MapReduce with P2P Implementing Self-* CS647 Pre-Proposal, Spring 2009

Omar Badran
Jordan Osecki
Bill Shaya

Research Area

- Exploring the MapReduce and P2P technologies and how they can be combined
- There are multiple Self-* issues that can be addressed to make this type of distributed system more efficient
    - Issues regarding how to make a P2P system perform MapReduce
    - Issues regarding if nodes perform horribly compared to their peer Worker nodes
    - Issues regarding if nodes (Masters or Workers) fail completely, not performing their workload

Problems and Interests

- When the MapReduce system is running on a P2P network, issues regarding how the mapping and reducing occur
  - How will the jobs be distributed out and then re-combined?
- When the work is split up between multiple nodes, only as good as the weakest link
  - How can we analyze this to make tasks resilient to slower nodes?
- When nodes go down, their work is lost, which can cause a huge delay
  - How can these nodes be watched to ensure if they fail, the delay time is not catastrophic? How do we handle a Worker versus a Master failing?

Relevant Work and Technologies

- Hadoop – Apache Java software framework
- Skynet – Ruby MapReduce framework

- Problems
  - Centralized command and control could be dangerous
    - If Master goes down, system fails
  - If certain nodes fail, priority needed to re-assign jobs
- Solutions
  - Dynamic masters
  - Scheduler that steps in if Master or Workers fail or slow

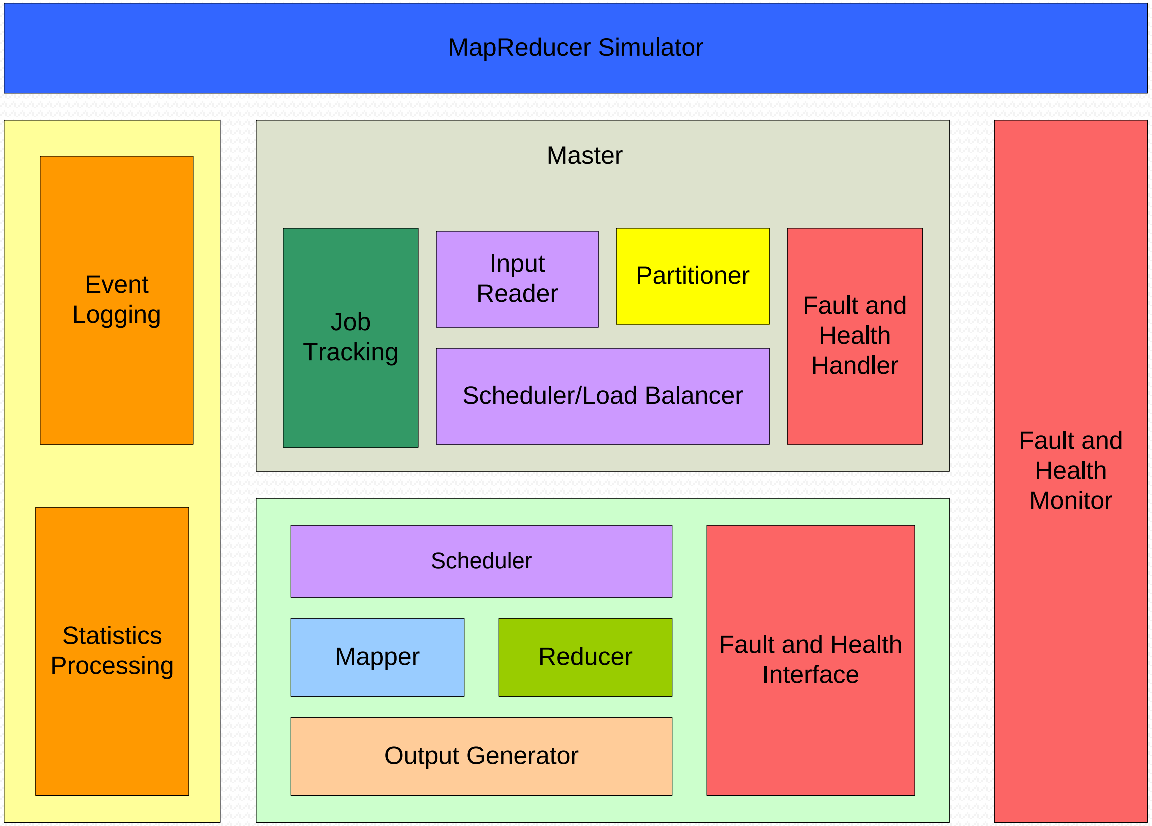
Self Adaptation Solutions

- Self Configuration:
  - One node assumes the role of the Master
  - Determines and distributes the number of mapper and reducer tasks
- Self Healing:
  - Monitor nodes
  - If Master fails, another peer takes over as Master
  - If Worker fails or runs poorly, reallocation of computation
  - Seamless completion despite failures

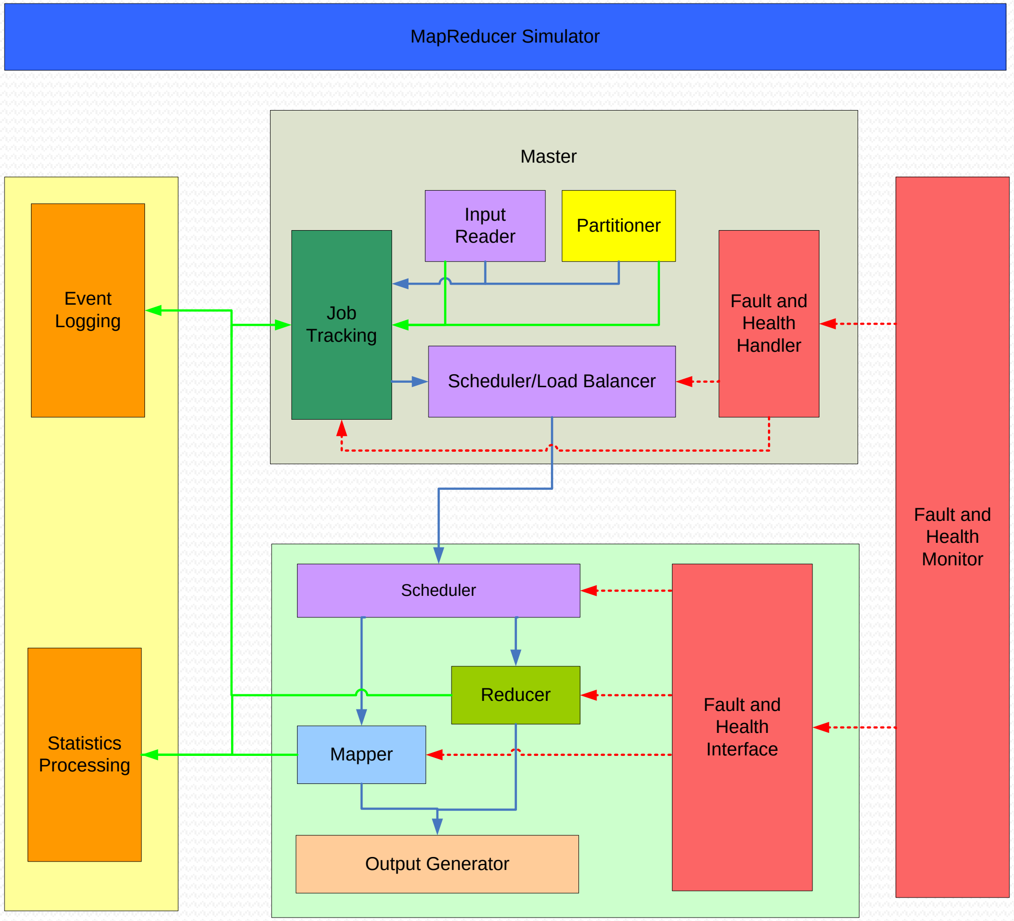
Implementation Solution Overview

- Create a Java console application
- Simulation of MapReduce system on a P2P-type network to count the number of words in a file
- Configurable system parameters to influence system behavior
- Simulate various scenarios such as faults, master failure, etc. to exercise self adaptation

Implementation Block Diagram



Implementation Flow Block Diagram


MapReducer Simulator
Master
Input Reader
Partitioner
Event Logging
Job Tracking
Fault and Health Handler
Scheduler/Load Balancer
Fault and Health Monitor
Scheduler
Reducer
Mapper
Fault and Health Interface
Statistics Processing
Output Generator