DryadLINQ

Overview

- Language (LINQ+)
  - A high-level programming language
  - Strongly typed
  - Like writing a sequential program
  - imperative+declarative
- Linking language and run-time together
  - EPG
- Run-time system (Dryad)
  - Scheduling, data partition, fault-tolerance
  - Dynamic optimization
- **Transparently** achieve **data parallelism**

Language

LINQ

- Language constructs that manipulate data sets
  - Select, join, groupby, orderby, where, …

Example 1

```
// Object-oriented syntax for the above join
var adjustedScoreTriples =
  scoreTriples.Join(staticRank,
    d => d.docID, r => r.key,
    (d, r) => new QueryScoreDocIDTriple(d, r));
var groupedQueries =
  adjustedScoreTriples.GroupBy(s => s.query);
var rankedQueries =
  groupedQueries.Select(
    g => TakeTopQueryResults(g));
```

Example 1

```
// Object-oriented syntax for the above join
var adjustedScoreTriples =
  scoreTriples.Join(staticRank,
    d => d.docID, r => r.key,
    (d, r) => new QueryScoreDocIDTriple(d, r));
var groupedQueries =
  adjustedScoreTriples.GroupBy(s => s.query);
var rankedQueries =
  groupedQueries.Select(
    g => TakeTopQueryResults(g));
```

Query	docID	Score
Q1	D1	s1
Q2	D1	
Q1	D2	

JOIN

Key	SScore
D1	ss1
D2	…

Query	docID
Q1	<D1, …>, <D2, …>
Q2	D1

Query	docID	NScore
Q1	D1	F(s1,ss1)
Q2	D1	
Q1	D2	

Example 2

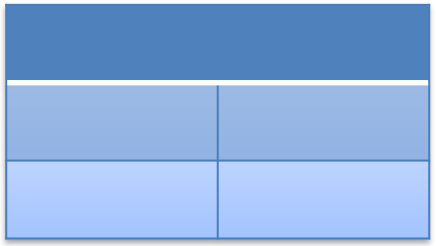
- LINQ version of MapReduce

map (<key, value>) => (<Ikey, value>, <Ikey, value>, …)

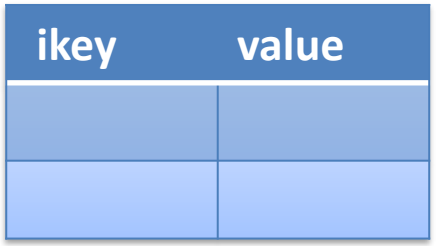
reduce (Ikey, <value, …>) => <Ikey, value>

```
Mapped=Source.selectMany(map);
Grouped =Mapped.groupBy(…);
Grouped.selectMany(reduce);
```

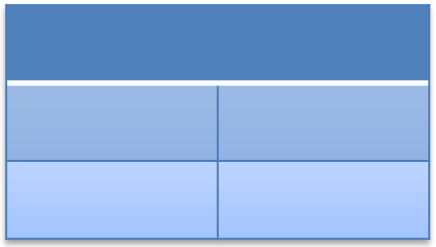
source



mapped



output



LINQ

- Language constructs that manipulate data sets
  - join, from, where, groupBy, orderBy, select, …
- Features
  - Can be written in either declarative/imperative way
  - Deferred execution
  - Strongly typed
  - **Flexible in how these computations are implemented**

DryadLINQ

- Data model
  - Distributed, partitioned implementation of LINQ collections
    - Stored by distributed file systems/NTFS/SQL tables
    - Three ways of partition
    - Corresponding metadata part of the object

DryadLINQ (2)

- Example

```
var input = GetTable<LineRecord>("file://in.tbl");
var result = MainProgram(input, ...); /*side-effect free*/
var output = ToDryadTable(result, "file://out.tbl");
```

DryadLINQ(3)

- Annotations
  - [Associative], [homomorphic], …
- Data re-partitioning
  - HashPartition<T,K>, RangePartition<T,K>
- …

How to go from LINQ to Dryad?

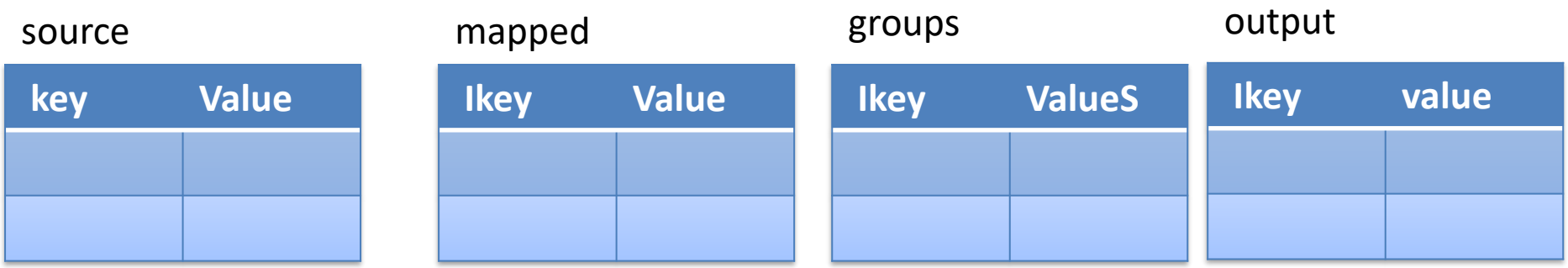
```
var input = GetTable<LineRecord>("file://in.tbl");
var result = MainProgram(input, ...); /*side-effect free*/
var output = ToDryadTable(result, "file://out.tbl");
```

Execution Plan Graph generation

- What is an EPG
  - Edge
  - Vertex (will be dynamically split)
  - Properties
  - Not a tree!
- EPG is a "**skeleton**" of the data-flow
  - Code is generated for each vertex
  - Multiple vertices per "stage"

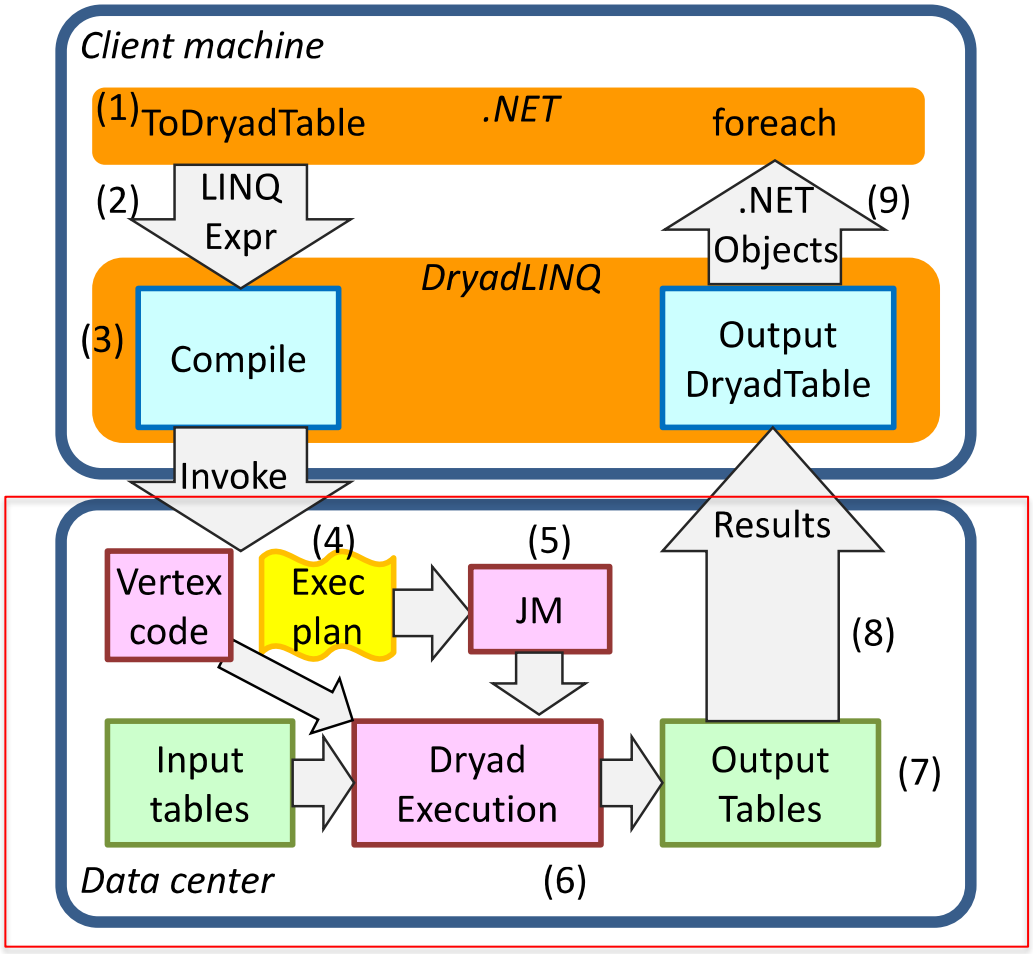
EPG of LINQ-MapReduce

```
var mapped = source.SelectMany(mapper);
var groups = mapped.GroupBy(keySelector);
return groups.SelectMany(reducer);
```

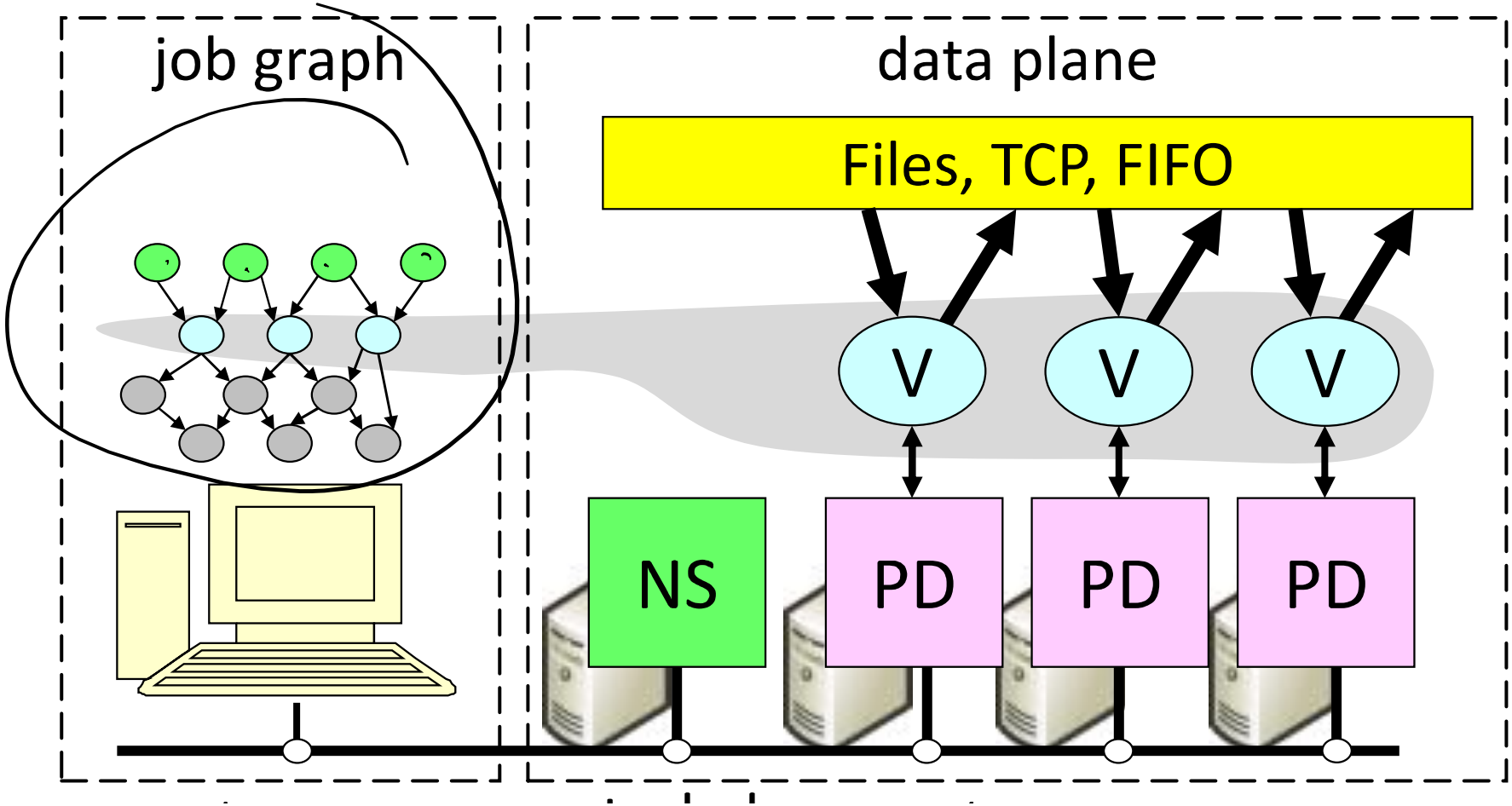


Dryad Run-Time

Flow


Client machine
(1) ToDryadTable
.NET
foreach
(2)
LINQ Expr
.NET Objects
(9)
DryadLINQ
(3)
Compile
Output DryadTable
Invoke
Results
(4)
(5)
Vertex code
Exec plan
JM
(8)
Input tables
Dryad Execution
Output Tables
(7)
Data center
(6)

Dryad architecture



Performance, failure tolerance, etc.

- Are all the workers instantiated at the same time?
- When does a worker started?

- How to find an appropriate worker for a vertex?
- What if a worker computer fails?
- What if the job manager fails?
- What if a worker is much slower than other worke in the same stage?

Optimization

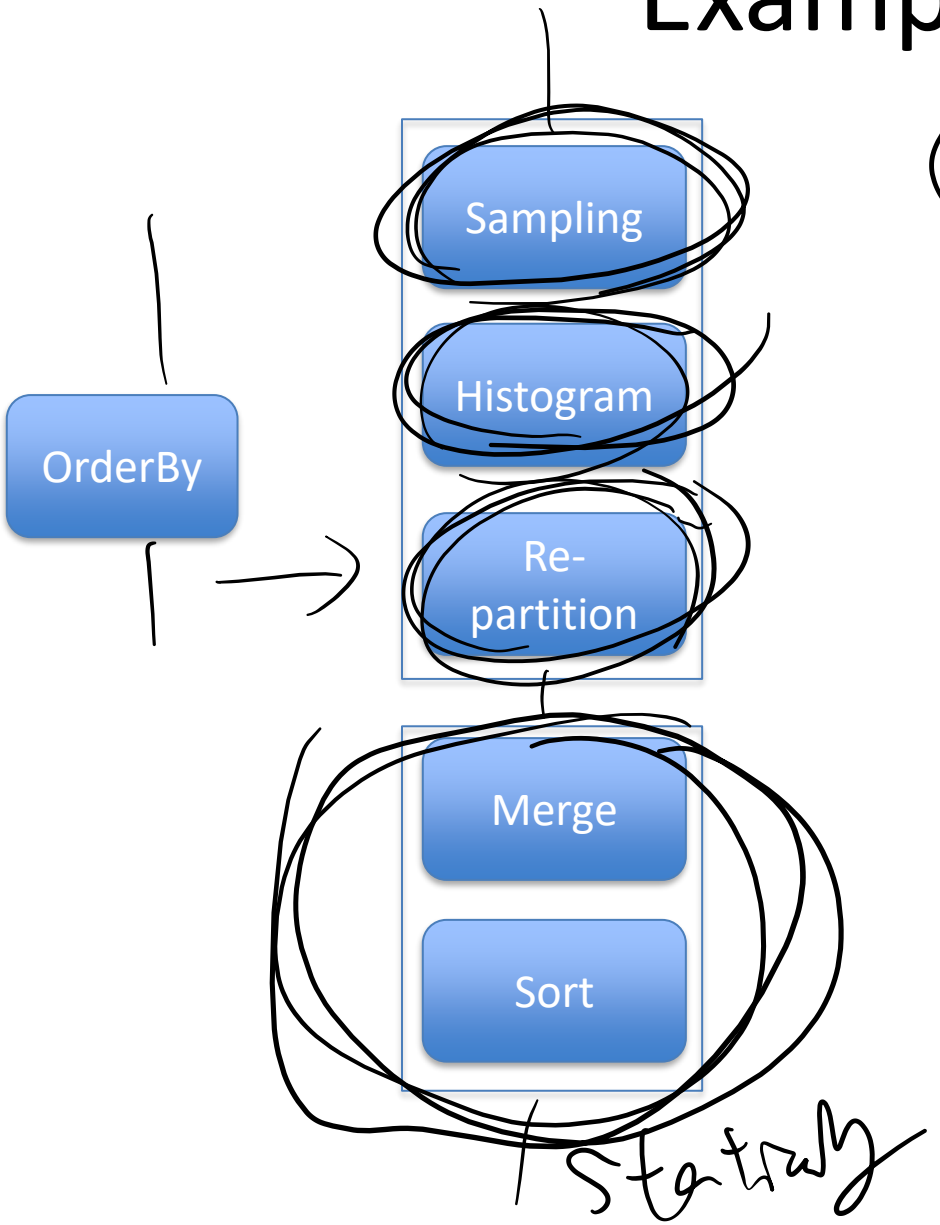
Static & Dynamic Optimization

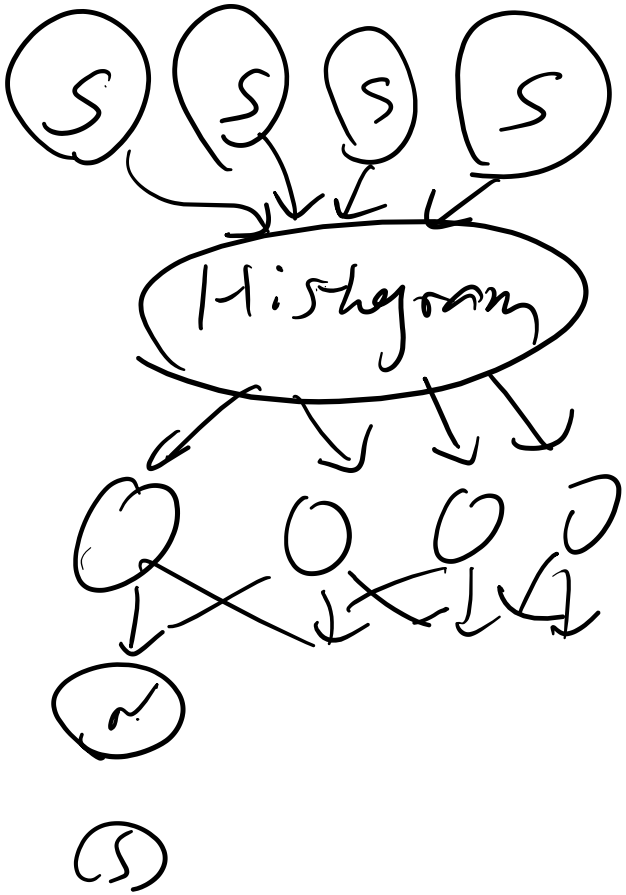
- Static optimization
  - Pipelining
  - Removing redundancy
  - Eager aggregation
  - I/O reduction
- Dynamic optimization
  - Adding partial aggregation
  - Change the number of instances/partitions

OrderBy<TSource,TKey>(IEnumerable<TSource>, Func<TSource,TKey>)

Sorts the elements of a sequence in ascending order according to a key.

Example 1



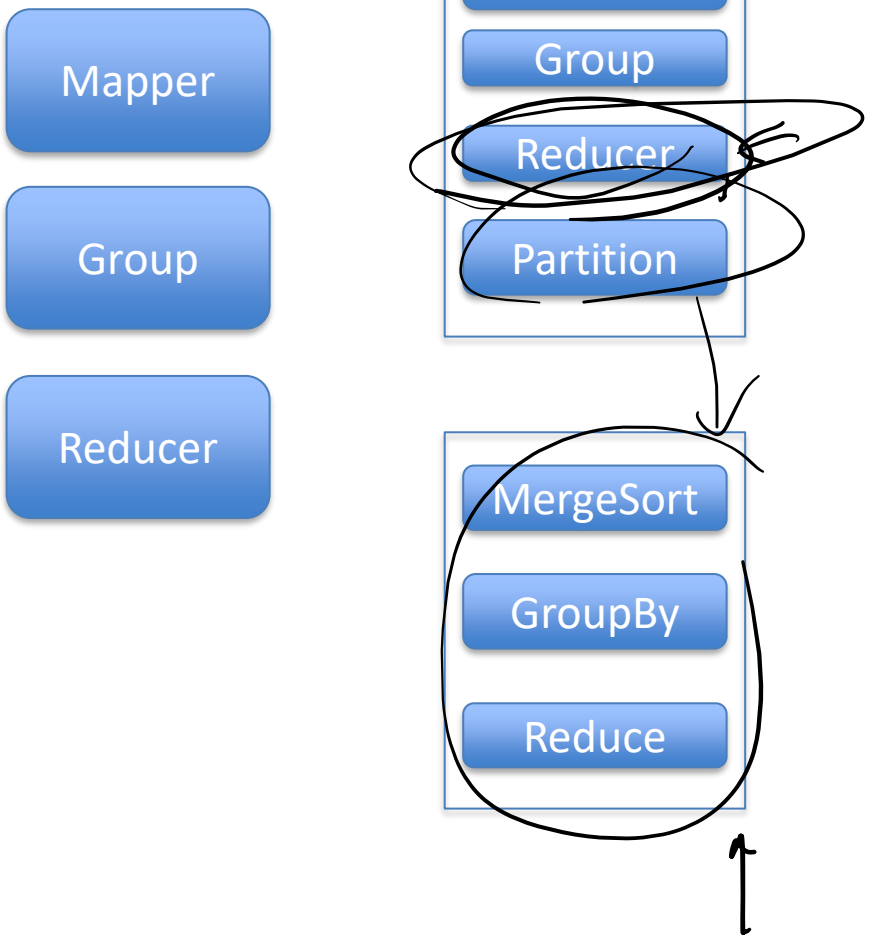


Examples

- How to optimize MapReduce

```
Mapped=Source.selectMany(map);
Grouped =Mapped.groupBy(…);
Grouped.selectMany(reduce);
```

Mapreduce


Mapper
Group
Reducer
Mapper
Group
Reducer
Partition
MergeSort
GroupBy
Reduce

Summary

- The flow of DryadLINQ execution
- The language
- The run-time
- The optimization

- Comparing with MapReduce …