Advanced C++ Course

Linaro Forge

CSCS

Introduction

Linaro Forge tools

- MAP
  - "Linaro MAP is a high-performance (sampling) profiler designed to optimize the efficiency of software running on multicore processors."
  - Designed for 'hot-spot' analysis with stack traces and performance metrics
  - Supports C/C++/Fortran/Python
  - Nvidia & AMD GPU profiling (AMD not supported on CSCS)
  - Performance Reports
- DDT
  - "Linaro DDT is an advanced debugger designed to simplify the troubleshooting and optimization of complex, high-performance computing (HPC) applications."
  - Supports C/C++/Fortran/Python
  - Nvidia & AMD GPUs (AMD not supported on CSCS)
  - Supports containers (to be tested with CE)

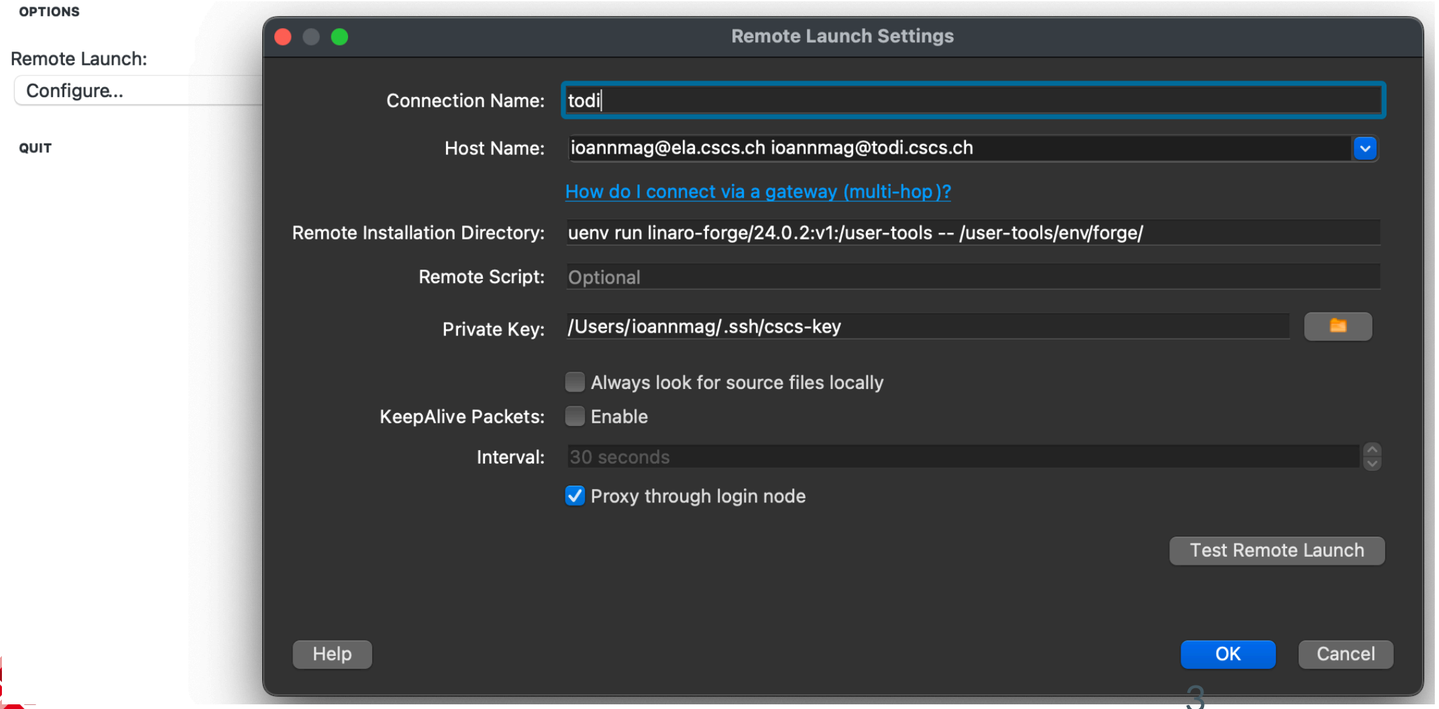
Documentation

- [Knowledge Base] Debugging
- Linaro Forge UENV

Forge client configuration

- Forge client on local machine (Linux, MacOS, Windows)
- License
  - Max processes 272 / Max users 256
  - We pay for support so if any feature is missing RFE is possible

TODI remote setup



MAP

Instructions

Todi

```
uenv start --view=linaro:forge,prgenv-gnu:default prgenv-gnu/24.7 linaro-forge/24.0.2
map -n 4 --mpi=slurm --mpiargs="-A csstaff -p debug -t 10" --profile ./matrix_mult 4
```

Local machine

- Connect to `todi`
- `LOAD PROFILE DATA FILE`

DDT

Instructions

Local machine

- Connect to `todi`
- `MANUAL LAUNCH (ADVANCED)`
- Create correct listener

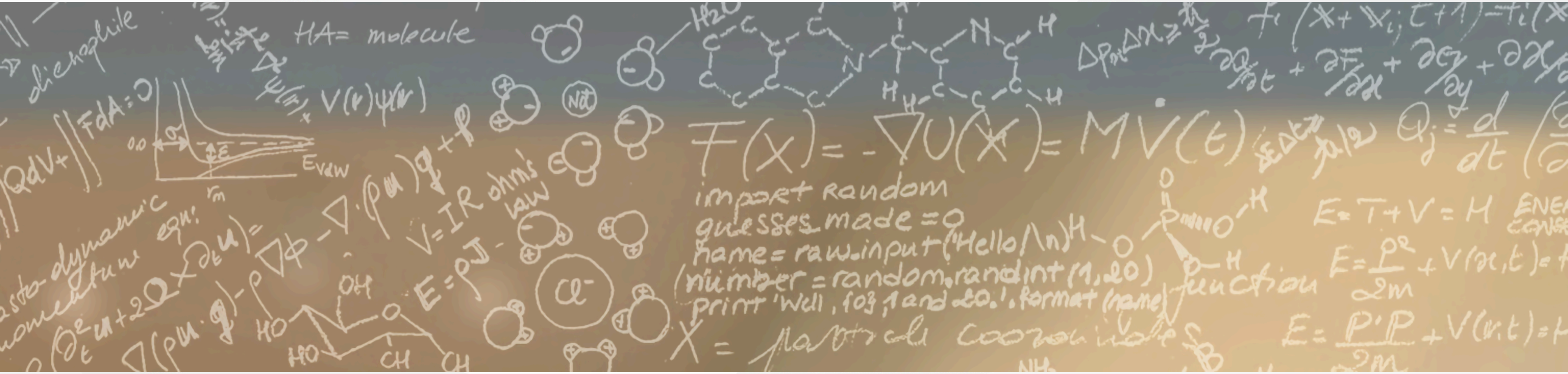
Todi

```
uenv start --view=linaro:forge,prgenv-gnu:default prgenv-gnu/24.7 linaro-forge/24.0.2
srun -n 4 -A csstaff -t30 -p debug ddt-client ./matrix_mult 4
```

Demo

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Questions?

Resources

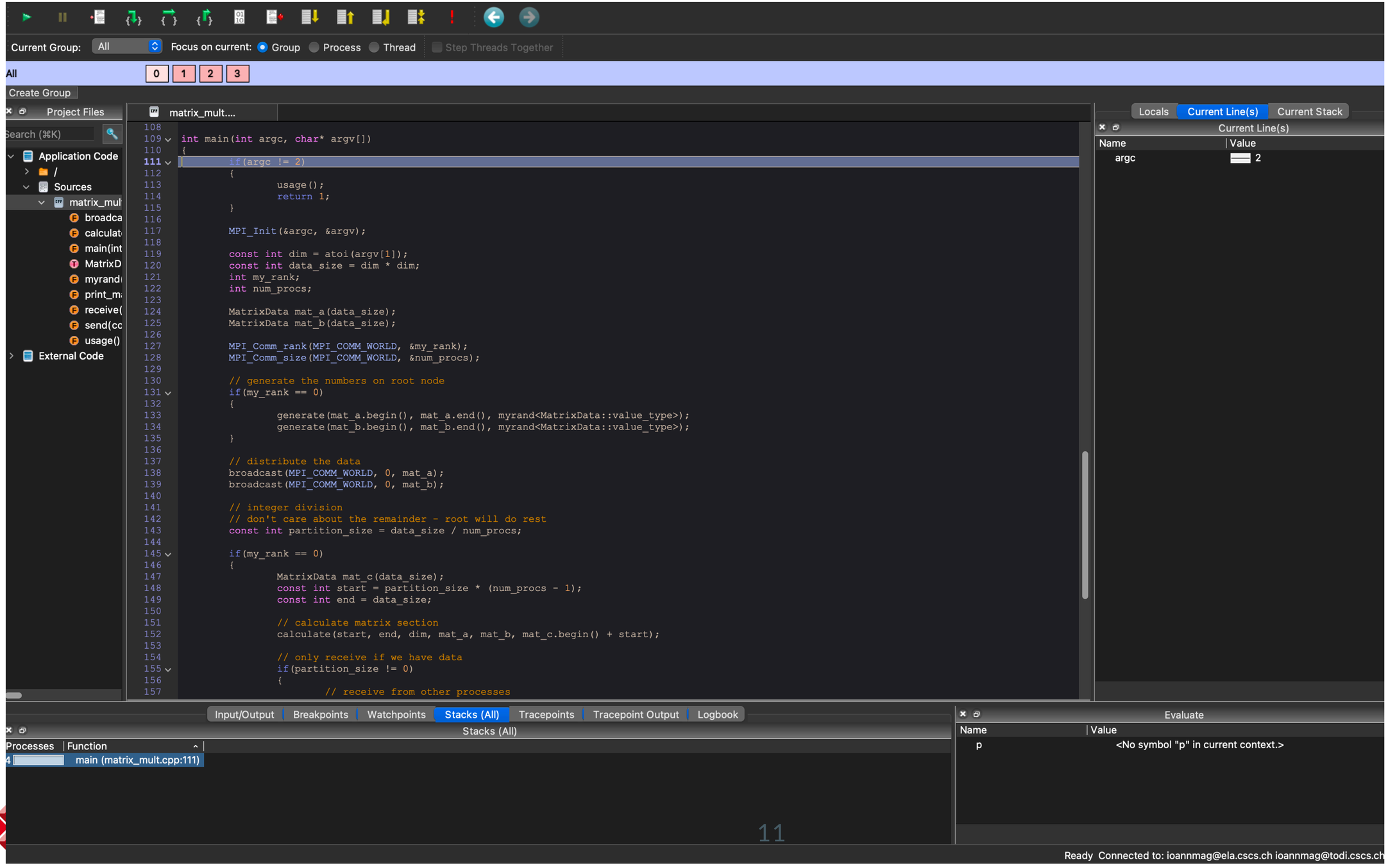
- Linaro MAP website
- Linaro DDT website
- Ensuring Program Correctness with Linaro DDT Rudy Shand

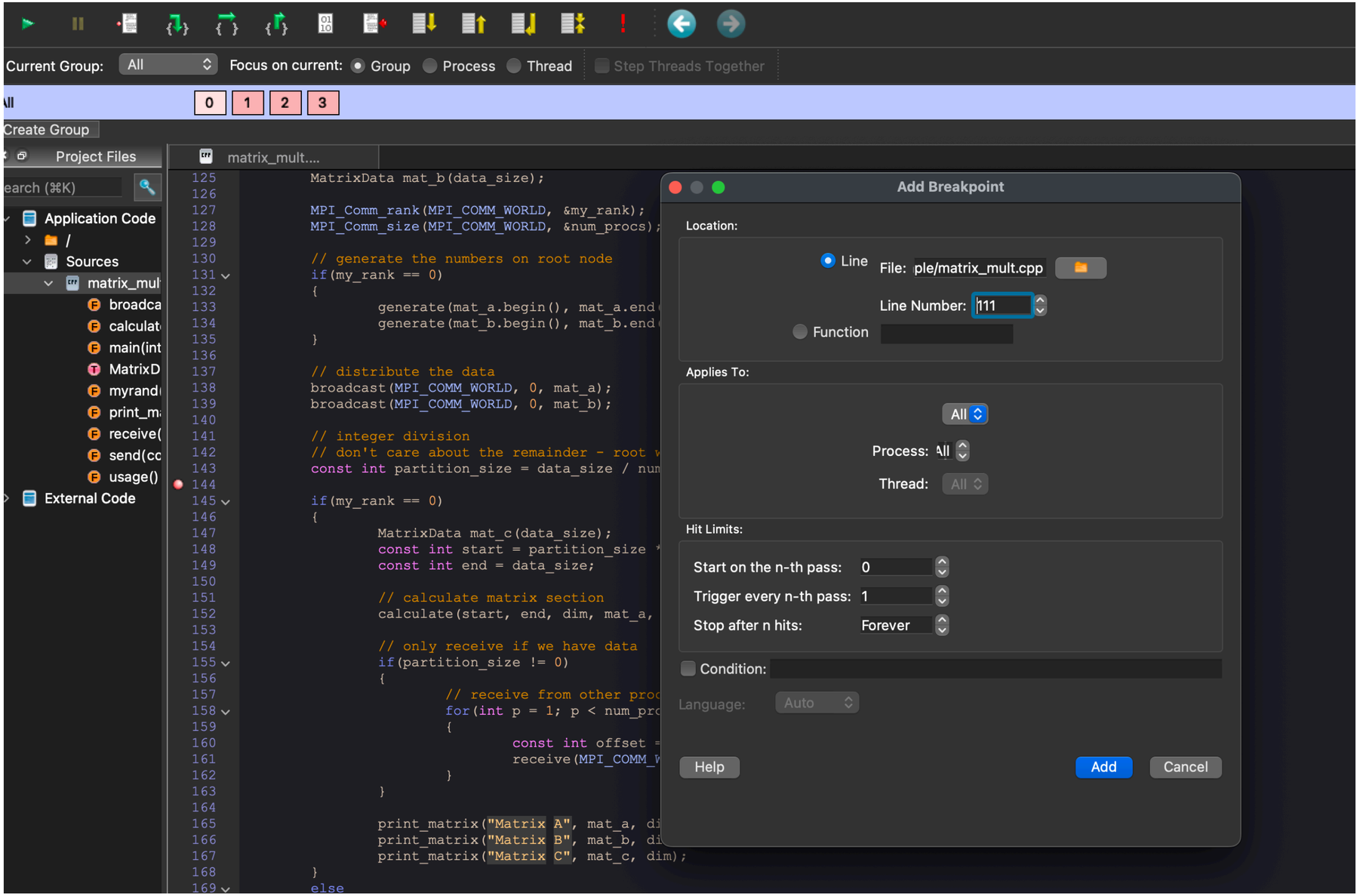
Backup slides

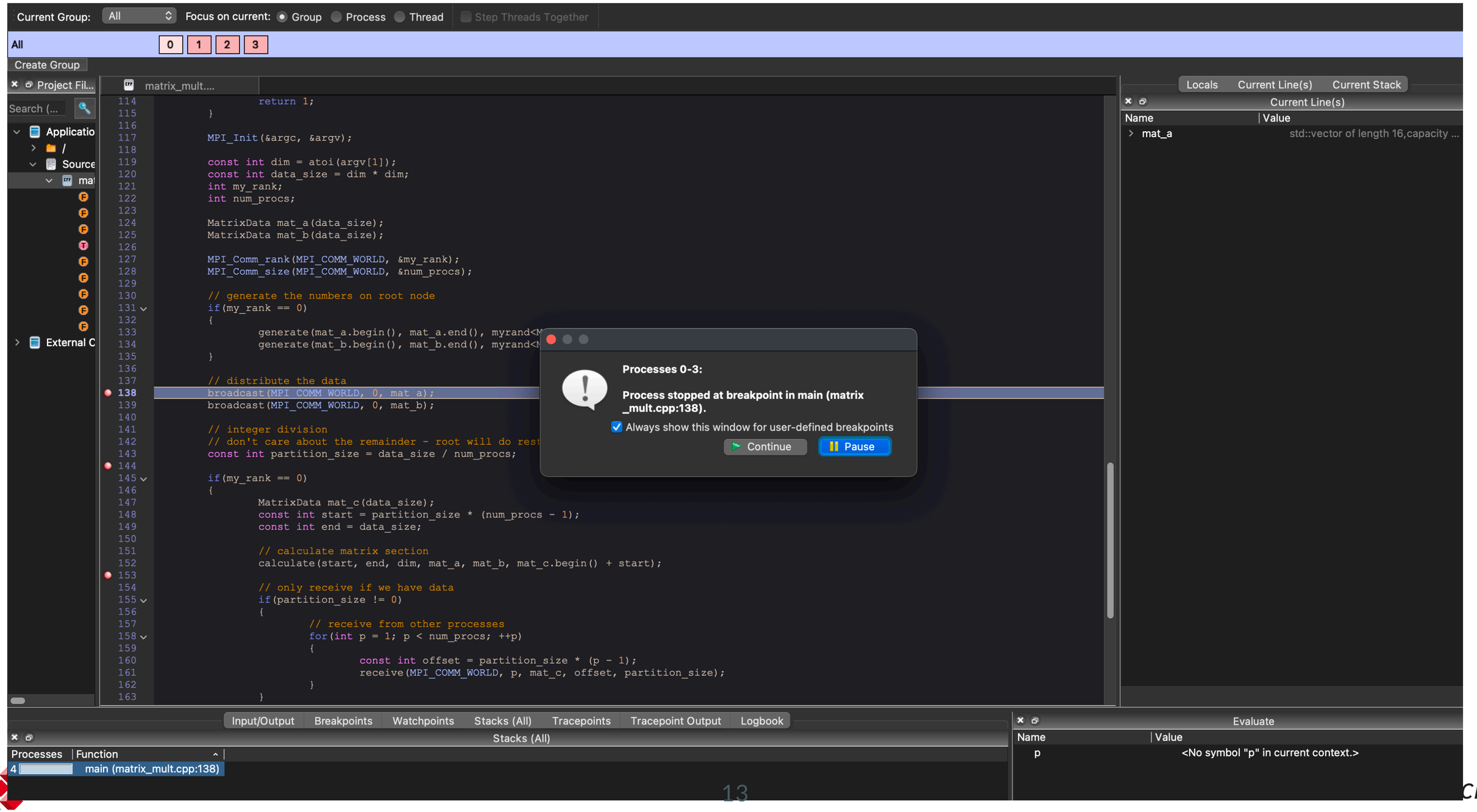
DDT

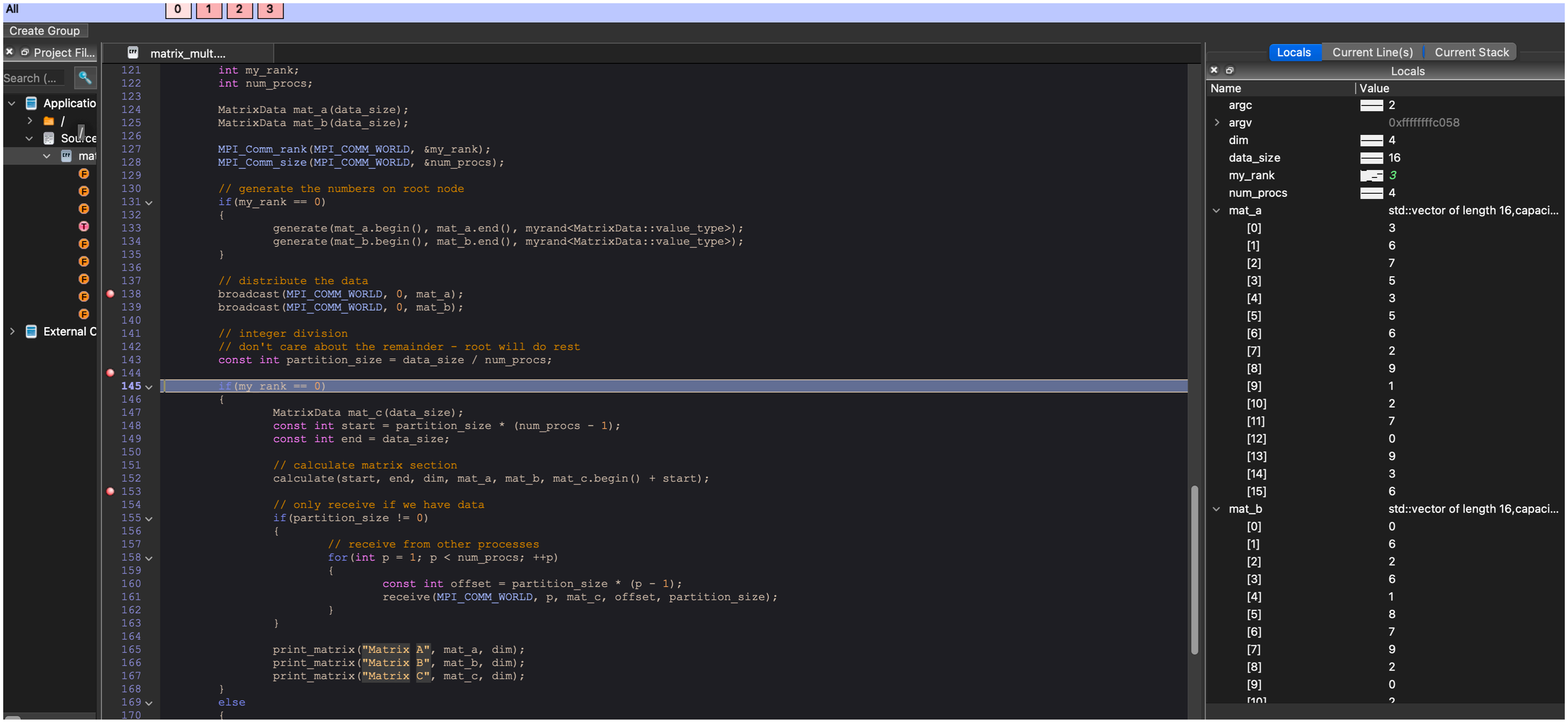

linaro forge
linaro DDT
linaro MAP
Get trial licence
Support
inaroforge.com
RUN
Run and debug a program.
ATTACH
Attach to an already running program.
OPEN CORE
Open a core file from a previous run.
MANUAL LAUNCH (ADVANCED)
Manually launch the backend yourself.
OPTIONS
Remote Launch:
todi
QUIT
Linaro Forge 24.0.3 Connected to: ioannmag@ela.cscs.ch ioannmag@todi.cscs.ch


Linaro DDT - Linaro Forge 24.0.3
linaro
forg
Runtime: manual launch
Details
Number of Processes: 4
CUDA
Details
ROCm
Details...
Intel Xe
Details...
Memory Debugging: Balanced,No guard pages,Backtraces,Preload
Details...
Plugins: none
Details
Help
Options
Listen
Cancel
Get trial licence
Support
linaroforge.com
Linaro Forge 24.0.3 Connected to: ioannmag@ela.cscs.ch ioannmag@todi.cscs.ch

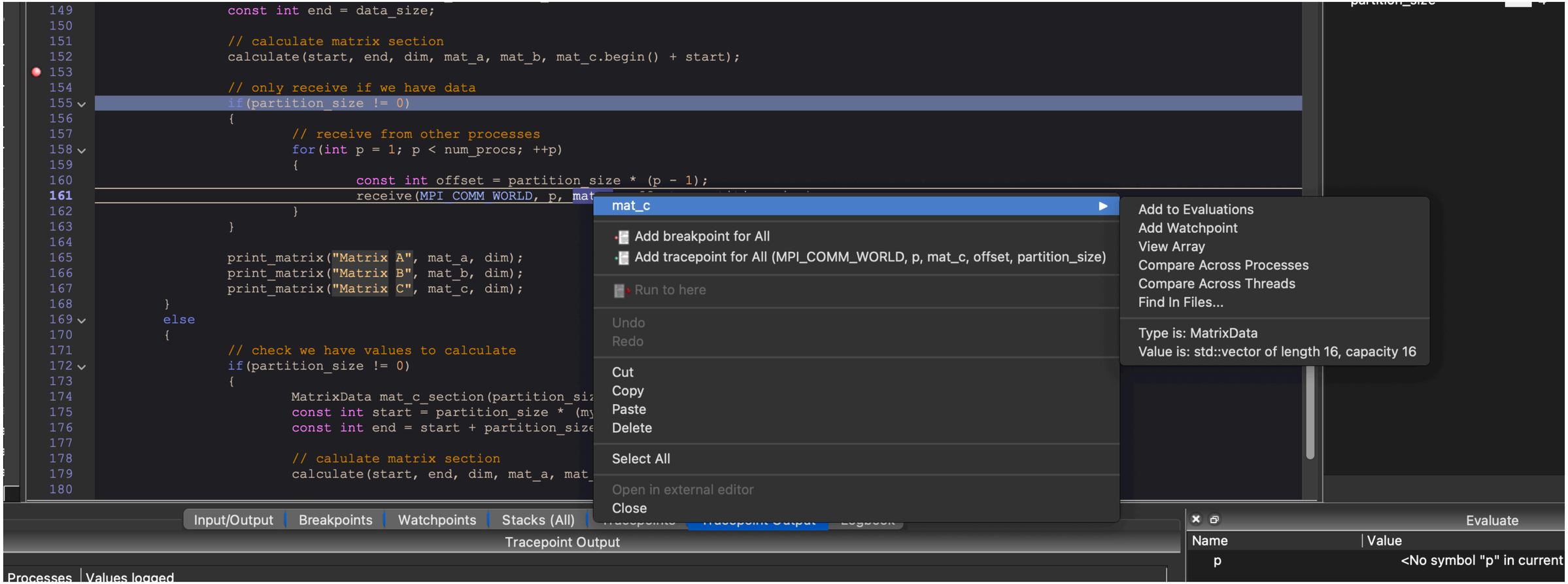

Current Group: All
Focus on current: Group Process Thread
Step Threads Together
All 0 1 2 3
Create Group
Project Files
Search (⌘K)
Application Code
/
Sources
matrix_mul
broadca
calculat
main(int
MatrixD
myrand
print_m
receive(
send(cc
usage()
External Code
matrix_mult....
int main(int argc, char* argv[])
{
if(argc != 2)
{
usage();
return 1;
}
MPI_Init(&argc, &argv);
const int dim = atoi(argv[1]);
const int data_size = dim * dim;
int my_rank;
int num_procs;
MatrixData mat_a(data_size);
MatrixData mat_b(data_size);
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
// generate the numbers on root node
if(my_rank == 0)
{
generate(mat_a.begin(), mat_a.end(), myrand<MatrixData::value_type>);
generate(mat_b.begin(), mat_b.end(), myrand<MatrixData::value_type>);
}
// distribute the data
broadcast(MPI_COMM_WORLD, 0, mat_a);
broadcast(MPI_COMM_WORLD, 0, mat_b);
// integer division
// don't care about the remainder - root will do rest
const int partition_size = data_size / num_procs;
if(my_rank == 0)
{
MatrixData mat_c(data_size);
const int start = partition_size * (num_procs - 1);
const int end = data_size;
// calculate matrix section
calculate(start, end, dim, mat_a, mat_b, mat_c.begin() + start);
// only receive if we have data
if(partition_size != 0)
{
// receive from other processes
Locals Current Line(s) Current Stack
Current Line(s)
Name Value
argc 2
Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook
Stacks (All)
Processes Function
4 main (matrix_mult.cpp:111)
Evaluate
Name Value
p <No symbol "p" in current context.>
Ready Connected to: ioannmag@ela.cscs.ch ioannmag@todi.cscs.ch

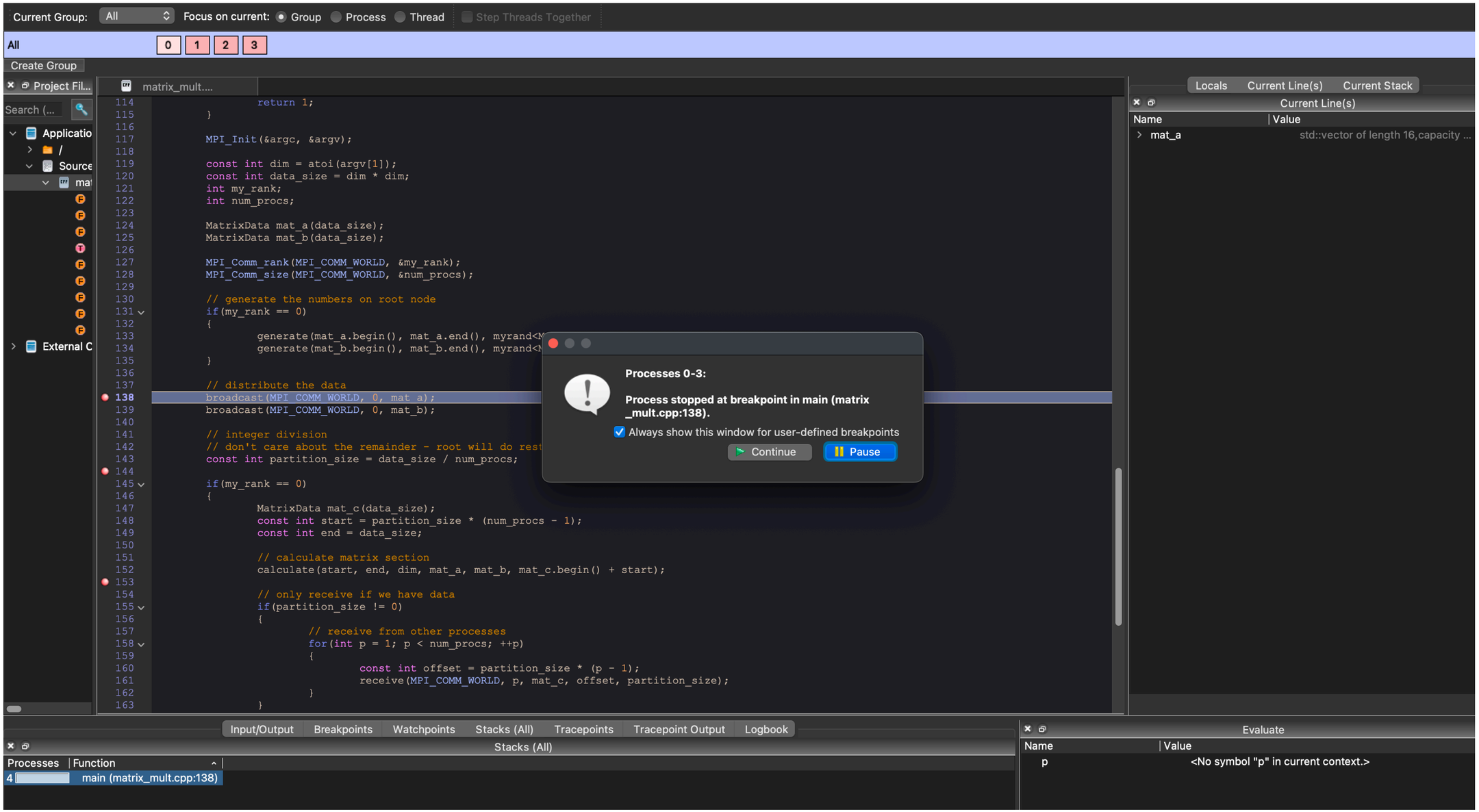

Current Group: All
Focus on current: Group Process Thread
Step Threads Together
0 1 2 3
Create Group
Project Files
matrix_mult....
Application Code
Sources
External Code
Add Breakpoint
Location:
Line
File: ple/matrix_mult.cpp
Line Number: 111
Function
Applies To:
All
Process: All
Thread: All
Hit Limits:
Start on the n-th pass: 0
Trigger every n-th pass: 1
Stop after n hits: Forever
Condition:
Language: Auto
Help
Add
Cancel

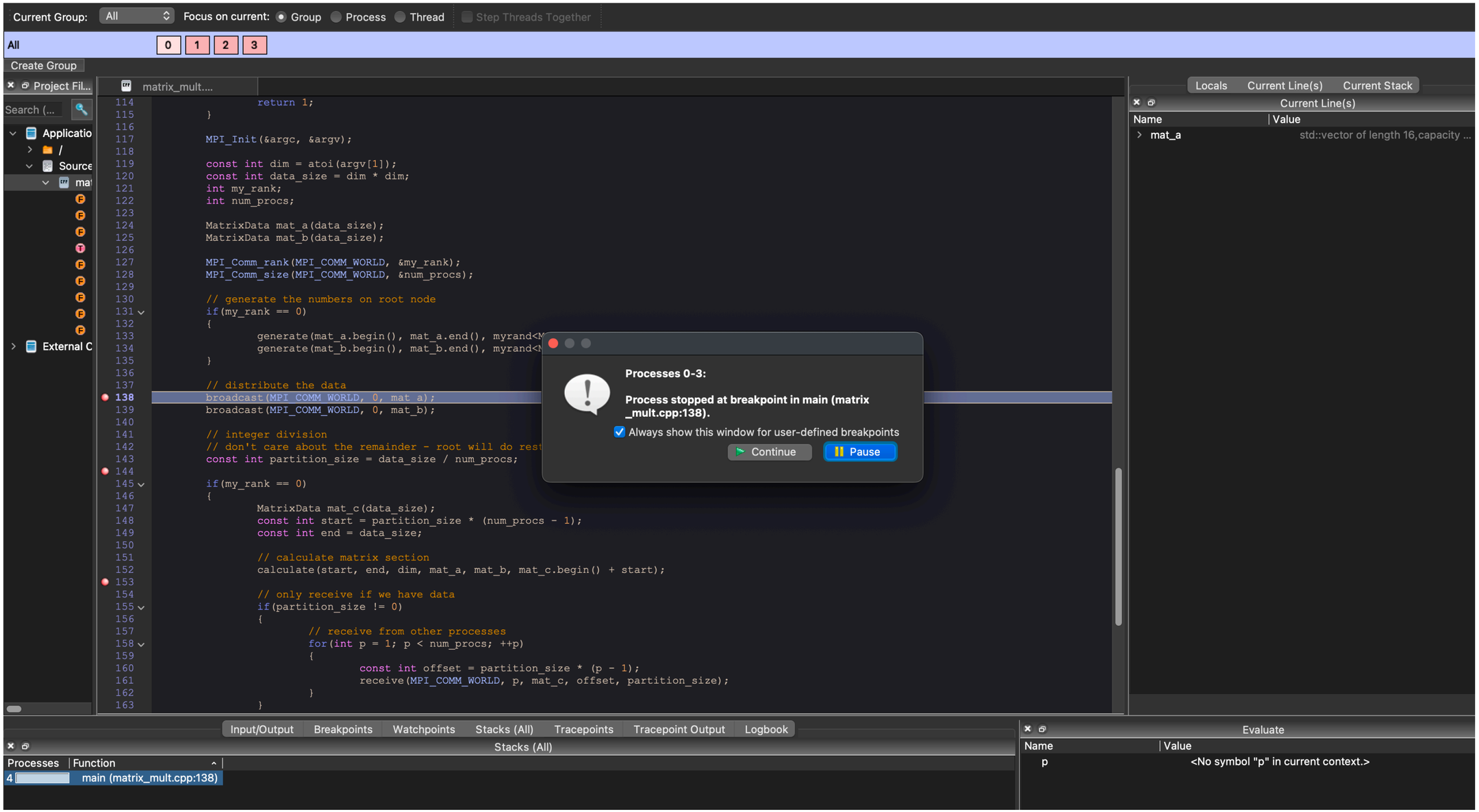

Current Group: All
Focus on current: Group Process Thread
Step Threads Together
All 0 1 2 3
Create Group
Project Fil...
matrix_mult....
Search (...
Applicatio
Source
External C
Locals
Current Line(s)
Current Stack
Name
Value
mat_a
std::vector of length 16,capacity ...
Processes 0-3:
Process stopped at breakpoint in main (matrix _mult.cpp:138).
Always show this window for user-defined breakpoints
Continue
Pause
Input/Output
Breakpoints
Watchpoints
Stacks (All)
Tracepoints
Tracepoint Output
Logbook
Stacks (All)
Processes
Function
4
main (matrix_mult.cpp:138)
Evaluate
Name
Value
p
<No symbol "p" in current context.>

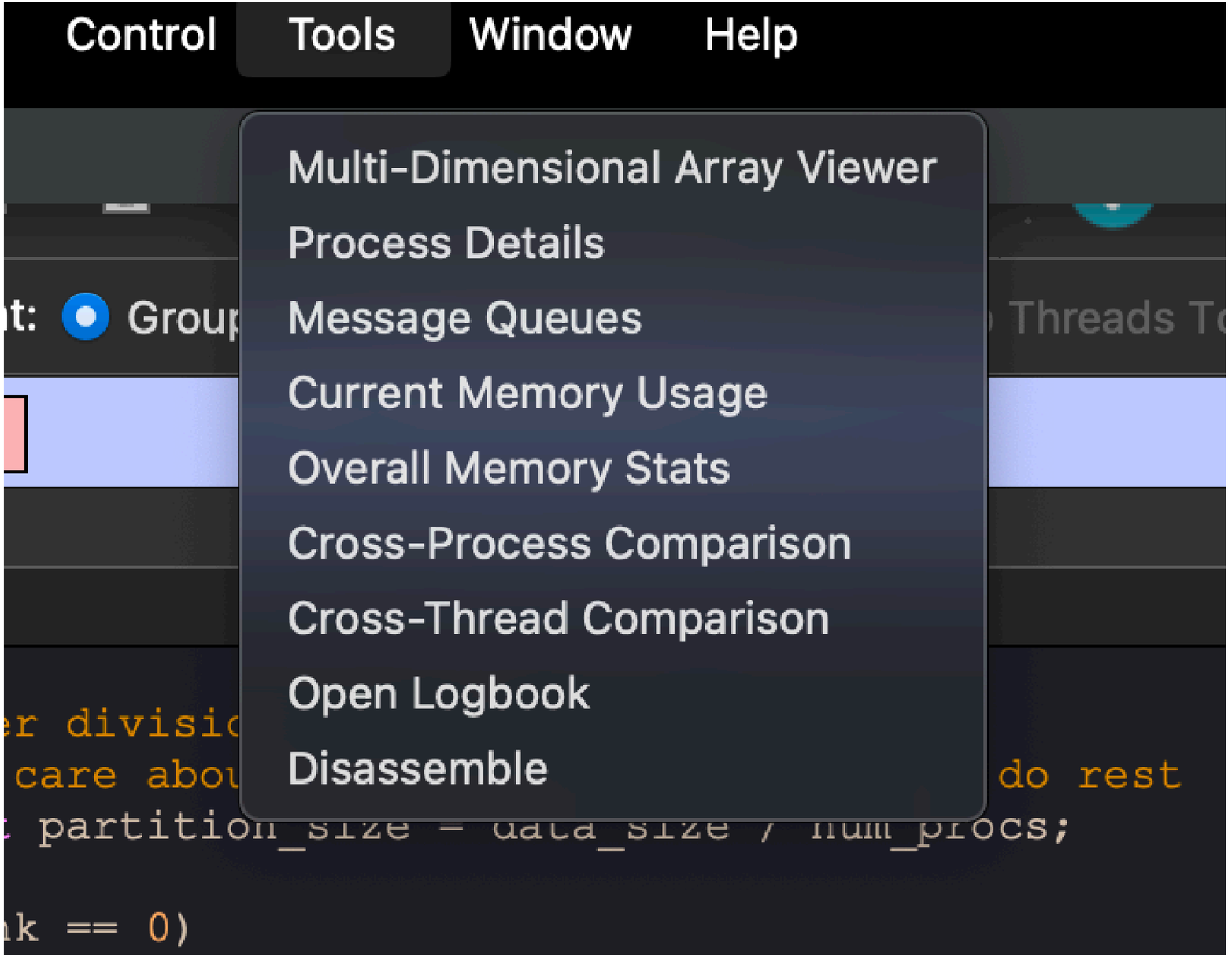


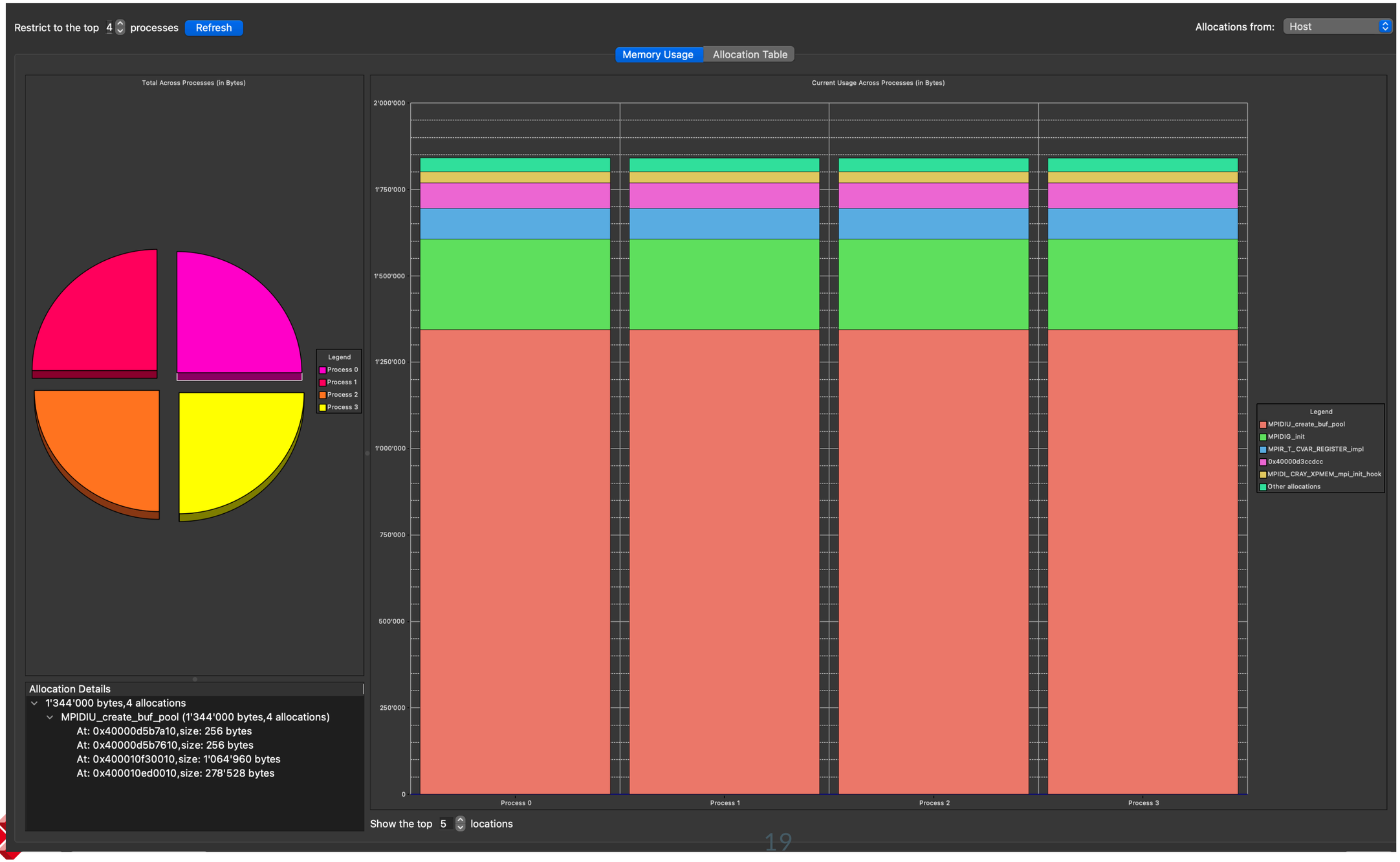

```
121         int my_rank;
122         int num_procs;
123
124         MatrixData mat_a(data_size);
125         MatrixData mat_b(data_size);
126
127         MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
128         MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
129
130         // generate the numbers on root node
131         if(my_rank == 0)
132         {
133                 generate(mat_a.begin(), mat_a.end(), myrand<MatrixData::value_type>);
134                 generate(mat_b.begin(), mat_b.end(), myrand<MatrixData::value_type>);
135         }
136
137         // distribute the data
138         broadcast(MPI_COMM_WORLD, 0, mat_a);
139         broadcast(MPI_COMM_WORLD, 0, mat_b);
140
141         // integer division
142         // don't care about the remainder - root will do rest
143         const int partition_size = data_size / num_procs;
144
145         if(my_rank == 0)
146         {
147                 MatrixData mat_c(data_size);
148                 const int start = partition_size * (num_procs - 1);
149                 const int end = data_size;
150
151                 // calculate matrix section
152                 calculate(start, end, dim, mat_a, mat_b, mat_c.begin() + start);
153
154                 // only receive if we have data
155                 if(partition_size != 0)
156                 {
157                         // receive from other processes
158                         for(int p = 1; p < num_procs; ++p)
159                         {
160                                 const int offset = partition_size * (p - 1);
161                                 receive(MPI_COMM_WORLD, p, mat_c, offset, partition_size);
162                         }
163                 }
164
165                 print_matrix("Matrix A", mat_a, dim);
166                 print_matrix("Matrix B", mat_b, dim);
167                 print_matrix("Matrix C", mat_c, dim);
168         }
169         else
170         {
```


const int end = data_size;
// calculate matrix section
calculate(start, end, dim, mat_a, mat_b, mat_c.begin() + start);
// only receive if we have data
if(partition_size != 0)
{
// receive from other processes
for(int p = 1; p < num_procs; ++p)
{
const int offset = partition_size * (p - 1);
receive(MPI_COMM_WORLD, p, mat
}
}
print_matrix("Matrix A", mat_a, dim);
print_matrix("Matrix B", mat_b, dim);
print_matrix("Matrix C", mat_c, dim);
}
else
{
// check we have values to calculate
if(partition_size != 0)
{
MatrixData mat_c_section(partition_si
const int start = partition_size * (my
const int end = start + partition_size
// calulate matrix section
calculate(start, end, dim, mat_a, mat_
mat_c
Add breakpoint for All
Add tracepoint for All (MPI_COMM_WORLD, p, mat_c, offset, partition_size)
Run to here
Undo
Redo
Cut
Copy
Paste
Delete
Select All
Open in external editor
Close
Add to Evaluations
Add Watchpoint
View Array
Compare Across Processes
Compare Across Threads
Find In Files...
Type is: MatrixData
Value is: std::vector of length 16, capacity 16
Input/Output
Breakpoints
Watchpoints
Stacks (All)
Tracepoint Output
Processes
Values logged
Evaluate
Name
Value
p
<No symbol "p" in current


Current Group: All
Focus on current: Group Process Thread
Step Threads Together
All 0 1 2 3
Create Group
Project Fil...
matrix_mult....
Search (...
Applicatio
Source
External C
Locals Current Line(s) Current Stack
Current Line(s)
Name Value
mat_a std::vector of length 16,capacity ...
Processes 0-3:
Process stopped at breakpoint in main (matrix _mult.cpp:138).
Always show this window for user-defined breakpoints
Continue
Pause
Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook
Stacks (All)
Processes Function
4 main (matrix_mult.cpp:138)
Evaluate
Name Value
p <No symbol "p" in current context.>


Current Group: All
Focus on current: Group Process Thread
Step Threads Together
All 0 1 2 3
Create Group
Project Fil...
matrix_mult....
Search (...
Applicatio
Source
External C
Locals
Current Line(s)
Current Stack
Current Line(s)
Name
Value
mat_a
std::vector of length 16,capacity ...
Processes 0-3:
Process stopped at breakpoint in main (matrix _mult.cpp:138).
Always show this window for user-defined breakpoints
Continue
Pause
Input/Output
Breakpoints
Watchpoints
Stacks (All)
Tracepoints
Tracepoint Output
Logbook
Stacks (All)
Processes
Function
4
main (matrix_mult.cpp:138)
Evaluate
Name
Value
p
<No symbol "p" in current context.>


Control
Tools
Window
Help
Multi-Dimensional Array Viewer
Process Details
Message Queues
Current Memory Usage
Overall Memory Stats
Cross-Process Comparison
Cross-Thread Comparison
Open Logbook
Disassemble


Restrict to the top 4 processes
Refresh
Allocations from: Host
Memory Usage
Allocation Table
Total Across Processes (in Bytes)
Legend
Process 0
Process 1
Process 2
Process 3
Current Usage Across Processes (in Bytes)
2'000'000
1'750'000
1'500'000
1'250'000
1'000'000
750'000
500'000
250'000
0
Process 0
Process 1
Process 2
Process 3
Legend
MPIDIU_create_buf_pool
MPIDIG_init
MPIR_T_CVAR_REGISTER_impl
0x40000d3ccdcc
MPIDI_CRAY_XPMEM_mpi_init_hook
Other allocations
Allocation Details
1'344'000 bytes,4 allocations
MPIDIU_create_buf_pool (1'344'000 bytes,4 allocations)
At: 0x40000d5b7a10,size: 256 bytes
At: 0x40000d5b7610,size: 256 bytes
At: 0x400010f30010,size: 1'064'960 bytes
At: 0x400010ed0010,size: 278'528 bytes
Show the top 5 locations

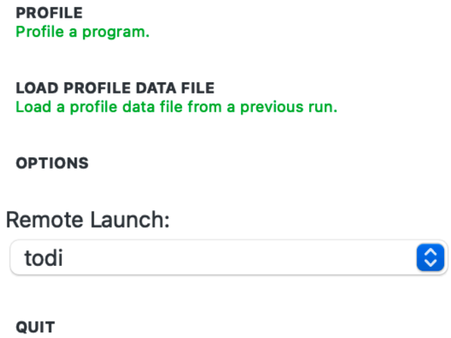

Graph View
Table View
Total Bytes
Total Calls
Current Bytes
Total bytes allocated/freed
2'000'000
1'750'000
1'500'000
1'250'000
1'000'000
750'000
500'000
250'000
0
Mean (All)
Mean (All)
Process 3
Process 3
Process 1
Process 1
Process 0
Process 0
Process 2
Process 2
Legend
allocation calls
deallocation calls

MAP

linaro forge

linaro DDT

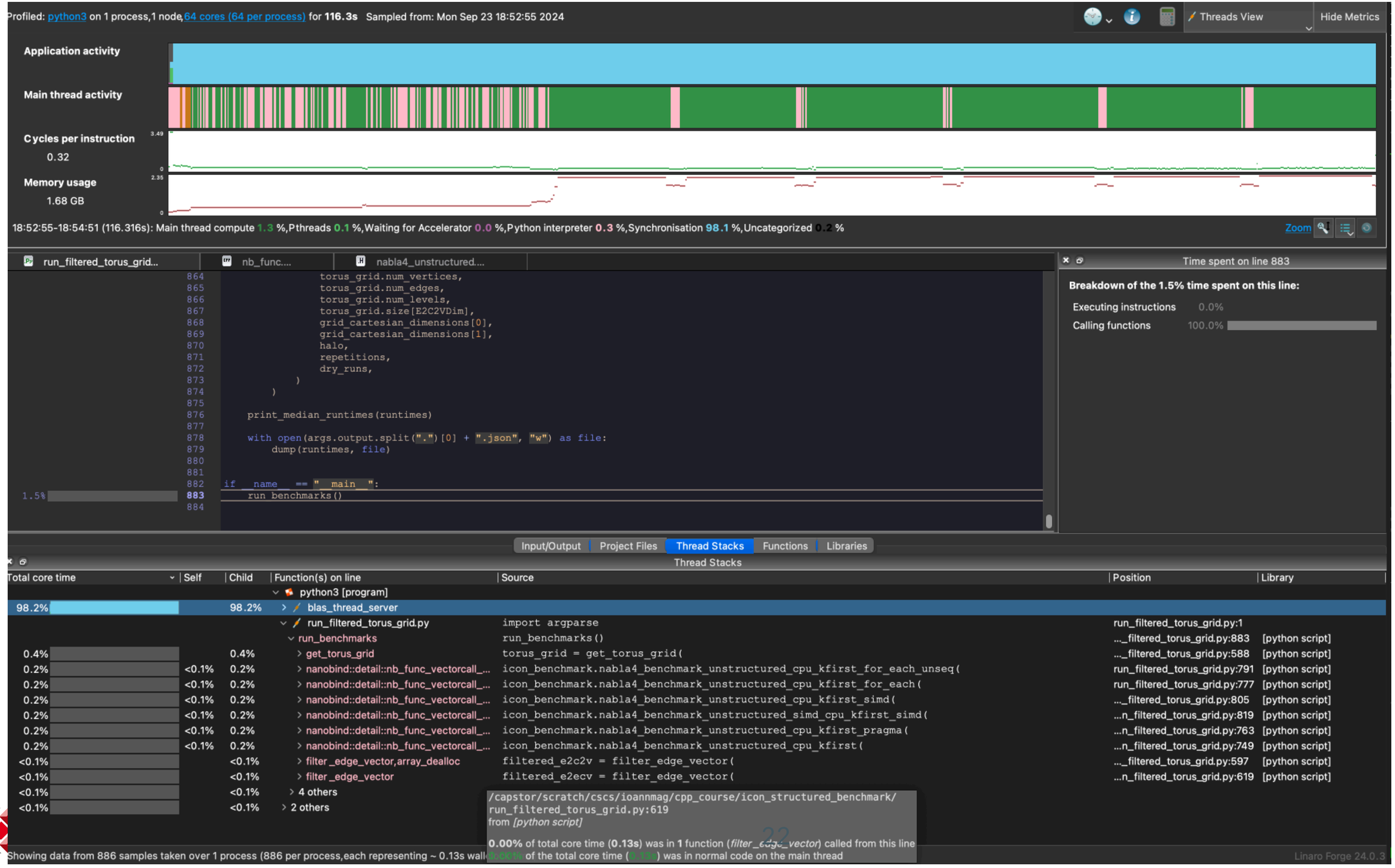
linaro MAP



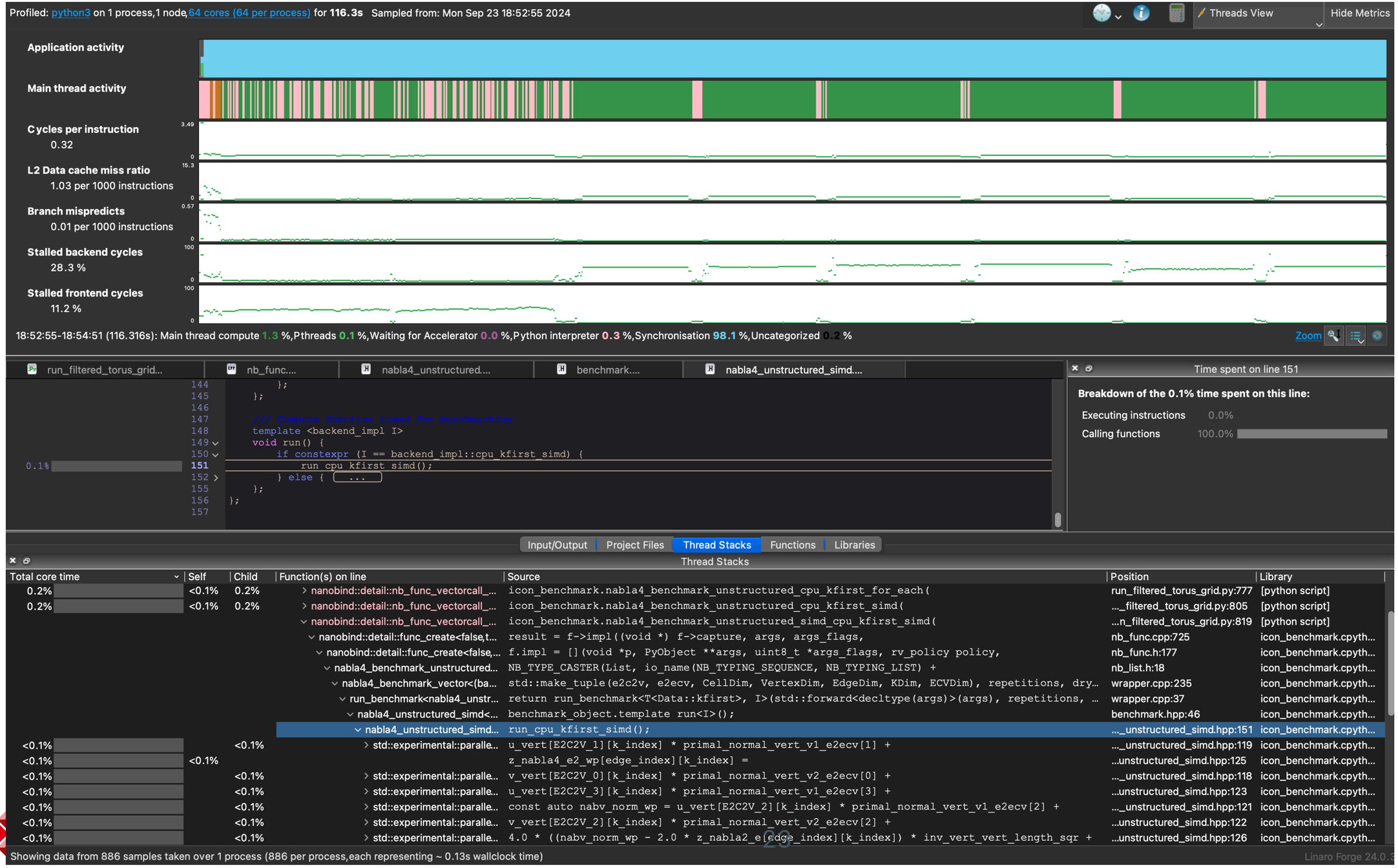
Get trial licence
Support
linaroforge.com

Licence Serial: 17741 ?

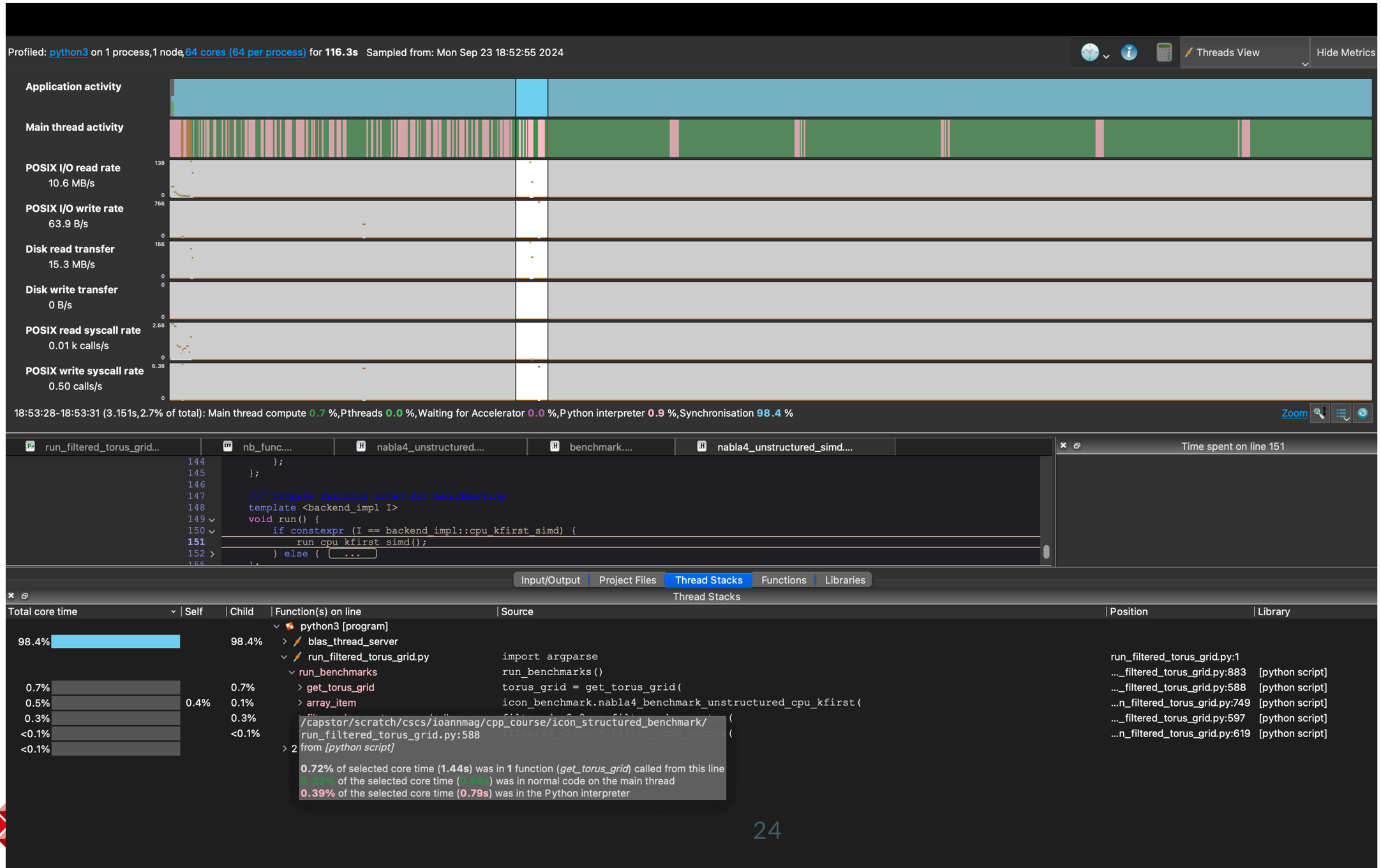
Linaro Forge 24.0.3

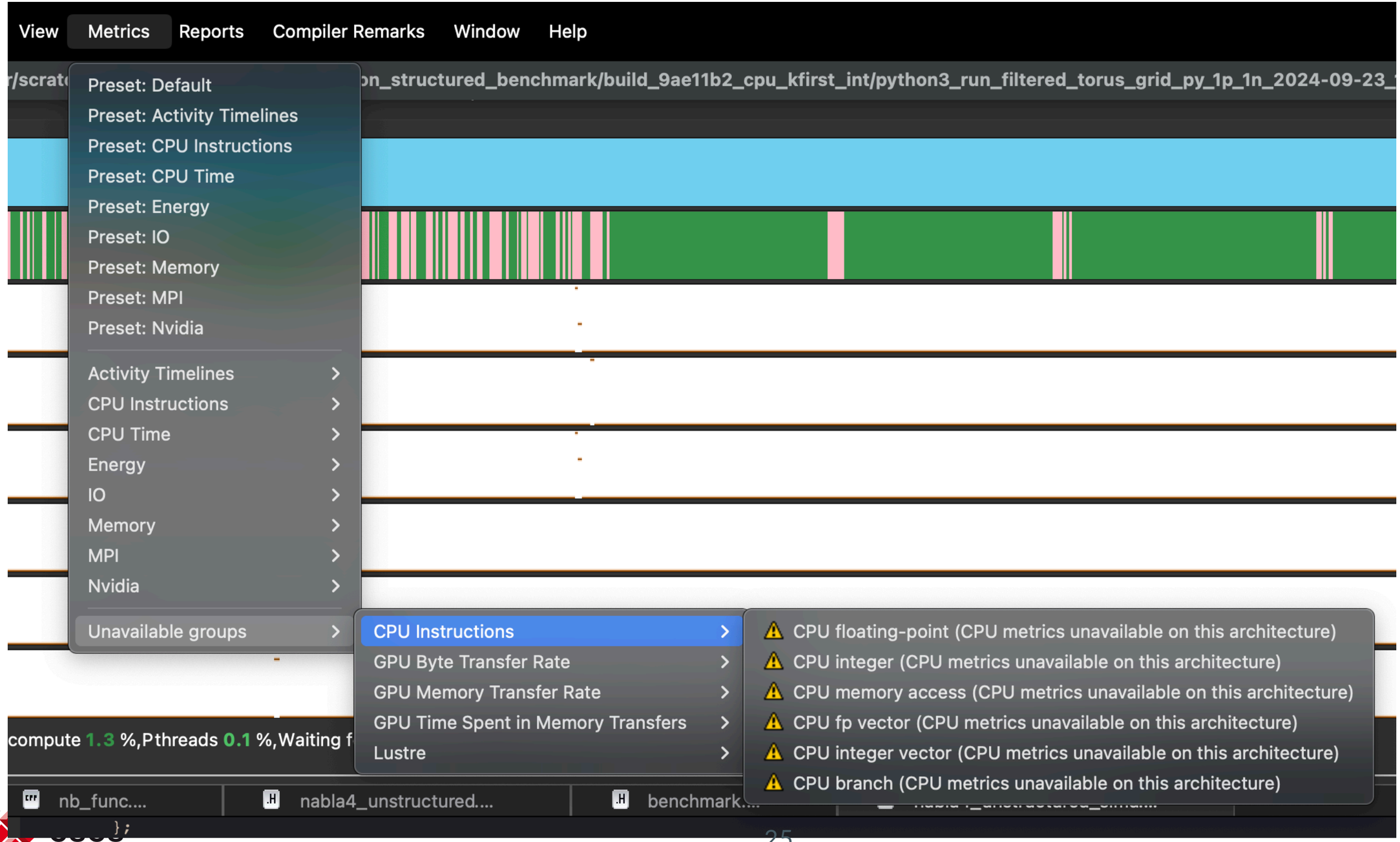

Profiled: python3 on 1 process,1 node,64 cores (64 per process) for 116.3s Sampled from: Mon Sep 23 18:52:55 2024
Threads View
Hide Metrics
Application activity
Main thread activity
Cycles per instruction
0.32
Memory usage
1.68 GB
18:52:55-18:54:51 (116.316s): Main thread compute 1.3 %,Pthreads 0.1 %,Waiting for Accelerator 0.0 %,Python interpreter 0.3 %,Synchronisation 98.1 %,Uncategorized 0.2 %
Zoom
run_filtered_torus_grid...
nb_func....
nabla4_unstructured....
torus_grid.num_vertices,
torus_grid.num_edges,
torus_grid.num_levels,
torus_grid.size[E2C2VDim],
grid_cartesian_dimensions[0],
grid_cartesian_dimensions[1],
halo,
repetitions,
dry_runs,
print_median_runtimes(runtimes)
with open(args.output.split(".")[0] + ".json", "w") as file:
dump(runtimes, file)
if __name__ == "__main__":
run_benchmarks()
1.5%
Time spent on line 883
Breakdown of the 1.5% time spent on this line:
Executing instructions 0.0%
Calling functions 100.0%
Input/Output
Project Files
Thread Stacks
Functions
Libraries
Thread Stacks
Total core time
Self
Child
Function(s) on line
Source
Position
Library
python3 [program]
98.2% 98.2% blas_thread_server
run_filtered_torus_grid.py import argparse run_filtered_torus_grid.py:1
run_benchmarks run_benchmarks() ..._filtered_torus_grid.py:883 [python script]
0.4% 0.4% get_torus_grid torus_grid = get_torus_grid(..._filtered_torus_grid.py:588 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst_for_each_unseq(run_filtered_torus_grid.py:791 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst_for_each(run_filtered_torus_grid.py:777 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst_simd(..._filtered_torus_grid.py:805 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_simd_cpu_kfirst_simd(...n_filtered_torus_grid.py:819 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst_pragma(...n_filtered_torus_grid.py:763 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst(...n_filtered_torus_grid.py:749 [python script]
<0.1% <0.1% filter _edge_vector,array_dealloc filtered_e2c2v = filter_edge_vector(..._filtered_torus_grid.py:597 [python script]
<0.1% <0.1% filter _edge_vector filtered_e2ecv = filter_edge_vector(...n_filtered_torus_grid.py:619 [python script]
<0.1% <0.1% 4 others
<0.1% <0.1% 2 others
/capstor/scratch/cscs/ioannmag/cpp_course/icon_structured_benchmark/
run_filtered_torus_grid.py:619
from [python script]
0.00% of total core time (0.13s) was in 1 function (filter_edge_vector) called from this line
0.00% of the total core time (0.13s) was in normal code on the main thread
Showing data from 886 samples taken over 1 process (886 per process,each representing ~ 0.13s wall
Linaro Forge 24.0.3

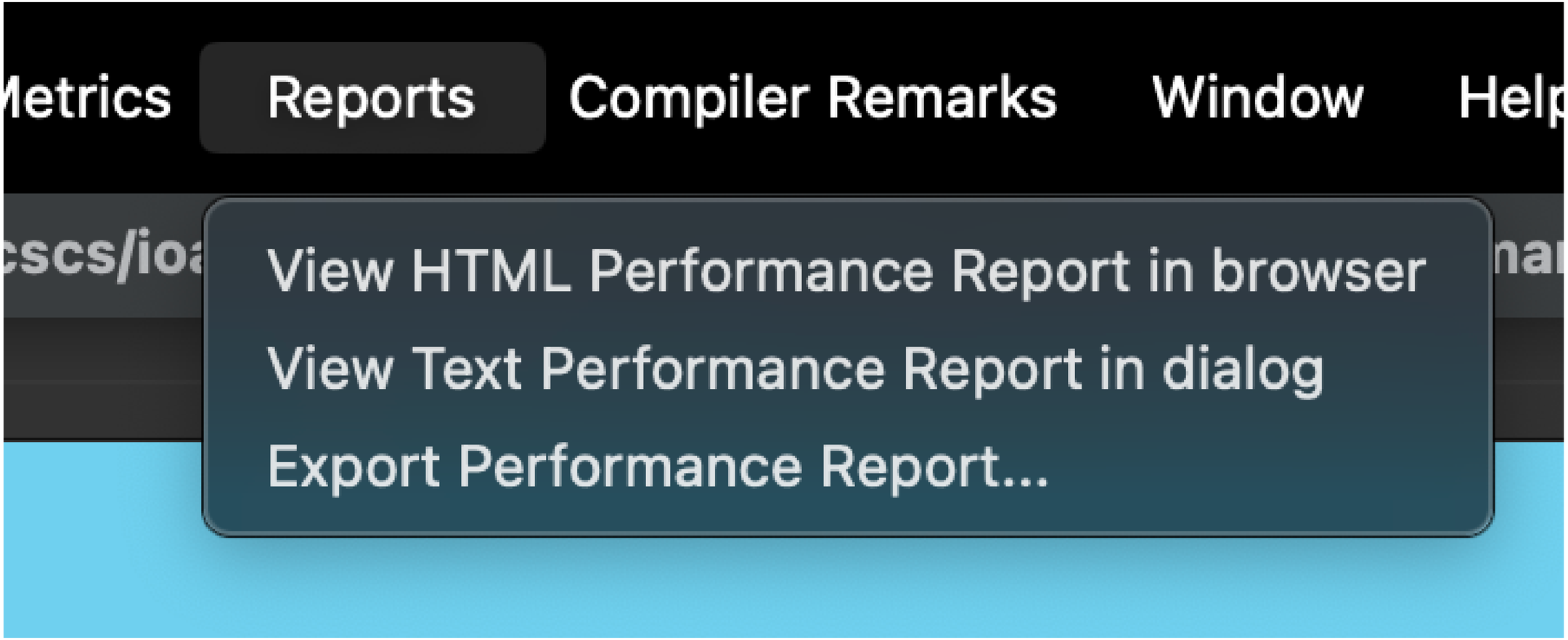
ETH zürich


Profiled: python3 on 1 process,1 node,64 cores (64 per process) for 116.3s Sampled from: Mon Sep 23 18:52:55 2024
Threads View
Hide Metrics
Application activity
Main thread activity
Cycles per instruction
0.32
L2 Data cache miss ratio
1.03 per 1000 instructions
Branch mispredicts
0.01 per 1000 instructions
Stalled backend cycles
28.3 %
Stalled frontend cycles
11.2 %
18:52:55-18:54:51 (116.316s): Main thread compute 1.3 %,Pthreads 0.1 %,Waiting for Accelerator 0.0 %,Python interpreter 0.3 %,Synchronisation 98.1 %,Uncategorized 0.2 %
Zoom
run_filtered_torus_grid...
nb_func....
nabla4_unstructured....
benchmark....
nabla4_unstructured_simd....
144 };
145 };
146
147 /// Compute function timed for benchmarking
148 template <backend_impl I>
149 void run() {
150 if constexpr (I == backend_impl::cpu_kfirst_simd) {
0.1% 151 run_cpu_kfirst_simd();
152 } else { ...
155 };
156 };
157
Time spent on line 151
Breakdown of the 0.1% time spent on this line:
Executing instructions 0.0%
Calling functions 100.0%
Input/Output | Project Files | Thread Stacks | Functions | Libraries
Thread Stacks
Total core time | Self | Child | Function(s) on line | Source | Position | Library
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst_for_each(run_filtered_torus_grid.py:777 [python script]
0.2% <0.1% 0.2% nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst_simd(..._filtered_torus_grid.py:805 [python script]
nanobind::detail::nb_func_vectorcall_... icon_benchmark.nabla4_benchmark_unstructured_simd_cpu_kfirst_simd(...n_filtered_torus_grid.py:819 [python script]
nanobind::detail::func_create<false,t... result = f->impl((void *) f->capture, args, args_flags, nb_func.cpp:725 icon_benchmark.cpyth...
nanobind::detail::func_create<false,... f.impl = [](void *p, PyObject **args, uint8_t *args_flags, rv_policy policy, nb_func.h:177 icon_benchmark.cpyth...
nabla4_benchmark_unstructured... NB_TYPE_CASTER(List, io_name(NB_TYPING_SEQUENCE, NB_TYPING_LIST) + nb_list.h:18 icon_benchmark.cpyth...
nabla4_benchmark_vector<(ba... std::make_tuple(e2c2v, e2ecv, CellDim, VertexDim, EdgeDim, KDim, ECVDim), repetitions, dry... wrapper.cpp:235 icon_benchmark.cpyth...
run_benchmark<nabla4_unstr... return run_benchmark<T<Data::kfirst>, I>(std::forward<decltype(args)>(args), repetitions, ... wrapper.cpp:37 icon_benchmark.cpyth...
nabla4_unstructured_simd<... benchmark_object.template run<I>(); benchmark.hpp:46 icon_benchmark.cpyth...
nabla4_unstructured_simd... run_cpu_kfirst_simd(); ..._unstructured_simd.hpp:151 icon_benchmark.cpyth...
<0.1% <0.1% std::experimental::paralle... u_vert[E2C2V_1][k_index] * primal_normal_vert_v1_e2ecv[1] + ..._unstructured_simd.hpp:119 icon_benchmark.cpyth...
<0.1% <0.1% z_nabla4_e2_wp[edge_index][k_index] = ...unstructured_simd.hpp:125 icon_benchmark.cpyth...
<0.1% <0.1% std::experimental::paralle... v_vert[E2C2V_0][k_index] * primal_normal_vert_v2_e2ecv[0] + ..._unstructured_simd.hpp:118 icon_benchmark.cpyth...
<0.1% <0.1% std::experimental::paralle... u_vert[E2C2V_3][k_index] * primal_normal_vert_v1_e2ecv[3] + ...unstructured_simd.hpp:123 icon_benchmark.cpyth...
<0.1% <0.1% std::experimental::paralle... const auto nabv_norm_wp = u_vert[E2C2V_2][k_index] * primal_normal_vert_v1_e2ecv[2] + ..._unstructured_simd.hpp:121 icon_benchmark.cpyth...
<0.1% <0.1% std::experimental::paralle... v_vert[E2C2V_2][k_index] * primal_normal_vert_v2_e2ecv[2] + ...unstructured_simd.hpp:122 icon_benchmark.cpyth...
<0.1% <0.1% std::experimental::paralle... 4.0 * ((nabv_norm_wp - 2.0 * z_nabla2_e[edge_index][k_index]) * inv_vert_vert_length_sqr + ...unstructured_simd.hpp:126 icon_benchmark.cpyth...
Showing data from 886 samples taken over 1 process (886 per process,each representing ~ 0.13s wallclock time)
Linaro Forge 24.0.3

ETH zürich

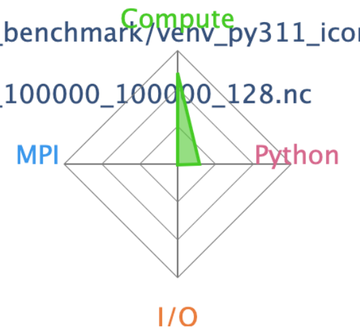

Profiled: python3 on 1 process,1 node,64 cores (64 per process) for 116.3s Sampled from: Mon Sep 23 18:52:55 2024
Threads View
Hide Metrics
Application activity
Main thread activity
POSIX I/O read rate
10.6 MB/s
POSIX I/O write rate
63.9 B/s
Disk read transfer
15.3 MB/s
Disk write transfer
0 B/s
POSIX read syscall rate
0.01 k calls/s
POSIX write syscall rate
0.50 calls/s
18:53:28-18:53:31 (3.151s,2.7% of total): Main thread compute 0.7 %,Pthreads 0.0 %,Waiting for Accelerator 0.0 %,Python interpreter 0.9 %,Synchronisation 98.4 %
Zoom
run_filtered_torus_grid...
nb_func....
nabla4_unstructured....
benchmark....
nabla4_unstructured_simd....
Time spent on line 151
144 };
145 };
146
147 /// Compute function timed for benchmarking
148 template <backend_impl I>
149 void run() {
150 if constexpr (I == backend_impl::cpu_kfirst_simd) {
151 run_cpu_kfirst_simd();
152 } else { ...
Input/Output
Project Files
Thread Stacks
Functions
Libraries
Thread Stacks
Total core time
Self
Child
Function(s) on line
Source
Position
Library
python3 [program]
98.4% 98.4% blas_thread_server
run_filtered_torus_grid.py import argparse run_filtered_torus_grid.py:1
run_benchmarks run_benchmarks() ..._filtered_torus_grid.py:883 [python script]
0.7% 0.7% get_torus_grid torus_grid = get_torus_grid(..._filtered_torus_grid.py:588 [python script]
0.5% 0.4% 0.1% array_item icon_benchmark.nabla4_benchmark_unstructured_cpu_kfirst(...n_filtered_torus_grid.py:749 [python script]
0.3% 0.3% ..._filtered_torus_grid.py:597 [python script]
<0.1% <0.1% ...n_filtered_torus_grid.py:619 [python script]
<0.1% 2
/capstor/scratch/cscs/ioannmag/cpp_course/icon_structured_benchmark/
run_filtered_torus_grid.py:588
from [python script]
0.72% of selected core time (1.44s) was in 1 function (get_torus_grid) called from this line
0.33% of the selected core time (0.66s) was in normal code on the main thread
0.39% of the selected core time (0.79s) was in the Python interpreter


View
Metrics
Reports
Compiler Remarks
Window
Help
Preset: Default
Preset: Activity Timelines
Preset: CPU Instructions
Preset: CPU Time
Preset: Energy
Preset: IO
Preset: Memory
Preset: MPI
Preset: Nvidia
Activity Timelines
CPU Instructions
CPU Time
Energy
IO
Memory
MPI
Nvidia
Unavailable groups
CPU Instructions
GPU Byte Transfer Rate
GPU Memory Transfer Rate
GPU Time Spent in Memory Transfers
Lustre
CPU floating-point (CPU metrics unavailable on this architecture)
CPU integer (CPU metrics unavailable on this architecture)
CPU memory access (CPU metrics unavailable on this architecture)
CPU fp vector (CPU metrics unavailable on this architecture)
CPU integer vector (CPU metrics unavailable on this architecture)
CPU branch (CPU metrics unavailable on this architecture)
compute 1.3 %,Pthreads 0.1 %,Waiting f
nb_func....
nabla4_unstructured....
benchmark....


Metrics
Reports
Compiler Remarks
Window
Help
View HTML Performance Report in browser
View Text Performance Report in dialog
Export Performance Report...

linaro PERFORMANCE REPORTS

Command:	/capstor/scratch/cscs/ioannmag/cpp_course/icon_structured_benchmark/venv_py311_icon4py/bin/python3 ../run_filtered_torus_grid.py /capstor/scratch/cscs/ioannmag/torus_files_strassman/torus_100000_100000_128.nc --backend cpu_kfirst --dry-run
Resources:	1 node (288 physical, 288 logical cores per node) 4 GPUs per node available
Memory:	857 GiB per node
Tasks:	1 process
Machine:	nid005037
Architecture:	aarch64
CPU Family:	neoverse-v2
Start time:	Mon Sep 23 18:52:55 2024
Total time:	117 seconds (about 2 minutes)
Full path:	.



Summary: run_filtered_torus_grid.py is Compute-bound in this configuration

Compute	80.0%	93.1s	Time spent running application code. High values are usually good. This is **high**; check the CPU and accelerators sections for advice
MPI	0.0%	0.0s	Time spent in MPI calls. High values are usually bad. This is **very low**; this code may benefit from a higher process count
I/O	0.7%	0.8s	Time spent in filesystem I/O. High values are usually bad. This is **very low**; however single-process I/O may cause MPI wait times
Python Interpreter	19.3%	22.4s	Time spent in the Python interpreter. Calling out to precompiled libraries may be more efficient. This is **very low**; focus on improving MPI or I/O performance first

This application run was Compute-bound (based on main thread activity). A breakdown of this time and advice for investigating further is in the CPU Metrics and Accelerators sections below.

ETH zürich

CPU Metrics

Linux perf event metrics:

Cycles per instruction	0.32
L2D cache miss ratio	1.03
Stalled backend cycles	28.3%
Stalled frontend cycles	11.2%

Cycles per instruction is low, which is good. Vectorization allows multiple instructions per clock cycle.

MPI

A breakdown of the 0.0% (0.0s) MPI time:

Time in collective calls	0.0%	0.0s
Time in point-to-point calls	0.0%	0.0s
Effective process collective rate	0.00 bytes/s	
Effective process point-to-point rate	0.00 bytes/s	

No time is spent in MPI operations. There's nothing to optimize here!

I/O

A breakdown of the 0.7% (0.8s) I/O time:

Time in reads	50.0%	0.4s
Time in writes	50.0%	0.4s
Effective process read rate	256 MB/s	
Effective process write rate	22.3 kB/s	

Most of the time is spent in write operations with a very low effective transfer rate. This may be caused by contention for the filesystem or inefficient access patterns. Use an I/O profiler to investigate which write calls are affected.

Threads

A breakdown of how multiple threads were used:

Computation	<0.1%	9.6s
Synchronization	99.9%	11622.0s
Physical core utilization	22.2%	
System load	0.5%	

Significant time is spent synchronizing threads. Check which locks cause the most overhead with a profiler.

This may be a sign of overly fine-grained parallelism or of workload imbalance between threads.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	1.56 GiB
Peak process memory usage	2.19 GiB
Peak node memory usage	2.0%

The peak node memory usage is very low. Larger problem sets

Accelerators

A breakdown of how CUDA accelerators were used:

GPU utilization	0.0%
Mean GPU memory usage	0.8%
Peak GPU memory usage	0.8%

GPUs are available but are not used. Identify suitable hot loops