

Developing Soft and Parallel Programming Skills Using Project-Based Learning

Spring-2019 **JupitorInk**

Nhi Ong, Alejandro Herbener, Alec Buchinski, Johnathan Moore

Table of Contents

I.	Project Description.....	2
	1.1 Purpose of the Project.....	2
II.	Introduction to the Team.....	2
	2.1 Members Roles and Assignments.....	2
III.	Planning and Scheduling.....	2-4
	3.1 Role as Coordinator.....	3
	3.2 Additional Tasks.....	3-4
IV.	Parallel Programming Skills.....	4-12
	5.1 Foundation.....	4-5
	5.2 Parallel Programming Basics Report.....	5-12
V.	Appendix.....	15

I. Project Description

1.1 Purpose of the Project

The purpose of this project is to 1) become familiar with unified, team-based project management tools and communication applications in order to develop soft skills, 2) identify and analyze the basics of parallel computing architectures and programming using OpenMP and C language 3) MapReduce

II. Introduction to the Team

2.1 Members' Roles and Assignments

Coordinator: Johnathon Jack Moore

He is the coordinator for this assignment. In addition to his responsibilities as a coordinator, he was also responsible for tasks one, two, and five. He created a new assignment task chart, sent an invitation to our Slack workspace to the TA, created a new GitHub repository and projects page, and edited our video and uploaded it to the YouTube channel.

Team Member: Alec Buchinski

This team member is responsible for task three. Alec answered the questions that for the Parallel Programming Foundation and completed the programming portion of this task. He also took screenshots and created a written report of his findings.

Team member: Nhi Ong

This team member is responsible for task five. I created this report after compiling each member's work from their corresponding tasks.

Team Member: Alejandro Herbener

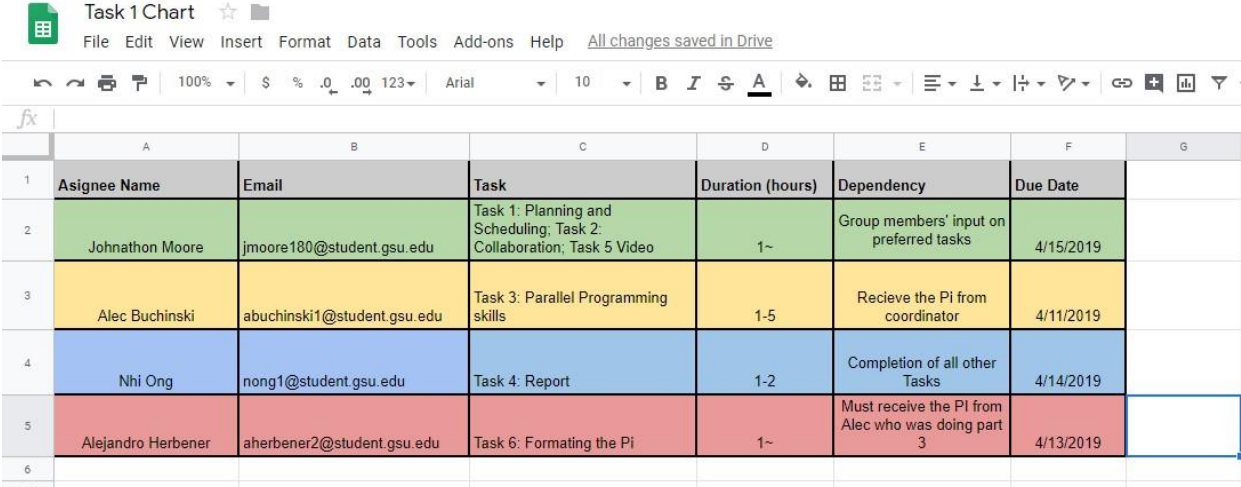
This team member is responsible for task six. He formatted the PI in preparation for its return. He will hand it to the coordinator who will complete the final hand off.

III. Planning and Scheduling

3.1 Role as Coordinator

As the coordinator, Johnathon was responsible for assigning us with our tasks, updating our assignment table, creating a new repository for the assignment, and

adding the TA to our Slack workspace. He first contacted us to discuss our preferences for our roles in this assignment, which we each quickly selected and promptly began working on our tasks. He updated the assignment task table soon thereafter and notified us of our responsibilities, dependencies, and due date. (see Figure 1.1)



	A	B	C	D	E	F	G
	Assignee Name	Email	Task	Duration (hours)	Dependency	Due Date	
2	Johnathon Moore	jmoore180@student.gsu.edu	Task 1: Planning and Scheduling; Task 2: Collaboration; Task 5 Video	1~	Group members' input on preferred tasks	4/15/2019	
3	Alec Buchinski	abuchinski1@student.gsu.edu	Task 3: Parallel Programming skills	1-5	Recieve the Pi from coordinator	4/11/2019	
4	Nhi Ong	nong1@student.gsu.edu	Task 4: Report	1-2	Completion of all other Tasks	4/14/2019	
5	Alejandro Herbener	aherbener2@student.gsu.edu	Task 6: Formating the Pi	1~	Must receive the PI from Alec who was doing part 3	4/13/2019	
6							

Figure 1.1 – Task One, Assignment Chart

3.2 Additional Tasks

Johnathon was responsible for creating a new repository for this assignment with a readme page. He added a projects page which details our tasks and allows for any member to update their status on their role. (see Figure 1.2) In addition to this, he also edited our video and uploaded it to YouTube. (see Appendix A, for link to video)

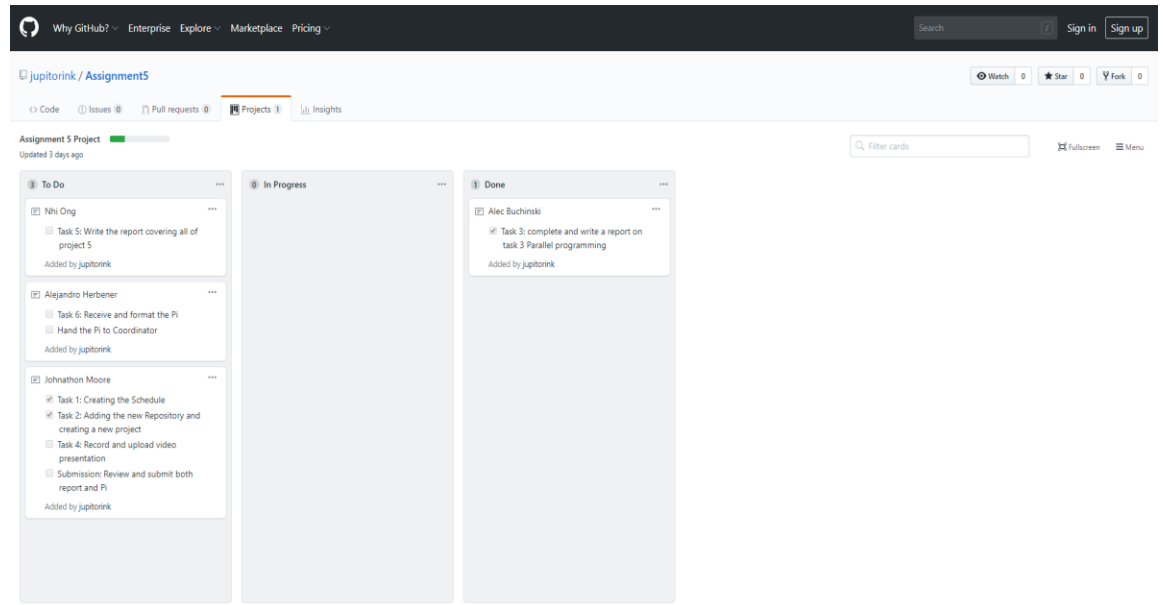


Figure 1.2 – Task Two, GitHub Projects Page Screenshot

IV. Parallel Programming Skills

5.1 Foundation

What are the basic steps in building a parallel program?

First, one must identify the tasks in a program that can run simultaneously. A task can run simultaneously if it has no dependencies or communications between tasks. Next, implementation techniques such as master/worker are used.

What is MapReduce?

MapReduce is a programming model that programmers often use when creating large sets of data on a cluster in a parallel and distributed algorithm.

Why MapReduce?

MapReduce can be useful because it allows for processing of large amounts of raw data while keeping the details of parallelization, data distribution, load balancing and fault tolerance hidden.

Show an example for MapReduce?

MapReduce can be used for a program that requires a term vector which summarizes the most important words that occur in a document stored in <word,frequency> pairs.

Explain in your own words how MapReduce is executed.

The library shards the files into a certain amount of pieces and begins creating many copies of the program on a number of machines. There is one copy designated master which assigns work to the workers. Workers are assigned a map task or a reduce task. The master is also responsible for forwarding locations of buffered pairs to the reduce workers. When these workers receive this information from the master, it reads the buffered data using a remote procedure call and sorts through the intermediate data and passes the key to the corresponding set of values in the Reduce function. When all tasks are finished, the master wakes up the program.

List and describe the three examples that are expressed as MapReduce computations.

Distributed Grep: The map function emits a line if it matches a given pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.

Count of URL Access Frequency: The map function processes logs of web page requests and outputs <URL, 1>. The reduce function adds together all values for the same URL and emits a <URL, total count> pair.

Reverse Web-Link Graph: The map function outputs <target, source> pairs for each link to a target URL found in a page named "source". The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: <target, list(source)>.

When do we use OpenMP, MPI, and MapReduce and why?

Because OpenMP is a directive based library, it is most desirable to use it when you want to include shared memory parallelism in your program. It makes use of threads. MPI on the other hand, is usually used in parallel scientific applications. It can run parallel code over many machines at once. Some programmers practice what is called “hybrid programming” and mix both MPI and OpenMP together, MapReduce is a technique used on large data where the data is first distributed and then reduced. It is also used for scientific applications but it does not have the best performance.

In your own words, explain what a Drug Design and DNA problem is in no more than 150 words.

A Drug Design and DNA problem usually has to do with the creation and design of medicine. Our bodies rely on proteins made by DNA in our body. The shape of the protein will decide what kind of function it performs in the body. When scientists design new drugs, they find what are called ligands to change a proteins shape. If there is a disease that requires a medicine, first the protein associated with the disease must be found and modeled through molecular modeling computations. They will then a set of computations is completed which tests collections of liqands for orientations that bind to the protein in satisfactory manners. Each of these computations are scored and the highest scoring sets will be tested further for drug development.

5.2 Parallel Programming Basics

2.2 Discussion Questions

Measure Run-Time

*** Real is the wall clock time, from start to finish of the call

*** User is the amount of CPU time spent outside the kernel within the process

*** Sys is the amount of CPU time spent in the kernel within the process

*** User+Sys is the total CPU time the process used. If multi-thread, might exceed the wall clock time.

Times for each implementation:

time -p ./dd_serial : 122.49

```

pi@raspberrypi:~/sequential $ ls
dd_serial  dd_serial.cpp  Makefile  Question2_WordCount.png
pi@raspberrypi:~/sequential $ time ./dd_serial
maximal score is 5, achieved by ligands
real 122.49
user 122.49
sys 0.00
pi@raspberrypi:~/sequential $ scrot

```

The screenshot shows a code editor with C++ code for a word count program. The code includes headers for `<iostream>`, `<string>`, `<vector>`, `<algorithm>`, and `<string_view>`. It defines constants for `max_ligand` and `DEFAULT_protein`. The `main` function uses `std::chrono::high_resolution_clock` to measure the execution time. The `Generate_tasks` function creates a vector of `Pair` objects representing word counts. The `Reduce` function calculates the maximal score by finding the maximum value in the `pairs` vector.

time -p ./dd_omp 1: Real: 0.02, User: 0.01

```

pi@raspberrypi:~/openmp1$ ls
2019-04-09-222052_1824x984_scr0t.png dd_omp Makefile tbb
current_vector.h dd_omp.cpp parallel_sort.h
pi@raspberrypi:~/openmp1$ time -p ./dd_omp 1
max ligand=1 nligands=120 nthreads=4
OMP defined
maximal score is 1, achieved by ligands
w w r n n r o c r c o c i h c y t t c e a c e y w p c a i c h y n t r o n r r
o i p w h p p r r
real 0.02
user 0.01
sys 0.01
pi@raspberrypi:~/openmp1$ scrot

```

time -p ./dd_threads 1: Real: 0.02, User: 0.01

```

pi@raspberrypi:~/cplusthreads1$ time -p ./dd_threads 1
max ligand=1 nligands=120 nthreads=4
c w h n r t r t i a w n n c o p p r e c o p p r i c h c c e o c y y r e h c o y
r a n w i p w r r
real 0.02
user 0.01
sys 0.01
pi@raspberrypi:~/cplusthreads1$ scrot

```


time -p ./dd_omp 2: Real: 0.02, User: 0.02, Sys: 0.02

```

File Edit Tabs Help
dd_serial.cpp:K dd_threads.cpp
pi@raspberrypi:~/openmp1
pi@raspberrypi1:~$ ls
2019-04-09-222052_1824x984_scr0t.png dd_omp parallel_sort.h
2019-04-10-232431_1824x984_scr0t.png dd_omp.cpp tbb
concurrent_vector.h Makefile
pi@raspberrypi1:~/openmp1$ time -p ./dd_omp 2
maximal score is 2, achieved by ligands
real 0.02
user 0.02
sys 0.02
pi@raspberrypi1:~/openmp1$ scrot

```

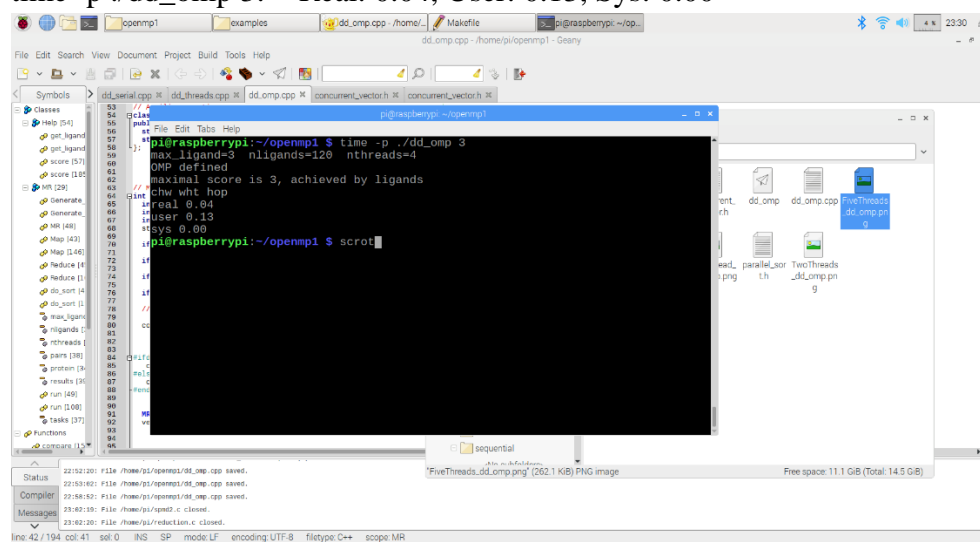
time -p ./dd_threads 2: Real: 0.02, User: 0.01, Sys: 0.01

```

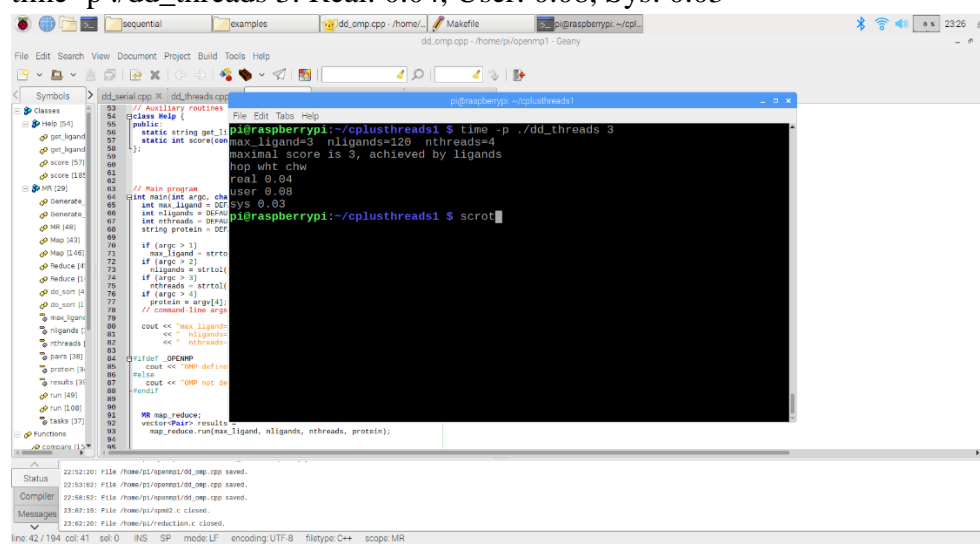
File Edit Tabs Help
dd_serial.cpp:K dd_threads.cpp
pi@raspberrypi:~/cplusthreads1
pi@raspberrypi1:~/cplusthreads1$ ls
2019-04-10-231416_1824x984_scr0t.png dd_threads.cpp
concurrent_queue.h Makefile
Cplusplus_OneThread.png parallel_sort.h
tbb
pi@raspberrypi1:~/cplusthreads1$ time -p ./dd_threads 2
maximal score is 2, achieved by ligands
real 0.02
user 0.01
sys 0.01
pi@raspberrypi1:~/cplusthreads1$ scrot

```

time -p ./dd_omp 3: Real: 0.04, User: 0.13, Sys: 0.00



time -p ./dd_threads 3: Real: 0.04, User: 0.08, Sys: 0.03



time -p ./dd_omp 4: Real: 0.21, User: 0.56, Sys: 0.01

```

1 // dd_omp.cpp
2 #include <iostream>
3 #include <vector>
4 #include <string>
5 #include <algorithm>
6 #include <set>
7
8 // Maximal score is 3, achieved by ligands
9 // orel ordt hkcic rdry hch cio ihjo wht
10 // real 0.21
11 // user 0.56
12 // sys 0.01
13
14 using namespace std;
15
16 // key-value pairs, used for sorting
17 struct Pair {
18     int key;
19     string val;
20 };
21 Pair(int k, const string& v) : key(k), val(v) {}
22
23 // no class provides a
24 // constructor for Pair
25 class MR {
26 private:
27     int max_ligand;
28     int nligands;
29     string protein;
30     vector<Pair> pairs;
31     vector<string> tasks;
32     void GenerateTasks() {
33         for (int i = 0; i < nligands; i++) {
34             string task = protein.substr(i, max_ligand);
35             tasks.push_back(task);
36         }
37     }
38     void Map(const string& task) {
39         int key = task.find(' ');
40         if (key != string::npos) {
41             int k = key;
42             string v = task.substr(k + 1, task.length() - k - 1);
43             pairs.push_back(Pair(k, v));
44         }
45     }
46 public:
47     MR(int k, const string& v) : max_ligand(k), nligands(v.length() / k), protein(v) {}
48     void Run() {
49         GenerateTasks();
50         Map(task);
51         sort(pairs.begin(), pairs.end(), [](const Pair& a, const Pair& b) { return a.key < b.key; });
52         int max_score = 0;
53         for (int i = 0; i < pairs.size(); i++) {
54             int key = pairs[i].key;
55             string val = pairs[i].val;
56             if (key < max_score) {
57                 max_score = key;
58                 string best_val = val;
59             }
60         }
61         cout << "Maximal score is " << max_score << ", achieved by ligands " << best_val << endl;
62     }
63 };
64
65 int main(int argc, char* argv[]) {
66     if (argc < 2) {
67         cout << "Usage: ./dd_omp k nligands\n";
68         return 1;
69     }
70     int k = atoi(argv[1]);
71     int nligands = atoi(argv[2]);
72     string protein = "orel ordt hkcic rdry hch cio ihjo wht";
73     MR mr(k, protein);
74     mr.Run();
75     return 0;
76 }

```

time -p ./dd_threads 4: Real: 0.16, User: 0.56, Sys: 0.01

```

1 // dd_threads.cpp
2 #include <iostream>
3 #include <vector>
4 #include <string>
5 #include <algorithm>
6 #include <set>
7
8 // Maximal score is 3, achieved by ligands
9 // hkcic ordt hkcic rdry ihjo hch cio wht ordt
10 // real 0.16
11 // user 0.56
12 // sys 0.01
13
14 using namespace std;
15
16 // key-value pairs, used for sorting
17 struct Pair {
18     int key;
19     string val;
20 };
21 Pair(int k, const string& v) : key(k), val(v) {}
22
23 // no class provides a
24 // constructor for Pair
25 class MR {
26 private:
27     int max_ligand;
28     int nligands;
29     string protein;
30     vector<Pair> pairs;
31     vector<string> tasks;
32     void GenerateTasks() {
33         for (int i = 0; i < nligands; i++) {
34             string task = protein.substr(i, max_ligand);
35             tasks.push_back(task);
36         }
37     }
38     void Map(const string& task) {
39         int key = task.find(' ');
40         if (key != string::npos) {
41             int k = key;
42             string v = task.substr(k + 1, task.length() - k - 1);
43             pairs.push_back(Pair(k, v));
44         }
45     }
46 public:
47     MR(int k, const string& v) : max_ligand(k), nligands(v.length() / k), protein(v) {}
48     void Run() {
49         GenerateTasks();
50         Map(task);
51         sort(pairs.begin(), pairs.end(), [](const Pair& a, const Pair& b) { return a.key < b.key; });
52         int max_score = 0;
53         for (int i = 0; i < pairs.size(); i++) {
54             int key = pairs[i].key;
55             string val = pairs[i].val;
56             if (key < max_score) {
57                 max_score = key;
58                 string best_val = val;
59             }
60         }
61         cout << "Maximal score is " << max_score << ", achieved by ligands " << best_val << endl;
62     }
63 };
64
65 int main(int argc, char* argv[]) {
66     if (argc < 2) {
67         cout << "Usage: ./dd_threads k nligands\n";
68         return 1;
69     }
70     int k = atoi(argv[1]);
71     int nligands = atoi(argv[2]);
72     string protein = "orel ordt hkcic rdry ihjo hch cio wht ordt";
73     MR mr(k, protein);
74     mr.Run();
75     return 0;
76 }

```

1) Which approach is the fastest?

Using the total of User and Sys time, dd_threads is the fastest. The user time increases less than the dd_omp user time, even with a longer sys time.

2) Determine the number of lines in each file using wc-l. How does the C++11 implementation compare to the OpenMP implementations?

```

pi@raspberrypi:~/sequential$ wc -l /home/pi/sequential/dd_serial /home/pi/sequential/dd_omp /home/pi/sequential/dd_threads
 67 /home/pi/sequential/dd_serial
176 /home/pi/sequential/dd_omp
159 /home/pi/sequential/dd_threads
402 total
pi@raspberrypi:~/sequential$ scrot
  
```

```

pi@raspberrypi:~/sequential$ wc -l /home/pi/sequential/dd_serial /home/pi/sequential/dd_omp /home/pi/sequential/dd_threads
 67 /home/pi/sequential/dd_serial
176 /home/pi/sequential/dd_omp
159 /home/pi/sequential/dd_threads
402 total
pi@raspberrypi:~/sequential$ scrot
  
```

3) Increase the number of threads to 5. What is the run time for each?

dd_omp 5: Real: 0.95, User: 2.6, Sys: 0.02

```

1 // dd_threads.cpp
2 #include <iostream>
3 #include <omp.h>
4 #include <vector>
5 #include <string>
6 #include <cstdlib>
7
8 #define DEFAULT_max_ligand 5
9 #define DEFAULT_nligands 120
10 #define DEFAULT_nthreads 4
11 #define DEFAULT_protein "hoach"
12
13 using namespace std;
14
15 // key-value pairs, e.g. {max_ligand, 5}
16 struct Pair {
17     int key;
18     string val;
19     Pair(int k, const string& v) : key(k), val(v) {}
20 };
21
22 // MR class provides a map-reduce interface
23 class MR {
24 private:
25     int max_ligand;
26     int nligands;
27     string protein;
28     queue<string> tasks;
29     vector<Pair> pairs;
30
31 public:
32     void GenerateTasks() {
33         for (int i = 0; i < nligands; i++) {
34             tasks.push(DEFAULT_protein + i);
35         }
36     }
37     void Map(const string& task) {
38         int key = task[0] - '0';
39         int Reduce(int key, const vector<Pair>& results, int count, string& s) {
40             MR m(max_ligand, nligands, protein);
41             m.Reduce(key, results, count, s);
42         }
43     };
44 };
45
46 int main() {
47     MR m(DEFAULT_max_ligand, DEFAULT_nligands, DEFAULT_protein);
48     m.GenerateTasks();
49     m.Reduce(0, m.pairs, m.pairs.size(), s);
50     return 0;
51 }

```

dd_threads 5: Real: 0.68, User: 2.58, Sys: 0.01

```

55 // dd_threads.cpp
56 #include <iostream>
57 #include <omp.h>
58 #include <vector>
59 #include <string>
60 #include <cstdlib>
61
62 #define DEFAULT_max_ligand 5
63 #define DEFAULT_nligands 120
64 #define DEFAULT_nthreads 4
65 #define DEFAULT_protein "hoach"
66
67 using namespace std;
68
69 // key-value pairs, e.g. {max_ligand, 5}
70 struct Pair {
71     int key;
72     string val;
73     Pair(int k, const string& v) : key(k), val(v) {}
74 };
75
76 // MR class provides a map-reduce interface
77 class MR {
78 private:
79     int max_ligand;
80     int nligands;
81     string protein;
82     queue<string> tasks;
83     vector<Pair> pairs;
84
85 public:
86     void GenerateTasks() {
87         for (int i = 0; i < nligands; i++) {
88             tasks.push(DEFAULT_protein + i);
89         }
90     }
91     void Map(const string& task) {
92         int key = task[0] - '0';
93     };
94 };
95
96 int main() {
97     MR m(DEFAULT_max_ligand, DEFAULT_nligands, DEFAULT_protein);
98     m.GenerateTasks();
99     m.Reduce(0, m.pairs, m.pairs.size(), s);
100     return 0;
101 }

```

The run times using User+Sys for dd_omp is 2.62, and for dd_threads is 2.59.

Using real, the times are 0.95 for dd_omp, and 0.68 for dd_threads.

4) Increase the maximum ligand length to 7, and rerun each program. What is the run time for each?

The default maximum ligand length is already 7 in each of the programs, so there are no reported changes.

V. Appendix

A. Link to Slack, GitHub, YouTube Channel, YouTube Video

Slack:

<https://csc-3210.slack.com>

GitHub:

<https://github.com/jupitorink>

GitHub Assignment Five Repository:

<https://github.com/jupitorink/Assignment5>

YouTube Channel:

<https://www.youtube.com/channel/UChTgBqHzG9G-Mvv-q7uK96g>

YouTube Video:

<https://www.youtube.com/watch?v=3XkongPbB1k&feature=youtu.be>

B. Task One Screenshot

Task Assignment Table:

<https://github.com/jupitorink/Assignment5/blob/master/Task%20%20Schedule.jpg>

C. Task Two Screenshot

GitHub Projects Page:

<https://github.com/jupitorink/Assignment5/blob/master/projectspage.png>

D. Task Three Parallel Programming Report, Screenshots

Parallel Programming Report, Screenshots

https://github.com/jupitorink/Assignment5/blob/master/FiveThreads_dd_omp.png

https://github.com/jupitorink/Assignment5/blob/master/FiveThreads_dd_threads.png

https://github.com/jupitorink/Assignment5/blob/master/FourThreads_dd_omp.png

https://github.com/jupitorink/Assignment5/blob/master/FourThreads_dd_threads.png

https://github.com/jupitorink/Assignment5/blob/master/OneThread_dd_omp.png

https://github.com/jupitorink/Assignment5/blob/master/OneThread_dd_threads.png

https://github.com/jupitorink/Assignment5/blob/master/Question2_WordCount.png

https://github.com/jupitorink/Assignment5/blob/master/RunTime_dd_serial.png

https://github.com/jupitorink/Assignment5/blob/master/ThreeThreads_dd_omp.png

https://github.com/jupitorink/Assignment5/blob/master/ThreeThreads_dd_threads.png

https://github.com/jupitorink/Assignment5/blob/master/TwoThreads_dd_omp.png

https://github.com/jupitorink/Assignment5/blob/master/TwoThreads_dd_threads.png