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, ...
// 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));
Example 2

• LINQ version of MapReduce

map (<key, value>) => (<lkey, value>, <lkey, value>, ...)
reduce (lkey, <value, ...>) => <lkey, value>

Mapped=Source.selectMany(map);
Grouped =Mapped.groupBy(...);
Grouped.selectMany(reduce);
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\langle T,K\rangle, RangePartition\langle T,K\rangle

• ...

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

<table>
<thead>
<tr>
<th>source</th>
<th>mapped</th>
<th>groups</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Value</td>
<td>Ikey</td>
<td>ValueS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ikey</td>
<td>value</td>
</tr>
</tbody>
</table>
Dryad Run-Time
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 workers 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

Diagram:
- OrderBy
- Sampling
- Histogram
- Repartition
- Merge
- Sort

Legend:
- S
- Histogram
Examples

• How to optimize MapReduce

\[
\text{Mapped} = \text{Source.selectMany(map)};
\]

\[
\text{Grouped} = \text{Mapped.groupBy(...)};
\]

\[
\text{Grouped.selectMany(reduce)};
\]
Mapreduce

Mapper
Group
Reducer
Partition

MergeSort
GroupBy
Reduce

Hello world!

Today is hot!
Summary

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

• Comparing with MapReduce ...