SCOPE and REEF

Quan Fang
Xiaofeng Tao

9/25/2013
Background

- Dryad - MS Research project to investigate programming models for running distributed programs
  - Similar goals to Hadoop and map reduce
- SCOPE - Query language powered by Dryad
  - Similar goals to Hive
- Dryad and SCOPE never commercialized!
- Microsoft developing open source technologies for Hadoop with REEF
SCOPE: Structured Computations Optimized for Parallel Execution

A declarative and extensible scripting language for efficient parallel processing on massive datasets
SCOPE

• Structured Computations Optimized for Parallel Execution
• Proprietary internal Microsoft technology for using COSMOS
• Declarative language resembling SQL
• Highly extensible with C#
• Compiled into Microsoft Dryad execution plan
COSMOS

- Internal Microsoft system for distributed storage and parallelized processing of big data
- Stores over 2 petabytes of new data per day
- SCOPE is the query language for Cosmos

(Chaiken 2008)
Writing a Select Query

- Syntax is similar to SELECT statement from SQL
- USING keyword used to specify C# extractor class to get the data

```
SELECT query, COUNT(*) AS count
FROM "search.log"
USING LogExtractor
GROUP BY query
HAVING count > 1000
ORDER BY count DESC;
OUTPUT TO "qcount.result";
```
C# SCOPE Extensions

- C# expressions are directly used in queries
- C# functions can be defined and used
- User defined type (C# class) can be a column

```csharp
SELECT Timestamp.ToUniversalTime() + "::" + Query AS Q
FROM "search.log" USING LogExtractor
WHERE StringOccurs(Query, "#") > 1

#CS
public static int StringOccurs(string str, string ptrn)
{
    int cnt=0;
    int pos=-1;
    while (pos+1 < str.Length)
    {
        pos = str.IndexOf(ptrn, pos+1);
        if (pos < 0) break; cnt++;
    }
    return cnt;
}
#EndCS
```

(Chaiken 2008)
SCOPE Process Command

- Process provides custom data transformation and mapping

```csharp
PROCESS Query, Timestamp, Market, User, Browser
USING TrimProcessor

#CS
public class TrimProcessor : Processor
{
    // This function trims all string valued columns
    public override IEnumerable<Row> Process(RowSet input, Row outRow, string[] args)
    {
        foreach (Row row in input.Rows)
        {
            row.Copy(outRow);
            for (int i = 0; i < row.Count; i++)
            {
                if (outRow.Schema[i].Type == ColumnDataType.String)
                {
                    outRow[i].Set(outRow[i].String.Trim());
                }
            }
            yield return outRow;
        }
    }
}

#ENDCS
(Chaiken 2008)
SCOPE Reduce/Combine Commands

- Reduce provides custom data grouping and aggregation
- Combine provides custom data joins

```csharp
REDUCE RowSet1
ON ColumnA
USING TrimProcessor

#CS
public class CountReducer : Reducer
{
    public override IEnumerable<Row> Reduce(RowSet input, Row outputRow, string[] args)
    {
        int count = 0;
        foreach (Row row in input.Rows)
        {
            if (count == 0)
                outputRow[0].Set(row[0].String);
            count++;
        }
        outputRow[1].Set(count.ToString());
        yield return outputRow;
    }
}
#ENDCS
```
SCOPE Compiler and Optimizer

- Compiled and optimized into Dryad execution plan based on directed acyclic graph
  - Edges are communication channels
  - Vertices are processing nodes

(Isard et al. 2007)
SCOPE Compiler and Optimizer

• Advantages of Dryad over Map Reduce
  • More general and subsumes all map reduce functionality
  • Less replicated storage in big jobs
  • Can “split” different output types from vertices
  • Data dependent re-partitioning at runtime
  Ex: “A” nodes split into “*” and “+” nodes by network

(Isard et al. 2007)
SCOPE Compiler and Optimizer

• PeriSCOPE for code optimizations of compiled SCOPE jobs
  • Column Reduction
  • Early Filtering
  • Smart Cuts

• Runtime optimizations
  • Fault tolerance
  • Data dependent re-partitioning

(Guo et al. 2012)
SCOPE Summary

• Never released Microsoft query language for using COSMOS
• Resembles SQL and very extensible with C#
• Process, Reduce, and Combine commands analogous to map, reduce, and merge
• Compiles into directed acyclic graph based execution plan
• Data dependent optimizations at runtime
REEF: Retainable Evaluator Execution Framework

Build persistent (cross-job) caches and cluster-wide services to support high-performance iterative graph processing and machine learning algorithms

Microsoft Cloud and Information Services Lab (CISL)
Hadoop

- **Hadoop Common**: the common utilities that support the other Hadoop facility
- **Hadoop Distributed File System (HDFS)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop MapReduce**: A software framework for distributed processing of large data sets on compute cluster.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
YARN

- Yet Another Resource Negotiator
- Resource manager for Hadoop2.x
- Allocates compute containers to competing jobs
REEF in the Stack

- SQL / Hive
- ... (omitted)
- ... (omitted)
- Machine Learning

REEF

YARN / HDFS
REEF in the Stack (Future)

SQL / Hive
...
...
Machine Learning

Logical Abstraction
Operator API and Library
REEF
YARN / HDFS
REEF builds on YARN

• **Retainability** of hardware resources across tasks and jobs

• **Composability** of operators written for multiple computational frameworks and storage backends.

• **Cost modeling** for data movement and single machine parallelism.

• **Fault handling** including checkpointing of task state, and deterministic task invocations.

• **Elasticity** REEF’s checkpointing and preemption mechanisms allow jobs to adapt as resource allocations changes.
Digital Shoebox Architecture

- Relational Queries
- Machine Learning
- Operators
  - REEF
  - YARN
  - HDFS-as-Cache
- WAS

Analysis Engine

Compute Fabric

Tiered Storage
REEF: Computation and Data Management

Extensible Control Flow

- **Job Driver**: Control plane implementation. 
  *User code* executed on YARN’s Application Master

- **Activity**: 
  *User code* executed within an **Evaluator**.

- **Evaluator**: Execution Environment for **Activities**. One **Evaluator** is bound to one YARN Container.

Data Management Services

- **Storage**: 
  Abstractions: Map and Spool
  Local and Remote

- **Network**: 
  Message passing
  Bulk Transfers
  Collective Communications

- **State Management**: 
  Fault Tolerance
  Checkpointing

Tang and Wake
Configuration Manager Event framework
REEF Key Abstractions

- **Job Driver**: The user-supplied control logic. There is exactly one Driver for each Job. The duration and characteristics of the Job are determined by this module.

- **Activity**: User-supplied logic that performs the data processing. Activities are a generalization of Hadoop’s Map and Reduce **Tasks**.

- **Evaluator**: The runtime for Activities.

- **Services**: Objects and daemon threads that are retained across Activities that run within an Evaluator.
Data Management Services

- A set of libraries that provides mechanisms for networking, storage, checkpointing, and other data processing infrastructure. **Identifier** is the basis for many mechanisms.

- **Two APIs:** Driver API requests and configures Services and Activity API provides access to the actual functionality.
Wake: Data And Control Planes

- A state-of-art event-driven programming framework
- Extensibility to multiple I/O subsystems
- Profiling and bottleneck analysis tools
- Inlining of event handlers
- Immutable dataflows
TANG: Job Configuration

- A configuration manager and dependency injector

- **Static checking** of configurations allows jobs to fail earlier.

- **Language independence** Drivers written in one language can configure Activities written in multiple languages.

- **Separation of concerns** between configuration processing and Driver implementations.

- **Graceful handling of ambiguity** for configurations that require further specialization.
- What they have built on top of REEF?
- **MapReduce Library**
  - Runs Hive and Pig
  - Excellent starting point for M/R optimizations: Caching, Shuffle, Map-Reduce-Reduce, Sessions...

- **Machine Learning algorithms**
  - Scalable implementations: Decision Trees, Linear Model. Soon: SVD
  - Excellent starting point for: fault awareness in ML
Questions ?
References


