TJHSST Senior Research Project
PRAM and the Ear Decomposition Algorithm
2007-2008

Alex Valentin

April 13, 2008

Abstract

This project takes the ear decomposition algorithm for partitioning maps and compares runtime efficiencies of different implementations. Four implementations are considered. Two implementations are run in a Parallel Random Access Machine (PRAM), while the other two are run serially. For each of these modes, one implementation uses only arrays and the other uses structures.

1 Introduction

Parallel programming is in no way a new concept. Unfortunately, for the past fifty years more emphasis has been put in improving run-time for serial programs. Now that hardware has almost hit its serial limit, it is turning to parallel implementations; dual cores, for example, are becoming more popular. More research will hopefully be put into parallel programming in the near future.

When serial algorithms do not parallelize well, new approaches are needed to tackle problems. In the case of the Depth-First Search, it does not convert to parallel well. The Ear Decomposition Search was created as the parallel equivalent of the DFS.

2 Background

2.1 PRAM

PRAM refers to an abstract machine for designing algorithms in parallel. It allows for an infinite number of processors that can each access the same memory in uniform time. The first known implementation of PRAM is the the University of Maryland A. James Clark School of Engineering's ParaLeap prototype 64-core supercomputer.

2.2 XMT-C

The language used on this supercomputer is called XMT-C, eXplicit multi-Threaded C. Simply, the language is C with two extra methods, SPAWN and PS. The spawn method allows the programmer to use multiple processors. While any number of processors can be called for, the computer only has 64 processors to run at any given moment. The ps method, short for prefix sum, allows for the different threads to communicate with each other.

2.3 Ear Decomposition

The Ear Decomposition algorithm is used for partitioning different maps into simple paths known as ears. Alone, the Ear Decomposition algorithm does not provide much information. Instead, it is used as an intermediate step for other algorithms. Some algorithms that use Ear Decomposition as an intermediate step include: [1]

1. st-numbering
2. search for tri-connected components
3. planarity testing
4. disjoint paths

3 Procedures

The four implementations were written in XMT-C, even though the serial versions do not use the parallel capabilities. The Ear Decomposition was

broken into three files, a main, span, and link. A convert file was also created for converting the input data from arrays to structures, if applicable. The span file finds a spanning tree of the data. The link file labels each edge and node with the correct ear. The main file runs the span and link files and then prints the ear of each edge and node.

Input data was given in the form of three arrays, vertices, degrees, and a two dimensional edges array. This data was quickly made with the help of the memMapCreate32 program in the XMT environment. See Appendix A for images of the five data sets used for testing (That is, when I make the Appendix.) Each data set was run on each of the four implementations of ear decomposition four times. The clock cycle counts were averaged and compared. The data is shown in Table 1 (which will arrive shortly.)

4 Results

The four ear decomposition implementations to be tested are the parallel using structures, parallel using only arrays, the serial using structures, and the serial using only arrays. The two parallel implementations should run faster than the serial implementations for obvious reasons. Of the the two parallel implementations, the one using structures is expected to run slower because of the overhead caused by using structures. Therefore, from fastest to slowest, the implementations are parallel arrays, parallel structures, serial arrays, and serial structures.

5 Conclusion

Comming Soon!

References

[1] Ibarra, Louis, and Dana Richards. "Efficient parallel graph algorithms based on open ear decomposition." Parallel Computing 19 (1993): 873-886

[2] Maon, Yael, Baruch Schieber, and Uzi Vishkin. "Parallel Ear Secomposition Search (EDS) and st-Numbering in Graphs." Theoretical Computer Science 47 (1986): 277-298