2017 Oct 12th
Yixue Wang

Report:

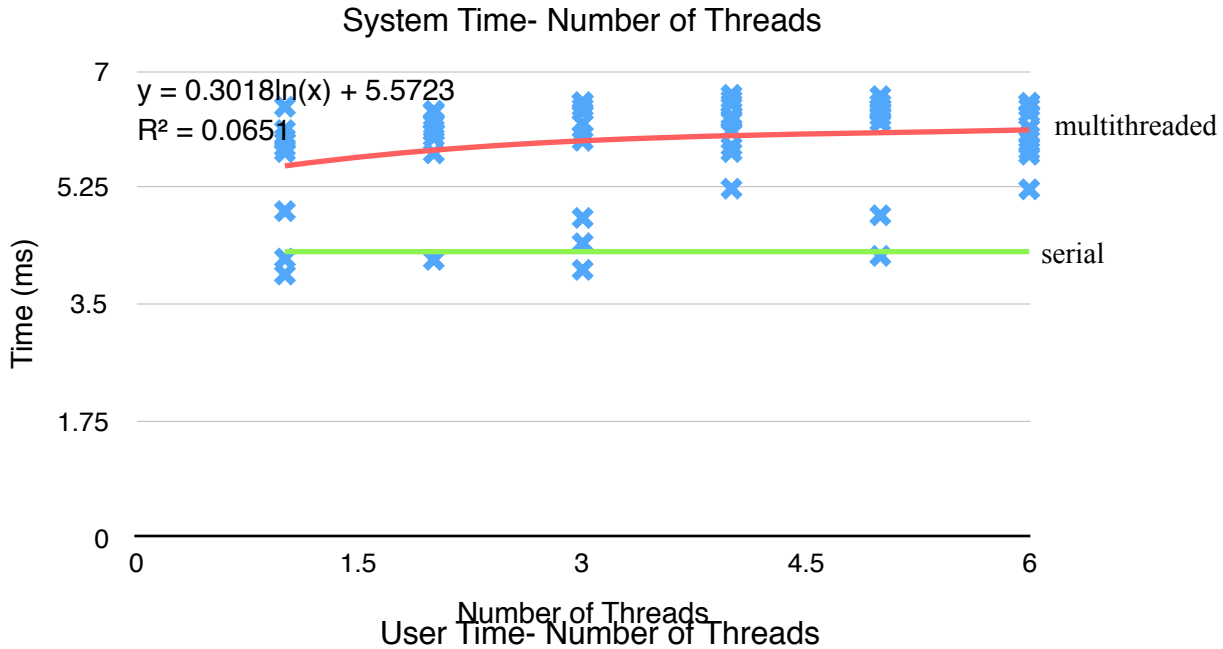
I test ./proj4 in macintosh O.S., which has 2 CPU cores. For each graph blow, the red lines correspond to the time cost in multithreaded architecture and the green lines correspond to the time cost in serial architecture.

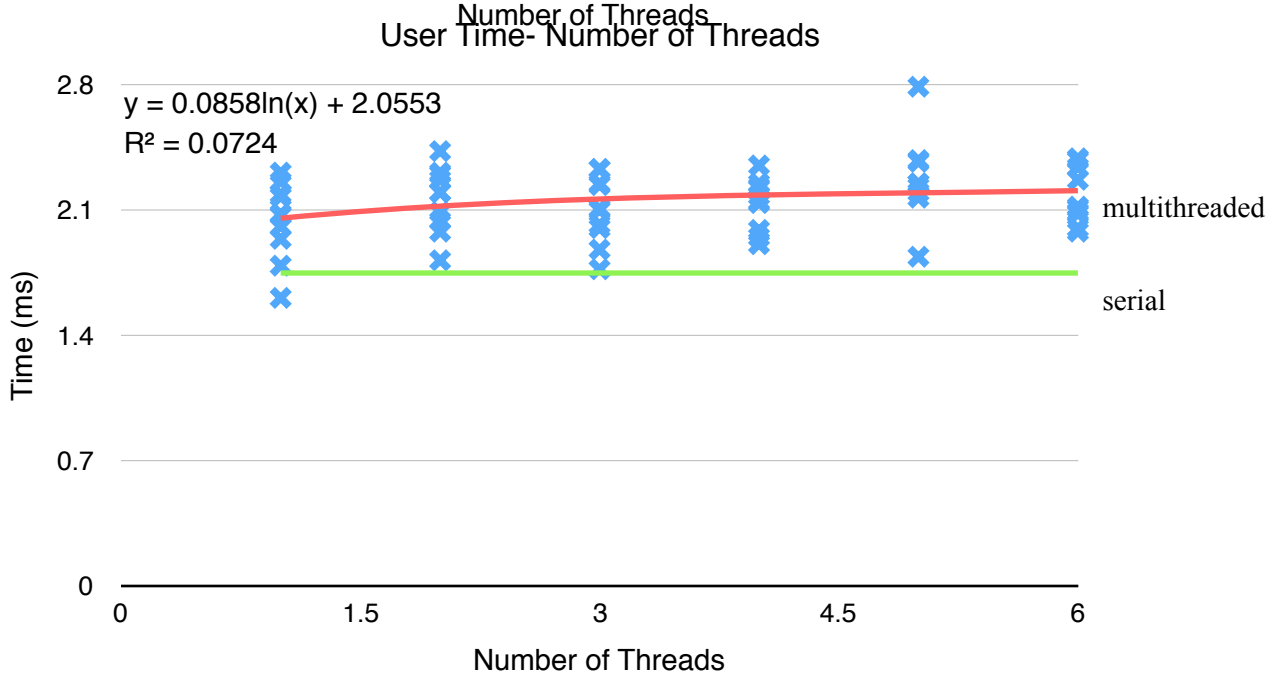
We can see that the system time cost in serial architecture is usually shorter, this is because in this architecture there is no need to spend time creating worker threads. Also since we only have the main thread doing all the things, there is no context switch happens and we can save time because we do not need to spend time paging the code and save modified data to disk.

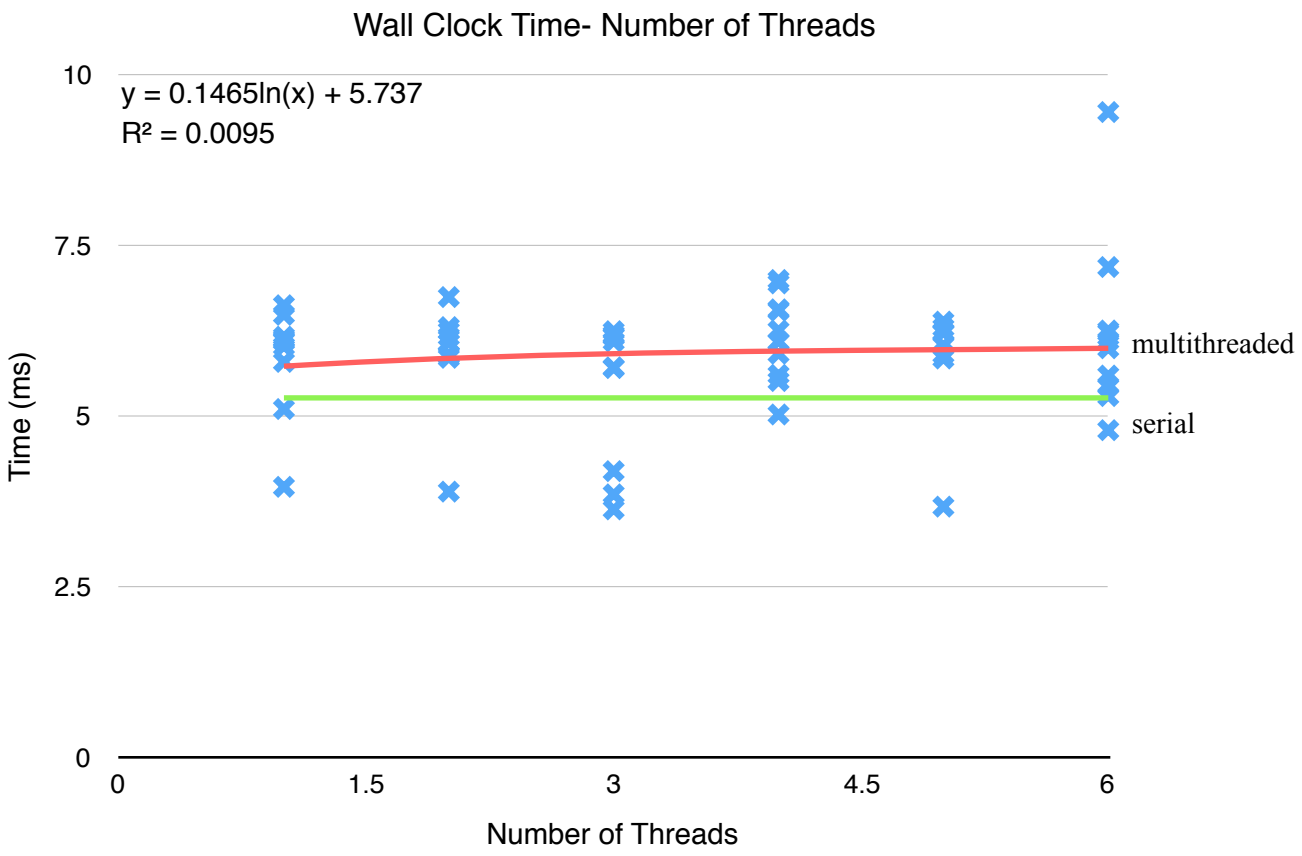
The serial architecture takes less user time because the code in user mode of this architecture is shorter therefore runs faster.The wall-clock time of both is similar because the project directory contains very small number of files.

Since the project directory only contains 3 files, the result using multi-threaded architecture might be inaccurate.

Graphs of running .proj4 on the folder containing proj4.c, makefile and executable:





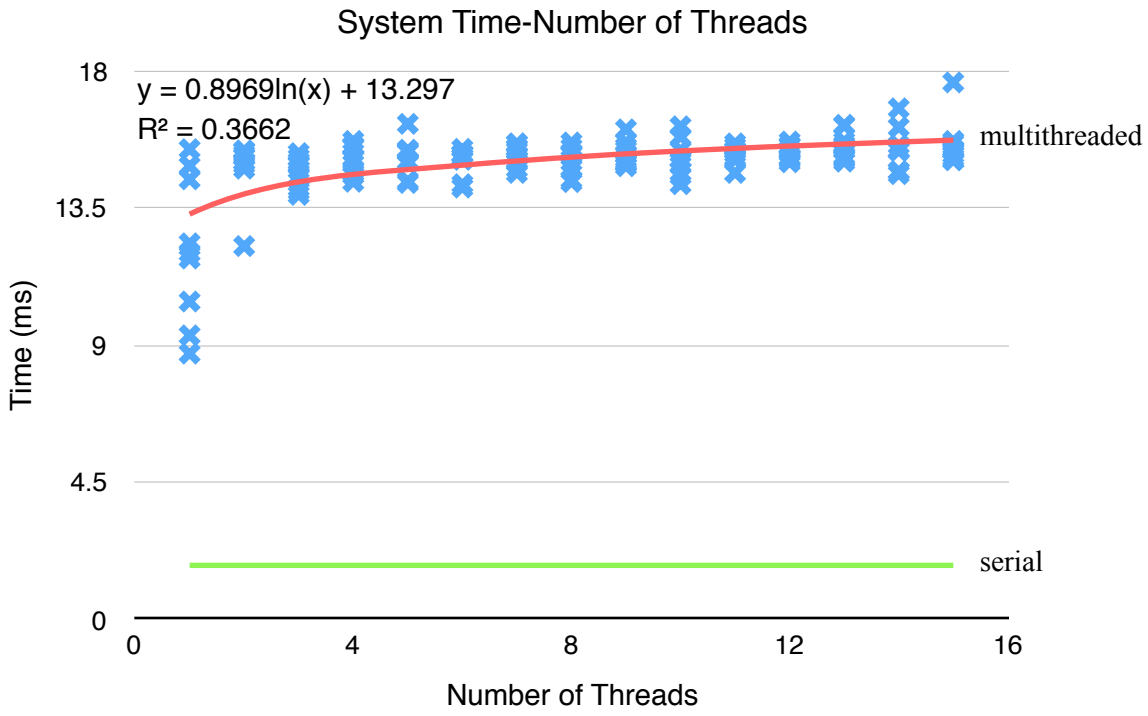


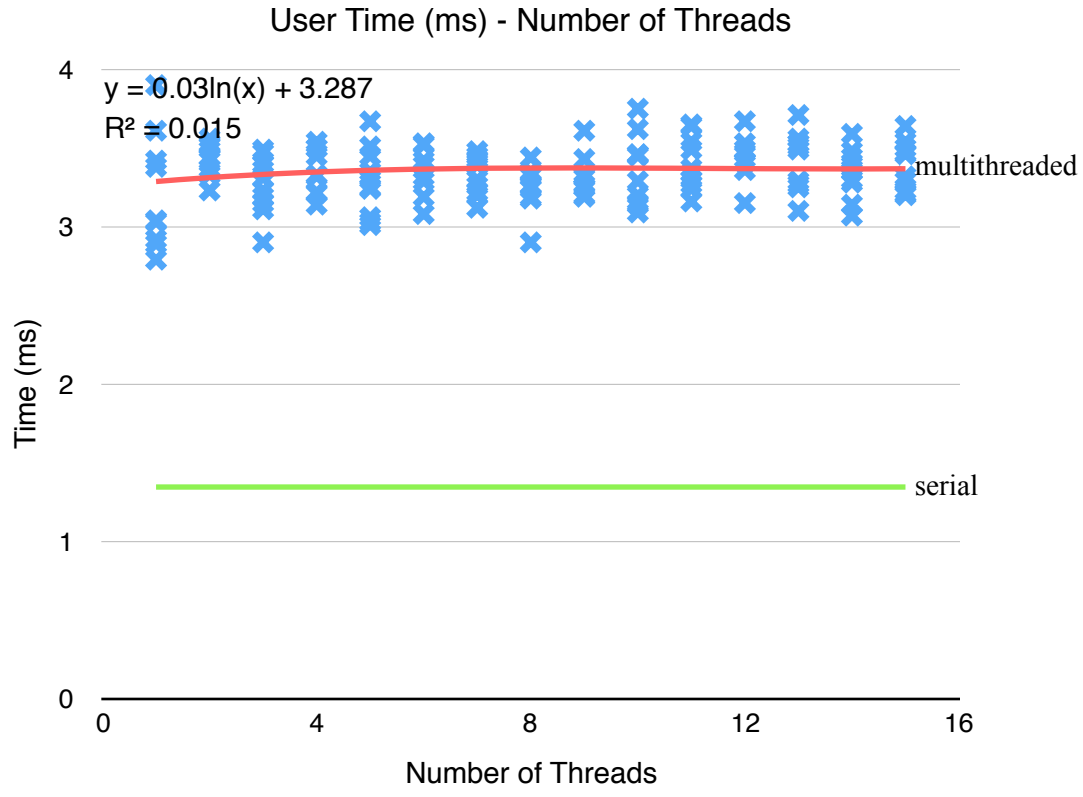
I tested proj4 on /dev/* and /bin/*, which contain more files than project directory does. From the graphs we can see that as the number of threads increases, the average system time also increases. This is because when we have more threads running at the same time, they may interrupt each other outside the critical zone, which causes context switch. If a context switch happens, the CPU will need time to do paging and save modified data, which takes some system CPU time.

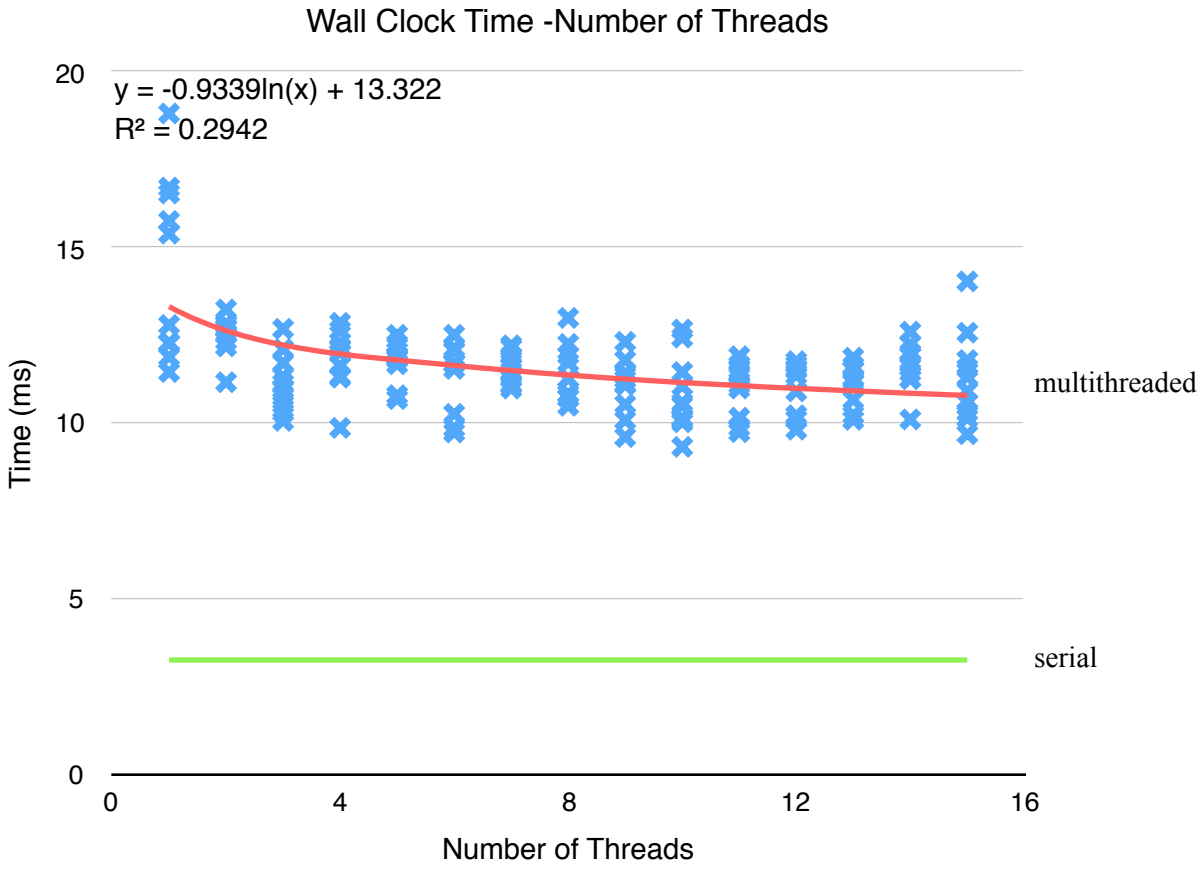
The CPU user time almost remains the same as the number of threads increases. Because the use-level executions which are depending on the code in either architecture are almost the same, the CPU user time should also be the same.

The wall-clock time decreases as we have more threads. The wall-clock time is the actually time taken by this process including the time periods that this process in run, ready, blocked states. Since I test proj4 on macintosh O.S. with 2 CPU cores, the CPU can allow multiple threads running as the same time. Therefore the multi-threaded architecture takes far less time than the serial architecture as far as it allows more than 1 requests to be handled at the same time. In multi-threaded architecture, as more files to be handled at the same time, the work can be done in a shorter time period. However since the increase of number of threads also increases the probability of context switching, the wall-clock time decreases slower and slower as the number of threads increases.

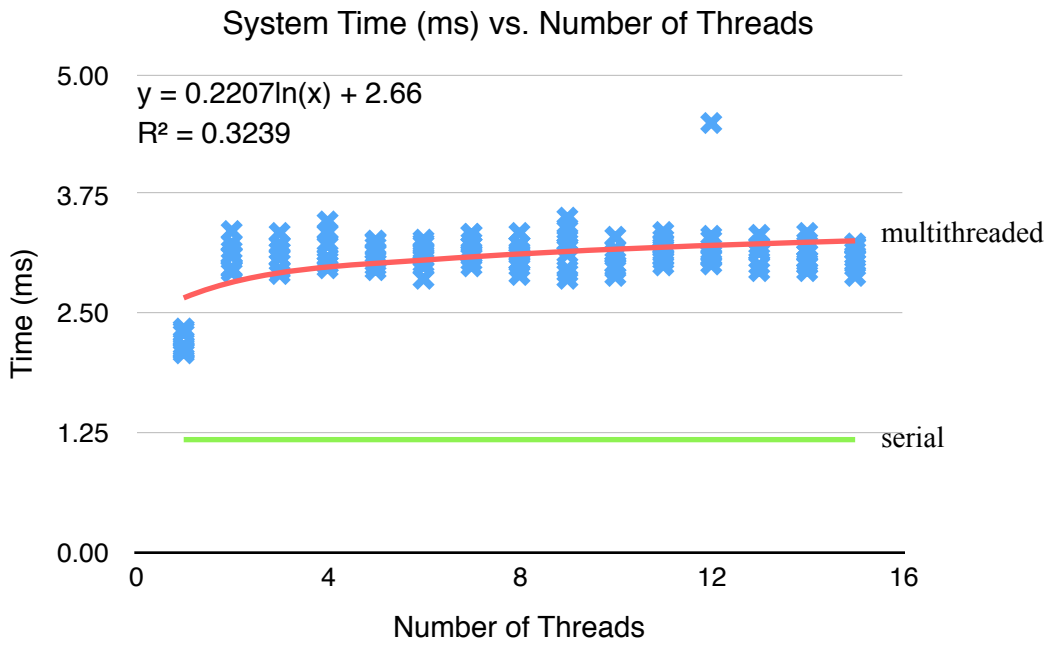
Graph of running proj4 on /dev/*:

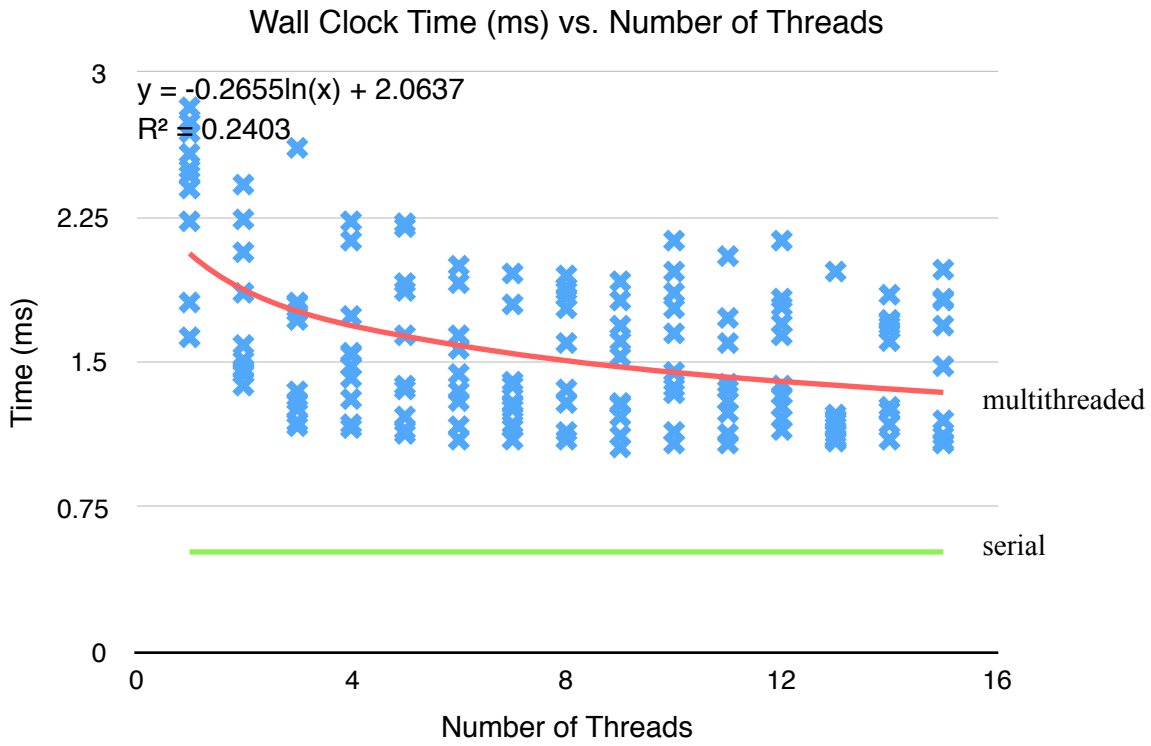


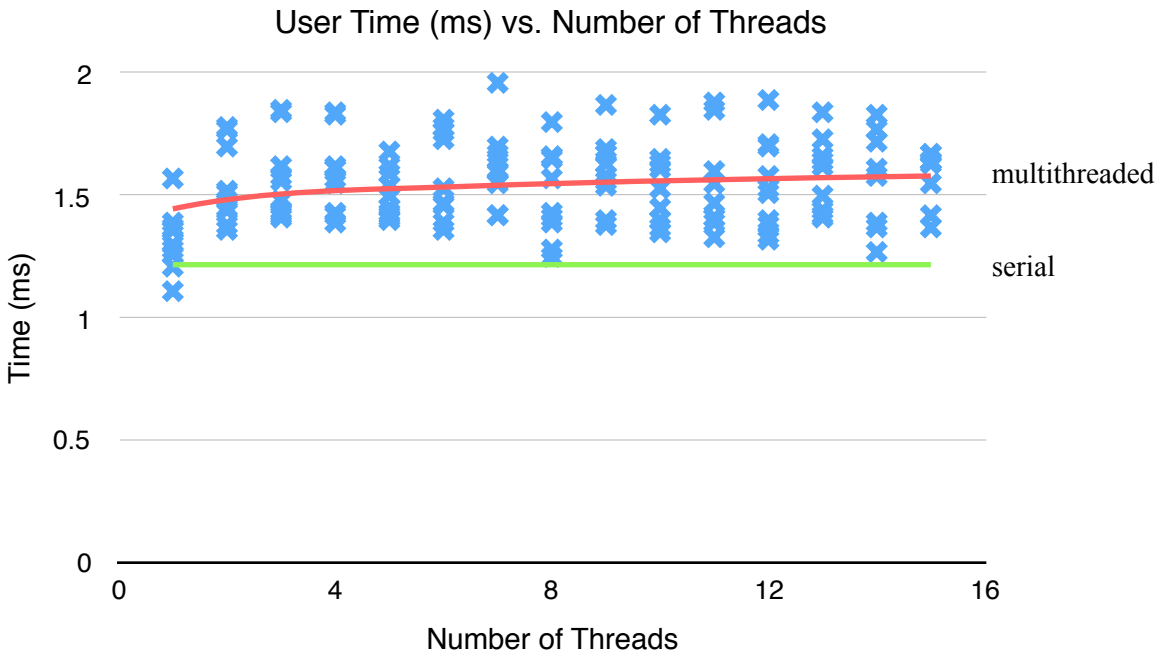



Wall Clock Time -Number of Threads
y = -0.9339ln(x) + 13.322
R² = 0.2942
20
15
10
5
0
Time (ms)
0
4
8
12
16
Number of Threads
multithreaded
serial

Graph of running proj4 on /bin/*:






User Time (ms) vs. Number of Threads
Time (ms)
2
1.5
1
0.5
0
0
4
8
12
16
Number of Threads
multithreaded
serial