COMS 4995 Parallel Functional Programming Project Proposal - MapReduce

Jian Song (js5316), Ziao Wang (zw2498)

November 23, 2019

1 Introduction

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key [1].

In our final project, we are planning to build a MapReduce framework in Haskell to provide parallel programming service for users to achieve several simple tasks. One thing to notice is that the networking between workers and master could be difficult to achieve in Haskell. We will look into existing packages about achieving socket networking between processes. If it's difficult to achieve, we will use shared memory to implement workers communication instead of networking. After implementing the server side of MapReduce in Haskell, we will also implement several typical MapReduce tasks including Word Count, Inverted Index and Distributed Grep. In the end, we will measure the performance of our implementation by running the above tasks in one core and multiple processes.

2 Implementation

On the server side, we will implement the key/value pairs parallel computation service in Haskell and provide two programming interfaces of Map and Reduce functions for users:

Map function
The Map function takes an input pair and produces a set of intermediate key/value pairs and groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function.

Reduce function
The Reduce function accepts an intermediate key and a set of values for that key and merges together these values to form a possibly smaller set of values.

Worker number

Users can specify the appropriate number of workers separately for Map task and Reduce task to best fit their machine condition and use cases.

On the user side, we plan to achieve the following tasks:

Word Count
The map function parses document and emits a sequence of [word, frequency] pairs. The reduce function accepts all pairs for a given word and output pairs of its total count.

Inverted Index
The map function parses each document, and emits a sequence of [word, document ID] pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document IDs and emits a [word,list(document ID)] pair. The set of all output pairs forms a simple inverted index [1].

Distributed Grep
The map function emits a line if it matches a supplied pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output [1].

3 Evaluation

We will be using a Ubuntu machine on cloud with 8 cores to evaluate the performance of our implementation. Experiments will be executed and corresponding performance figures will be drawn to indicate how much better our solution runs in parallel.

References

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.