MapReduce Simulator

Andrey Kurenkov
Troy O'Neal
Matthew O'Shaughnessy
Mentor: Xiao Yu
Advisor: Dr. Bo Hong

Introduction

- MapReduce provides paradigm to easily process data in parallel
- Parallel processing example: word count
- How it works: Map & Reduce tasks
  - Map tasks transform data to key/value pairs
  - Reduce tasks sort and combine pairs
- Apache Hadoop MapReduce implementation used as model
- Problem: unable to estimate operation expense effectively
- Solution: a simulator

Problem Statement

- Design and implement an extensible MapReduce simulation framework
  - Improve abstraction of MapReduce job specification
  - Determine granularity needed for sufficient accuracy

Significance

- Assists Hadoop cluster administrators
  - Cluster configuration
  - Scheduling policy
- Helps programmers write efficient Map and Reduce functions
- Improve Hadoop itself
  - Custom scheduling policies
  - Failure handling
  - Data placement
- Eases research in related areas
  - Use of GPUs in MapReduce context

Goals

- More powerful
  - Does not rely on external trace
- More extensible
  - No hardcoded functionality
- More configurable
  - More general job description
- Maintaining accuracy
  - Comparable to existing works

Challenges

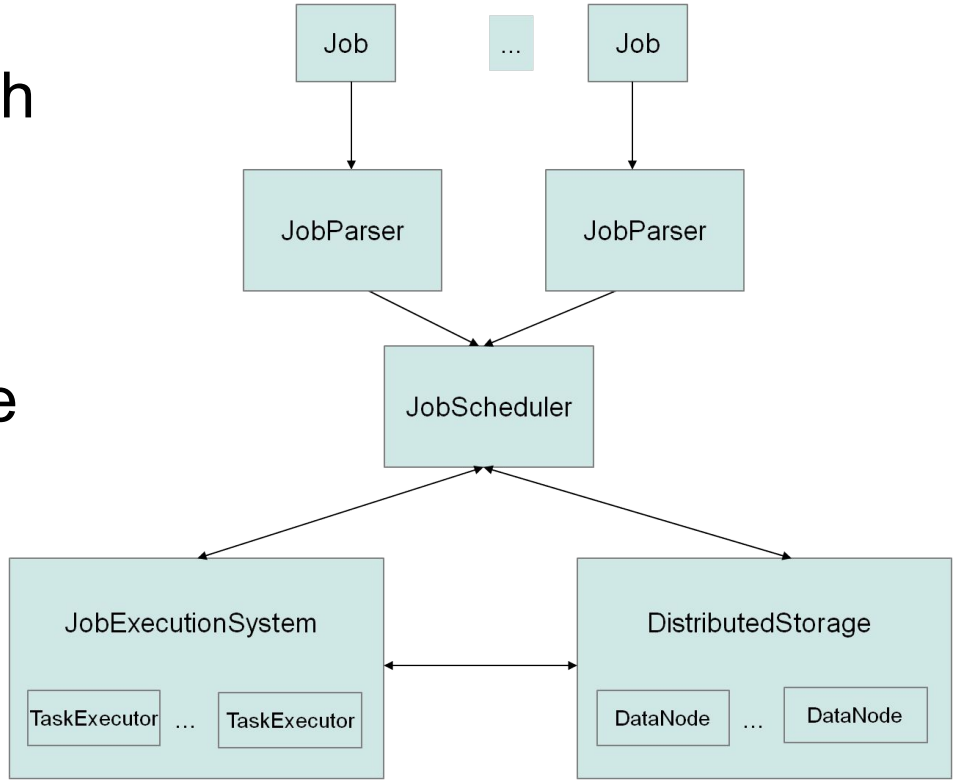
- Breadth of MapReduce uses requires careful framework design
- Balance between accuracy and performance
  - Disk I/O
  - OS scheduling and multithreading
  - Network performance
- Choice of heuristic representations
  - Mathematical model necessary
- Evaluation necessitates actual MapReduce cluster

Methods and tasks

- Methods involve typical OOP concepts
  - Make use of Reflection and Java 7.0 API
  - Allow extension of classes to ensure flexibility
  - Thorough documentation
- Tasks broken down into three stages
  - Develop a robust simulation framework
  - Implement base modules to enable simulation
  - Continually revise and test for stability and accuracy

Current work

- Selection of simulation engine
- Maximum flexibility through a custom-built simulation engine
- Entities in engine correspond to MapReduce components
  - Task executors
  - Schedulers
  - Job descriptors
  - Storage system



Questions?

MapReduce Operation

- Input file distributed into splits
- Master node assigns Map and Reduce tasks to worker nodes
- Map parses input split and produces intermediate key/value pairs
- Reduce sorts and then combines intermediate key/value pairs

Related Works

- SimMR: Univ. of Illinois Urbana-Champaign; HP Labs
  - Narrowly focused on scheduler selection and Map/Reduce slot allocation
- Rumen/Mumak: Hadoop
  - Powered by a previous job trace
  - Narrowly focused on scheduler selection
- MRSim: Brunel University
  - Many hardcoded properties (limited extensibility)
- MRPerf: Virginia Tech; IBM
  - Flexibility limited to certain configurable items