Introduction to Performance Tools

Samuel Khuvis

Scientific Applications Engineer, OSC

Goals of the Breakout Session

- Let you know what tools are available at OSC
- Suggest when you should use each of them
- Give an overview of usage for each
  - Including a demo or sample output
- Show you where to find more information

For More Information

- Visit the software pages on our website
  www.osc.edu
  Resource → Available Software
- Contact the help desk (OSC Help)
  oschelp@osc.edu
  614-292-1800
  1-800-686-6472
- Optimization and Performance Tuning Workshop on May 21, 2019 at 1-4 pm
  `osc.edu/calendar/events/2019_05_21-optimization_performance_tuning_workshop`

Profiling/Debugging Tools Available at OSC

- Parallel debugging tools
  - ARM DDT
- Profiling tools
  - ARM Performance Reports
  - ARM MAP
  - Intel VTune
  - Intel Trace Analyzer and Collector (ITAC)
  - Intel Advisor
  - TAU Commander
  - HPCToolkit

What can a debugger do for you?

- Debuggers let you
  - execute your program one line at a time ("step")
  - inspect variable values
  - stop your program at a particular line ("breakpoint")
  - open a "core" file (after program crashes)
- HPC debuggers
  - support multithreaded code
  - support MPI code
  - support GPU code
  - provide a nice GUI

Compilation flags for debugging

For debugging:

- Use -g flag
- Remove optimization or set to -O0
- Examples:
  - `icc -g -o mycode mycode.c`
  - `gcc -g -O0 -o mycode mycode.c`
- Use `icc -help diag` to see what compiler warnings and diagnostic options are available for the Intel compiler
- Diagnostic options can also be found by reading the `man` page of `gcc` with `man gcc`

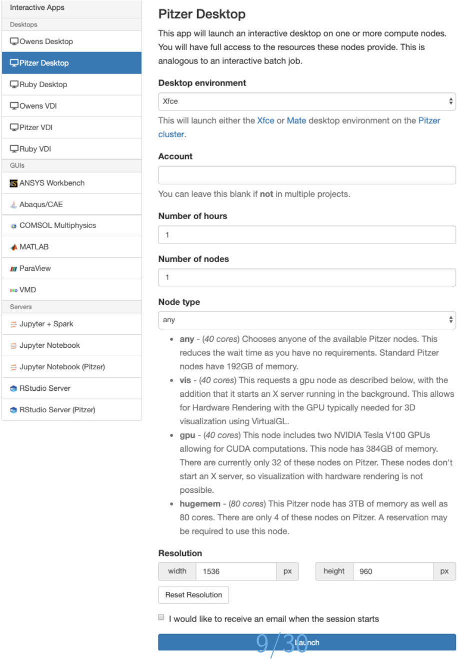
ARM DDT

- Available on all OSC clusters
  - `module load arm-ddt`
- To run a non-MPI program from the command line:
  - `ddt --offline --no-mpi ./mycode [args]`
- To run a MPI program from the command line:
  - `ddt --offline -np num_procs ./mycode [args]`

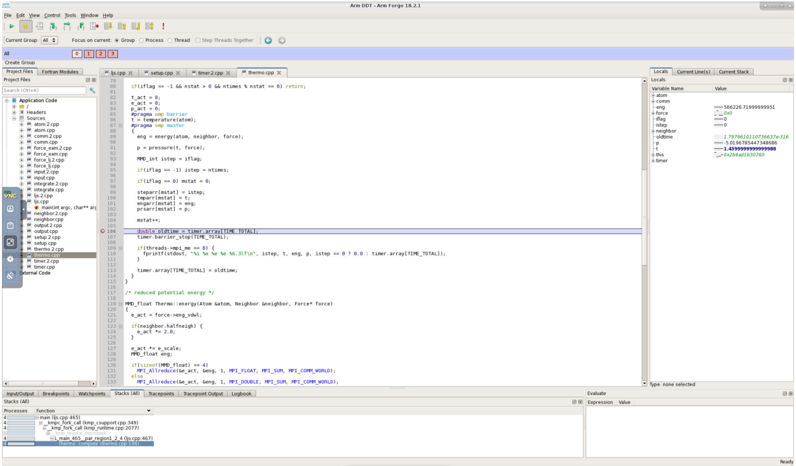
ARM DDT GUI

- To run ARM DDT as a GUI, login to OnDemand at `ondemand.osc.edu`
- To get an interactive session on a compute node, select "Pitzer Desktop" under "Interactive Apps"
- Enter information and click "Launch"
- Click "Launch noVNC in New Tab" to launch the desktop in a new tab
- From there you can open a terminal and run DDT as a GUI
- For a non-MPI program:
  - `ddt --no-mpi ./mycode [args]`
- For a MPI program:
  - `ddt -np num_procs ./mycode [args]`
- More information on using OnDemand is available at `osc.edu/resources/online_portals/ondemand`

OnDemand Screenshot



ARM DDT


Arm DDT - Arm Forge 18.2.1
File Edit View Control Tools Window Help
Current Group: All
Focus on current: Group Process Thread Step Threads Together
All
Create Group
Project Files Fortran Modules
Application Code
Headers
Sources
ljs.cpp setup.cpp timer.2.cpp thermo.cpp
if(iflag == -1 && nstat > 0 && ntimes % nstat == 0) return;
t_act = 0;
e_act = 0;
p_act = 0;
#pragma omp barrier
t = temperature(atom);
#pragma omp master
eng = energy(atom, neighbor, force);
p = pressure(t, force);
MMD_int istep = iflag;
if(iflag == -1) istep = ntimes;
if(iflag == 0) mstat = 0;
steparr[mstat] = istep;
tmparr[mstat] = t;
engarr[mstat] = eng;
prsarr[mstat] = p;
mstat++;
double oldtime = timer.array[TIME_TOTAL];
timer.barrier_stop(TIME_TOTAL);
if(threads->mpi_me == 0) {
fprintf(stdout, "%i %e %e %e %6.3lf\n", istep, t, eng, p, istep == 0 ? 0.0 : timer.array[TIME_TOTAL]);
timer.array[TIME_TOTAL] = oldtime;
/* reduced potential energy */
MMD_float Thermo::energy(Atom &atom, Neighbor &neighbor, Force* force)
e_act = force->eng_vdwl;
if(neighbor.halfneigh) {
e_act *= 2.0;
e_act *= e_scale;
MMD_float eng;
if(sizeof(MMD_float) == 4)
MPI_Allreduce(&e_act, &eng, 1, MPI_FLOAT, MPI_SUM, MPI_COMM_WORLD);
else
MPI_Allreduce(&e_act, &eng, 1, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
Locals Current Line(s) Current Stack
Variable Name Value
atom
comm
eng 566226.71999999951
force 0x0
iflag 0
istep 0
neighbor
oldtime 1.7979610110736637e-316
p -5.0196785447348686
t 1.4399999999999988
this 0x2b8ad1b30760
timer
Type: none selected
Input/Output Breakpoints Watchpoints Stacks (All) Tracepoints Tracepoint Output Logbook
Stacks (All)
Processes Function
main (ljs.cpp:465)
__kmpc_fork_call (kmp_csupport.cpp:349)
__kmp_fork_call (kmp_runtime.cpp:2077)
L_main_465__par_region1_2_4 (ljs.cpp:467)
Thermo::compute (thermo.cpp:106)
Evaluate
Expression Value
Ready

What can a profiler show you?

- Whether code is
  - compute-bound
  - memory-bound
  - communication-bound
- How well the code uses available resources
  - Multiple cores
  - Vectorization
- How much time is spent in different parts of the code

Compilation flags for profiling

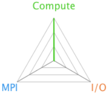
- For profiling
  - Use -g flag
  - Explicitly specify optimization level -On
  - Example: `icc -g -O3 -o mycode mycode.c`
- Use the same level optimization as you normally do
  - Bad example: `icc -g -o mycode mycode.c`
    - Equivalent to `-O0`

ARM Performance Reports

- Easy to use
  - "–g" flag not needed - works on precompiled binaries
- Gives a summary of your code's performance
  - view report with browser
- For a non-MPI program:
  - `module load arm-pr`
  - `perf-report --no-mpi ./mycode [args]`
- For an MPI program:
  - `perf-report -np num_procs ./mycode [args]`

arm PERFORMANCE REPORTS

Command:	/fs/project/PZS0720/skhuvis/SETSM/setsm dataset/WV01_15MAY080613301-P1BS-102001003C02A600.tif dataset/WV01_15MAY080614188-P1BS-102001003EA5DA00.tif out -outres 8 -projection ps
Resources:	1 node (40 physical, 40 logical cores per node)
Tasks:	1 process, OMP_NUM_THREADS was 28
Machine:	p0165.ten.osc.edu
Start time:	Fri Dec 28 2018 14:13:20 (UTC-05)
Total time:	372 seconds (about 6 minutes)
Full path:	/fs/project/PZS0720/skhuvis/SETSM



Summary: setsm is Compute-bound in this configuration

Compute	99.3%	Time spent running application code. High values are usually good. This is **very high**; check the CPU performance section for advice
MPI	0.0%	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%	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

This application run was Compute-bound. A breakdown of this time and advice for investigating further is in the CPU section below.

As very little time is spent in MPI calls, this code may also benefit from running at larger scales.

CPU

A breakdown of the 99.3% CPU time:

Single-core code	44.5%
OpenMP regions	55.5%
Scalar numeric ops	21.8%
Vector numeric ops	4.4%
Memory accesses	43.7%

The per-core performance is memory-bound. Use a profiler to identify time-consuming loops and check their cache performance.

Little time is spent in vectorized instructions. Check the compiler's vectorization advice to see why key loops could not be vectorized.

MPI

A breakdown of the 0.0% MPI time:

Time in collective calls	0.0%
Time in point-to-point calls	0.0%
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% I/O time:

Time in reads	71.4%
Time in writes	28.6%
Effective process read rate	2.88 GB/s
Effective process write rate	3.23 GB/s

Most of the time is spent in read operations with a high effective transfer rate. It may be possible to achieve faster effective transfer rates using asynchronous file operations.

Memory

Per-process memory usage may also affect scaling:

Mean process memory usage	1.16 GiB
Peak process memory usage	3.70 GiB
Peak node memory usage	8.0%

The peak node memory usage is very low. Larger problem sets can be run before scaling to multiple nodes.

OpenMP

A breakdown of the 55.5% time in OpenMP regions:

Computation	78.5%
Synchronization	21.5%
Physical core utilization	70.0%
System load	57.9%

OpenMP thread performance looks good. Check the CPU breakdown for advice on improving code efficiency.

Energy

A breakdown of how the 19.1 Wh was used:

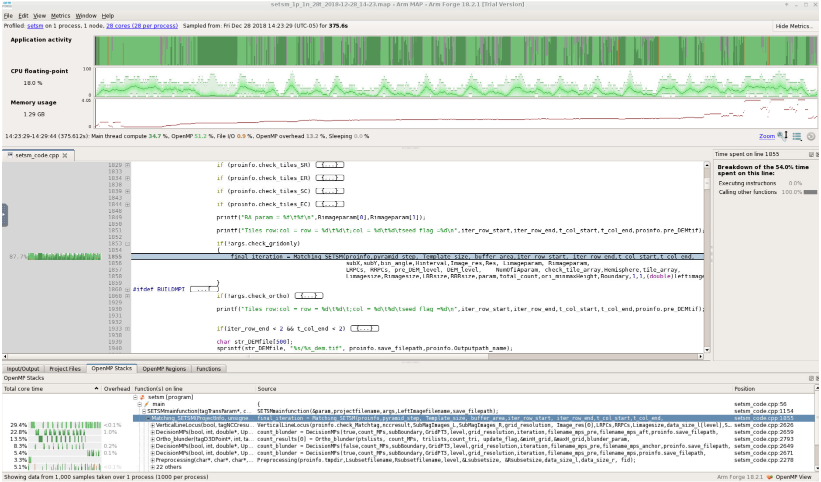
CPU	100.0%
System	not supported %
Mean node power	not supported W
Peak node power	0.00 W

The whole system energy has been calculated using the CPU energy usage.

System power metrics: No Arm IPMI Energy Agent config file found in /var/spool/ipmi-energy-agent. Did you start the Arm IPMI Energy Agent?

ARM MAP

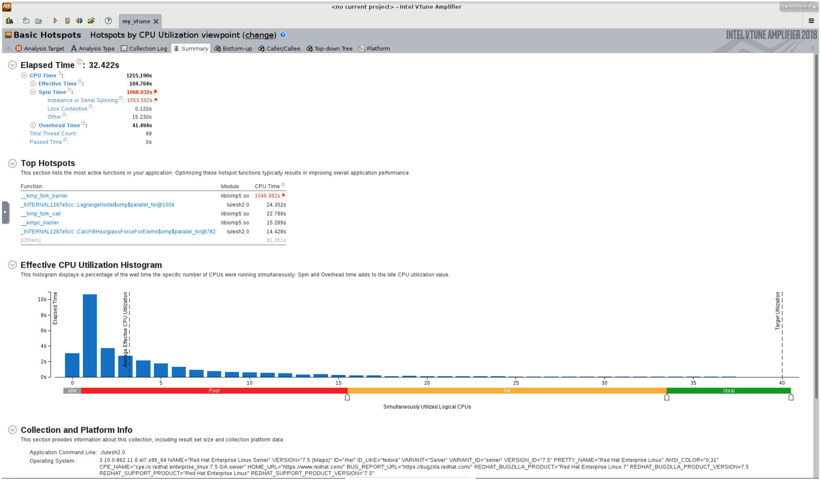
- Interpretation of profile requires some expertise
- Gives details about your code's performance
- For a non-MPI program:
  - `module load arm-map`
  - `map --profile --no-mpi ./mycode [args]`
- For an MPI program:
  - `map --profile -np num_procs ./mycode [args]`
- View and explore resulting profile using ARM client
  - Download remote client to view profiles on local machine at `developer.arm.com/products/software-development-tools/hpc/downloads/download-arm-forge`
  - Information on transferring files to your local machine at `osc.edu/resources/online_portals/ondemand/file_transfer_and_management`


setsm_1p_1n_28t_2018-12-28_14-23.map - Arm MAP - Arm Forge 18.2.1 [Trial Version]
File Edit View Metrics Window Help
Profiled: setsm on 1 process, 1 node, 28 cores (28 per process) Sampled from: Fri Dec 28 2018 14:23:29 (UTC-05) for 375.6s
Hide Metrics...
Application activity
CPU floating-point
18.0 %
Memory usage
1.29 GB
14:23:29-14:29:44 (375.612s): Main thread compute 34.7 %, OpenMP 51.2 %, File I/O 0.9 %, OpenMP overhead 13.2 %, Sleeping 0.0 %
Zoom
setsm_code.cpp
if (proinfo.check_tiles_SR)
if (proinfo.check_tiles_ER)
if (proinfo.check_tiles_SC)
if (proinfo.check_tiles_EC)
printf("RA param = %f\t%f\n",Rimageparam[0],Rimageparam[1]);
printf("Tiles row:col = row = %d\t%d\t;col = %d\t%d\tseed flag =%d\n",iter_row_start,iter_row_end,t_col_start,t_col_end,proinfo.pre_DEMtif);
if(!args.check_gridonly)
final_iteration = Matching_SETSM(proinfo,pyramid_step, Template_size, buffer_area,iter_row_start, iter_row_end,t_col_start,t_col_end,
subX,subY,bin_angle,Hinterval,Image_res,Res, Limageparam, Rimageparam,
LRPCs, RRPCs, pre_DEM_level, DEM_level, NumOfIAparam, check_tile_array,Hemisphere,tile_array,
Limagesize,Rimagesize,LBRsize,RBRsize,param,total_count,ori_minmaxHeight,Boundary,1,1,(double)leftimage
#ifdef BUILDMPI
if(!args.check_ortho)
printf("Tiles row:col = row = %d\t%d\t;col = %d\t%d\tseed flag =%d\n",iter_row_start,iter_row_end,t_col_start,t_col_end,proinfo.pre_DEMtif);
if(iter_row_end < 2 && t_col_end < 2)
char str_DEMfile[500];
sprintf(str_DEMfile, "%s/%s_dem.tif", proinfo.save_filepath,proinfo.Outputpath_name);
87.7%
Time spent on line 1855
Breakdown of the 54.0% time spent on this line:
Executing instructions 0.0%
Calling other functions 100.0%
Input/Output Project Files OpenMP Stacks OpenMP Regions Functions
OpenMP Stacks
Total core time Overhead Function(s) on line Source Position
setsm [program]
main setsm_code.cpp:56
SETSMmainfunction(tagTransParam*, c... SETSMmainfunction(&param,projectfilename,args,LeftImagefilename,save_filepath); setsm_code.cpp:1154
Matching_SETSM(ProjectInfo, unsigne... final_iteration = Matching_SETSM(proinfo,pyramid_step, Template_size, buffer_area,iter_row_start, iter_row_end,t_col_start,t_col_end, setsm_code.cpp:1855
29.4% <0.1% VerticalLineLocus(bool, tagNCCresu... VerticalLineLocus(proinfo.check_Matchtag,nccresult,SubMagImages_L,SubMagImages_R,grid_resolution, Image_res[0],LRPCs,RRPCs,Limagesize,data_size_l[level],S... setsm_code.cpp:2626
22.8% 1.0% DecisionMPs(bool, int, double*, Up... count_blunder = DecisionMPs(true,count_MPs,subBoundary,GridPT3,level,grid_resolution,iteration,filename_mps_pre,filename_mps_aft,proinfo.save_filepath, setsm_code.cpp:2659
13.5% Ortho_blunder(tagD3DPoint*, int, ta... count_results[0] = Ortho_blunder(ptslists, count_MPs, trilists,count_tri, update_flag,&minH_grid,&maxH_grid,blunder_param, setsm_code.cpp:2793
8.3% 0.2% DecisionMPs(bool, int, double*, Up... count_blunder = DecisionMPs(false,count_MPs,subBoundary,GridPT3,level,grid_resolution,iteration,filename_mps_pre,filename_mps_anchor,proinfo.save_filepath, setsm_code.cpp:2649
5.4% 0.1% DecisionMPs(bool, int, double*, Up... count_blunder = DecisionMPs(true,count_MPs,subBoundary,GridPT3,level,grid_resolution,iteration,filename_mps_pre,filename_mps,proinfo.save_filepath, setsm_code.cpp:2671
3.3% Preprocessing(char*, char*, char*,... Preprocessing(proinfo.tmpdir,Lsubsetfilename,Rsubsetfilename,level,&Lsubsetsize, &Rsubsetsize,data_size_l,data_size_r, fid); setsm_code.cpp:2278
5.1% <0.1% 22 others
Showing data from 1,000 samples taken over 1 process (1000 per process)
Arm Forge 18.2.1 OpenMP View

Intel VTune

- A profiler that can work with C, C++, Fortran programs
- Works best on a single node
- For using a GUI (use for small problems $<$ 5 minutes):
  - `amplxe-gui`
- For non-interactive usage:
  - `amplxe-cl -r my_vtune -collect hotspots -no-auto-finalize ./mycode`
  - `amplxe-cl -report hotspots -r my_vtune`
- View and explore existing results with `amplxe-gui`

VTune GUI

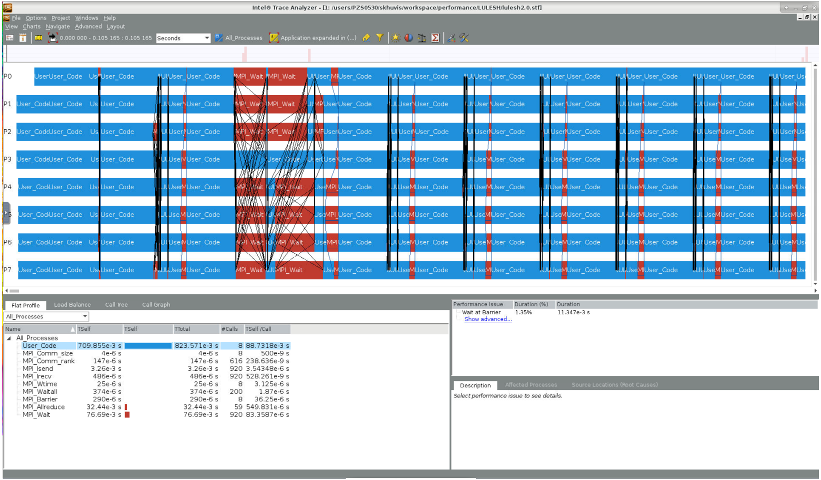


<no current project> - Intel VTune Amplifier
my_vtune
Basic Hotspots Hotspots by CPU Utilization viewpoint (change)
INTEL VTUNE AMPLIFIER 2018
Analysis Target
Analysis Type
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Elapsed Time: 32.422s
CPU Time: 1215.190s
Effective Time: 104.764s
Spin Time: 1068.932s
Imbalance or Serial Spinning: 1053.582s
Lock Contention: 0.120s
Other: 15.230s
Overhead Time: 41.494s
Total Thread Count: 69
Paused Time: 0s
Top Hotspots
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.
Function Module CPU Time
__kmp_fork_barrier libiomp5.so 1046.982s
_INTERNAL1297e5cc::LagrangeNodalompparallel_for@1009 lulesh2.0 24.352s
__kmp_fork_call libiomp5.so 22.788s
__kmpc_barrier libiomp5.so 15.289s
_INTERNAL1297e5cc::CalcFBHourglassForceForElemsompparallel_for@782 lulesh2.0 14.428s
[Others] 91.351s
Effective CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.
Elapsed Time
Average Effective CPU Utilization
Target Utilization
10s
8s
6s
4s
2s
0s
0
5
10
15
20
25
30
35
40
Idle
Poor
Ok
Ideal
Simultaneously Utilized Logical CPUs
Collection and Platform Info
This section provides information about this collection, including result set size and collection platform data.
Application Command Line: ./lulesh2.0
Operating System: 3.10.0-862.11.6.el7.x86_64 NAME="Red Hat Enterprise Linux Server" VERSION="7.5 (Maipo)" ID="rhel" ID_LIKE="fedora" VARIANT="Server" VARIANT_ID="server" VERSION_ID="7.5" PRETTY_NAME="Red Hat Enterprise Linux" ANSI_COLOR="0;31" CPE_NAME="cpe:/o:redhat:enterprise_linux:7.5:GA:server" HOME_URL="https://www.redhat.com/" BUG_REPORT_URL="https://bugzilla.redhat.com/" REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 7" REDHAT_BUGZILLA_PRODUCT_VERSION=7.5 REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux" REDHAT_SUPPORT_PRODUCT_VERSION="7.5"

Intel Trace Analyzer and Collector (ITAC)

- Graphical tool for profiling MPI code (Intel MPI)
- To use:
  - `module load intelmpi # then compile (-g) code`
  - `mpiexec -trace ./mycode`
- View and explore existing results using GUI with `traceanalyzer`:
  - `traceanalyzer <mycode>.stf`

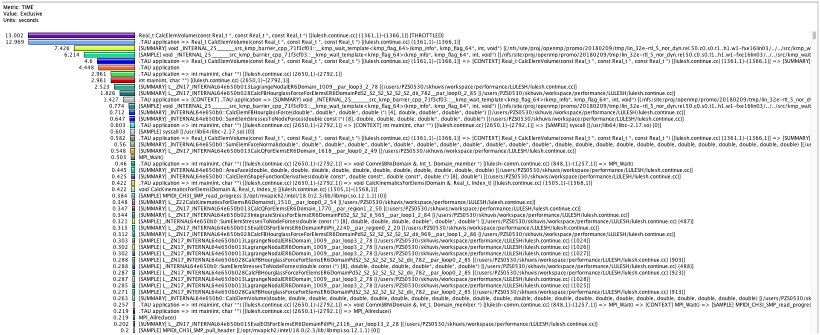
ITAC GUI


Intel® Trace Analyzer - [1: /users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh2.0.stf]
File Options Project Windows Help
View Charts Navigate Advanced Layout
0.000 000 - 0.105 165 : 0.105 165
Seconds
All_Processes
Application expanded in (...)
P0
P1
P2
P3
P4
P6
P7
User_Code
MPI_Wait
Flat Profile
Load Balance
Call Tree
Call Graph
All_Processes
Name
TSelf
TSelf
TTotal
#Calls
TSelf /Call
All_Processes
User_Code 709.855e-3 s 823.571e-3 s 8 88.7318e-3 s
MPI_Comm_size 4e-6 s 4e-6 s 8 500e-9 s
MPI_Comm_rank 147e-6 s 147e-6 s 616 238.636e-9 s
MPI_Isend 3.26e-3 s 3.26e-3 s 920 3.54348e-6 s
MPI_Irecv 486e-6 s 486e-6 s 920 528.261e-9 s
MPI_Wtime 25e-6 s 25e-6 s 8 3.125e-6 s
MPI_Waitall 374e-6 s 374e-6 s 200 1.87e-6 s
MPI_Barrier 290e-6 s 290e-6 s 8 36.25e-6 s
MPI_Allreduce 32.44e-3 s 32.44e-3 s 59 549.831e-6 s
MPI_Wait 76.69e-3 s 76.69e-3 s 920 83.3587e-6 s
Performance Issue
Duration (%)
Duration
Wait at Barrier 1.35% 11.347e-3 s
Show advanced...
Description
Affected Processes
Source Locations (Root Causes)
Select performance issue to see details.

TAU Commander

- Tool that can be used to profile, trace, or sample your application
- Load with `taucmdr` module
- Requires moderate amount of setup:
  - Create a project with information about the application
    - For example, `tau initialize --mpi --compilers Intel`
  - Select appropriate measurement:
    - `tau select sample`
- To use:
  - `tau mpiexec ./mycode`
- To view and explore profile data:
  - `tau trial show <trial_number>`

TAU Profile

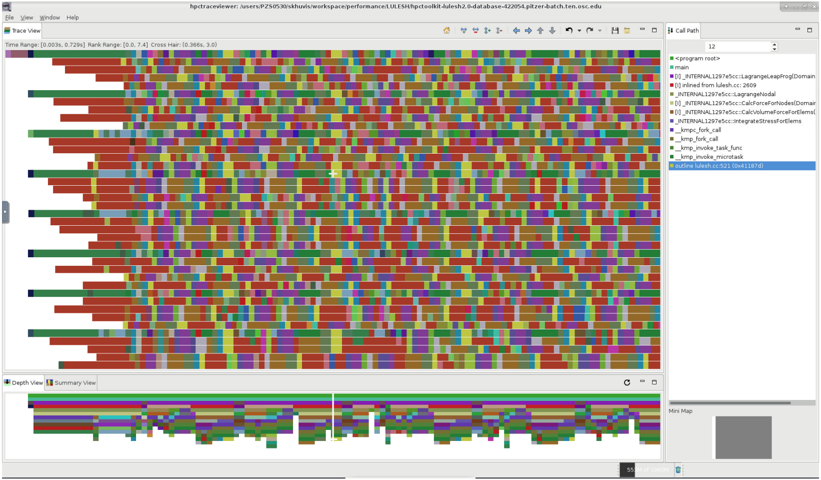


Metric: TIME
Value: Exclusive
Units: seconds
13.002 Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] [THROTTLED]
12.969 .TAU application => Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}]
7.426 [SUMMARY] void _INTERNAL_25______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dyn.rel.50.c0.s0.t1..h1.w1-fxe16lin03/../../src/kmp_w
6.214 [SAMPLE] void _INTERNAL_25______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dyn.rel.50.c0.s0.t1..h1.w1-fxe16lin03/../../src/kmp_wait
4.6 .TAU application => Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [CONTEXT] Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [SUMMARY]
4.448 .TAU application
2.961 .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}]
2.961 int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}]
2.523 [SUMMARY] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
1.826 [SUMMARY] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
1.427 .TAU application => [CONTEXT] .TAU application => [SUMMARY] void _INTERNAL_25______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dy
0.774 [SAMPLE] void _INTERNAL_25______src_kmp_barrier_cpp_71f3cf03::__kmp_wait_template<kmp_flag_64>(kmp_info*, kmp_flag_64*, int, void*) [{/nfs/site/proj/openmp/promo/20180209/tmp/lin_32e-rtl_5_nor_dyn.rel.50.c0.s0.t1..h1.w1-fxe16lin03/../../src/kmp_wait
0.712 [SUMMARY] _INTERNAL64e650b0::CalcElemFBHourglassForce(double*, double*, double*, double (*) [4], double, double*, double*, double*) [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.647 [SUMMARY] _INTERNAL64e650b0::SumElemStressesToNodeForces(double const (*) [8], double, double, double, double*, double*, double*) [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.603 .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => [CONTEXT] int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => [SAMPLE] syscall [{/usr/lib64/libc-2.17.so} {0}]
0.603 [SAMPLE] syscall [{/usr/lib64/libc-2.17.so} {0}]
0.582 .TAU application => Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [CONTEXT] Real_t CalcElemVolume(const Real_t *, const Real_t *, const Real_t *) [{lulesh.continue.cc} {1361,1}-{1366,1}] => [SUMMARY]
0.56 [SUMMARY] _INTERNAL64e650b0::SumElemFaceNormal(double*, double*, double*, double*, double*, double*, double*, double*, double*, double*, double*, double*, double, double, double, double, double, double, double, double, double, double, double, double) [{/us
0.548 [SUMMARY] L__ZN17_INTERNAL64e650b013CalcQForElemsER6Domain_1618__par_loop0_2_49 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.503 MPI_Wait()
0.46 .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => void CommSBN(Domain &, Int_t, Domain_member *) [{lulesh-comm.continue.cc} {848,1}-{1257,1}] => MPI_Wait()
0.445 [SUMMARY] _INTERNAL64e650b0::AreaFace(double, double, double, double, double, double, double, double, double, double, double, double) [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.425 [SUMMARY] _INTERNAL64e650b0::CalcElemShapeFunctionDerivatives(double const*, double const*, double const*, double (*) [8], double*) [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.422 .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => void CalcKinematicsForElems(Domain &, Real_t, Index_t) [{lulesh.continue.cc} {1505,1}-{1568,1}]
0.422 void CalcKinematicsForElems(Domain &, Real_t, Index_t) [{lulesh.continue.cc} {1505,1}-{1568,1}]
0.384 [SAMPLE] MPIDI_CH3I_SMP_read_progress [{/opt/mvapich2/intel/18.0/2.3/lib/libmpi.so.12.1.1} {0}]
0.348 [SUMMARY] L__Z22CalcKinematicsForElemsR6Domaindi_1510__par_loop0_2_54 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.347 [SUMMARY] L__ZN17_INTERNAL64e650b013CalcQForElemsER6Domain_1770__par_region1_2_50 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.344 [SUMMARY] L__ZN17_INTERNAL64e650b023IntegrateStressForElemsER6DomainPdS2_S2_S2_ii_565__par_loop1_2_84 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.321 [SAMPLE] _INTERNAL64e650b0::SumElemStressesToNodeForces(double const (*) [8], double, double, double, double*, double*, double*) [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {487}]
0.315 [SUMMARY] L__ZN17_INTERNAL64e650b015EvalEOSForElemsER6DomainPdiPii_2240__par_region0_2_20 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.312 [SUMMARY] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_dii_969__par_loop1_2_86 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.303 [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1024}]
0.302 [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1026}]
0.302 [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1027}]
0.288 [SAMPLE] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {903}]
0.288 [SAMPLE] _INTERNAL64e650b0::SumElemStressesToNodeForces(double const (*) [8], double, double, double, double*, double*, double*) [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {488}]
0.287 [SAMPLE] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {923}]
0.287 [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1028}]
0.285 [SAMPLE] L__ZN17_INTERNAL64e650b013LagrangeNodalER6Domain_1009__par_loop3_2_78 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {1025}]
0.271 [SAMPLE] L__ZN17_INTERNAL64e650b028CalcFBHourglassForceForElemsER6DomainPdS2_S2_S2_S2_S2_dii_782__par_loop0_2_85 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc} {913}]
0.263 [SUMMARY] _INTERNAL64e650b0::CalcElemVolume(double, double) [{/users/PZS0530/sk
0.257 .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => void CommSBN(Domain &, Int_t, Domain_member *) [{lulesh-comm.continue.cc} {848,1}-{1257,1}] => MPI_Wait() => [CONTEXT] MPI_Wait() => [SAMPLE] MPIDI_CH3I_SMP_read_progres
0.219 .TAU application => int main(int, char **) [{lulesh.continue.cc} {2650,1}-{2792,1}] => MPI_Allreduce()
0.219 MPI_Allreduce()
0.202 [SUMMARY] L__ZN17_INTERNAL64e650b015EvalEOSForElemsER6DomainPdiPii_2116__par_loop13_2_28 [{/users/PZS0530/skhuvis/workspace/performance/LULESH/lulesh.continue.cc}]
0.2 [SAMPLE] MPIDI_CH3I_SMP_pull_header [{/opt/mvapich2/intel/18.0/2.3/lib/libmpi.so.12.1.1} {0}]

HPCtoolkit

- Suite of tools that can be used to profile or trace your application
- Load with `hpctoolkit` module
- To profile your application: `mpiexec hpcrun ./mycode`
- This will produce a directory with a name of the form `hpctoolkit-mycode-measurements-pid.nodeid` containg profile data
- To convert the output to a format than be viewed by the `hpcviewer` tool, run `hpcprof hpctoolkit-mycode-measurements-pid.nodeid`
- To view the profile data generated during the run in a GUI, call `hpcviewer hpctoolkit-mycode-database-pid.nodeid`

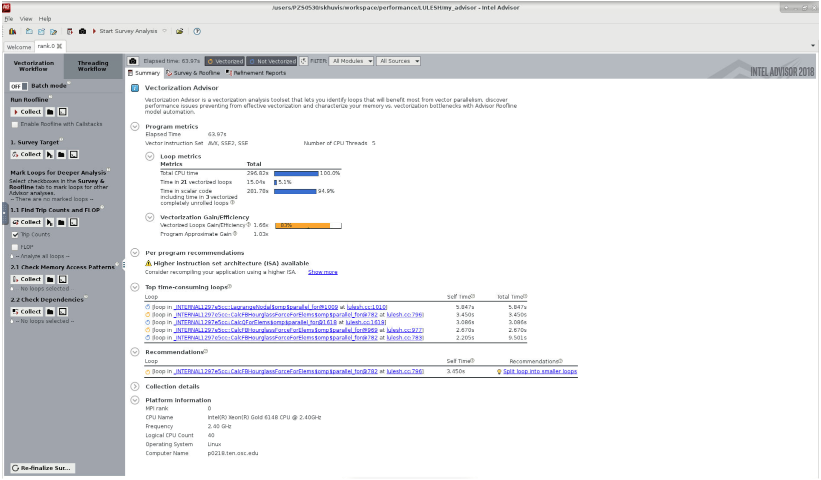
HPCToolkit Profile



Intel Advisor

- Graphical tool for optimizing vectorization and threading
- For using a GUI (use for small problems $<$ 5 minutes):
  - `advixe-gui`
- For non-MPI non-interactive usage:
  - `advixe-cl -collect survey -project-dir ./my_advisor ./mycode`
- For MPI non-interactive usage:
  - `mpirun -n <mpi_tasks> advixe-cl -collect survey -project-dir ./my_advisor ./mycode`
- View and explore existing results with `advixe-gui`

Intel Advisor GUI



Resources to get your questions answered

FAQs: osc.edu/resources/getting_started/supercomputing_faq
HOW TOs: osc.edu/resources/getting_started/howto

Performance Collection Guide:
osc.edu/resources/getting_started/howto/howto_collect_performance_data_for_your_program

Office Hours:
go.osu.edu/rc-osc Tuesdays 1-3 p.m. or Wednesdays and Fridays 1-2:30 p.m. at Pomerene Hall

System updates:

- Read Message of the Day on login
- Follow @HPCNotices on Twitter

Optimization and Performance Tuning Workshop

- May 21, 2019 at 1-4 pm
- Present techniques for improving the performance of scientific software on High Performance Computing (HPC) systems such as those available at OSC.
- The focus will be on serial performance, including vectorization and cache utilization, with a brief mention of parallel computing.
- Topics covered:
  - Hardware overview
  - Important factors for good performance
  - Compiler optimizations
  - Profiling tools
- `osc.edu/calendar/events/2019_05_21-optimization_performance_tuning_workshop`

OH·TECH

Ohio Technology Consortium

A Division of the Ohio Department of Higher Education

info@osc.edu

twitter.com/osc

facebook.com/ohiosupercomputercenter

osc.edu

oh-tech.org/blog

linkedin.com/company/ohio-supercomputer-center