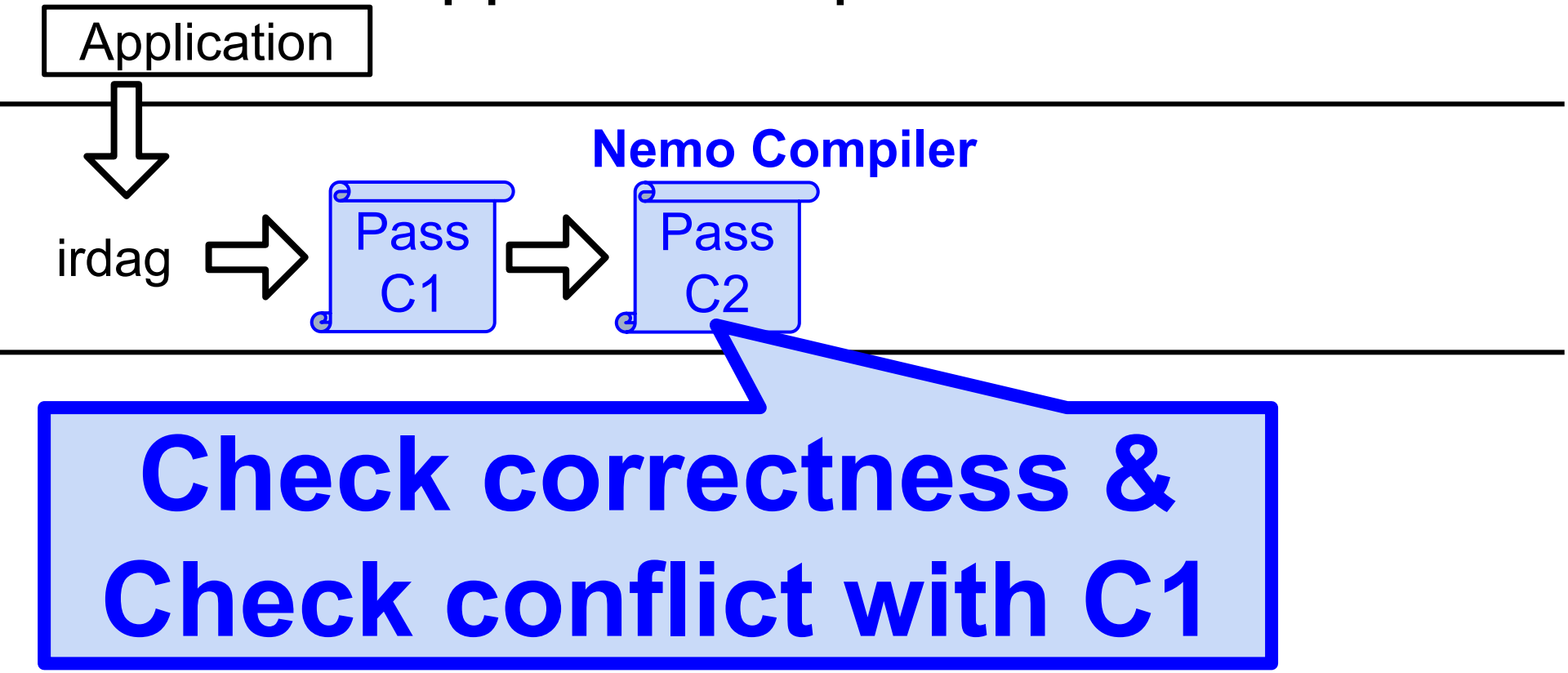
Nemo Applies Compile-time Passes



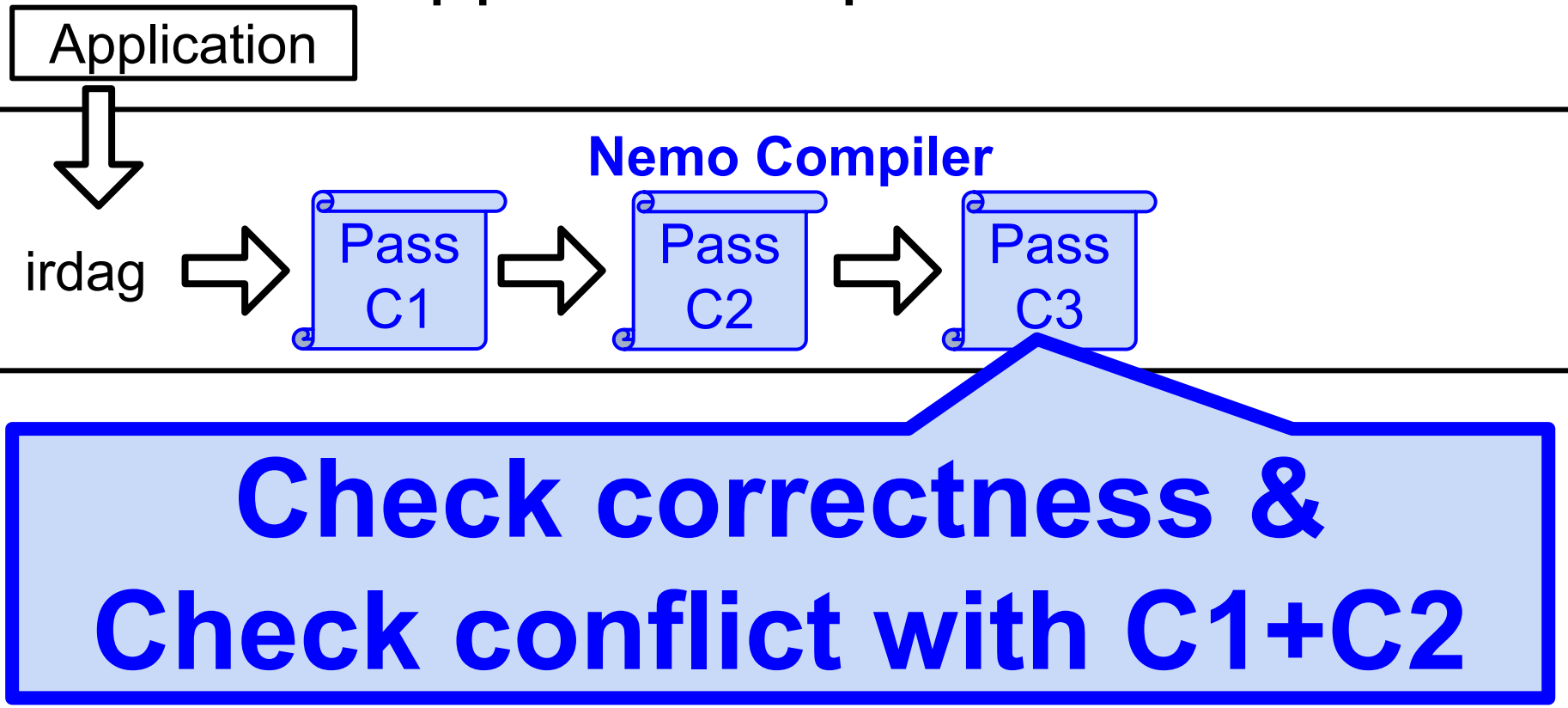
Nemo Applies Compile-time Passes



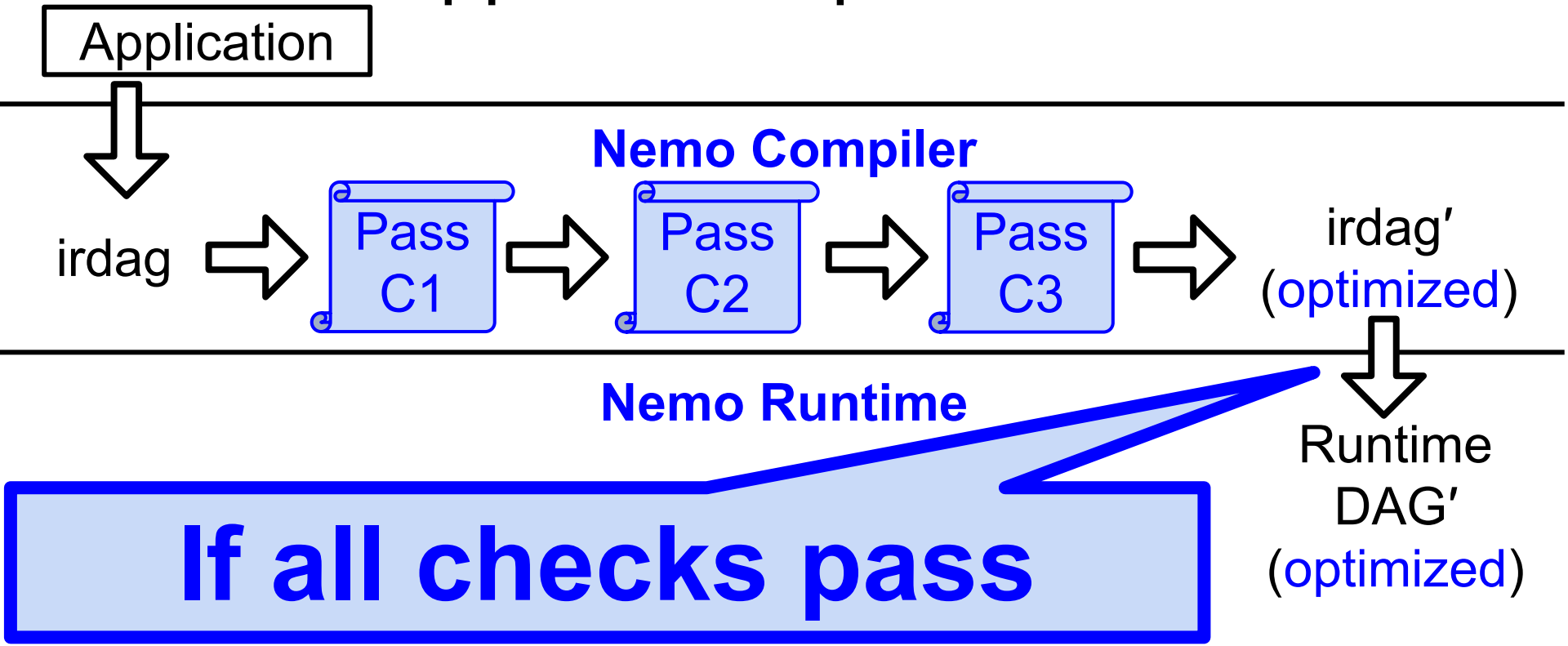
Nemo Applies Compile-time Passes



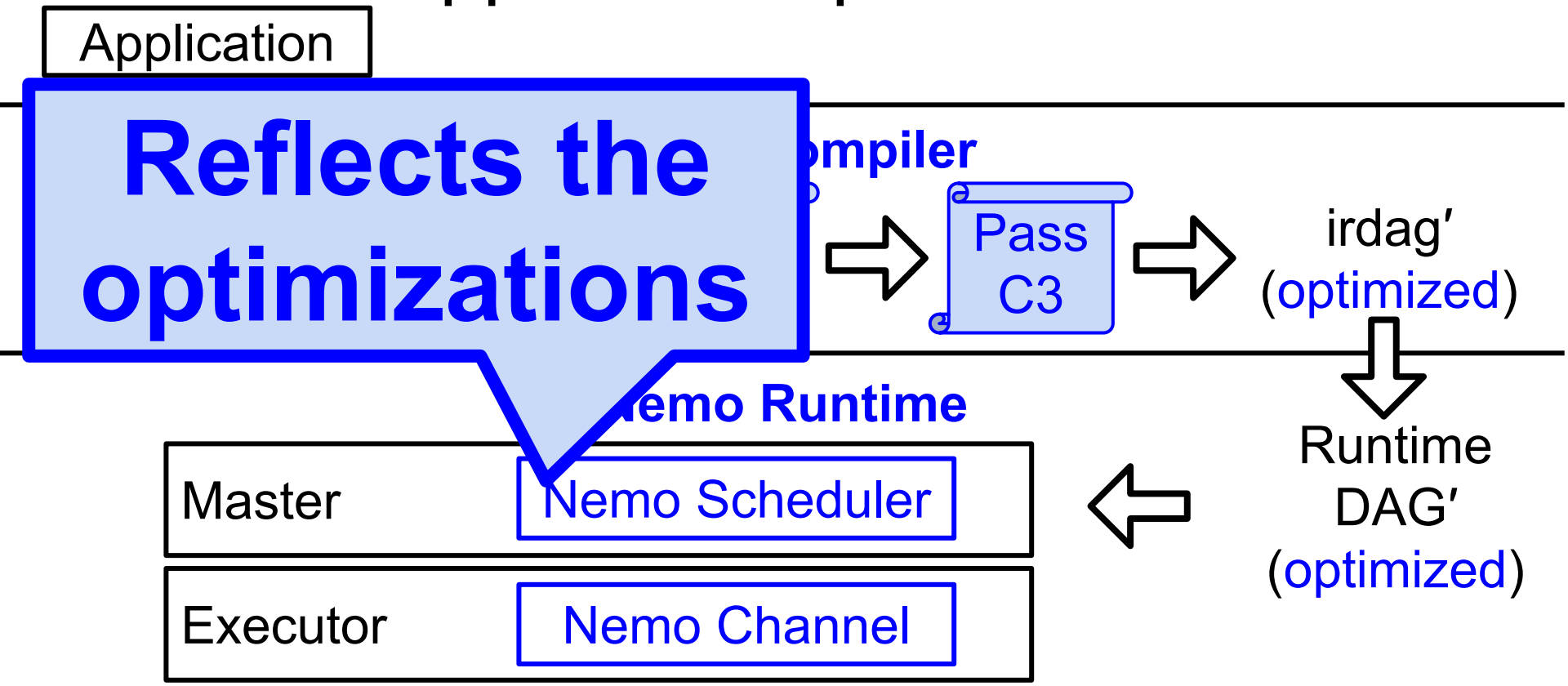
Nemo Applies Compile-time Passes



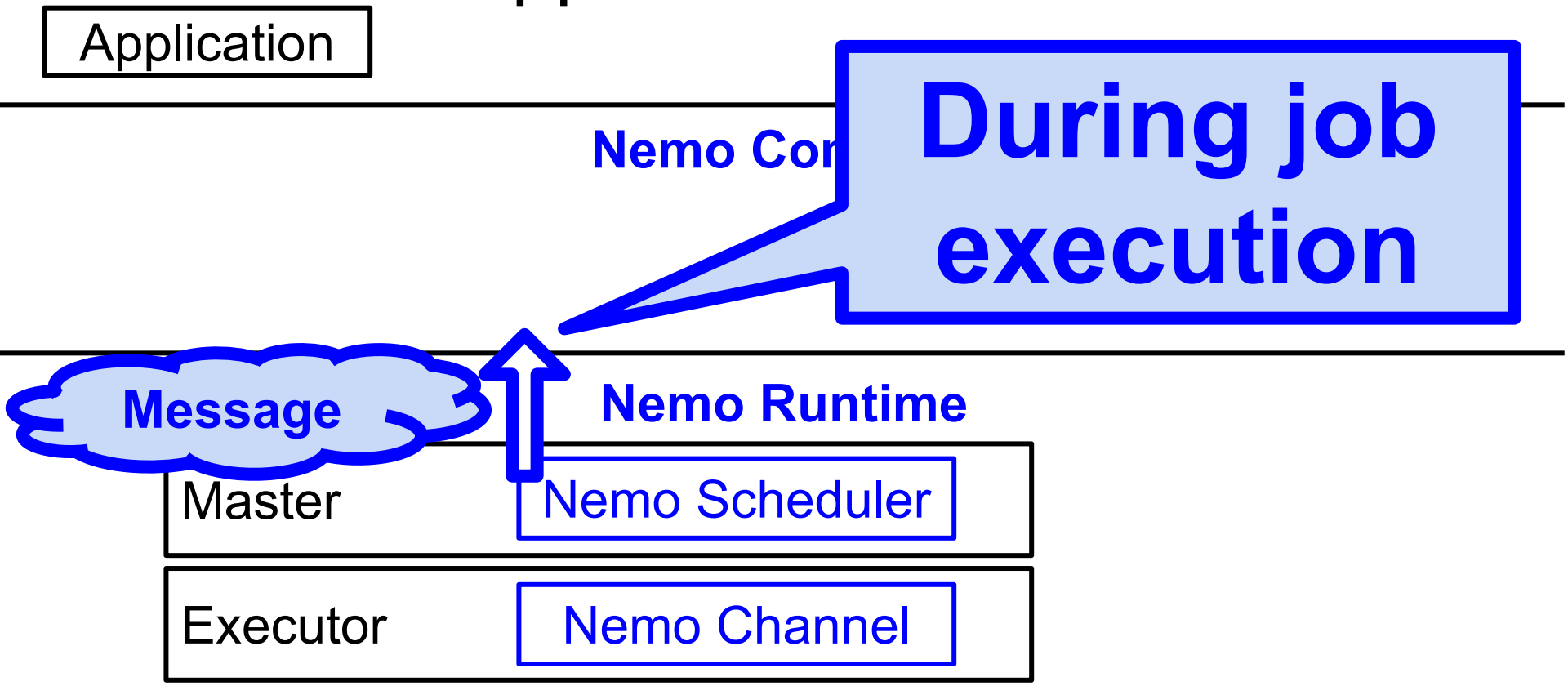
Nemo Applies Compile-time Passes



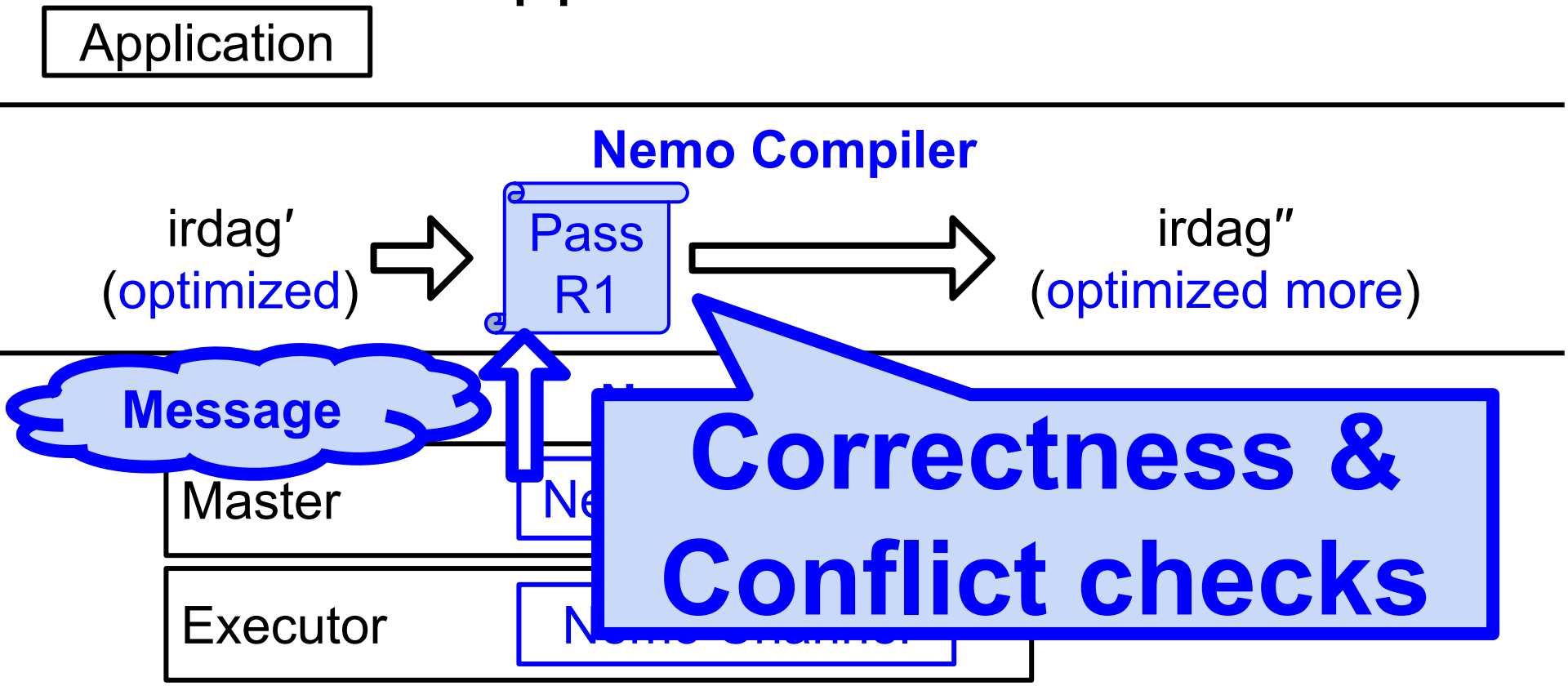
Nemo Applies Compile-time Passes



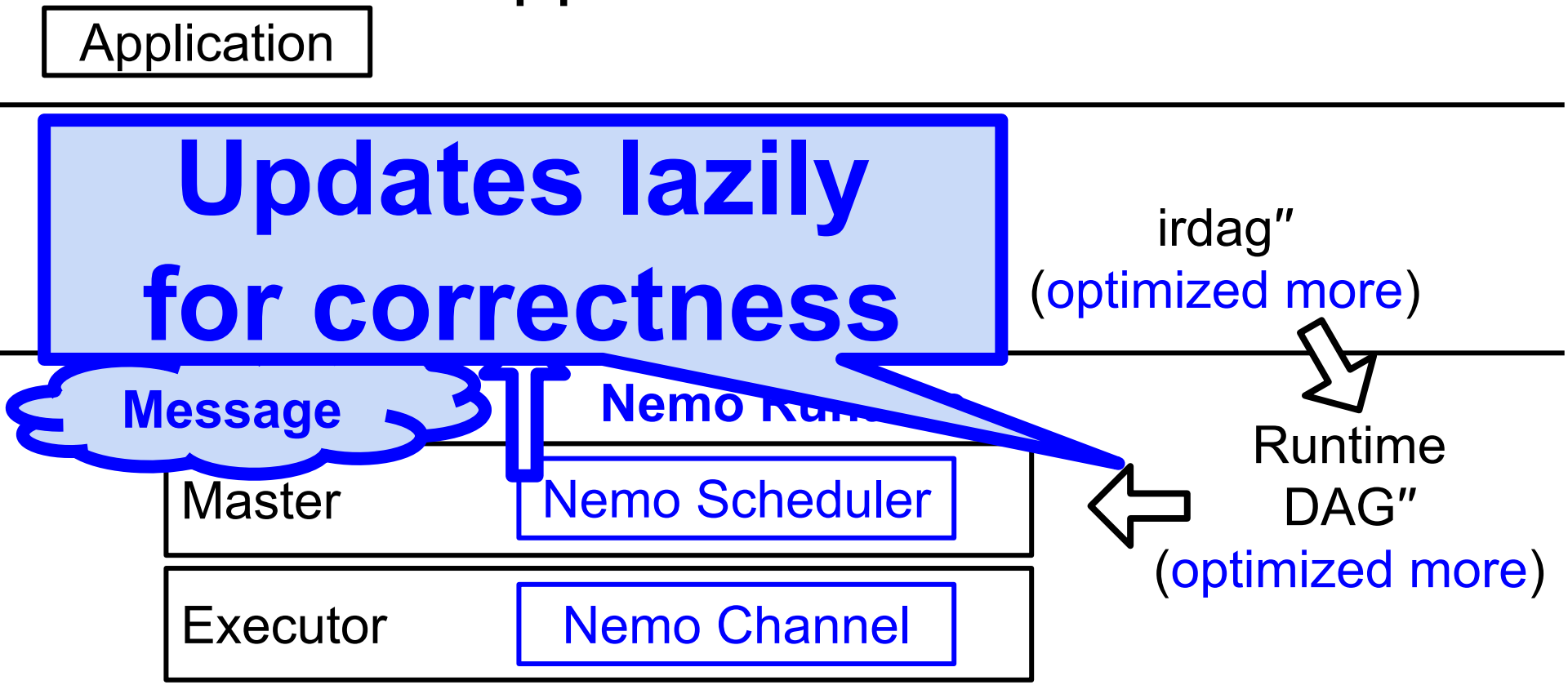
Nemo Applies Run-time Passes



Nemo Applies Run-time Passes



Nemo Applies Run-time Passes



Example Apache Nemo Optimization Passes

What A Pass Does

While traversing the input IR DAG,

(1) Inserts **Utility Vertices**

(2) Annotates **Execution Properties**

What A Pass Does

Applies a specific function

Wh… …g the input IR DAG,

(1) Inserts **Utility Vertices**

(2) Annotates **Execution Properties**

What A Pass Does

While traversing the input IR DAG,

(1) Inserts **Utility Vertices**

(2) Annotates **Execution Properties**

Scheduling/Communication

Passes We Implemented & Evaluated

GeoDistResourcePass

LargeShufflePass

TransientResourcePass

SkewCTPass

SkewRTPass

SkewSamplingPass

In This Talk (See Paper for Others)

GeoDistResourcePass

(1) LargeShufflePass

(2) TransientResourcePass

SkewCTPa

SkewRTPa

SkewSampling

Both are compile time passes

(1) LargeShufflePass: Goal

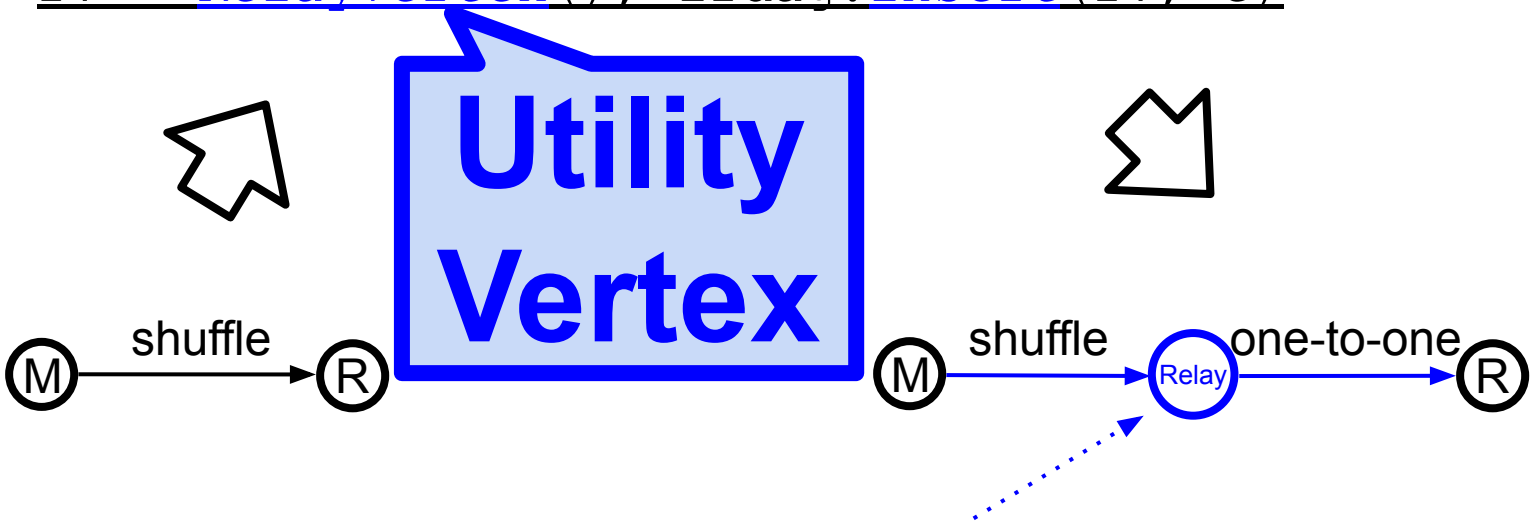
Avoid on-disk data shuffle!

- Shuffle data in memory
- Write shuffled data to disks
- Read from disks sequentially

Related Work: Riffle (EuroSys18)

(1) LargeShufflePass: Algorithm

```
for each shuffle edge e in irdag:
 rv = RelayVertex(), irdag.insert(rv, e)
```



Applies an identity function

(1) LargeShufflePass: Algorithm

```
for each s
 rv = Rela
 rv.inEdge.set(DataFlow.Push,
  DataStore.Memory, Persistence.Discard)
```

Execution Properties

In-memory shuffle

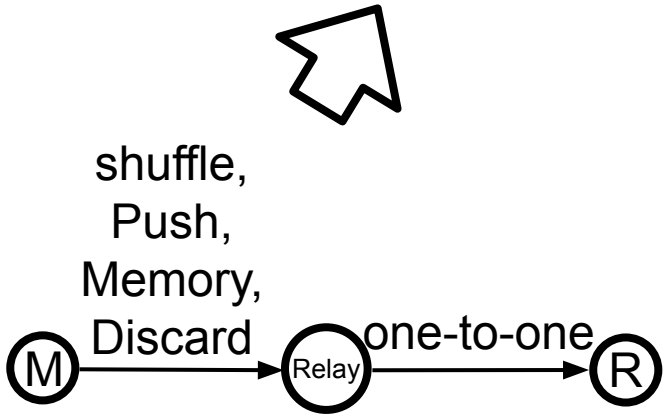
Execute M and Relay concurrently

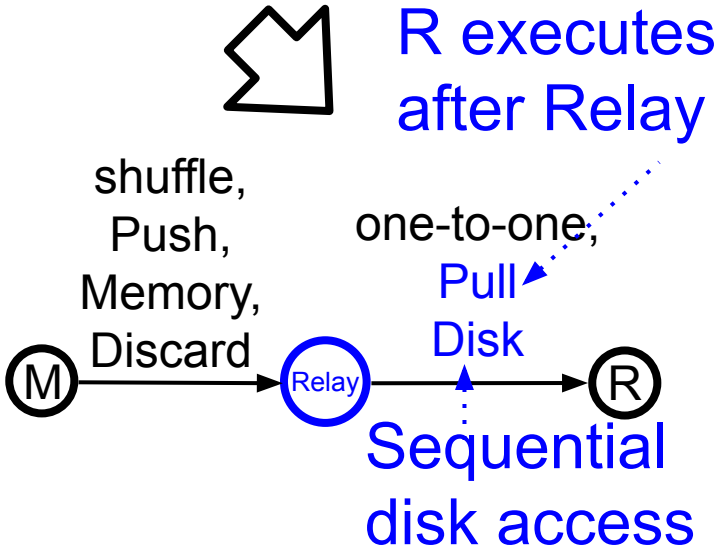
shuffle, Push, Memory, Discard

Do not persist data in memory

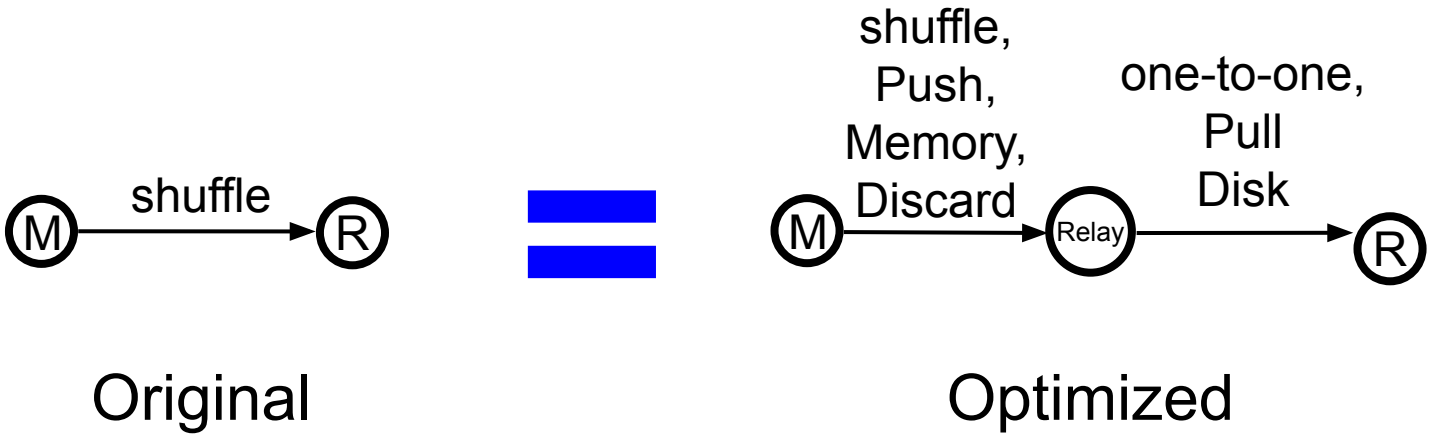
(1) LargeShufflePass: Algorithm

```
for each shuffle edge e in irdag:
 rv = RelayVertex(), irdag.insert(rv, e)
 rv.inEdge.set(DataFlow.Push,
  DataStore.Memory, Persistence.Discard)
 rv.outEdge.set(DataFlow.Pull, DataStore.Disk)
```



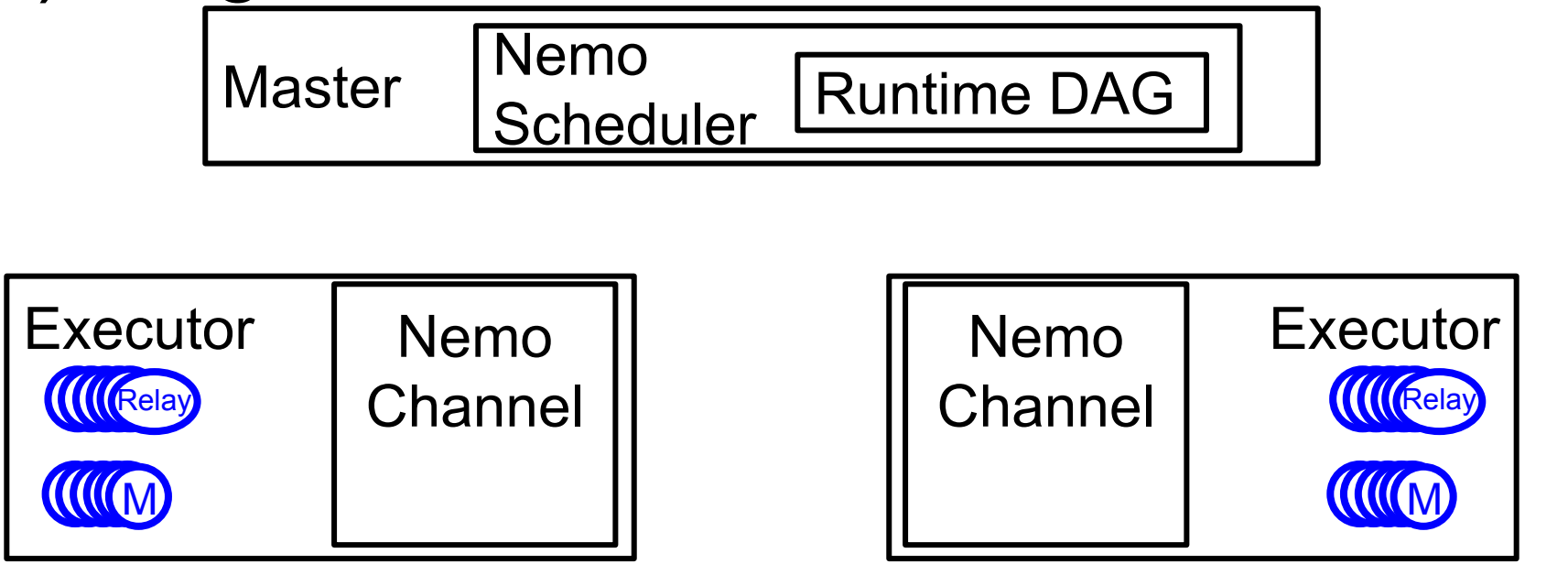


(1) LargeShufflePass: Correctness

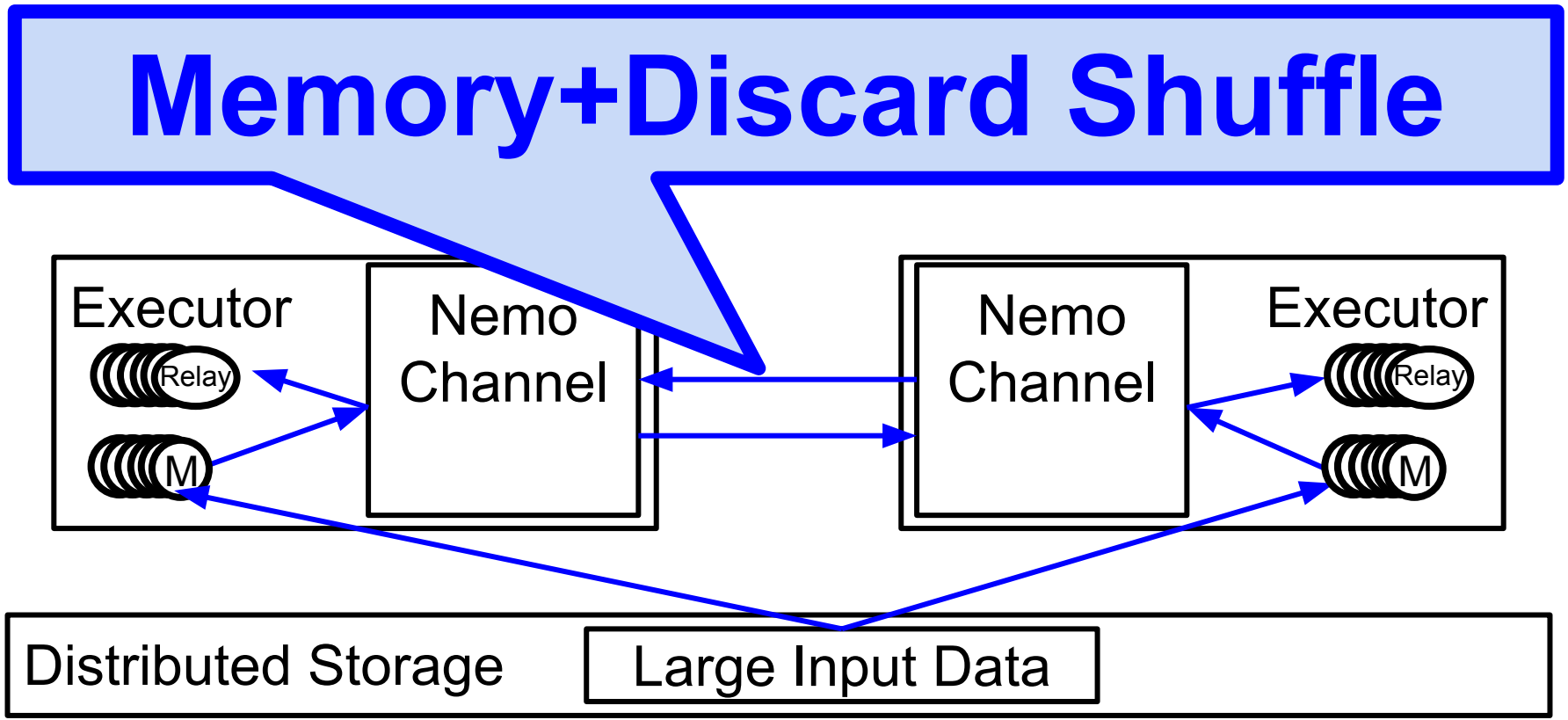


Equivalent final outputs!

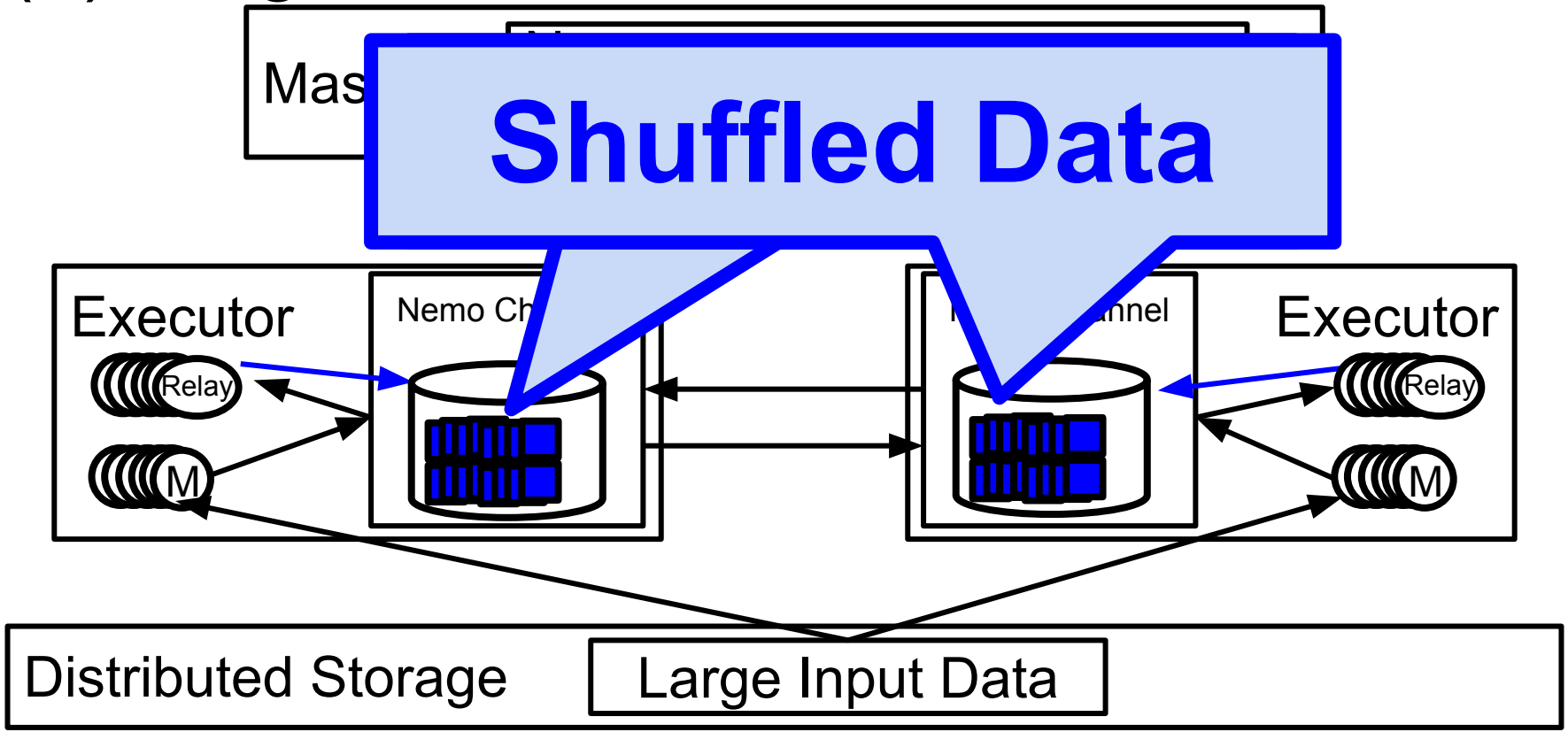
(1) LargeShufflePass: Runtime Execution



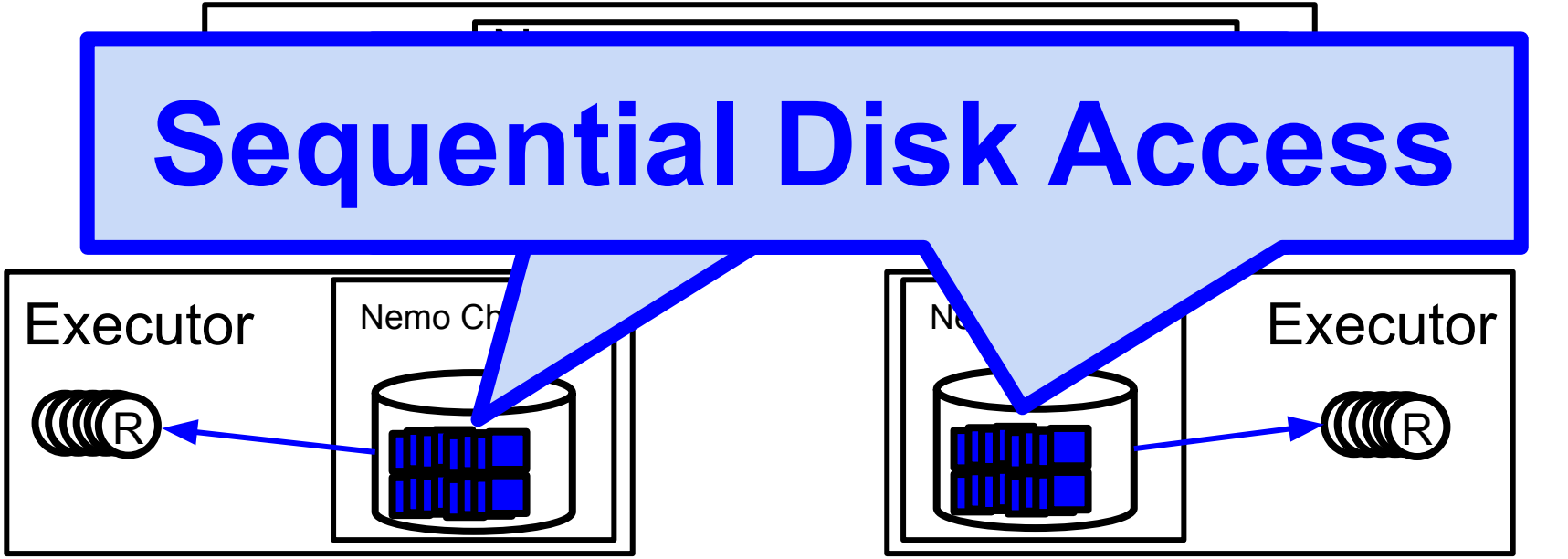
(1) LargeShufflePass: Runtime Execution



(1) LargeShufflePass: Runtime Execution



(1) LargeShufflePass: Runtime Execution



Distributed Storage | Large Input Data

(2) TransientResourcePass: Goal

Minimize recomputations!

- Place on Transient/Reserved judiciously
- Push data from Transient to Reserved

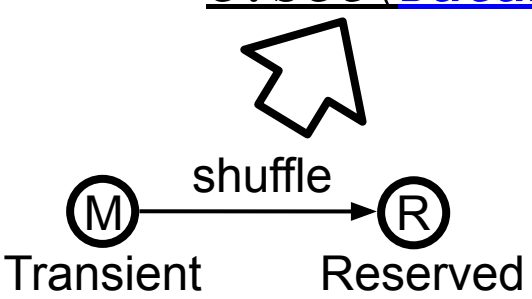
Related Work: Pado (EuroSys17)

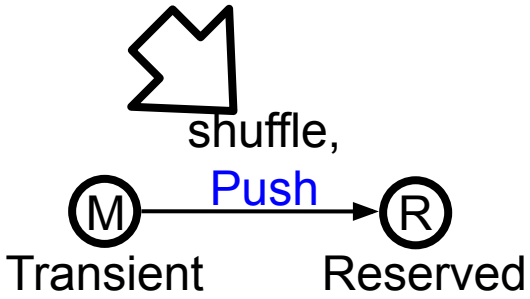
(2) TransientResourcePass: Algorithm

```
for each vertex v in topologicalSort(irdag):
 if (containsShuffle(v.inEdges) || ...):
  v.set(ResourcePriority.Reserved)
 else:
  v.set(ResourcePriority.Transient)
```

(2) TransientResourcePass: Algorithm

```
for each vertex v in topologicalSort(irdag):
 if (containsShuffle(v.inEdges) || ...):
  v.set(ResourcePriority.Reserved)
 else:
  v.set(ResourcePriority.Transient)
 for e in v.inEdges:
  if fromTransientToReserved(e.src, e.dst):
   e.set(DataFlow.Push)
```





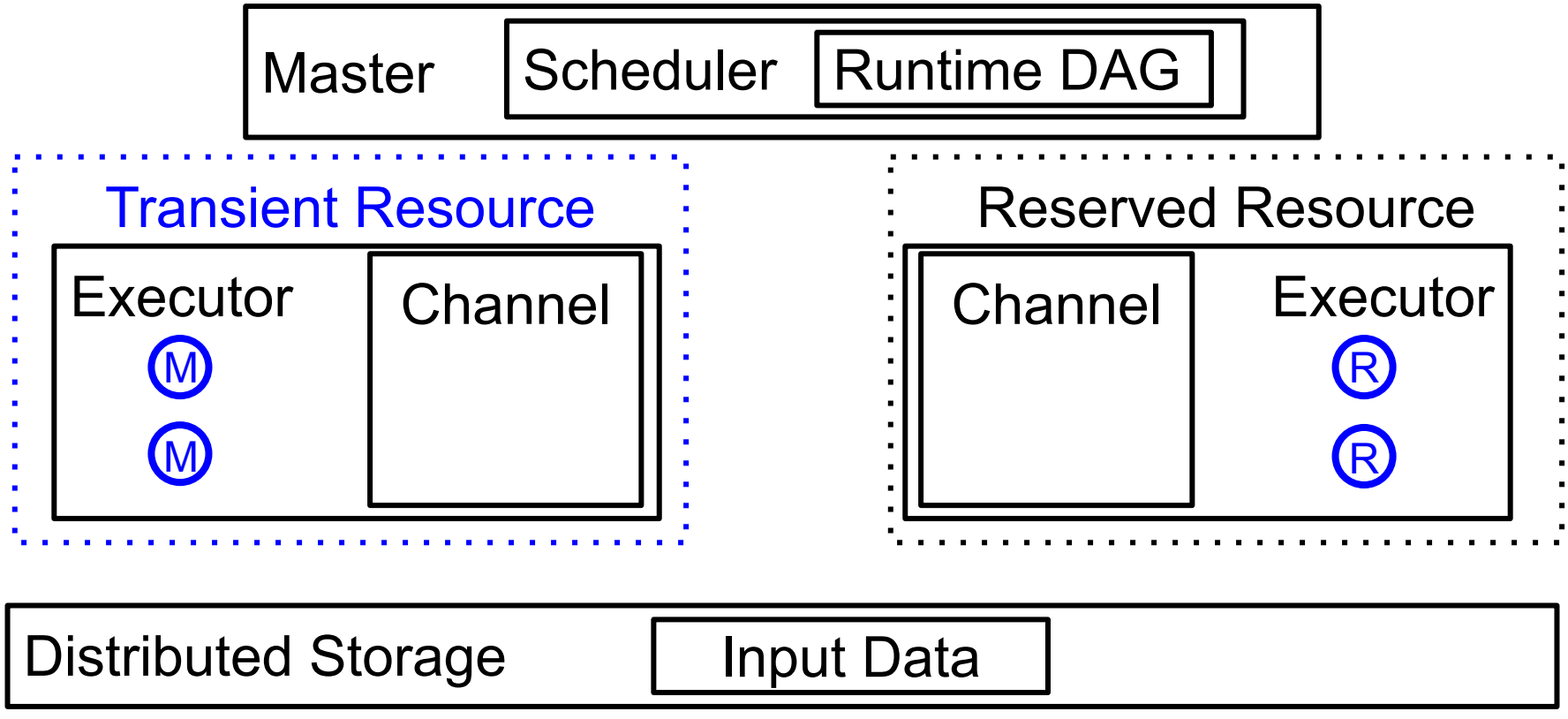
(2) TransientResourcePass: Corectness

Original

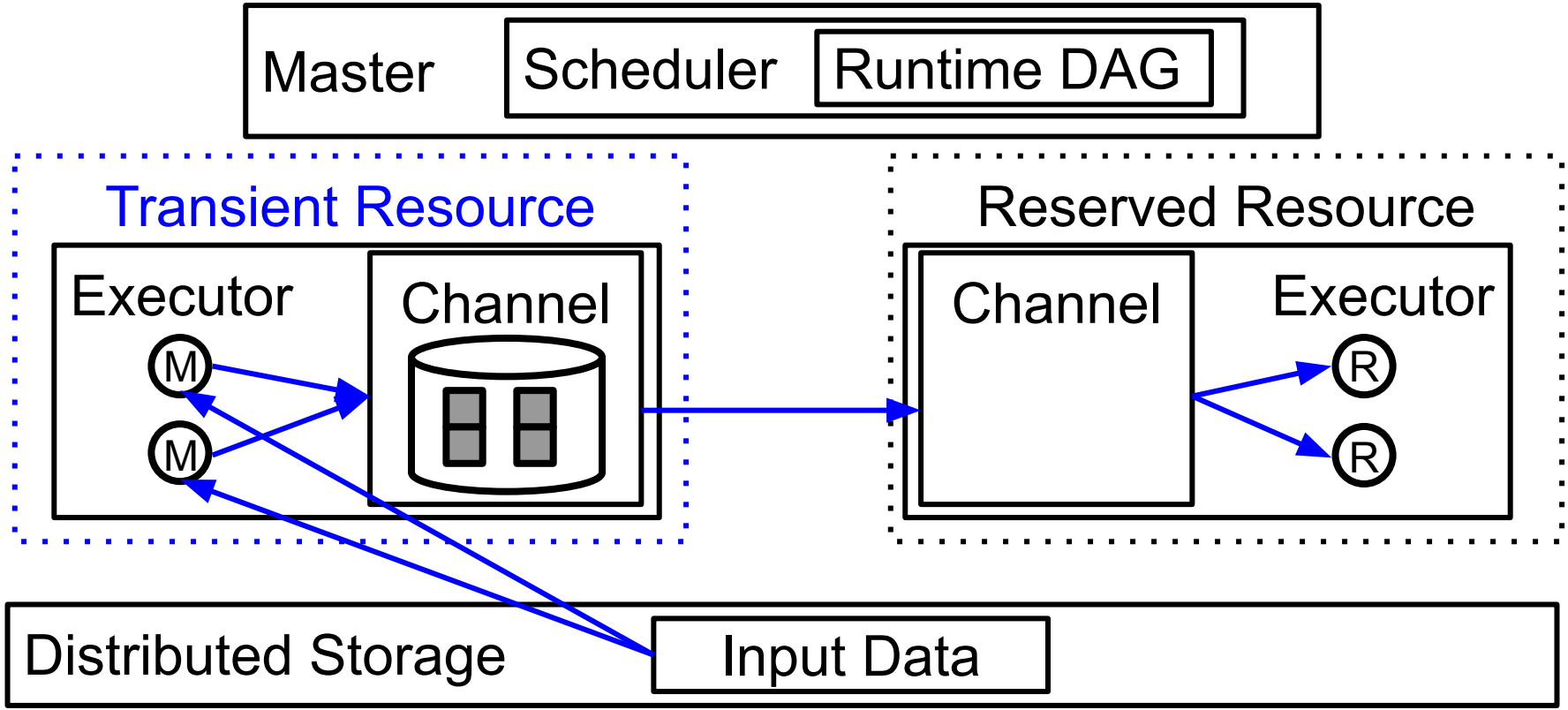
Optimized

Equivalent final outputs!

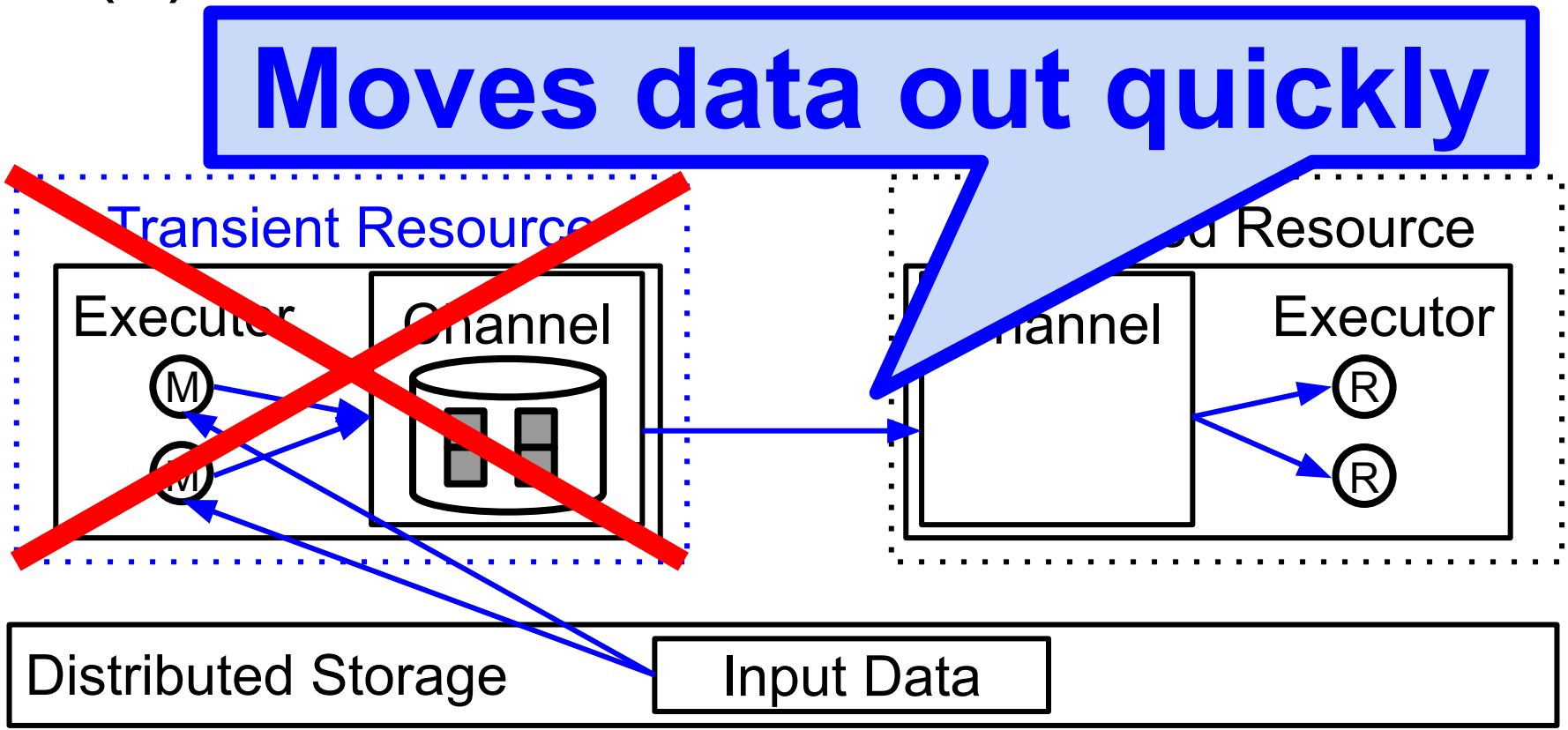
(2) TransientResourcePass: Runtime



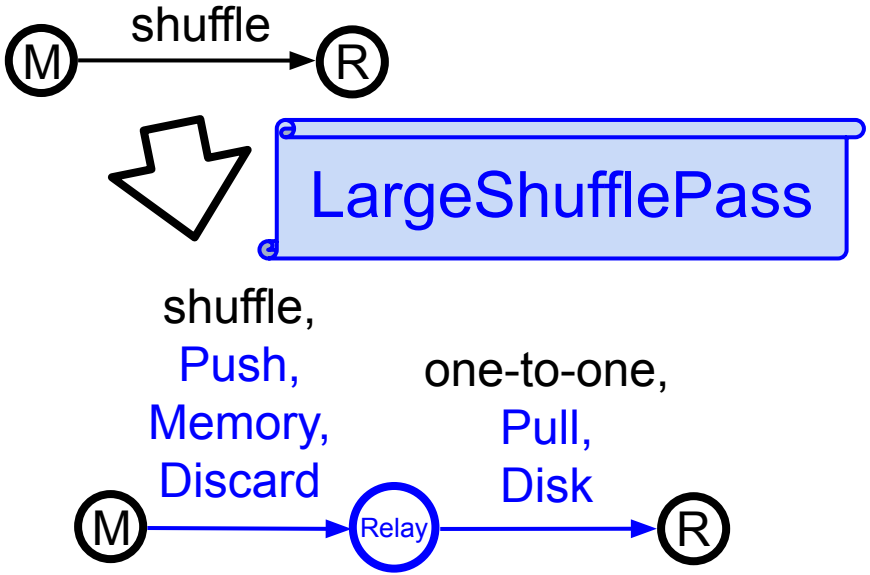
(2) TransientResourcePass: Runtime



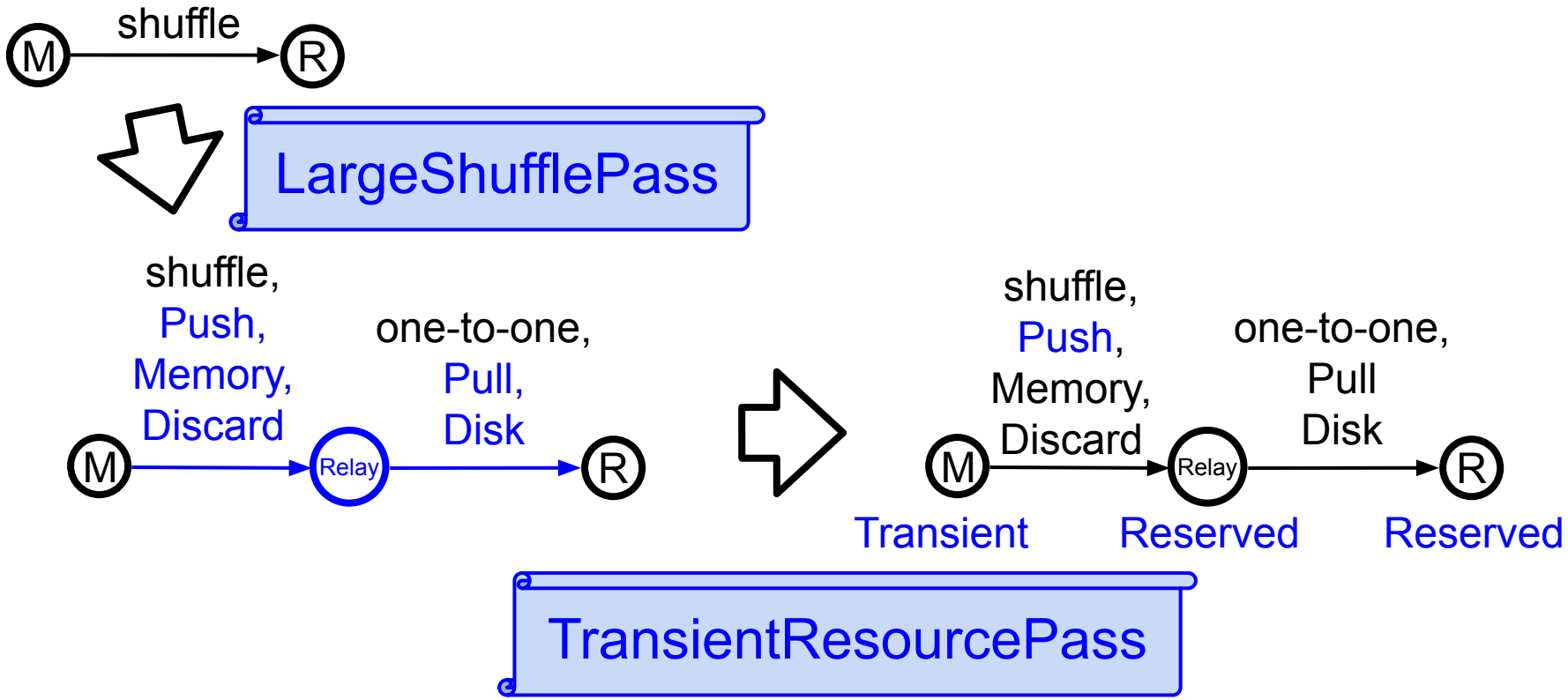
(2) TransientResourcePass: Runtime



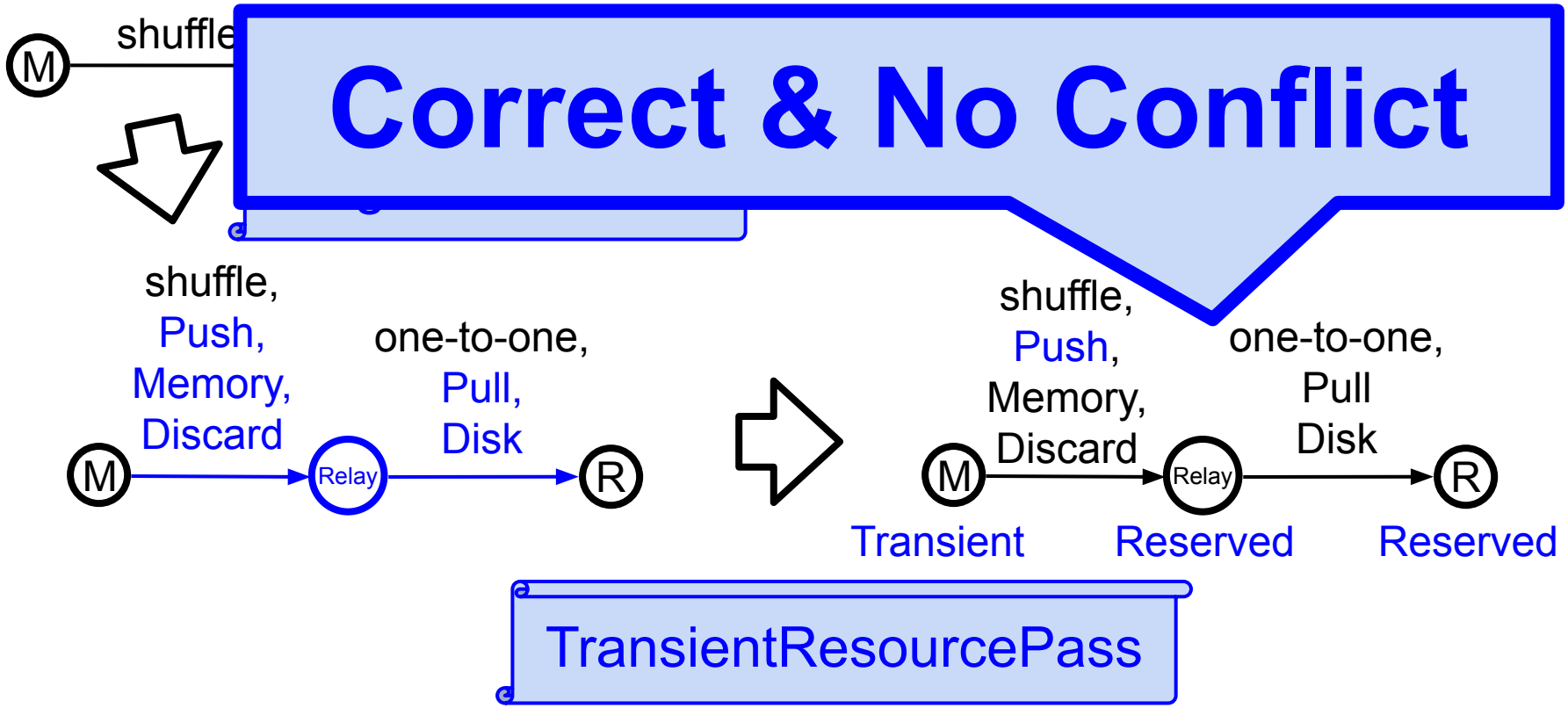
LargeShufflePass+TransientResourcePass



LargeShufflePass+TransientResourcePass



LargeShufflePass+TransientResourcePass



Implementation & Evaluation

Nemo Implementation

- Open source (https://nemo.apache.org)
- 32K lines of Java code, including its own runtime
- Good integration with other Apache Big Data projects
  - Supported applications

  - Supported cluster resource managers

(thanks to  REEF)

What We Evaluated: Scenarios

Large Data Shuffle

Transient Resources

Geo-distributed Resources

Skewed Data

Large Shuffle on Transient Resources

Skewed Data on Geo-distributed Resources

Large Shuffle with Skewed Data

In This Talk (See Paper for Others)

Large Data Shuffle

Transient Resources

Geo-distributed Resources

Skewed Data

Large Shuffle on Transient Resources

Skewed Data on Geo-distributed Resources

Large Shuffle with Skewed Data

What We Evaluated: Systems

Apache Nemo

Apache Spark: A state-of-the-art runtime

Pado (EuroSys17): Specialized for transient resources

Hurricane (EuroSys18): Specialized for data skew

Iridium (SIGCOMM15): Specialized for geo analytics

In This Talk (See Paper for Others)

Apache Nemo

Apache Spark: A state-of-the-art runtime

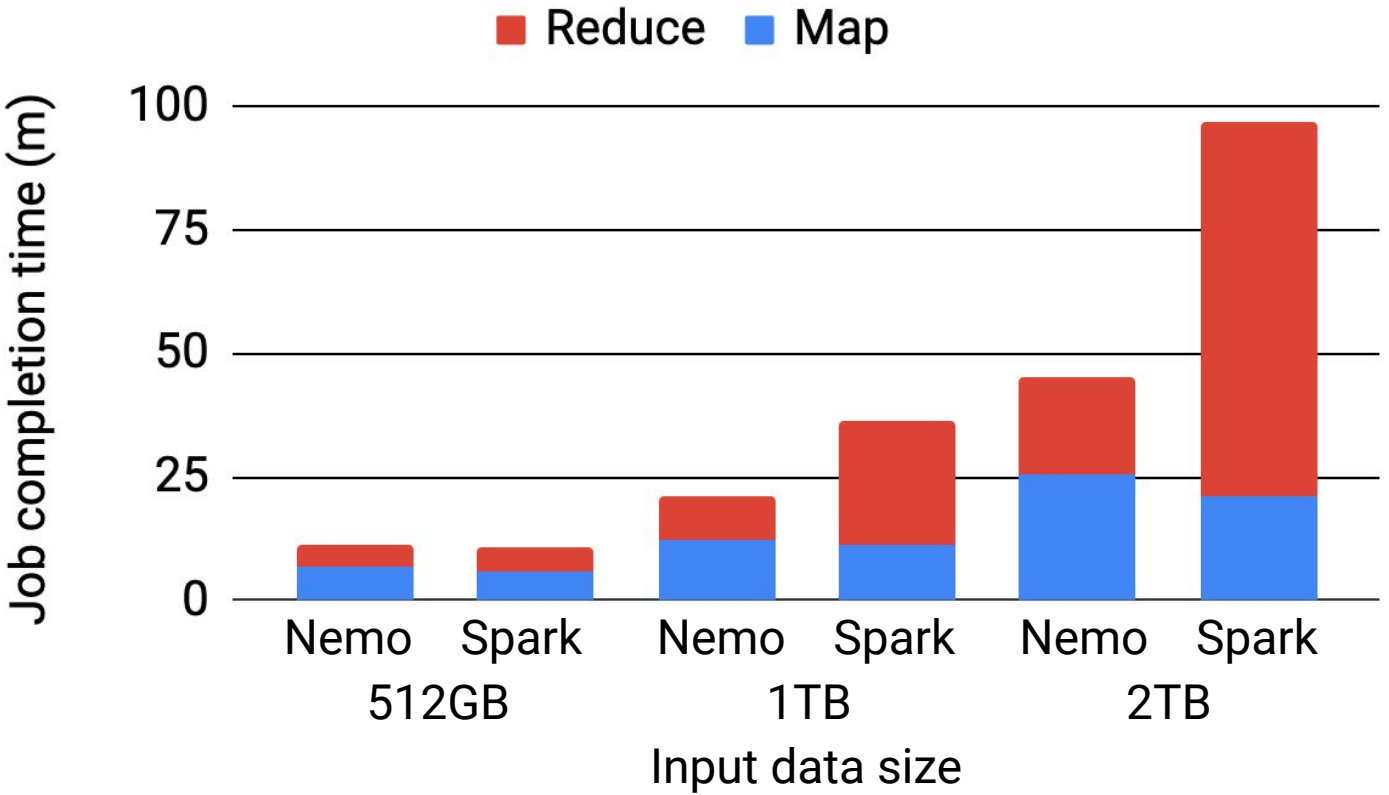
Pado (EuroSys17): Specialized for transient resources

Hurricane (EuroSys18): Specialized for data skew

Iridium (SIGCOMM15): Specialized for geo analytics

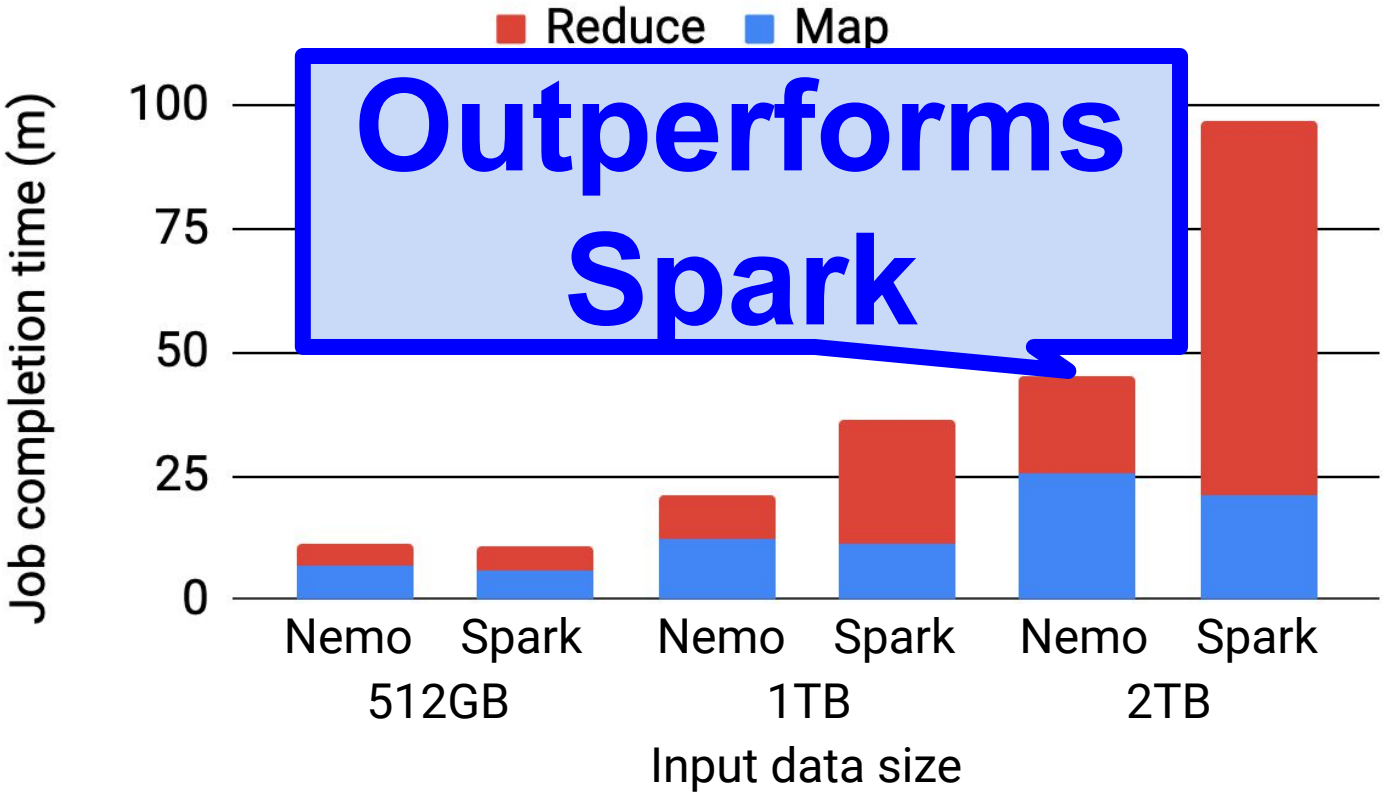
Large Shuffle (Lower is Better)

⇒ MapReduce on 20 AWS EC2 h1.4xlarge instances



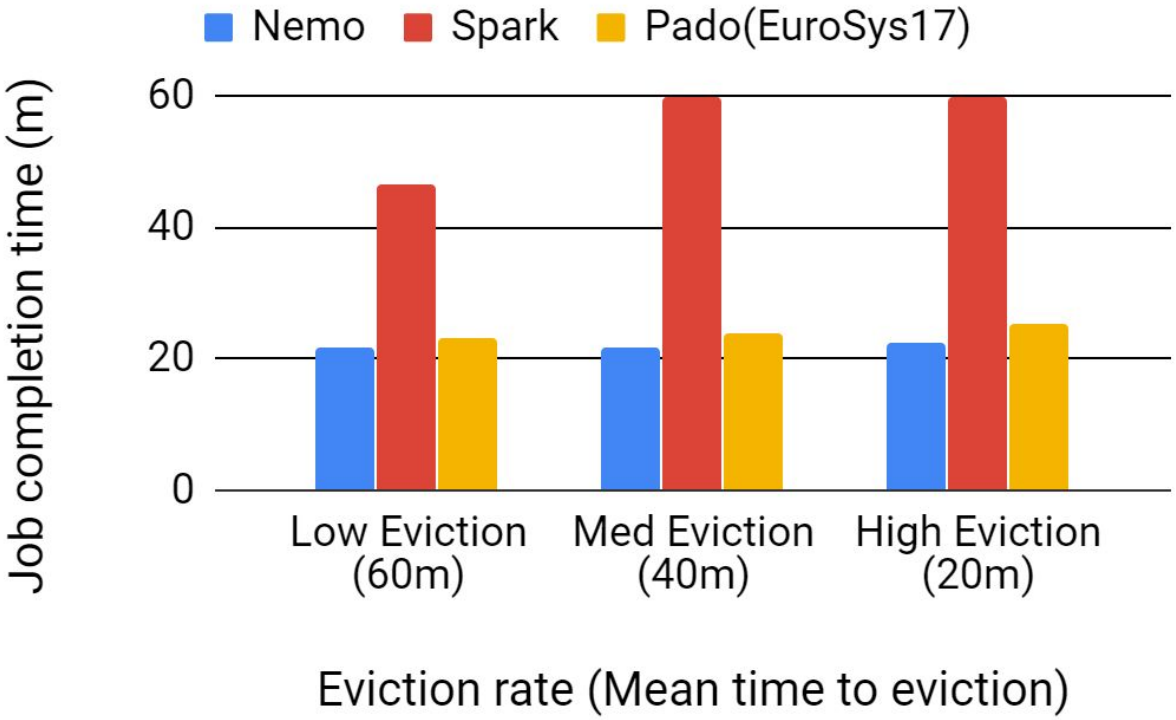
Large Shuffle (Lower is Better)

⇒ MapReduce on 20 AWS EC2 h1.4xlarge instances



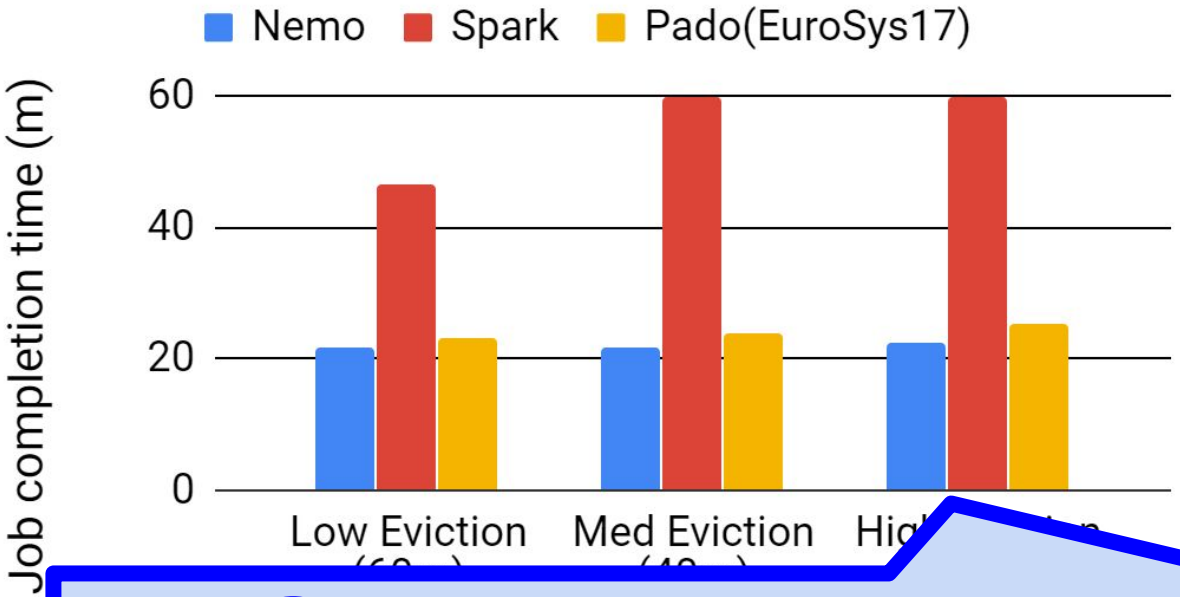
Transient Resources (Lower is Better)

⇒ ALS on 10 transient + 2 reserved EC2 instances



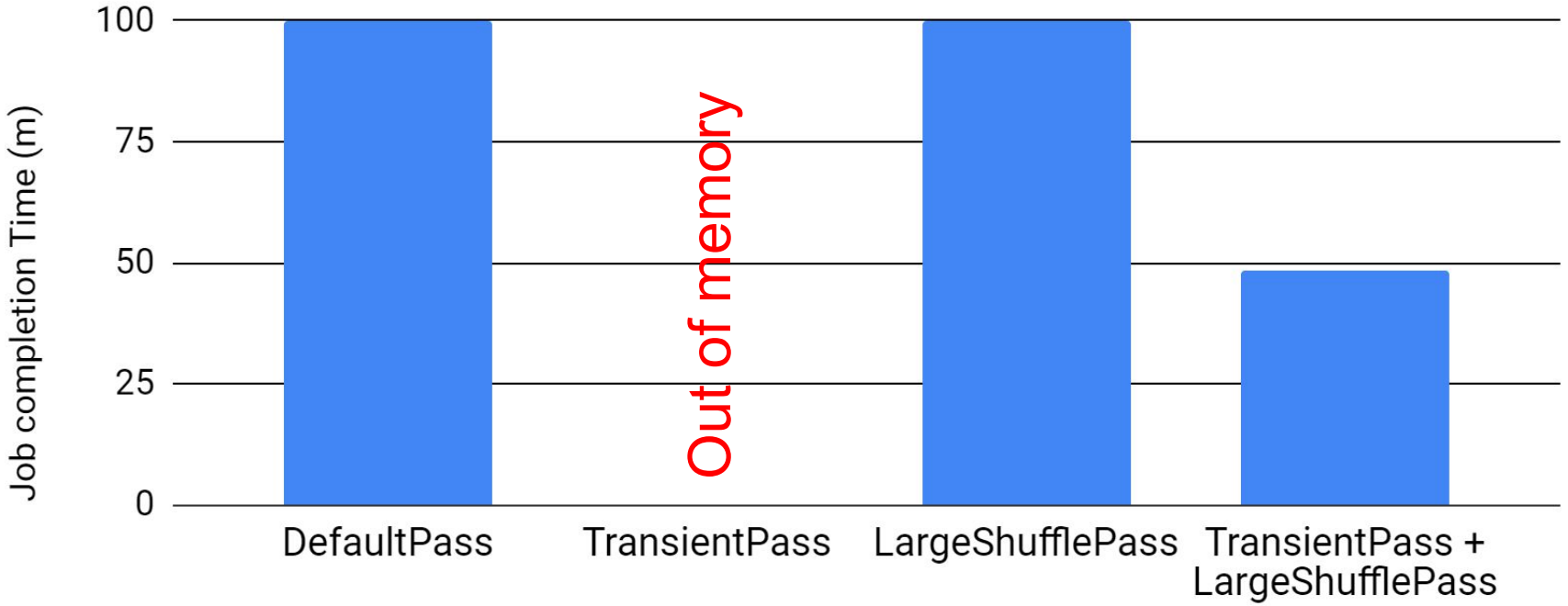
Transient Resources (Lower is Better)

⇒ ALS on 10 transient + 2 reserved EC2 instances



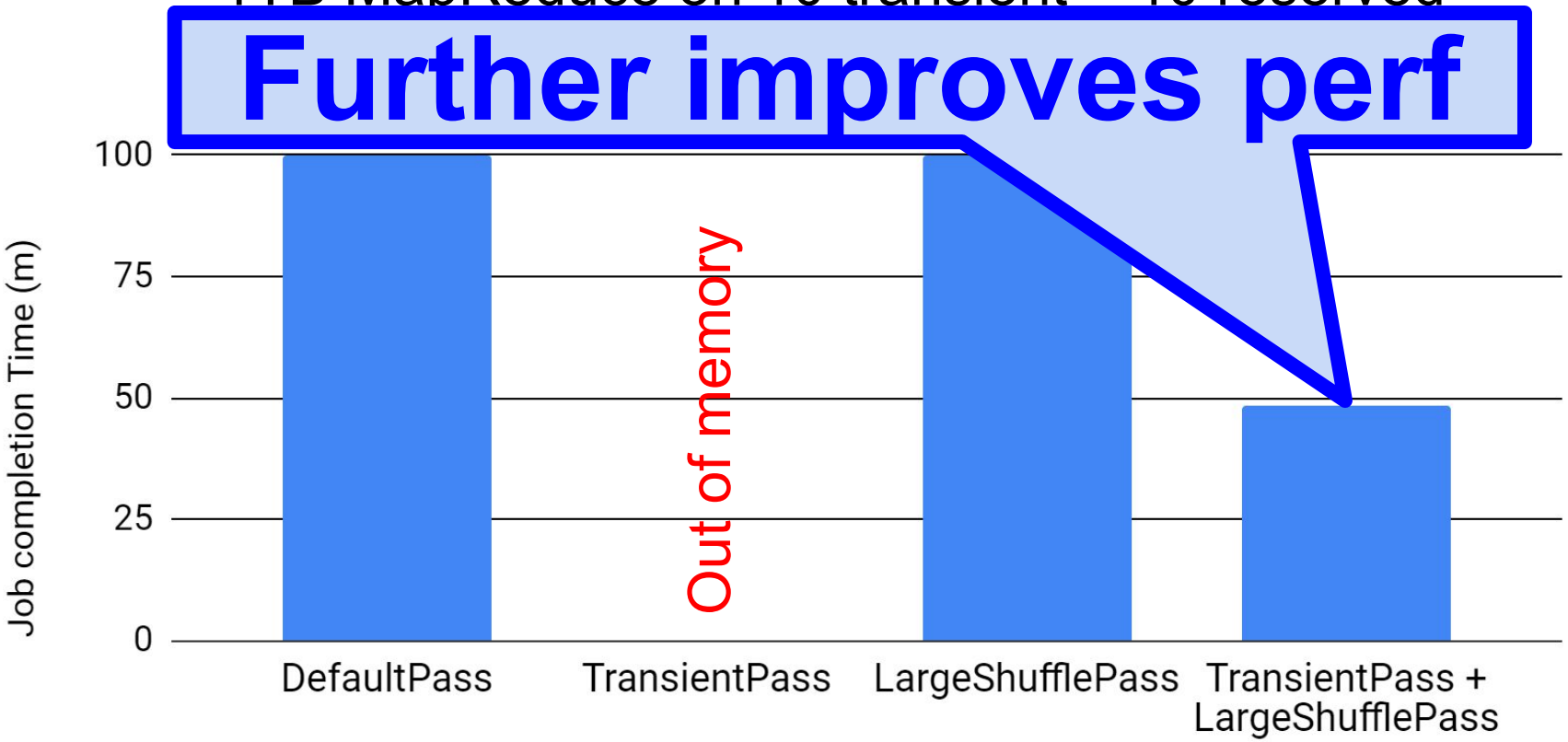
Large Shuffle on Transient Resources

⇒ 1TB MapReduce on 10 transient + 10 reserved



Large Shuffle on Transient Resources

⇒ 1TB MapReduce on 10 transient + 10 reserved



Summary: Apache Nemo

- Problem: Make it easy to optimize distributed dataflows
- Solution: Optimization passes that transform an intermediate representation (IR) DAG
- Result
  - Nemo outperforms a state-of-the-art Apache Spark with clean and simple optimization passes
  - Nemo is on par with specialized runtimes
  - Nemo further improves performance for scenarios with combined resource and data characteristics

https://nemo.apache.org

https://github.com/apache/incubator-nemo

Build Your Own Passes,
For Your Dataflow Research!