DryadLINQ

A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language

Overview

- Motivation for DryadLINQ
- Design
- Implementation
- Performance
- Q & A

Motivation

- More machines + more code = more problems
- Need to simplify!
- **Solution → Higher-level Language**

Design Goals

- Easy to write
- General Purpose
- Efficient

Existing Solutions

- SQL
  - Difficult to express common programming constructs
- MapReduce
  - Not flexible enough
  - Inefficient for some use cases
- Dryad
  - Have to specify DAG
  - Harder to write

DryadLINQ

- **Dryad**
  - Execution Engine
- **L**anguage **IN**tegrated **Q**uery
  - Declarative + Imperative + Object Oriented

LINQ vs. SQL

- Expressions can be directly embedded in code
- Allow direct calls to C#, F#, ... functions
- Evaluated by Dryad

LINQ expressions

- Declarative

```
var adjustedScoreTriples =
    from d in scoreTriples
    join r in staticRank on d.docID equals r.key
    select new QueryScoreDocIDTriple(d,r);
```

- OO

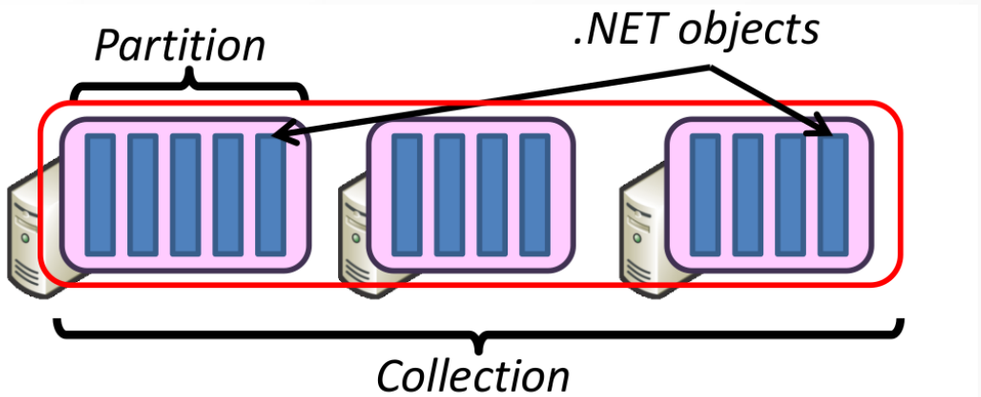
```
var adjustedScoreTriples =
    scoreTriples.Join(staticRank,
    d => d.docID, r => r.key,
    (d, r) => new QueryScoreDocIDTriple(d, r));
```

API

- Compatible with many .NET Languages (e.g. C#)
- DryadLINQ vs. SPARK
  - Language embedded
  - Compiler Hints
  - Functions must have no side effects
  - Non-interactive

Data Model

- IEnumberable<T> vs. RDD's
  - Distributed
  - Strongly typed
  - Mutable
  - Nested generics
  - Lazy Evaluation



Execution

- Similar to SQL query plan
- Create execution plan graph
- Static Optimizations
- Pass to Dryad Job Manager
- Dynamic Optimzations

Expression Execution

```
// Do Stuff ...

var DT = ToDryadTable(X);

foreach (row in DT) {

      // Do more stuff ...

}
```

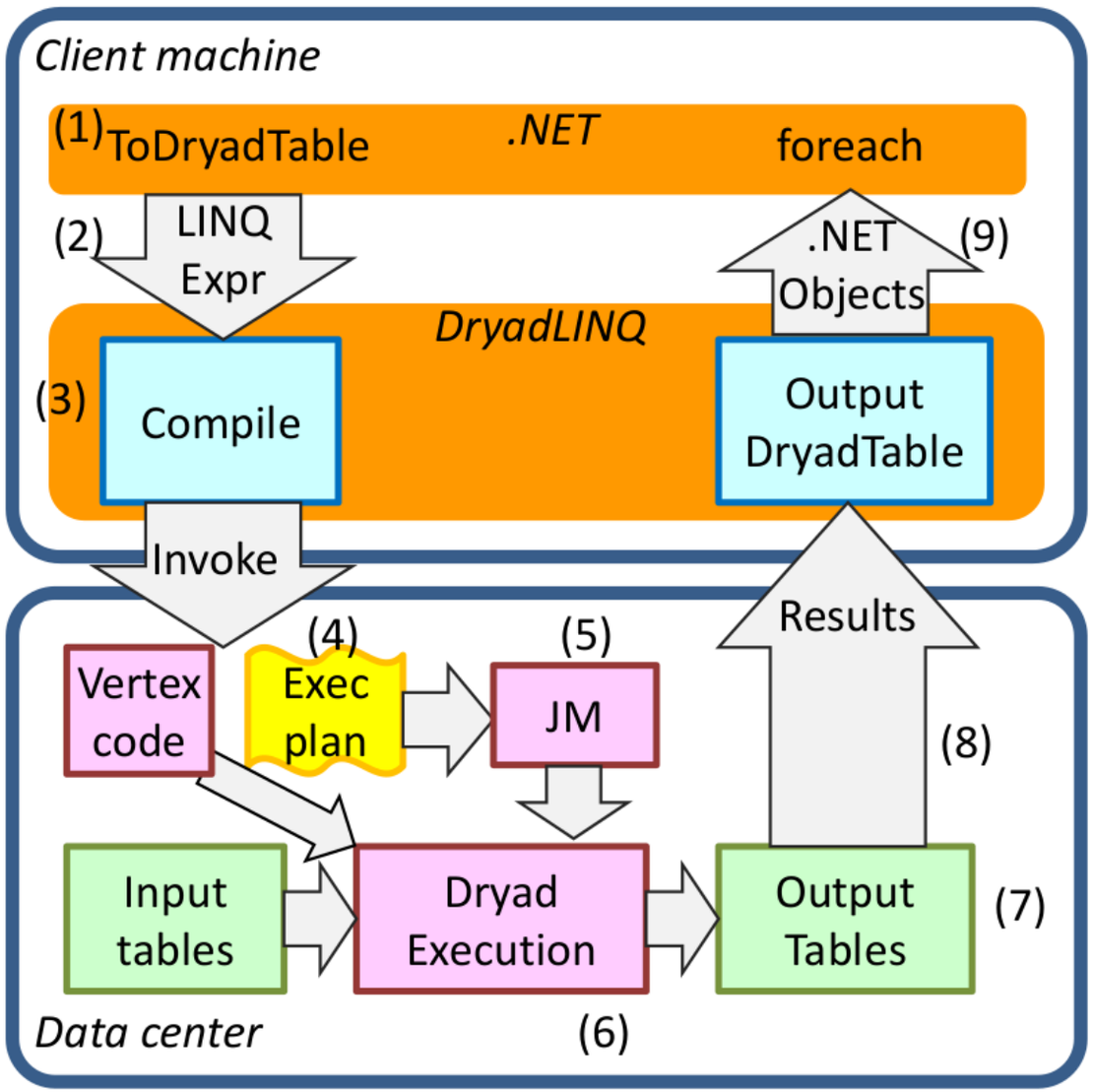
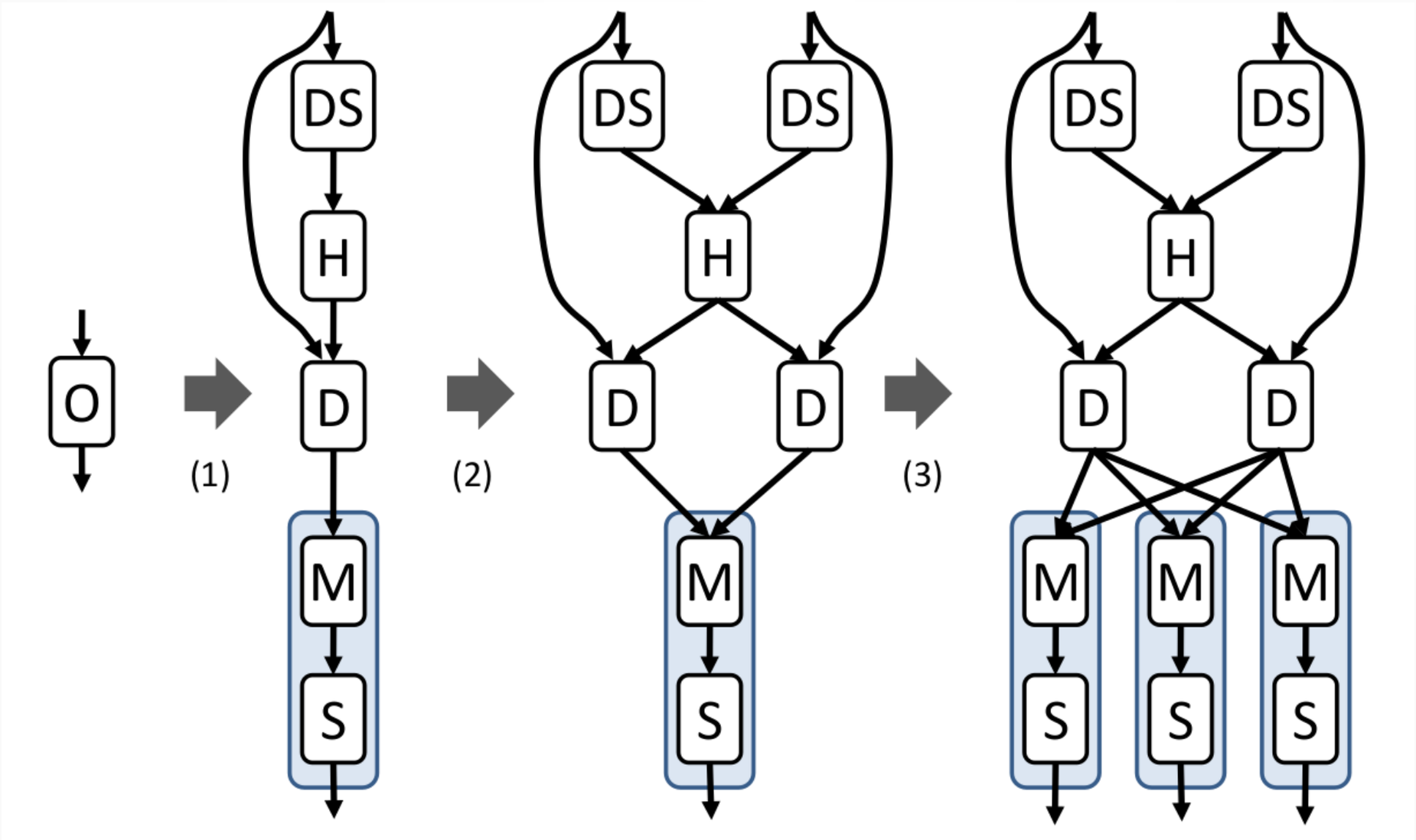


Figure 2: *LINQ-expression execution in DryadLINQ.*

Optimizations

- Static
  - I/O reduction
  - Pipelining
  - Eager aggregation
- Dynamic
  - Partitioning
  - Topology aware aggregation
  - Lazy evaluation

Example: OrderBy

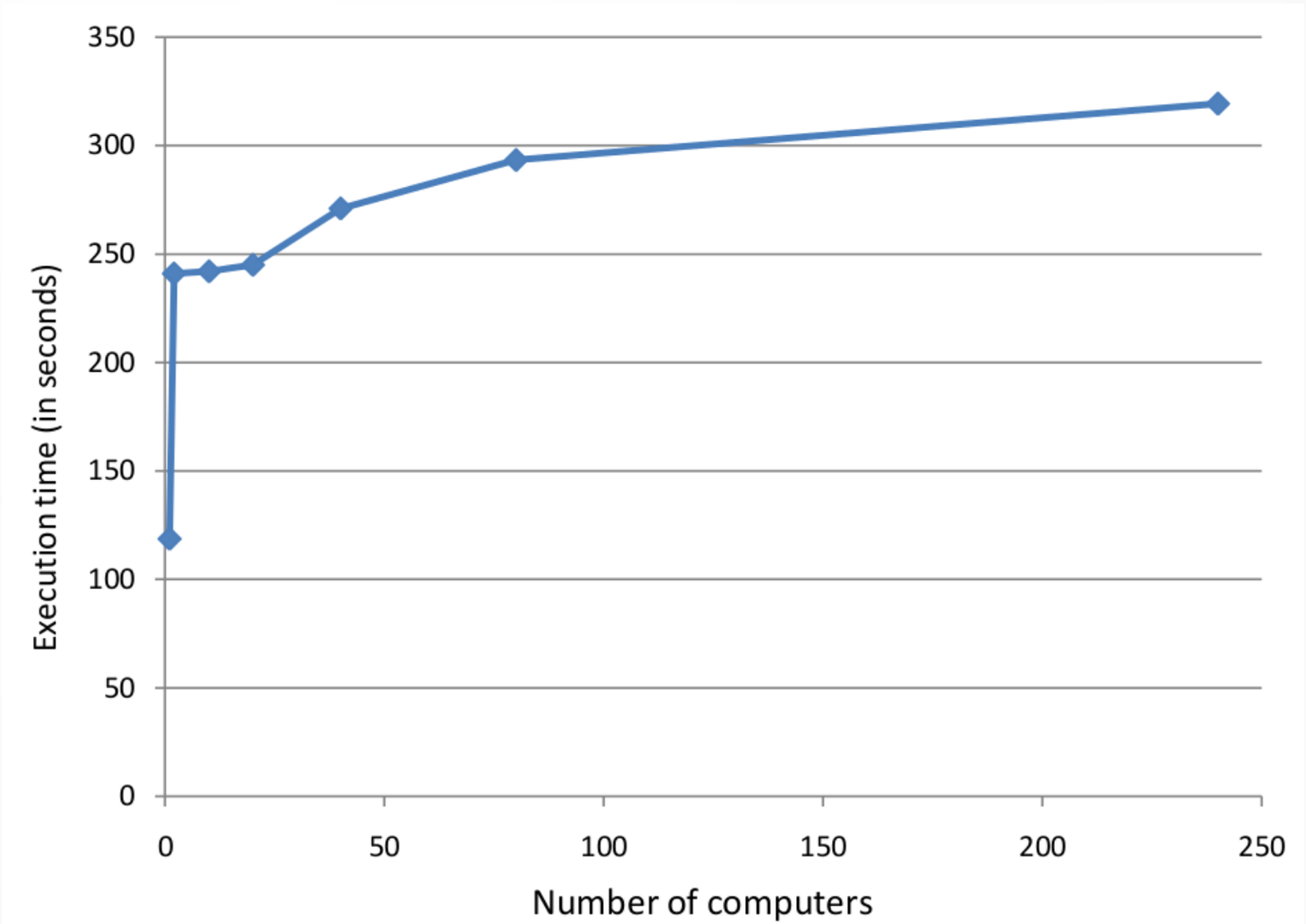


Performance

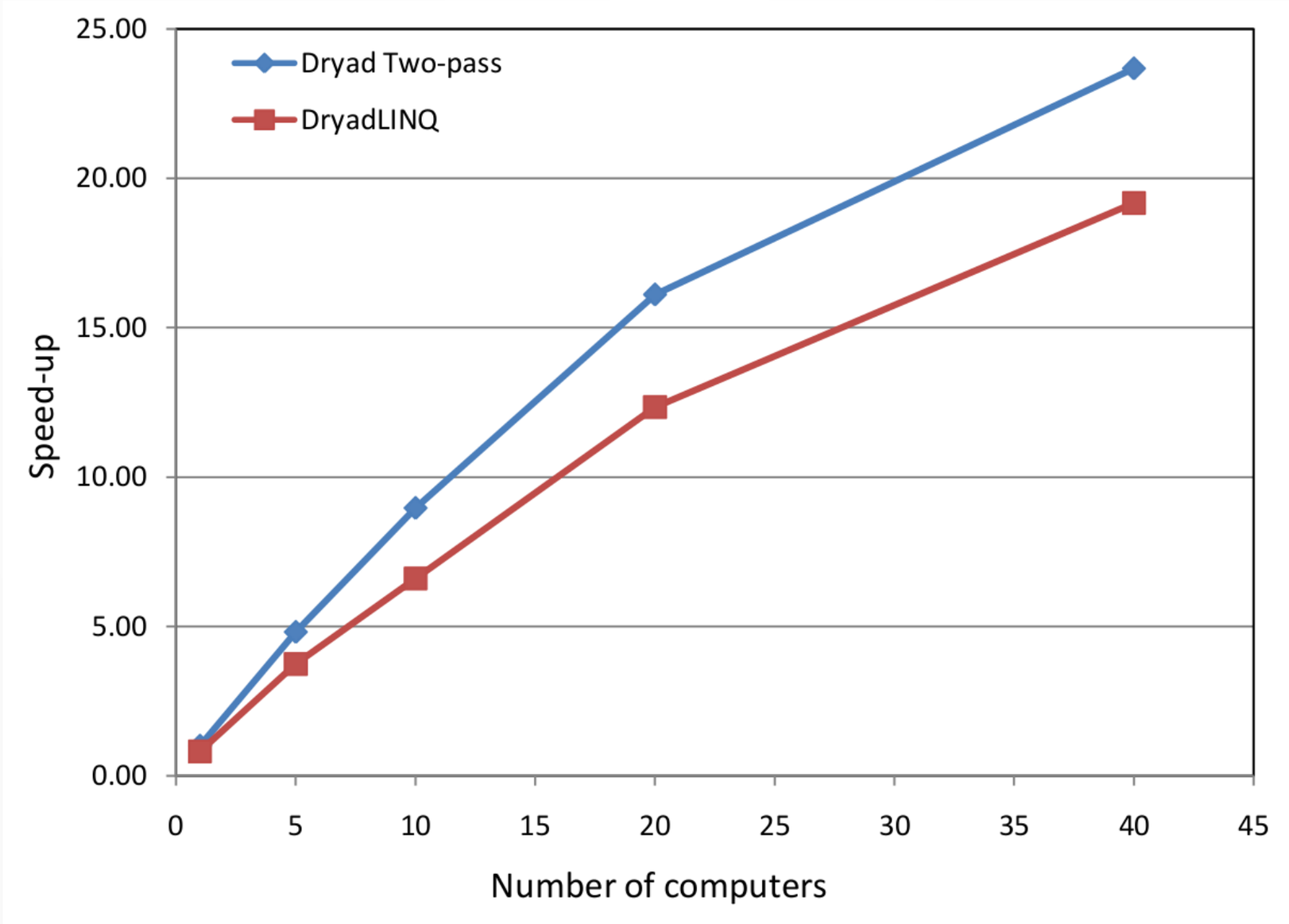
- TeraSort
- Skyserver Q18 computation

TeraSort

~ 3.87 Gb per machine



Comparison


25.00
20.00
15.00
10.00
5.00
0.00
Speed-up
Dryad Two-pass
DryadLINQ
0
5
10
15
20
25
30
35
40
45
Number of computers

Q & A