Apache Nemo: A Framework for Building Distributed Dataflow Optimization Policies

Youngseok Yang¹ Jeongyoon Eo¹ Geon-Woo Kim² Joo Yeon Kim³
Sanha Lee⁴ Jangho Seo¹ Won Wook Song¹ Byung-Gon Chun¹

¹Seoul National University ²Viva Republica ³Samsung Electronics ⁴Naver Corp.
Execution of Distributed WordCount

Application

dataset.map(word ⇒ (word, 1))
  .reduceByKey((l, r) ⇒ l + r)
Execution of Distributed WordCount

dataSet.map(word ⇒ (word, 1))
  .reduceByKey((l, r) ⇒ l + r)
Execution of Distributed WordCount

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

Application

Compiler

Runtime

![Diagram showing the execution of a distributed WordCount application with a Master, Executor, and DAG structure. The application processes a dataset by mapping each word to a tuple (word, 1) and then reducing these values by key to sum them up.]
Execution of Distributed WordCount

Master  Scheduler  Runtime DAG

Distributed Storage  Input Data
Execution of Distributed WordCount

Master  Scheduler  Runtime DAG

Datacenter Resource
Executor

Datacenter Resource
Executor

M

M

Distributed Storage
Input Data
Execution of Distributed WordCount

Datacenter Resource

Executor

Channel

M

Distributed Storage

Input Data

Master

Scheduler

Runtime DAG

Executor

Channel

M
Execution of Distributed WordCount

Master  Scheduler  Runtime DAG

Datacenter Resource

Executor (R)  Channel  Executor (R)

Distributed Storage  Input Data
Our Observations: **Resources & Data**

**Master**  
**Scheduler**  
**Runtime DAG**

**Datacenter Resource**

- **Executor**
- **Channel**
- **Distributed Storage**
- **Input Data**

**Datacenter Resource**

- **Executor**
- **Channel**
- **Distributed Storage**
- **Input Data**
Trend: Diverse Characteristics

Datacenter Resources
- Geographically-distributed
- Cheap transient

Input Data
- Large-scale
- 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

Master  Scheduler  Runtime DAG

Transient Resource

Executive  Channel

Reserved Resource

Executive  Channel

Distributed Storage  Input Data
(1) Cheap Transient Resources

Resource Eviction!
(2) Large-scale Data Shuffle

Distributed Storage

Large Input Data

Datacenter Resource
Executor
Channel
Datacenter Resource
Executor
Channel

Master
Scheduler
Runtime DAG
(2) Large-scale Data Shuffle

Many Disk Seeks!
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)**

- Apache Spark WordCount
- Apache Beam WordCount
- ....
Our Idea: Intermediate Representation (IR)

Apache Spark WordCount

IR

DAG

shuffle

M

R

Apache Beam WordCount

....
Our Idea: Intermediate Representation (IR)

Apache Spark WordCount  Apache Beam WordCount  ....

IR DAG  M shuffle  R

Optimization Pass f: irdag → irdag’

Easy! (Think Functions)
Our Idea: **Intermediate Representation (IR)**

![Diagram showing IR DAG and Optimized DAG with optimization pass](image)

- **Apache Spark** WordCount
- **Apache Beam** WordCount

- **IR DAG**
  - Master
  - Executor
  - Optimization Pass $f: \text{irdag} \rightarrow \text{irdag}'$

- **Runtime DAG**

---

23
Overall Workflow of Apache Nemo
Nemo User Job Submission (Easy!)

Application

[Pass C1][Pass C2][Pass C3]

Compile-time Passes
(List)

{Pass R1}

Run-time Passes
(Set)

e.g., Spark/Beam Application
## Nemo Applies Compile-time Passes

**Application**

<table>
<thead>
<tr>
<th>Nemo Compiler</th>
<th></th>
<th></th>
</tr>
</thead>
</table>
Nemo Applies Compile-time Passes

Application

irdag

Nemo Compiler
Nemo Applies Compile-time Passes

Application → irdag → Nemo Compiler

Pass C1 → Check correctness of the output IR DAG
Nemo Applies Compile-time Passes

Application

irdag

Nemo Compiler

Pass C1

Pass C2

Check correctness & Check conflict with C1
Nemo Applies Compile-time Passes

Application

irdag

Nemo Compiler

Pass C1

Pass C2

Pass C3

Check correctness & Check conflict with C1+C2
Nemo Applies Compile-time Passes

If all checks pass
Nemo Applies Compile-time Passes

Reflects the optimizations

Application

Compiler

Pass C3

irdag′ (optimized)

Master

Nemo Scheduler

Nemo Runtime

Executor

Nemo Channel

Runtime DAG′ (optimized)

Nemo Applies Compile-time Passes

Reflects the optimizations

Application

Compiler

Pass C3

irdag′ (optimized)

Master

Nemo Scheduler

Nemo Runtime

Executor

Nemo Channel

Runtime DAG′ (optimized)
Nemo Applies Run-time Passes

During job execution

Application

Nemo Compiler

Nemo Runtime

Master

Nemo Schedulor

Executor

Nemo Channel

Message
Nemo Applies Run-time Passes

Application

Nemo Compiler

irdag' (optimized) → Pass R1 → i rdag'' (optimized more)

Message

Correctness & Conflict checks

Master

Executor

Nemo Applies Run-time Passes

Nemo Compiler

Pass R1

Message

Correctness & Conflict checks

Master

Executor
Nemo Applies Run-time Passes

Updates lazily for correctness

Message
Master
Executor
Nemo Compiler
Nemo Runtime
Nemo Scheduler
Nemo Channel

irdag′ (optimized more)

Runtime DAG′ (optimized more)
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

When iterating down the input IR DAG,

1. Inserts Utility Vertices
2. Annotates Execution Properties

Applies a specific function
What A Pass Does

While traversing the input IR DAG,

(1) Inserts **Utility Vertices**

(2) Annotates **Execution Properties**
Passes We Implemented & Evaluated

GeoDistResourcePass
LargeShufflePass
TransientResourcePass
SkewCTPass
SkewRTPass
SkewSamplingPass
In This Talk (See Paper for Others)

GeoDistResourcePass

(1) LargeShufflePass

(2) TransientResourcePass

SkewCTPass

SkewRTPass

SkewSamplingPass

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
for each shuffle edge \( e \) in \( \text{irdag} \):
rv = RelayVertex()
rv.inEdge.set(DataFlow.Push, DataStore.Memory, Persistence.Discard)

(1) LargeShufflePass: Algorithm Execution Properties

- **Execution Properties**

  - **Shuffle, one-to-one**
  - **Relay**

  - **Push, Memory, Discard**

  - **In-memory shuffle**

  - **Execute \( M \) and Relay concurrently**

- **Do not persist data in memory**
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: Algorithm

shuffle, Push, Memory, Discard

one-to-one

M Relay R

R executes after Relay

one-to-one, Pull

Disk

Sequential disk access
(1) LargeShufflePass: Correctness

Original

Equivalent final outputs!

Optimized
(1) **LargeShufflePass**: Runtime Execution

**Master**

- Nemo Scheduler
- Runtime DAG

**Executor**

- Nemo Channel

**Distributed Storage**

- Large Input Data
(1) LargeShufflePass: Runtime Execution

Memory+Discard Shuffle

Distributed Storage

Large Input Data

Executor

Nemo Channel

Nemo Channel

Executor

Relay

M

Relay

M

Relay

M

Relay

M

Relay

M

Relay

M

Relay

M
(1) LargeShufflePass: Runtime Execution

[Diagram of distributed storage and large input data with Executor, Nemo Channel, and Relay connecting.]
(1) LargeShufflePass: Runtime Execution

Sequential Disk Access

Executor

Nemo Channel

Executor

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  Equivalent final outputs!  Optimized

shuffle → Reserved

shuffle, Push
(2) TransientResourcePass: Runtime

Master  Scheduler  Runtime DAG

Transient Resource
Executor  Channel
\(\text{M}\)  \(\text{M}\)

Reserved Resource
Channel  Executor
\(\text{R}\)  \(\text{R}\)

Distributed Storage  Input Data
(2) TransientResourcePass: Runtime

Master  Scheduler  Runtime DAG

Transient Resource

Executor
M
M

Channel

Reserved Resource

Executor
R
R

Channel

Distributed Storage

Input Data
(2) TransientResourcePass: Runtime

Moves data out quickly

Distributed Storage Input Data
LargeShufflePass + TransientResourcePass

shuffle

M → R

LargeShufflePass

shuffle, Push, Memory, Discard

one-to-one, Pull, Disk

M → R
LargeShufflePass + TransientResourcePass

LargeShufflePass

shuffle, Push, Memory, Discard

one-to-one, Pull, Disk

M → R

TransientResourcePass

Transient

Reserved

M → R

one-to-one, Pull, Disk

Reserved

M → R

M → R

shuffle, Push, Memory, Discard

Relay

Transplant

Reserved

Relay

Reserved

Relay

R

M

High

Low
LargeShufflePass + TransientResourcePass

Correct & No Conflict

shuffle, Push, Memory, Discard

one-to-one, Pull, Disk

Transist
Reserved

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
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 \textbf{(Lower is Better)}

⇒ MapReduce on 20 AWS EC2 h1.4xlarge instances

![Bar chart showing job completion times for different input data sizes and frameworks.](image-url)
Large Shuffle (Lower is Better)

⇒ MapReduce on 20 AWS EC2 h1.4xlarge instances

Outperforms Spark
Transient Resources \textbf{(Lower is Better)}

⇒ ALS on 10 transient + 2 reserved EC2 instances
Transient Resources *(Lower is Better)*

⇒ ALS on 10 transient + 2 reserved EC2 instances

On par with Pado
Large Shuffle on Transient Resources

⇒ 1TB MapReduce on 10 transient + 10 reserved
Large Shuffle on Transient Resources

⇒ 1TB MapReduce on 10 transient + 10 reserved

Further improves perf
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!