Abstract (leave blank for now)

Introduction

- The SSI experience (learning to conduct research & learning about parallelism)

-10 weeks

-on parallelism/research

-“full time” job

-

-          TSP problem (references)

-NP complete

-n! possible paths

-

What You Learned

-          TSP Algorithms (sequential & parallel)

-Brute force

-Easy to parallelize on all platforms

-embarrassingly parallel

-finding permutations of paths

-optimizations(?)

-          Parallelism in general (hardware, concepts like protecting shared data, splitting up big tasks so multiple processors/cores/nodes can work on it)

-          OpenMP

-Easiest to work with, both debugging and coding (Jerome)

-Requires little software configuration (Jerome)

-common hardware setup

-works on multi-core systems as well as the coprocessor

-          CUDA

-Hardest to work with, especially debugging (Jerome)

-also require learning C

-Requires moderate configuration to run (linking the appropriate libraries) (Jerome)

-hardest to optimize(Jerome)

-most difficult to grasp architecture (both memory and thread distribution) (Jerome)

-Cannot use many C library functions

-          MPI

-Most arduous configuration (Jerome)

-BCCD does most of the software config, but still difficult overall (Zach)

- Coprocessor (openMP)

-Even more difficult to set up than MPI, but more understandable (Jerome)

Experiments with LittleFe & Results

Conclusions

-openMP is a great start to parallel computing

-Parallel computing requires good knowledge of scope and copying of variables

-Focus on optimization is interesting, but can make parallel computing seem somewhat overwhelming

-We had more hardware and config issues than actual coding issues

-difficulty debugging encourages small steps in working up to complex algorithms

-VERY VIABLE TO TEACH PARALLEL COMPUTING EARLY IN THE MAJOR

Future Work

References