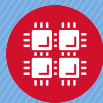


Introduction to Performance Tools

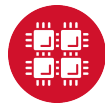
Samuel Khuvis

Scientific Applications Engineer, OSC



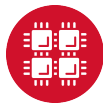
Goals of the Breakout Session

- ▶ 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



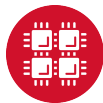
For More Information

- ▶ 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



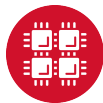
Profiling/Debugging Tools Available at OSC

- ▶ 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?

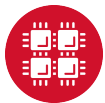
- ▶ Debuggers let you
 - ▶ execute your program one line at a time (“step”)
 - ▶ inspect variable values
 - ▶ stop your program at a particular line (“breakpoint”)
 - ▶ open a “core” file (after program crashes)
- ▶ HPC debuggers
 - ▶ support multithreaded code
 - ▶ support MPI code
 - ▶ support GPU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



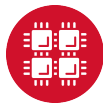
Compilation flags for debugging

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`

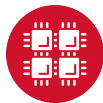


- ▶ Available on all OSC clusters
 - ▶ `module load arm-ddt`
- ▶ To run a non-MPI program from the command line:
 - ▶ `ddt --offline --no-mpi ./mycode [args]`
- ▶ To run a MPI program from the command line:
 - ▶ `ddt --offline -np num_procs ./mycode [args]`



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

Interactive Apps
Desktops
<input type="checkbox"/> Owens Desktop
<input checked="" type="checkbox"/> Pitzer Desktop
<input type="checkbox"/> Ruby Desktop
<input type="checkbox"/> Owens VDI
<input type="checkbox"/> Pitzer VDI
<input type="checkbox"/> Ruby VDI
GUIs
<input type="checkbox"/> ANSYS Workbench
<input type="checkbox"/> Abaqus/CAE
<input type="checkbox"/> COMSOL Multiphysics
<input type="checkbox"/> MATLAB
<input type="checkbox"/> ParaView
<input type="checkbox"/> VMD
Servers
<input type="checkbox"/> Jupyter + Spark
<input type="checkbox"/> Jupyter Notebook
<input type="checkbox"/> Jupyter Notebook (Pitzer)
<input type="checkbox"/> RStudio Server
<input type="checkbox"/> RStudio Server (Pitzer)

Pitzer Desktop

This app will launch an interactive desktop on one or more compute nodes. You will have full access to the resources these nodes provide. This is analogous to an interactive batch job.

Desktop environment

This will launch either the [Xfce](#) or [Mate](#) desktop environment on the [Pitzer cluster](#).

Account

You can leave this blank if not in multiple projects.

Number of hours

Number of nodes

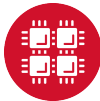
Node type

- any - (40 cores)** Chooses anyone of the available Pitzer nodes. This reduces the wait time as you have no requirements. Standard Pitzer nodes have 192GB of memory.
- vis - (40 cores)** This requests a gpu node as described below, with the addition that it starts an X server running in the background. This allows for Hardware Rendering with the GPU typically needed for 3D visualization using VirtualGL.
- gpu - (40 cores)** This node includes two NVIDIA Tesla V100 GPUs allowing for CUDA computations. This node has 384GB of memory. There are currently only 32 of these nodes on Pitzer. These nodes don't start an X server, so visualization with hardware rendering is not possible.
- hugemem - (80 cores)** This Pitzer node has 3TB of memory as well as 80 cores. There are only 4 of these nodes on Pitzer. A reservation may be required to use this node.

Resolution

width	<input type="text" value="1536"/>	px	height	<input type="text" value="960"/>	px
-------	-----------------------------------	----	--------	----------------------------------	----

I would like to receive an email when the session starts

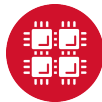


ARM DDT

The screenshot displays the ARM DDT interface with the following components:

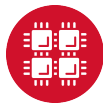
- Project Files:** A tree view on the left showing the project structure with files like 'l2.cpp', 'l2ehp.cpp', 'l2ee2.cpp', and 'l2ee3.cpp'.
- Code Editor:** The main window showing C++ code for 'l2.cpp'. The code includes headers, constants, and a main function that simulates a system with energy and force interactions. A breakpoint is set at line 136.
- Locals:** A window on the right showing the current state of local variables, such as 'e', 'p', 'm', and 't', with their corresponding values.
- Stack:** A window at the bottom showing the current call stack, including the main function and various system calls.

```
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



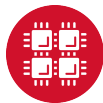
What can a profiler show you?

- ▶ 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



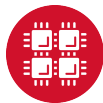
Compilation flags for profiling

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



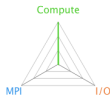
ARM Performance Reports

- ▶ 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/skhuis/SETSM/setsm
dataset/WV01_15MAY080613301-
P1B5-102001003C02A600.tif
dataset/WV01_15MAY080614188-
P1B5-102001003EA5DA00.tif out -outr5 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/skhuis/SETSM
```



Summary: setsm is **Compute-bound** in this configuration

Compute	99.3%	<div style="width: 99.3%; height: 10px; background-color: green;"></div>	Time spent running application code. High values are usually good. This is very high ; check the CPU performance section for advice
MPI	0.0%	<div style="width: 0.0%; height: 10px; background-color: blue;"></div>	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%	<div style="width: 0.7%; height: 10px; background-color: orange;"></div>	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%	<div style="width: 44.5%; height: 10px; background-color: green;"></div>
OpenMP regions	55.5%	<div style="width: 55.5%; height: 10px; background-color: green;"></div>
Scalar numeric ops	21.8%	<div style="width: 21.8%; height: 10px; background-color: green;"></div>
Vector numeric ops	4.4%	<div style="width: 4.4%; height: 10px; background-color: green;"></div>
Memory accesses	43.7%	<div style="width: 43.7%; height: 10px; background-color: green;"></div>

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

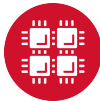
Little time is spent in **vectorized instructions**. Check the compiler's vectorization advice to see why key loops could not be vectorized.

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.

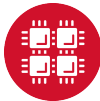
Energy

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	

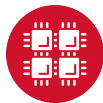
The **whole system energy** has been calculated using the **CPU** energy usage.

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?



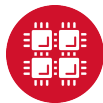
ARM MAP

- ▶ 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



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`



Intel VTune Amplifier 2018

Basic Hotspots Hotspots by CPU Utilization viewpoint (change)

Analysis Target: Analysis Type: Collection Log Summary Bottom-up Caller/Caller Top-down Tree Platform

Elapsed Time: 32.422s

- CPU Time: 1225.190s
- Effective Time: 184.764s
- Spin Time: 1008.932s
 - Wastage at Serial Spinning: 1293.582s
 - Lock Contention: 9.120s
 - Other: 15.230s
- Overhead Time: 41.494s
 - Total Thread Count: 69
 - Passed 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_bsk_barrier	libomp5.so	1046.882s
INTERNAL1297d5cc:Kspargelbdsdmp\$parah2_to@1188	libsh2.so	24.252s
__kmp_bsk_csk	libomp5.so	22.788s
__kmp_barrier	libomp5.so	15.288s
INTERNAL1297d5cc:CallOfBlissgastForceOfEvmsdmp\$parah2_to@792	libsh2.so	14.428s
[2]@940		91.551s

Effective CPU Utilization Histogram

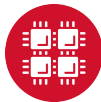
This histogram displays a percentage of the real time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the idle CPU utilization value.

Collection and Platform Info

This section provides information about this collection, including result set size and collection platform data.

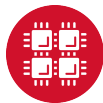
```

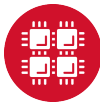
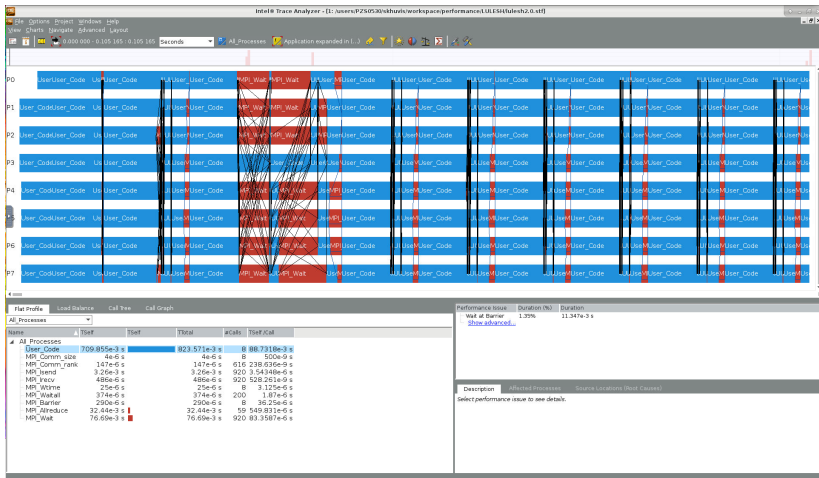
Application Command Line: ./sh2.0
Operating System: 3.10.0-11.6.el7.x86_64 NAME="Red Hat Enterprise Linux Server" VERSION="7.5" ANSIBLE_ID="rhel" ID="rhel" ID_LINUX="rhel" VARIANT="Server" VARIANT_ID="server" VERSION_ID="7.5" PRETTY_NAME="Red Hat Enterprise Linux" ANSIBLE_COLOR="031"
CPU_NAME="Type: Intel(R) Xeon(R) Platinum 8269C CPU @ 2.90GHz" CPU_VENDOR="Intel(R) Xeon(R) Platinum 8269C CPU @ 2.90GHz" CPU_MODEL="8269C" CPU_FAMILY="605" CPU_FEATURES="cx16 cx8 cx8_1" CPU_FLAGS_1="fpu_eme" CPU_FLAGS_2="cmov clflush clflushopt clflushwb clwb clzero clzero_1 clzero_2 clzero_3 clzero_4 clzero_5 clzero_6 clzero_7 clzero_8 clzero_9 clzero_10 clzero_11 clzero_12 clzero_13 clzero_14 clzero_15 clzero_16 clzero_17 clzero_18 clzero_19 clzero_20 clzero_21 clzero_22 clzero_23 clzero_24 clzero_25 clzero_26 clzero_27 clzero_28 clzero_29 clzero_30 clzero_31 clzero_32 clzero_33 clzero_34 clzero_35 clzero_36 clzero_37 clzero_38 clzero_39 clzero_40 clzero_41 clzero_42 clzero_43 clzero_44 clzero_45 clzero_46 clzero_47 clzero_48 clzero_49 clzero_50 clzero_51 clzero_52 clzero_53 clzero_54 clzero_55 clzero_56 clzero_57 clzero_58 clzero_59 clzero_60 clzero_61 clzero_62 clzero_63 clzero_64 clzero_65 clzero_66 clzero_67 clzero_68 clzero_69 clzero_70 clzero_71 clzero_72 clzero_73 clzero_74 clzero_75 clzero_76 clzero_77 clzero_78 clzero_79 clzero_80 clzero_81 clzero_82 clzero_83 clzero_84 clzero_85 clzero_86 clzero_87 clzero_88 clzero_89 clzero_90 clzero_91 clzero_92 clzero_93 clzero_94 clzero_95 clzero_96 clzero_97 clzero_98 clzero_99 clzero_100 clzero_101 clzero_102 clzero_103 clzero_104 clzero_105 clzero_106 clzero_107 clzero_108 clzero_109 clzero_110 clzero_111 clzero_112 clzero_113 clzero_114 clzero_115 clzero_116 clzero_117 clzero_118 clzero_119 clzero_120 clzero_121 clzero_122 clzero_123 clzero_124 clzero_125 clzero_126 clzero_127 clzero_128 clzero_129 clzero_130 clzero_131 clzero_132 clzero_133 clzero_134 clzero_135 clzero_136 clzero_137 clzero_138 clzero_139 clzero_140 clzero_141 clzero_142 clzero_143 clzero_144 clzero_145 clzero_146 clzero_147 clzero_148 clzero_149 clzero_150 clzero_151 clzero_152 clzero_153 clzero_154 clzero_155 clzero_156 clzero_157 clzero_158 clzero_159 clzero_160 clzero_161 clzero_162 clzero_163 clzero_164 clzero_165 clzero_166 clzero_167 clzero_168 clzero_169 clzero_170 clzero_171 clzero_172 clzero_173 clzero_174 clzero_175 clzero_176 clzero_177 clzero_178 clzero_179 clzero_180 clzero_181 clzero_182 clzero_183 clzero_184 clzero_185 clzero_186 clzero_187 clzero_188 clzero_189 clzero_190 clzero_191 clzero_192 clzero_193 clzero_194 clzero_195 clzero_196 clzero_197 clzero_198 clzero_199 clzero_200 clzero_201 clzero_202 clzero_203 clzero_204 clzero_205 clzero_206 clzero_207 clzero_208 clzero_209 clzero_210 clzero_211 clzero_212 clzero_213 clzero_214 clzero_215 clzero_216 clzero_217 clzero_218 clzero_219 clzero_220 clzero_221 clzero_222 clzero_223 clzero_224 clzero_225 clzero_226 clzero_227 clzero_228 clzero_229 clzero_230 clzero_231 clzero_232 clzero_233 clzero_234 clzero_235 clzero_236 clzero_237 clzero_238 clzero_239 clzero_240 clzero_241 clzero_242 clzero_243 clzero_244 clzero_245 clzero_246 clzero_247 clzero_248 clzero_249 clzero_250 clzero_251 clzero_252 clzero_253 clzero_254 clzero_255 clzero_256 clzero_257 clzero_258 clzero_259 clzero_260 clzero_261 clzero_262 clzero_263 clzero_264 clzero_265 clzero_266 clzero_267 clzero_268 clzero_269 clzero_270 clzero_271 clzero_272 clzero_273 clzero_274 clzero_275 clzero_276 clzero_277 clzero_278 clzero_279 clzero_280 clzero_281 clzero_282 clzero_283 clzero_284 clzero_285 clzero_286 clzero_287 clzero_288 clzero_289 clzero_290 clzero_291 clzero_292 clzero_293 clzero_294 clzero_295 clzero_296 clzero_297 clzero_298 clzero_299 clzero_300 clzero_301 clzero_302 clzero_303 clzero_304 clzero_305 clzero_306 clzero_307 clzero_308 clzero_309 clzero_310 clzero_311 clzero_312 clzero_313 clzero_314 clzero_315 clzero_316 clzero_317 clzero_318 clzero_319 clzero_320 clzero_321 clzero_322 clzero_323 clzero_324 clzero_325 clzero_326 clzero_327 clzero_328 clzero_329 clzero_330 clzero_331 clzero_332 clzero_333 clzero_334 clzero_335 clzero_336 clzero_337 clzero_338 clzero_339 clzero_340 clzero_341 clzero_342 clzero_343 clzero_344 clzero_345 clzero_346 clzero_347 clzero_348 clzero_349 clzero_350 clzero_351 clzero_352 clzero_353 clzero_354 clzero_355 clzero_356 clzero_357 clzero_358 clzero_359 clzero_360 clzero_361 clzero_362 clzero_363 clzero_364 clzero_365 clzero_366 clzero_367 clzero_368 clzero_369 clzero_370 clzero_371 clzero_372 clzero_373 clzero_374 clzero_375 clzero_376 clzero_377 clzero_378 clzero_379 clzero_380 clzero_381 clzero_382 clzero_383 clzero_384 clzero_385 clzero_386 clzero_387 clzero_388 clzero_389 clzero_390 clzero_391 clzero_392 clzero_393 clzero_394 clzero_395 clzero_396 clzero_397 clzero_398 clzero_399 clzero_400 clzero_401 clzero_402 clzero_403 clzero_404 clzero_405 clzero_406 clzero_407 clzero_408 clzero_409 clzero_410 clzero_411 clzero_412 clzero_413 clzero_414 clzero_415 clzero_416 clzero_417 clzero_418 clzero_419 clzero_420 clzero_421 clzero_422 clzero_423 clzero_424 clzero_425 clzero_426 clzero_427 clzero_428 clzero_429 clzero_430 clzero_431 clzero_432 clzero_433 clzero_434 clzero_435 clzero_436 clzero_437 clzero_438 clzero_439 clzero_440 clzero_441 clzero_442 clzero_443 clzero_444 clzero_445 clzero_446 clzero_447 clzero_448 clzero_449 clzero_450 clzero_451 clzero_452 clzero_453 clzero_454 clzero_455 clzero_456 clzero_457 clzero_458 clzero_459 clzero_460 clzero_461 clzero_462 clzero_463 clzero_464 clzero_465 clzero_466 clzero_467 clzero_468 clzero_469 clzero_470 clzero_471 clzero_472 clzero_473 clzero_474 clzero_475 clzero_476 clzero_477 clzero_478 clzero_479 clzero_480 clzero_481 clzero_482 clzero_483 clzero_484 clzero_485 clzero_486 clzero_487 clzero_488 clzero_489 clzero_490 clzero_491 clzero_492 clzero_493 clzero_494 clzero_495 clzero_496 clzero_497 clzero_498 clzero_499 clzero_500 clzero_501 clzero_502 clzero_503 clzero_504 clzero_505 clzero_506 clzero_507 clzero_508 clzero_509 clzero_510 clzero_511 clzero_512 clzero_513 clzero_514 clzero_515 clzero_516 clzero_517 clzero_518 clzero_519 clzero_520 clzero_521 clzero_522 clzero_523 clzero_524 clzero_525 clzero_526 clzero_527 clzero_528 clzero_529 clzero_530 clzero_531 clzero_532 clzero_533 clzero_534 clzero_535 clzero_536 clzero_537 clzero_538 clzero_539 clzero_540 clzero_541 clzero_542 clzero_543 clzero_544 clzero_545 clzero_546 clzero_547 clzero_548 clzero_549 clzero_550 clzero_551 clzero_552 clzero_553 clzero_554 clzero_555 clzero_556 clzero_557 clzero_558 clzero_559 clzero_560 clzero_561 clzero_562 clzero_563 clzero_564 clzero_565 clzero_566 clzero_567 clzero_568 clzero_569 clzero_570 clzero_571 clzero_572 clzero_573 clzero_574 clzero_575 clzero_576 clzero_577 clzero_578 clzero_579 clzero_580 clzero_581 clzero_582 clzero_583 clzero_584 clzero_585 clzero_586 clzero_587 clzero_588 clzero_589 clzero_590 clzero_591 clzero_592 clzero_593 clzero_594 clzero_595 clzero_596 clzero_597 clzero_598 clzero_599 clzero_600 clzero_601 clzero_602 clzero_603 clzero_604 clzero_605 clzero_606 clzero_607 clzero_608 clzero_609 clzero_610 clzero_611 clzero_612 clzero_613 clzero_614 clzero_615 clzero_616 clzero_617 clzero_618 clzero_619 clzero_620 clzero_621 clzero_622 clzero_623 clzero_624 clzero_625 clzero_626 clzero_627 clzero_628 clzero_629 clzero_630 clzero_631 clzero_632 clzero_633 clzero_634 clzero_635 clzero_636 clzero_637 clzero_638 clzero_639 clzero_640 clzero_641 clzero_642 clzero_643 clzero_644 clzero_645 clzero_646 clzero_647 clzero_648 clzero_649 clzero_650 clzero_651 clzero_652 clzero_653 clzero_654 clzero_655 clzero_656 clzero_657 clzero_658 clzero_659 clzero_660 clzero_661 clzero_662 clzero_663 clzero_664 clzero_665 clzero_666 clzero_667 clzero_668 clzero_669 clzero_670 clzero_671 clzero_672 clzero_673 clzero_674 clzero_675 clzero_676 clzero_677 clzero_678 clzero_679 clzero_680 clzero_681 clzero_682 clzero_683 clzero_684 clzero_685 clzero_686 clzero_687 clzero_688 clzero_689 clzero_690 clzero_691 clzero_692 clzero_693 clzero_694 clzero_695 clzero_696 clzero_697 clzero_698 clzero_699 clzero_700 clzero_701 clzero_702 clzero_703 clzero_704 clzero_705 clzero_706 clzero_707 clzero_708 clzero_709 clzero_710 clzero_711 clzero_712 clzero_713 clzero_714 clzero_715 clzero_716 clzero_717 clzero_718 clzero_719 clzero_720 clzero_721 clzero_722 clzero_723 clzero_724 clzero_725 clzero_726 clzero_727 clzero_728 clzero_729 clzero_730 clzero_731 clzero_732 clzero_733 clzero_734 clzero_735 clzero_736 clzero_737 clzero_738 clzero_739 clzero_740 clzero_741 clzero_742 clzero_743 clzero_744 clzero_745 clzero_746 clzero_747 clzero_748 clzero_749 clzero_750 clzero_751 clzero_752 clzero_753 clzero_754 clzero_755 clzero_756 clzero_757 clzero_758 clzero_759 clzero_760 clzero_761 clzero_762 clzero_763 clzero_764 clzero_765 clzero_766 clzero_767 clzero_768 clzero_769 clzero_770 clzero_771 clzero_772 clzero_773 clzero_774 clzero_775 clzero_776 clzero_777 clzero_778 clzero_779 clzero_780 clzero_781 clzero_782 clzero_783 clzero_784 clzero_785 clzero_786 clzero_787 clzero_788 clzero_789 clzero_790 clzero_791 clzero_792 clzero_793 clzero_794 clzero_795 clzero_796 clzero_797 clzero_798 clzero_799 clzero_800 clzero_801 clzero_802 clzero_803 clzero_804 clzero_805 clzero_806 clzero_807 clzero_808 clzero_809 clzero_810 clzero_811 clzero_812 clzero_813 clzero_814 clzero_815 clzero_816 clzero_817 clzero_818 clzero_819 clzero_820 clzero_821 clzero_822 clzero_823 clzero_824 clzero_825 clzero_826 clzero_827 clzero_828 clzero_829 clzero_830 clzero_831 clzero_832 clzero_833 clzero_834 clzero_835 clzero_836 clzero_837 clzero_838 clzero_839 clzero_840 clzero_841 clzero_842 clzero_843 clzero_844 clzero_845 clzero_846 clzero_847 clzero_848 clzero_849 clzero_850 clzero_851 clzero_852 clzero_853 clzero_854 clzero_855 clzero_856 clzero_857 clzero_858 clzero_859 clzero_860 clzero_861 clzero_862 clzero_863 clzero_864 clzero_865 clzero_866 clzero_867 clzero_868 clzero_869 clzero_870 clzero_871 clzero_872 clzero_873 clzero_874 clzero_875 clzero_876 clzero_877 clzero_878 clzero_879 clzero_880 clzero_881 clzero_882 clzero_883 clzero_884 clzero_885 clzero_886 clzero_887 clzero_888 clzero_889 clzero_890 clzero_891 clzero_892 clzero_893 clzero_894 clzero_895 clzero_896 clzero_897 clzero_898 clzero_899 clzero_900 clzero_901 clzero_902 clzero_903 clzero_904 clzero_905 clzero_906 clzero_907 clzero_908 clzero_909 clzero_910 clzero_911 clzero_912 clzero_913 clzero_914 clzero_915 clzero_916 clzero_917 clzero_918 clzero_919 clzero_920 clzero_921 clzero_922 clzero_923 clzero_924 clzero_925 clzero_926 clzero_927 clzero_928 clzero_929 clzero_930 clzero_931 clzero_932 clzero_933 clzero_934 clzero_935 clzero_936 clzero_937 clzero_938 clzero_939 clzero_940 clzero_941 clzero_942 clzero_943 clzero_944 clzero_945 clzero_946 clzero_947 clzero_948 clzero_949 clzero_950 clzero_951 clzero_952 clzero_953 clzero_954 clzero_955 clzero_956 clzero_957 clzero_958 clzero_959 clzero_960 clzero_961 clzero_962 clzero_963 clzero_964 clzero_965 clzero_966 clzero_967 clzero_968 clzero_969 clzero_970 clzero_971 clzero_972 clzero_973 clzero_974 clzero_975 clzero_976 clzero_977 clzero_978 clzero_979 clzero_980 clzero_981 clzero_982 clzero_983 clzero_984 clzero_985 clzero_986 clzero_987 clzero_988 clzero_989 clzero_990 clzero_991 clzero_992 clzero_993 clzero_994 clzero_995 clzero_996 clzero_997 clzero_998 clzero_999 clzero_1000
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux" REDHAT_SUPPORT_PRODUCT_VERSION="7.5"
    
```



Intel Trace Analyzer and Collector (ITAC)

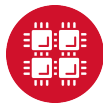
- ▶ Graphical tool for profiling MPI code (Intel MPI)
- ▶ To use:
 - ▶ `module load intelmpi # then compile (-g) code`
 - ▶ `mpiexec -trace ./mycode`
- ▶ View and explore existing results using GUI with `traceanalyzer`:
 - ▶ `traceanalyzer <mycode>.stf`





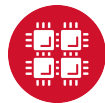
TAU Commander

- ▶ 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>`

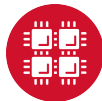
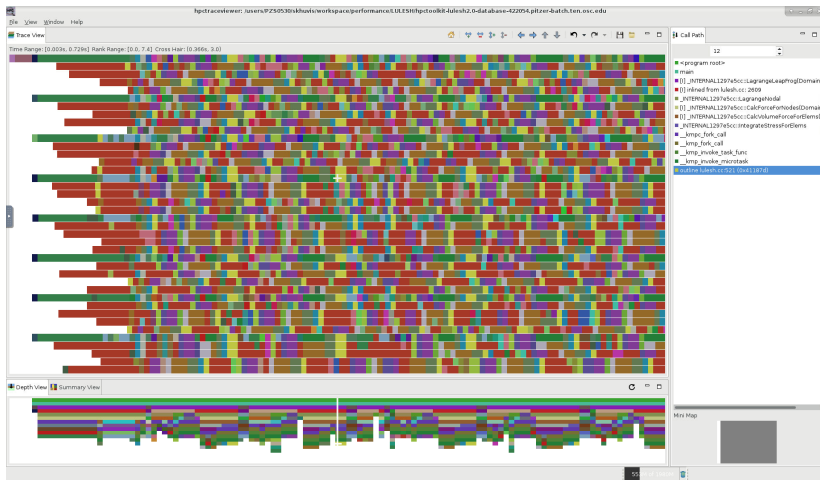


HPCToolkit

- ▶ 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` containing profile data
- ▶ To convert the output to a format that can 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`

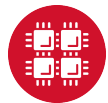


HPCToolkit Profile



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

Intel Advisor GUI showing performance analysis results for a program.

Vectorization Advisor

Vectorization Advisor is a vectorization analysis toolkit that lets you identify loops that will benefit most from vector parallelism, discover performance issues preventing from effective vectorization and characterize your memory vs. vectorization bottlenecks with Advisor Rofline model automation.

Program metrics

Elapsed Time: 63.97s
Vector Instruction Set: AVX, SSE2, SSE
Number of CPU Threads: 5

Loop metrics

Metrics

Metric	Total
Total CPU Time	295.32s
Time in 2L vectorized loops	15.04s (5.1%)
Time in scalar code	281.78s (94.9%)

including time in 3 vectorized completely unrolled loops

Vectorization Gain/Efficiency

Vectorized Loops Gain Efficiency: 1.66x
Program Approximate Gain: 1.03x

Per program recommendations

Higher instruction set architecture (ISA) available
Consider recompiling your application using a higher ISA. [Show more](#)

Top time-consuming loops

Loop	Self Time	Total Time
Loop in _NTRMALL207f5ccc-LagrangeMethodCompSparseFor@1009 at Mesh.cc:1010	5.847s	5.847s
Loop in _NTRMALL207f5ccc-CalcOfHourglassElemForElemCompSparseFor@782 at Mesh.cc:798	3.450s	3.450s
Loop in _NTRMALL207f5ccc-CalcOfHourglassElemForElemCompSparseFor@1018 at Mesh.cc:1018	3.086s	3.086s
Loop in _NTRMALL207f5ccc-CalcOfHourglassElemForElemCompSparseFor@995@ at Mesh.cc:977	2.670s	2.670s
Loop in _NTRMALL207f5ccc-CalcOfHourglassElemForElemCompSparseFor@782 at Mesh.cc:782	2.205s	9.503s

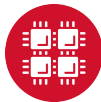
Recommendations

Loop	Self Time	Recommendations
Loop in _NTRMALL207f5ccc-CalcOfHourglassElemForElemCompSparseFor@782 at Mesh.cc:798	3.450s	Fold loop into smaller loops

Collection details

Platform information

MPI rank: 0
CPU Name: Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz
Frequency: 2.40 GHz
Logical CPU Count: 40
Operating System: Linux
Computer Name: p02L8.tah.ncc.edu



Resources to get your questions answered

FAQs: osc.edu/resources/getting_started/supercomputing_faqs

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

Performance Collection Guide:

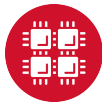
osc.edu/resources/getting_started/howto/howto_collect_performance_data_for_your_program

Office Hours:

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

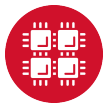
System updates:

- ▶ Read Message of the Day on login
- ▶ Follow [@HPCNotices](https://twitter.com/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

