CS 267 Homework 0

Xiaoqin Jimmy Zhou

February 7, 2013

Biography I am a third-year undergraduate student at UC Berkeley studying EECS and Applied Mathematics. I am interested in many aspects of computer science and mathematics, including: artificial intelligence, algorithms, parallelism, and scientific computing. From this class, I wish to learn the applications of parallel programming as well as some of the basic parallel programing techniques. I wish I could help people to process large-scale data more efficiently in the future.

Intro to Mapreduce The only parallel programing technique that I have encountered before taking this class is Mapreduce. I was asked to use Mapreduce to count how many times each word has appeared in a certain document. To be brief, 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] Notice that the program itself is automatic parallelized and will be executed in a cluster of machines.

Graph Nowadays, more and more data needs to be analyzed because of the development of internet. For example, Twitter processes 7 terabytes of everyday and Facebook processes 10 terabytes of data everyday[2]. Therefore, solve a graph problem using Mapreduce can be both interesting and important. One of the application of Mapreduce is to identify maximal independent set. A maximal independent set or maximal stable set is an independent set that is not a subset of any other independent set[3]. Finding a maximal independent set (MIS) is useful, some of the applications are: Pattern Recognition, Map Labeling, Molecular Biology, Scheduling.

Now, I will be presenting a research conducted by the Sandier National Laboratories at Albuquerque, NM.

Platform Their simulation uses Mapreduce version of Luby's Algortithm[5] to find a MIS. The algorithms is tested on arbitrarily large artificial graphs. The application is targeted at distributed parallel platform, using C++ library built on top of Message Passing Interface (MPI). The test was run on Sandias Cray XMT, a multi-threaded parallel computer with 500 MHz processors and a 3D-Torus network[4], which is not listed at a top 500 machine.

Performance The execution times at the Sandier National Laboratories for the maximal independent set algorithm are shown below:

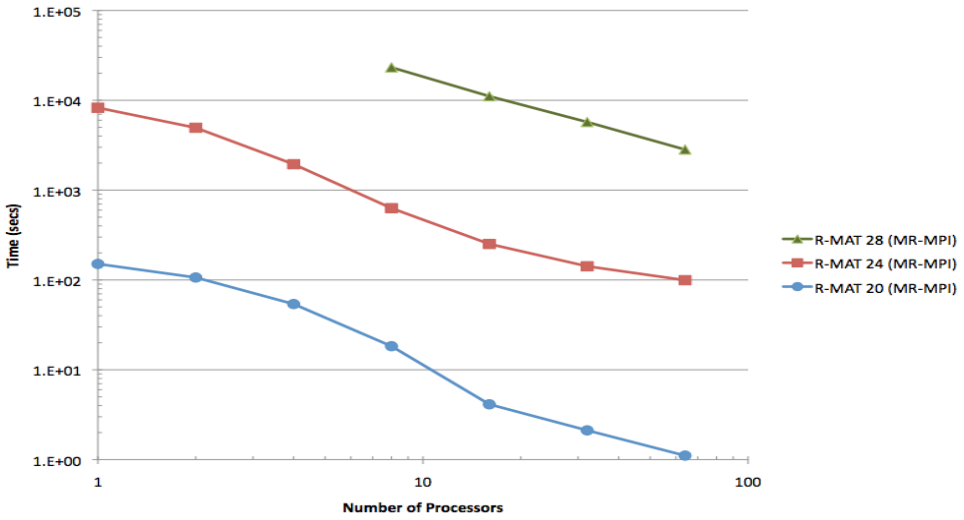

Figure 1: Performance of the MR-MPI maximal independent set algorithm

Based on the graph there is a superlinear speed-up[6] of the algorithm occurs for RMAT-20 (368K, vertices) and RMAT-24 (5.18M, vertices), as more of the graph fits into processor memory and less file I/O is needed. For RMAT-28 (72.4M, vertices), the algorithm requires significant out-of-core operations. In this case, parallel efficiency is nearly perfect going from 8 to 64 processors. The number of iterations required by the algorithm ranged from five for

RMAT-20 to nine for RMAT-28[4].

Note the above graph and performance analysis is provided in the paper **MapReduce in MPI for Large-scale Graph Algorithms** by Plimpton and Devine at Sandia National Lab.

Limitation There are two limitation to the MPI implementations for the above case particularly. One of them is its incapability of detecting dead processors; therefore, if a processor goes away the parallel program will crash. The other limitation is that it doesn't not provide data redundancy[4].

Interesting Topic
Why Facebook ditched Hadoops MapReduce and built a better mousetrap called Corona to handle its data.
Please Visit: http://thenextweb.com/facebook/2012/11/08/facebook-engineering-team-builds-corona-for-mapreduce-jobs/

References
1. http://research.google.com/archive/mapreduce.html
2. http://almaden.ibm.com/colloquium/resources/Why%20Big%20Data%20Krishna.PDF
3. http://en.wikipedia.org/wiki/Maximal_independent_set
4. http://mapreduce.sandia.gov/pdf/pc11.pdf
5. http://www.cs.cornell.edu/courses/CS6820/2012sp/Handouts/191-200.pdf
6. http://en.wikipedia.org/wiki/Speedup