Apache Nemo: A Framework for Building Distributed Dataflow Optimization Policies

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

\textsuperscript{1}Seoul National University  \textsuperscript{2}Viva Republica  \textsuperscript{3}Samsung Electronics  \textsuperscript{4}Naver Corp.
Distributed data processing applications and runtime

Application: \(\text{dataCollection.map(..).reduce(..)}\)
(e.g., DryadLINQ, Spark RDD, Beam)

Runtime:
Directed-Acyclic Graph (DAG)
(e.g., Dryad, Tez, Spark, …)
Policy interfaces for scheduling/communication

**Application-level** (e.g., Optimus)

```java
register(stat, subqueryA, subqueryB)
```

- Correctness: O
- Fine Control: X

**Runtime-level** (e.g., Dryad)

```java
onEvent(event) {
    modify(DAG, event)
}
```

- Correctness: X
- Fine Control: O
Our idea: Transform an intermediate representation

- **IR DAG**: Provides both correctness and fine control
- **Optimization pass**: A function to transform the IR DAG
- **Runtime extensions**: Apply the optimizations
Experimental Evaluation

● **Deployment scenarios**
  ○ Resources: Transient, Geo-distributed
  ○ Data: Large, Skewed

● **Results**
  ○ Nemo is on par with specialized runtimes
  ○ Nemo further improves performance for scenarios with combinations of resource/data characteristics
USENIX ATC 2019, 2:20PM, Track II, on July 10

https://nemo.apache.org/