CIS 520 Project 4 Performance Analysis

Authors: Blake A Holman, Gabriel Serrano, Cullen Povilonis
Date: 5/6/2019

STATEMENT OF PURPOSE

We have been tasked with creating a program that finds the longest common substring among pairs of Wikipedia entries. The program prints out a list of lines, in order, with the longest common substring between the current and next string. It then prints that common substring into a line and then does the process again for the next line. The data set that we used for the Wikipedia entries contained approximately 1 million wikipedia entries.

Our goal for the project was to make the entire process as fast as possible and compare the runtimes of the different multi-threading/parallelization libraries. The ones we used followL

- Pthreads
- MPI
- OpenMP

This document is written for the purpose of analyzing the performances of each version of the program.

Experimental Configuration

Hardware

For this experiment, we used Beocat. Beocat is a High-Performance Computing (HPC) cluster located at Kansas State University. The execution was limited to the “Elves” computer nodes on Beocat. The Elve nodes adhere to the following specifications.

Specification 1:

Processors	2x 8-Core Xeon E5-2690
Ram	64GB
Hard Drive	1x 250GB 7,200 RPM SATA
NICs	4x Intel I350
10GbE and QDR Infiniband	Mellanox Technologies MT27500 Family

	[ConnectX-3]

Specification 2:

Processors	2x 10-Core Xeon E5-2690 v2
Ram	96GB
Hard Drive	1x 250GB 7,200 RPM SATA
NICs	4x Intel I350
10GbE and QDR Infiniband	Mellanox Technologies MT27500 Family [ConnectX-3]

Specification 3:

Processors	2x 10-Core Xeon E5-2690 v2
Ram	384GB
Hard Drive	1x 250GB 7,200 RPM SATA
NICs	4x Intel I350
10GbE and QDR Infiniband	Mellanox Technologies MT27500 Family [ConnectX-3]

Specification 4:

Processors	2x 10-Core Xeon E5-2690v2
Ram	64GB
Hard Drive	1x 250GB 7,200 RPM SATA
NICs	4x Intel I350
10GbE and QDR Infiniband	Mellanox Technologies MT27500 Family

	[ConnectX-3]

Software

For completing the project necessarily, our team utilized C code with all standard C libraries included. And beocat also runs on x86_64 Linux for its operating system. We also utilized OpenMP, MPI, and Pthreads for multithreading and parallelization. When it came to running our C code, our team utilized gcc for both our OpenMP and Pthread C files, while utilizing MPICC for the MPI solution. Utilizing these technologies, we were able to complete our tests with Beocat's scheduling system.

Experiment Procedures

Each implementation of the program (MPI, OpenMP, and PThreads) all used multi-threading/parallelization libraries. The purpose, as stated above, was to find the longest common substring between line pairs.

File Access: The file accessed was located on Beocat, and it contained approximately one million wikipedia entries.

RAM Usage: Every test was allocated at least 64GB of RAM. However, the tests utilized less than that. It utilized less than 64GB because all of the longest common substrings were stored in memory. Any amount less than 64GB would potentially prevent the caching of the entire working set.

For running the programs, we needed the following specified:

1. Number of machines
2. Number of cores
3. Number of threads

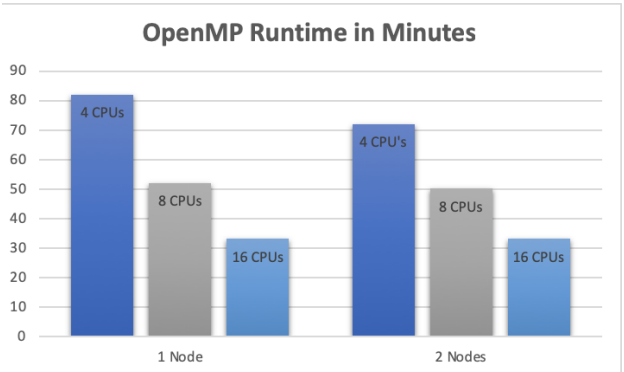
We also ran the tests on different file sizes in order to understand the effect file size has on performance. We ran it on 1,000, 10,000, 100,000, and 1,000,000 amount of lines in the file.

MPI Implementation

For running MPI, it appears beocat would not run our mpi solution, however worked for us in proper testing before being utilized in beocat. We are not sure about the failure of the execution, possibly due to our time constraint enabled for that bash file. It also might have something to do with the mpicc command which was used, with beocat possibly not having the files needed to run the command. The working C file and bash file are included in the full .tgz file.

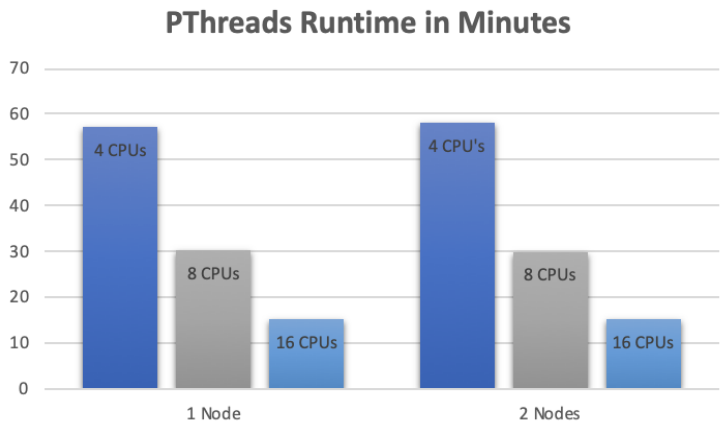
OpenMP Implementation
This implementation parallelize the programing using the OpenMP library. All threads ran on a single machine with different combinations of threads and cores.

We noticed that using openmp with a single core runs slower than other tests. We also noted that the more cores that were added, the faster the program ran.



PThreads Implementation
The PThreads implementation used the PThreads library in order to parallelize the program. All threads ran on a single machine with different combinations of threads and cores.

We noticed that using pthreads with a single core runs slower than other implementations and tests. We also noted that the more cores that were added, the faster the program ran.



Conclusion

For this project, we were able to understand how multi-threading/parallelization libraries can significantly improve performance or undermine it. For openmp and pthreads, when we ran tests only using a single core, the performance from it was not worth the parallelization. However, when four or more cores were added, these libraries can significantly improve performance.

First 100 Lines

```
0-1: </title><text>\'\'\'aa
1-2: </text> </page>
2-3: }}</text> </page>
3-4: <page> <title>a
4-5: \n|foundation = 19
5-6: <page> <title>abc_
6-7: <page> <title>abc_
7-8:  the [[australian broadcasting corporation]]
8-9:  the [[australian broadcasting corporation]]
9-10:  [[australian broadcasting corporation]]
10-11: </title><text>{{infobox
11-12: <page> <title>ab
12-13: </title><text>\'\'\'ab
13-14: </title><text>\'\'\'a
14-15: <page> <title>ac
15-16: <page> <title>acc
16-17: \n\n{{disambig}}</text> </page>
17-18: }}</text> </page>
18-19: \n\n{{disambiguation}}</text> </page>
19-20: </title><text>\'\'\'ac
20-21: <page> <title>ac
21-22: <page> <title>ac_
22-23: <page> <title>ac_
23-24: </text> </page>
24-25: \'\'\' may refer to:\n
25-26: \'\'\' may refer to:\n\n
26-27: <page> <title>ad
27-28: <page> <title>ad
28-29: <page> <title>a
29-30: \n\n{{disambig}}</text> </page>
30-31: <page> <title>afc
31-32: <page> <title>af
32-33: </text> </page>
33-34: <page> <title>a
34-35: </title><text>{{infobox
35-36: </title><text>{{
36-37: <page> <title>a
37-38: <page> <title>aid
38-39: <page> <title>ai
39-40: [[australian institute of
40-41: <page> <title>a
```

```
41-42: \n\n{{disambig}}</text> </page>
42-43: ]]\n\n{{disambig}}</text> </page>
43-44: </text> </page>
44-45: n-stub}}</text> </page>
45-46: </text> </page>
46-47: </text> </page>
47-48: <page> <title>alar
48-49: <page> <title>al
49-50: <page> <title>alp
50-51: <page> <title>a
51-52: <page> <title>am
52-53: <page> <title>am
53-54: <page> <title>am
54-55: <page> <title>am
55-56: }}\n\n{{defaultsort:am
56-57: </title><text>{{unreferenced
57-58: class=\"wikitable\"
58-59: <page> <title>an
59-60: <page> <title>a
60-61: <page> <title>a
61-62: <page> <title>ap
62-63: <page> <title>ap
63-64: <page> <title>ap
64-65: <page> <title>ap
65-66: \n\n==external links==\n*
66-67: <page> <title>ap
67-68: </title><text>{{
68-69: {{cite web|url=http://www.
69-70: <page> <title>ar
70-71: <page> <title>a
71-72: ==\n{{reflist}}\n\n==
72-73: <page> <title>asa_
73-74: <page> <title>as
74-75: <page> <title>as
75-76: <page> <title>as
76-77: <page> <title>as
77-78: </text> </page>
78-79: <page> <title>as
79-80: american society for
80-81:  [[association fo
81-82: \'\'\' is a [[france|french]] [[association football]]
```

```
82-83: <page> <title>a
83-84: <page> <title>at
84-85: <page> <title>at
85-86: <page> <title>at
86-87: <page> <title>at
87-88: <page> <title>a
88-89: {{cite web|url=http://www.
89-90: <page> <title>a
90-91: <page> <title>a
91-92: ]]\n\n{{disambig}}</text> </page>
92-93: }}</text> </page>
93-94: </title><text>{{
94-95: {{unreferenced|date=
95-96: <page> <title>a_b
96-97: <page> <title>a_be
97-98: <page> <title>a_be
98-99:  2012}}\n{{infobox album
99-100:  name        = a big
```