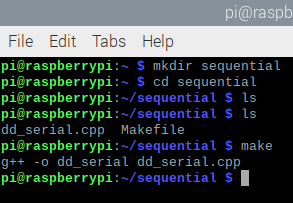
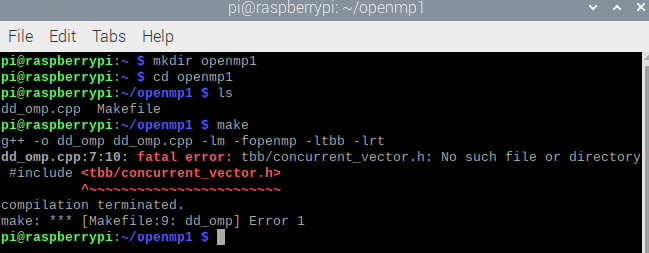
Tony Derado

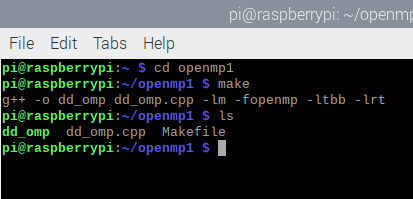
In the part we learned about the design of drugs using molecular model computation. The programs match proteins with ligands to find the best pair. The codes apply map-reduce, in which the ligands are mapped using their binding score and reduce pulls together the sequences to find the highest scoring ligand pair.

The first code uses sequential sorting giving us an idea on how much of a difference there is between it and parallel. It was set up easily as once the directory was created the files were just copied in and we ran Makefile to create an executable. Sequential by far to the longest to run with over 2 minutes wait period. By only comparing the pairs one by one we can see that sequential doesn’t use most of the Raspberry’s capabilities.

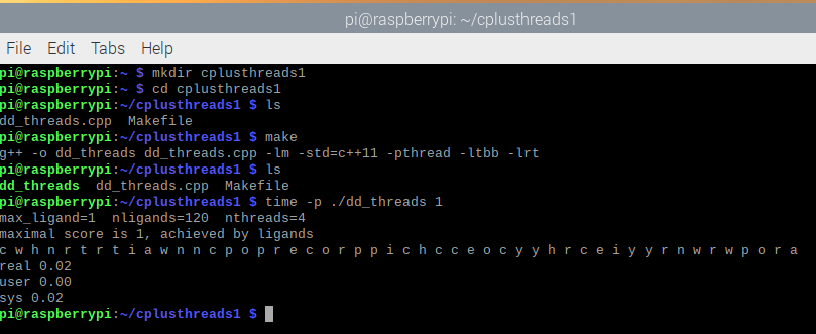


The second code we used for searching to the top ligands was OpenMP. It took a lot longer to implement this one as it used TBB (Thread Building Blocks) which isn’t part of the basic libraries. Figuring out where to get TBB and how to add it took some time, and even after getting installed it had to have a little modification as an error would pop up when making the dd\_omp. Once the TBB was done OpenMP ran quickly taking .02 seconds to do just what sequential did.

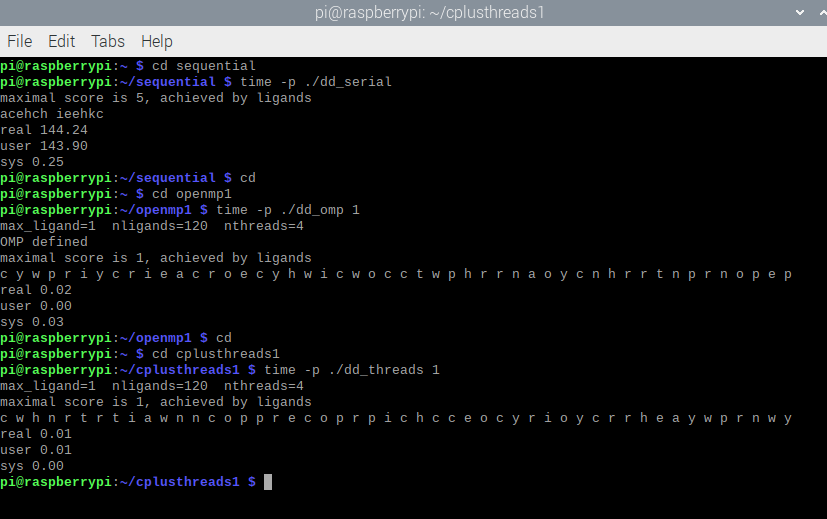




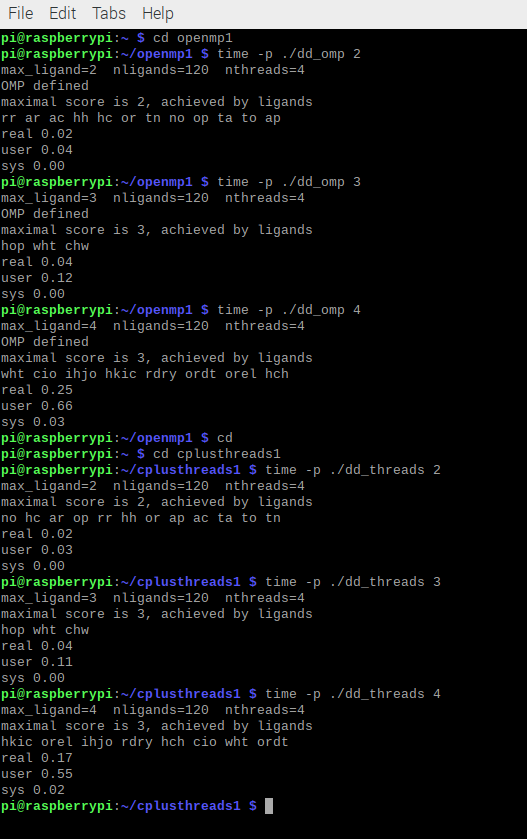
The C++11 solution was as simple to implement as sequential as TBB was set up and the rest had no problems. It ran .01 which was a little faster than OpenMP but hard to compare with such close numbers.



Run Times:



|  |  |
| --- | --- |
| Implementation | Time(s) |
| dd\_serial | 144.24 |
| dd\_omp | .02 |
| dd\_threads | .01 |



|  |  |  |  |
| --- | --- | --- | --- |
| Implementation | Time (s) 2 Threads | Time (s) 3 Threads | Time (s) 4 Threads |
| dd\_omp | .02 | .04 | .25 |
| dd\_threads | .02 | .04 | .17 |

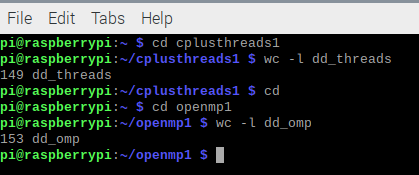
Questions:

1. What approach is the Fastest?

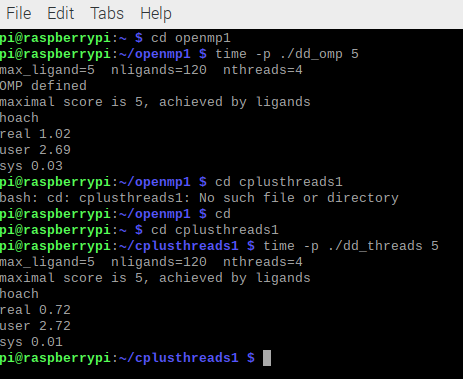
* The fastest running code is C++11, with the time at 4 threads beating OpenMP and sequential isn’t even close ever.

1. Determine the number of lines in each file. How does the C++11 implementation compare to the OpenMP implementation?

* The C++11 code has fewer lines of code, with 149 to the OpenMP’s 153 lines.



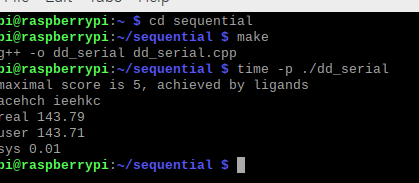
1. Increase the number of threads to 5 threads, what is the run time for each?

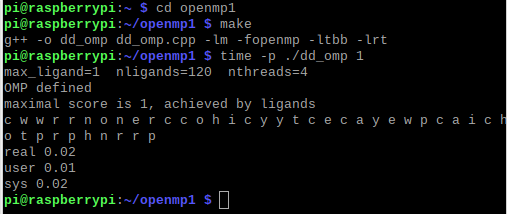


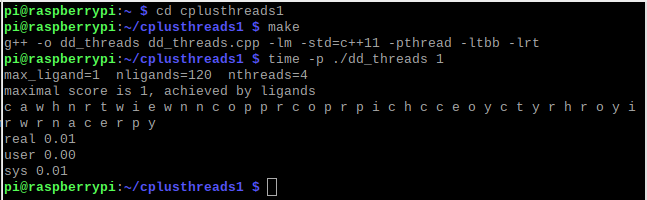
|  |  |
| --- | --- |
| Implementation | Time (s) 5 Threads |
| dd\_omp | 1.02 |
| dd\_threads | .72 |

The time appear of both appear to increase by around 4 times.

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







|  |  |
| --- | --- |
| Implementation | Time (s) |
| dd\_serial | 143.79 |
| dd\_omp | .02 |
| dd\_threads | .01 |

The times appear to be the same or close to the original. But with all the tests done on these three methods its clear that C++11 is the best, as it uses less lines and time to complete the job.