Universität Basel | Spring Semester 2018
High Performance Computing | Lukas Valatka
Ahmed Eleliemy, M.Sc., Ali Mohammed, M.Sc. | May 22, 2018

Assignment 5: Correctness, Fault tolerance, Performance analysis, and Reproducibility

1. Code correctness

See MUST_OUTPUT.html. As the MUST tool suggests, one request is not freed before the finalize.

Cause: After investigation, it seems the issue is that the *request* variable, referencing *MPI_Isend* handle (which is used for checking status of an established asynchronous communication), is not freed.

Solution: Calling *MPI_Request_free(&request)* before finalize solves the issue

See MUST_OUTPUT_corrected.html for the final "green" MUST report.

2. Fault tolerance

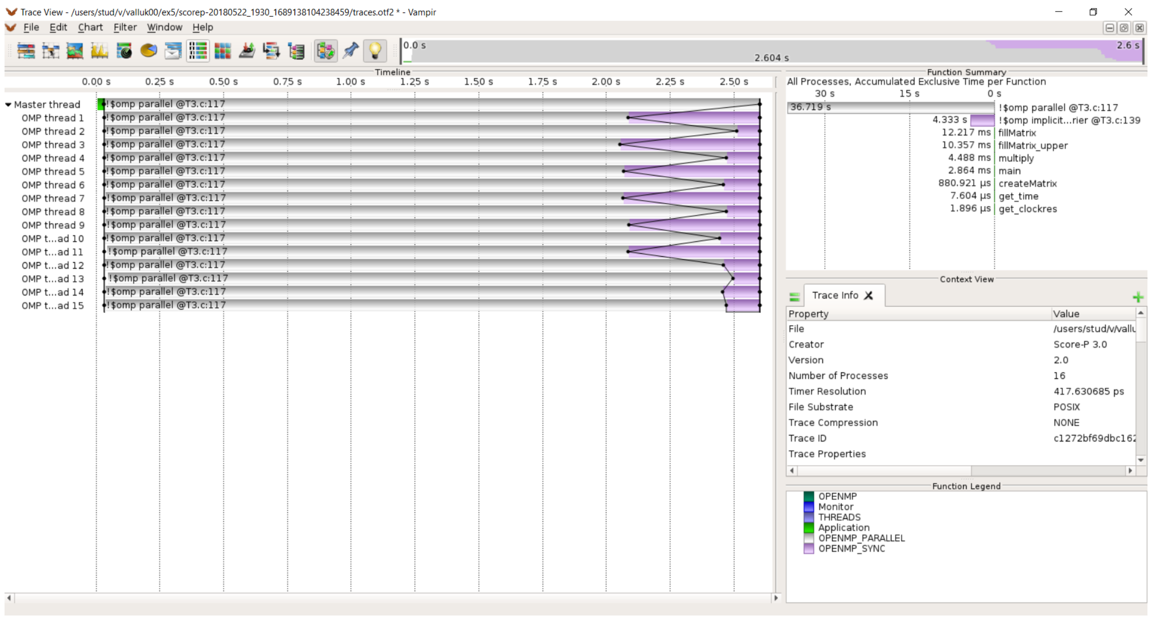
See the T2.c code for implementation.

3. Performance analysis

a)

Timer resolution is 1 nano seconds.
Starting benchmark with mSize = 1000.
Number of threads = 16
MATRIX SIZE: 1000, GFLOPS: 0.715400
Mean execution time: 2.603643

b)



Trace View - /users/stud/v/valluk00/ex5/scorep-20180522_1930_1689138104238459/traces.otf2 * - Vampir

All Processes, Accumulated Exclusive Time per Function

36.719 s	!$omp parallel @T3.c:117
4.333 s	!$omp implicit...rier @T3.c:139
12.217 ms	fillMatrix
10.357 ms	fillMatrix_upper
4.488 ms	multiply
2.864 ms	main
880.921 µs	createMatrix
7.604 µs	get_time
1.896 µs	get_clockres

Property	Value
File	/users/stud/v/vallu
Creator	Score-P 3.0
Version	2.0
Number of Processes	16
Timer Resolution	417.630685 ps
File Substrate	POSIX
Trace Compression	NONE
Trace ID	c1272bf69dbc162
Trace Properties	

🟩	OPENMP
🟦	Monitor
🟦	THREADS
🟩	Application
⬜	OPENMP_PARALLEL
🟪	OPENMP_SYNC

Figure 1: Vampir visualization of T3 execution.

c)

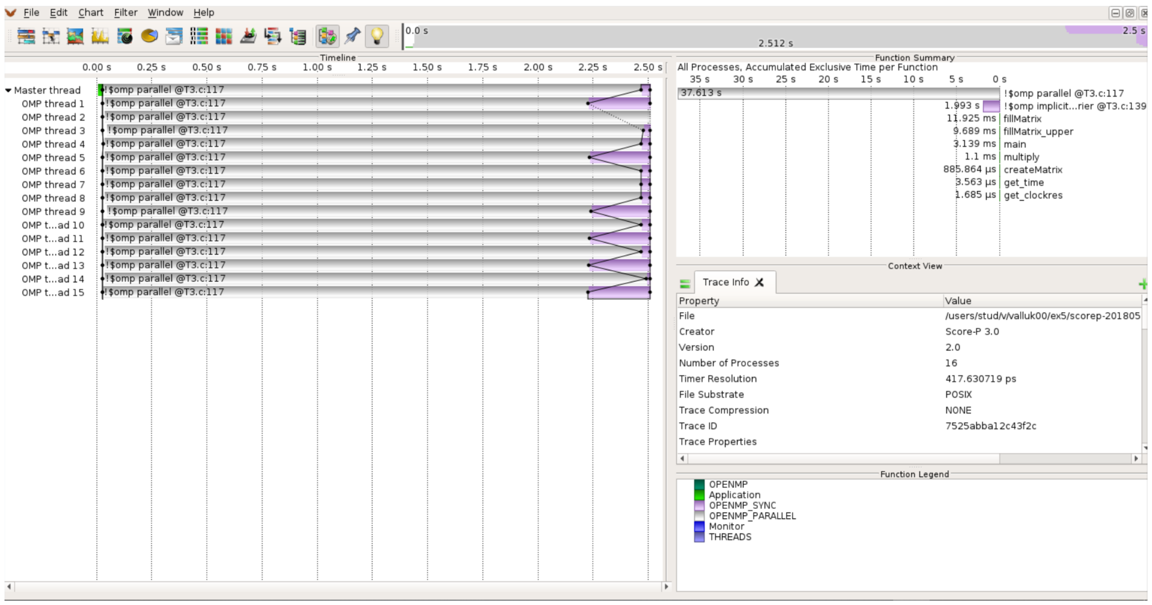
The main issue is that matrices are sparse. It is clear that every second thread finishes much quicker than the following one (i.e. there's irregular workload for each thread, as some matrix cells contain 0.0)

d)

Timer resolution is 1 nano seconds. Starting benchmark with mSize = 1000. Number of threads = 16 MATRIX SIZE: 1000, GFLOPS: 0.776621 Mean execution time: 2.398396

Added schedule(guided) to regularize the workload across threads.

e)



All Processes, Accumulated Exclusive Time per Function

37.613 s	!$omp parallel @T3.c:117
1.993 s	!$omp implicit...rier @T3.c:139
11.925 ms	fillMatrix
9.689 ms	fillMatrix_upper
3.139 ms	main
1.1 ms	multiply
885.864 µs	createMatrix
3.563 µs	get_time
1.685 µs	get_clockres

Property	Value
File	/users/stud/v/valluk00/ex5/scorep-201805
Creator	Score-P 3.0
Version	2.0
Number of Processes	16
Timer Resolution	417.630719 ps
File Substrate	POSIX
Trace Compression	NONE
Trace ID	7525abba12c43f2c
Trace Properties	

🟩	OPENMP
🟩	Application
🟪	OPENMP_SYNC
⬜	OPENMP_PARALLEL
🟦	Monitor
🟦	THREADS

Figure 2: Vampir visualization of T3 execution (optimized)

4. Reproducibility

Reproduced results vastly differ from the original ones. In general, the runtimes I have received are much lower than the one's of the author of the script.

Specifically, it is clear that for the OpenMP scenario, every runtime should be lower, except for the last one (it's too low in the author's report).

Regarding rank=2 scenario, it nearly matches mine, though 8 thread run seems unreasonably lower in the author's graph. Somehow, there is a huge gap from 8 thread to 16 thread scenarios runtime results, which I was not able to reproduce (i.e. the runtime always decreases as the thread count increases).

Also, the rank=3 scenario seems unreasonably flat in the author's graph. I have measured a decrease as the number of threads increases.

Solution: The author should have specified the hardware parameters (they might have changed in a year). Also, the batching scripts should have been included, as the author may have used slightly different configuration.

Regarding the third run, it seems the results were interpolated (i.e. not actually measured for 4 and 8 thread runs), as the curve is too flat.

In addition, it may seem the author may have used several runs to average the runtime - this should have been specified as well, if so.

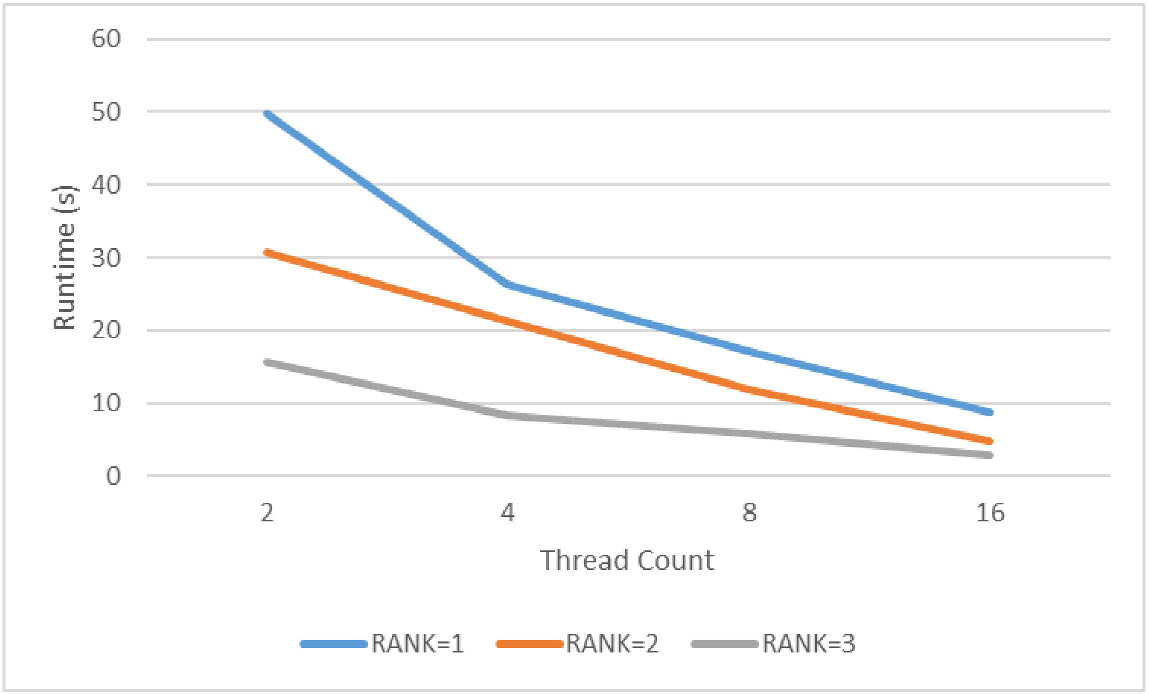


Figure 3: Reproduced T4.c runtime results.