*Bristol Miller, Shreya Kumar, Corey Vessar*

*bristolnmiller@ksu.edu, shreyak@ksu.edu, coreyvessar@ksu.edu Manhattan, KS 66503*

*PArallelization using three different methods*

Kansas State University: Department of Computer Science

Table of Contents

[Abstract 2](#_Toc512420976)

[Introduction 2](#_Toc512420977)

[Related Work 3](#_Toc512420978)

[Implementation 3](#_Toc512420979)

[Evaluation 4](#_Toc512420980)

[Conclusions & Future Work 5](#_Toc512420981)

[Appendix 5](#_Toc512420982)

[OpenMP 5](#_Toc512420983)

[OpenMP Shell Script 6](#_Toc512420984)

[MPI 6](#_Toc512420985)

[MPI Shell Script 6](#_Toc512420986)

[PThreads 6](#_Toc512420987)

[PThreads Shell Script 6](#_Toc512420988)

[References 6](#_Toc512420989)

# Abstract

Implementing programs without parallelization can make even simple solutions quite slow. When reading a larger file and searching it line by line for information, it is necessary to parallelize your program in order to have the most efficient solution. Through our trial of the three different methods of parallelization: OpenMP, MPI, and PThreads, we have found that one method outperforms the others and improves overall timing by (XXX).

# Introduction

A program that we have written reads in a 1.7 GB file, called wiki\_dump.txt, and compares the first two lines, followed by the second and third lines, then the third and fourth lines, and so on, searching for the longest common substring between each pairing and outputting those substrings to a file. The major problem with this program is that it is inefficient.

Initially, we had not implemented any parallelization in order to not complicate the ode. Avoiding parallelization from the start led to our program being very slow. From start to end, accomplishing the tasks of reading in the file and searching for the longest common substrings took (XXX) amount of time. When faced with a heavy task such as reading in a large file and searching through each line of that large file, this can be detrimental if the information is needed quickly. We have solved this problem by parallelizing the program in three different ways, in order to figure out which method provides the quickest way to determine the longest common substrings. These three methods are OpenMP, MPI, and PThreads.

By creating solutions using the OpenMP, MPI, and PThreads methods all separately, we were able to discern which method is the most efficient for this application. Each of the three methods presented a considerable decrease in runtime, and thus a dramatic increase in efficiency. Also, having the comparisons between the three different methods allowed us to make an educated and unbiased decision as to which solution is the best for this specific application. After experimentation with all three methods, we came to the conclusion that the (XXX) method is the best for our program because of (X, Y, and Z).

# Related Work

Other efforts that exist to solve this problem and why are they less effective than our method

• Resist the urge to point out only flaws in other work. Do your best to point out both the strengths and weaknesses to provide as well rounded a view of how your idea relates to other work as possible

• In a social and political sense it is very smart as well as ethically superior to say good things, which are true, about other people’s work. A major motivation for this is that editors and program committee members have to get a set of reviews for your paper. The easiest way for them to decide who should review it is to look at the set of references to related work (e.g., [1,2, 3]) to find people who are likely to be competent to review your paper. The people whose work you talk about are thus likely to be reading what you say about their work while deciding what to say about your work.

• Clear enough? Speak the truth, say what you have to say, but be generous to the efforts of others.

Other efforts that exist to solve related problems that are relevant, how are they relevant, and why are they less effective than our solution for this problem

• Many times no one has solved your exact problem before, but others have solved closely related problems or problems with aspects that are strongly analogous to aspects of your problem

# Implementation

What we (will do | did): Our Solution

• Another way to look at this section is as a paper, within a paper, describing your implementation. That viewpoint makes this the introduction to the subordinate paper, which should describe the overall structure of your implementation and how it is designed to address the problem effectively.

• Then, describe the structure of the rest of this section, and what each subsection describes.

How our solution (will | does) work

• This is the body of the subordinate paper describing your solution. It may be divided into several subsections as required by the nature of your implementation.

• The level of detail about how the solution works is determined by what is appropriate to the type of paper (conference, journal, technical report)

• This section can be fairly short for conference papers, fairly long for journal papers, or quite long in technical reports. It all depends on the purpose of the paper and the target audience

• Proposals are necessarily a good deal more vague in this section since you have to convince someone you know enough to have a good chance of building a solution, but that you have not already done so.

# Evaluation

How we tested our solution

• Performance metrics

• Performance parameters

• Experimental design

How our solution performed, how its performance compared to that of other solutions mentioned in related work, and how these results show that our solution is effective

• Presentation and Interpretation

• Why, how, and to what degree our solution is better

• Why the reader should be impressed with our solution

• Comments

Context and limitations of our solution as required for summation

• What the results do and do not say

# Conclusions & Future Work

The problem we have solved

• The most succinct statement of the problem in the paper. Ideally one sentence. More realistically two or three. Remember that you simply state it without argument. If you have written a good paper you are simply reminding the reader of what they now believe and of how much they agree with you.

Our solution to the problem

• Again, the succinct statement that you have presented a solution

• Sometimes it works well to leave it at that and not even describe your solution here. If you do, then again state your solution in one or two sentences taking the rhetorical stance that this is all obvious. If you have a good solution and have written an effective paper, then the reader already agrees with you.

Why our solution is worthwhile in some significant way

• Again, a succinct restatement in just a few sentences of why your solution is worthwhile assuming the reader already agrees with you

Why the reader should be impressed and/or pleased to have read the paper

• A few sentences about why your solution is valuable, and thus why the reader should be glad to have read the paper and why they should be glad you did this work.

What we will (or could) do next

• Improve our solution

• Apply our solution to harder or more realistic versions of this problem

• Apply our solution or a related solution to a related problem

# Appendix

In this section, we have included copies of our code, along with links to the files on github (Not included yet).

## OpenMP

OpenMP Base Code goes here

### OpenMP Shell Script

Open MP Shell scripts here we can rename the above header to be exactly what it is

## MPI

Same deal as before

### MPI Shell Script

Same

## PThreads

More

### PThreads Shell Script

And finally more

# References

[1] Hughes, Christian J. “ChristianJHughes/Pintos-project4.” GitHub, 9 May 2017, github.com/ChristianJHughes/pintos-project4.

[2] Barney, Blaise. “Open MP.” OpenMP, U.S. Department of Energy, 17 July 2017, computing.llnl.gov/tutorials/openMP/.

[3] Barney, Blaise. “POSIX Threads Programming.” POSIX Threads Programming, U.S. Department of Energy, 7 Mar. 2017, computing.llnl.gov/tutorials/pthreads/.

[4] Barney, Blaise. “Message Passing Interface (MPI).” Message Passing Interface (MPI), U.S. Department of Energy, 19 June 2017, computing.llnl.gov/tutorials/mpi/.

[5] “Longest Common Substring Problem.” Wikipedia, Wikimedia Foundation, 9 Nov. 2017, en.wikipedia.org/wiki/Longest\_common\_substring\_problem.

[6] Hancy. “Hancyxhx/Longest-Common-Substring.” GitHub, 20 Mar. 2014, github.com/hancyxhx/Longest-Common-Substring.