Software

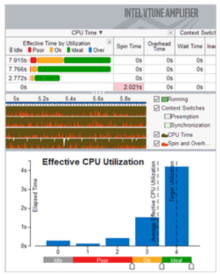
Profiling your application with Intel® Vtune™ Amplifier

Paulius Velesko

Getting a local copy of Vtune™

Get a free version of vtune for your PC/Mac/Linux machine:

https://software.intel.com/en-us/vtune/choose-download



Only Intel® VTune™ Amplifier

This advanced profiler helps you increase application performance on modern hardware.

Free Download

Download a free copy backed by community forum support.

Download

Buy It Now

A paid product entitles you to:

- Private Priority Support from Intel's engineers
- The ability to submit confidential code samples
- Responsive help with technical questions
- Access to older versions
- Support renewals at a lower rate

Buy the software from a number of resellers or directly from the online store. Special pricing for academic research is available.

Find a Reseller

Buy Now

Pricing for Academic Research

Tuning at Multiple Hardware Levels

Exploiting all features of modern processors requires good use of the available resources

- Core
  - Vectorization is critical with 512bit FMA vector units (32 DP ops/cycle)
  - Targeting the current ISA is fundamental to fully exploit vectorization
- Socket
  - Using all cores in a processor requires parallelization (MPI, OMP, …)
  - Up to 64 Physical cores and 256 logical processors per socket on Theta!
- Node
  - Minimize remote memory access (control memory affinity)
  - Minimize resource sharing (tune local memory access, disk IO and network traffic)

Intel® Compiler Reports

FREE* performance metrics

Compile with -qopt-report=5

- Which loops were vectorized
  - Vector Length
  - Estimated Gain
  - Alignment
  - Scatter/Gather
- Prefetching
- Issues preventing vectorization
- Inline reports
- Interprocedural optimizations
- Register Spills/Fills

```
LOOP BEGIN at ../src/timestep.F(4835,13)
   remark #15389: vectorization support: reference nbd_(i) has unaligned access   [ ../src/timestep.F(4836,16) ]
   remark #15381: vectorization support: unaligned access used inside loop body
   remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use vector always directive or -vec-threshold0 to override
   remark #15329: vectorization support: irregularly indexed store was emulated for the variable <coefd_(nbd_(i))>, part of index is read from memory
   remark #15305: vectorization support: vector length 2
   remark #15399: vectorization support: unroll factor set to 4
   remark #15309: vectorization support: normalized vectorization overhead 0.139
   remark #15450: unmasked unaligned unit stride loads: 1
   remark #15463: unmasked indexed (or scatter) stores: 1
   remark #15475: --- begin vector cost summary ---
   remark #15476: scalar cost: 4
   remark #15477: vector cost: 4.500
   remark #15478: estimated potential speedup: 0.880
   remark #15488: --- end vector cost summary ---
   remark #25439: unrolled with remainder by 2
LOOP END
```

Getting your application ready for profiling

- Always add -g
  - No performance penalty
- On Cray Systems (Theta) add -dynamic
- E.x:
  - cc -g -xMIC-AVX512 -dynamic -c test.cpp

Intel® Application Performance Snapshot

Profile your entire application at scale

VTune™ Amplifier's Application Performance Snapshot

High-level overview of application performance

- Identify primary optimization areas
- Recommend next steps in analysis
- Extremely easy to use
- Informative, actionable data in clean HTML report
- Detailed reports available via command line
- Low overhead, high scalability
- Can be run at scale - python ML/AI profiles

Usage on Theta

Launch all profiling jobs from **/projects** rather than **/home**

```
$ module load aps
```

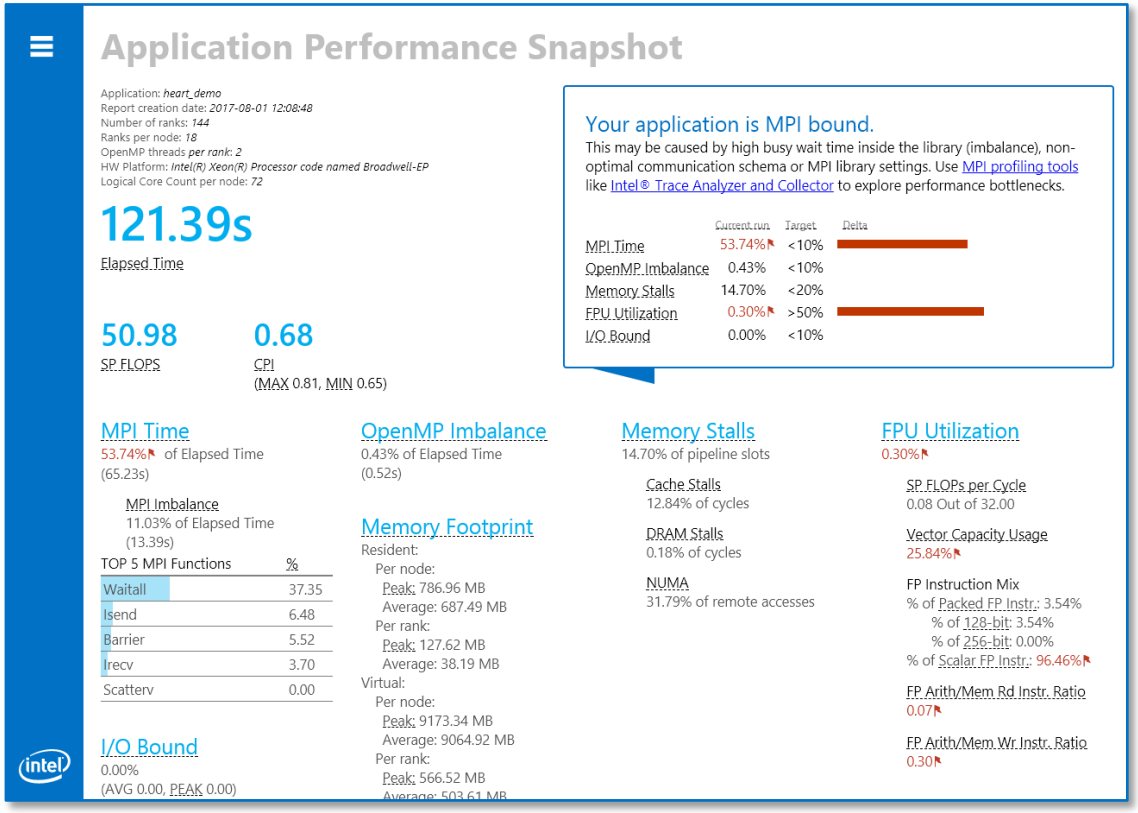
Launch your job in interactive or batch mode:

```
$ aprun -N <ppn> -n <totRanks> [affinity opts] aps ./exe
```

Produce text and html reports:

```
$ aps -report=./aps_result_ ….
```

APS HTML Report



Application Performance Snapshot

Application: *heart_demo*
Report creation date: *2017-08-01 12:08:48*
Number of ranks: *144*
Ranks per node: *18*
OpenMP threads *per rank: 2*
HW Platform: *Intel(R) Xeon(R) Processor code named Broadwell-EP*
Logical Core Count per node: *72*

121.39s
Elapsed Time

50.98
SP FLOPS

0.68
CPI
(MAX 0.81, MIN 0.65)

Your application is MPI bound.

This may be caused by high busy wait time inside the library (imbalance), non-optimal communication schema or MPI library settings. Use MPI profiling tools like Intel® Trace Analyzer and Collector to explore performance bottlenecks.

	Current run	Target	Delta
MPI Time	53.74%	<10%	
OpenMP Imbalance	0.43%	<10%	
Memory Stalls	14.70%	<20%	
FPU Utilization	0.30%	>50%	
I/O Bound	0.00%	<10%	

MPI Time
53.74% of Elapsed Time
(65.23s)

MPI Imbalance
11.03% of Elapsed Time
(13.39s)

TOP 5 MPI Functions	%
Waitall	37.35
Isend	6.48
Barrier	5.52
Irecv	3.70
Scatterv	0.00

I/O Bound
0.00%
(AVG 0.00, PEAK 0.00)

OpenMP Imbalance
0.43% of Elapsed Time
(0.52s)

Memory Footprint
Resident:
Per node:
Peak: 786.96 MB
Average: 687.49 MB
Per rank:
Peak: 127.62 MB
Average: 38.19 MB
Virtual:
Per node:
Peak: 9173.34 MB
Average: 9064.92 MB
Per rank:
Peak: 566.52 MB
Average: 503.61 MB

Memory Stalls
14.70% of pipeline slots

Cache Stalls
12.84% of cycles

DRAM Stalls
0.18% of cycles

NUMA
31.79% of remote accesses

FPU Utilization
0.30%

SP FLOPs per Cycle
0.08 Out of 32.00

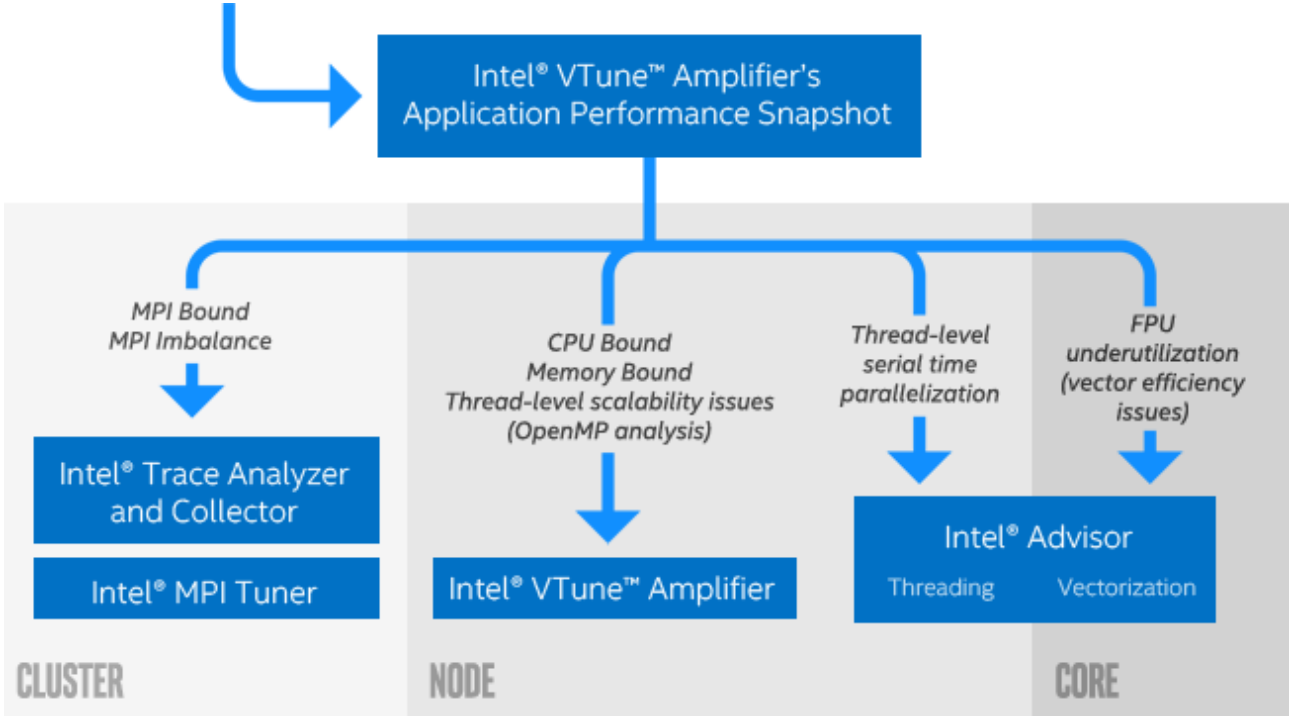
Vector Capacity Usage
25.84%

FP Instruction Mix
% of Packed FP Instr.: 3.54%
% of 128-bit: 3.54%
% of 256-bit: 0.00%
% of Scalar FP Instr.: 96.46%

FP Arith/Mem Rd Instr. Ratio
0.07

FP Arith/Mem Wr Instr. Ratio
0.30

Tuning Workflow



Intel® VTUNE™ Amplifier

Core-level hardware metrics

https://www.alcf.anl.gov/user-guides/vtune-xc40

Intel® VTune™ Amplifier

VTune Amplifier is a full system profiler

- Accurate
- Low overhead
- Comprehensive (microarchitecture, memory, IO, treading, …)
- Highly customizable interface
- Direct access to source code and assembly
- User-mode driverless sampling
- Event-based sampling

Analyzing code access to shared resources is critical to achieve good performance on multicore and manycore systems

Predefined Collections

Many available analysis types:

- uarch-exploration General microarchitecture exploration
- hpc-performance HPC Performance Characterization
- memory-access Memory Access
- disk-io Disk Input and Output
- concurrency Concurrency
- gpu-hotspots GPU Hotspots
- gpu-profiling GPU In-kernel Profiling
- hotspots Basic Hotspots
- locksandwaits Locks and Waits
- memory-consumption Memory Consumption
- system-overview System Overview
- …

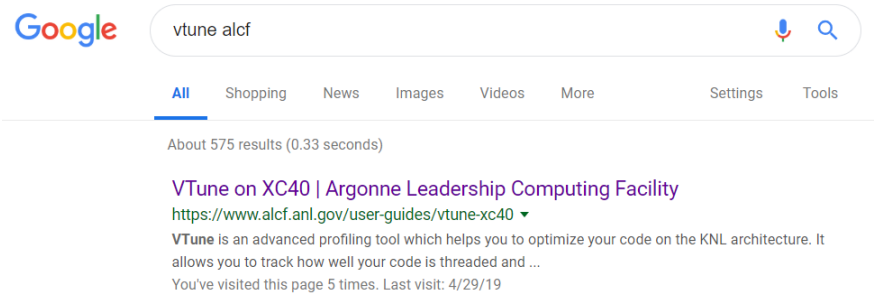
Python Support

Running on Theta

- Cray systems (such as Theta) use aprun instead of mpirun
  - No SPMD notation
    - mpirun -n 1 amplxe-cl -c hotspots ./exe : -n <N-1> ./exe
  - Use $PE_RANK in a bash script instead
    - If $PE_RANK==0 amplxe-cl -c hotspots ./exe; else …
  - PMI_NO_FORK
- Disable darshan
- Dynamic Linking

amplxe.qsub Script

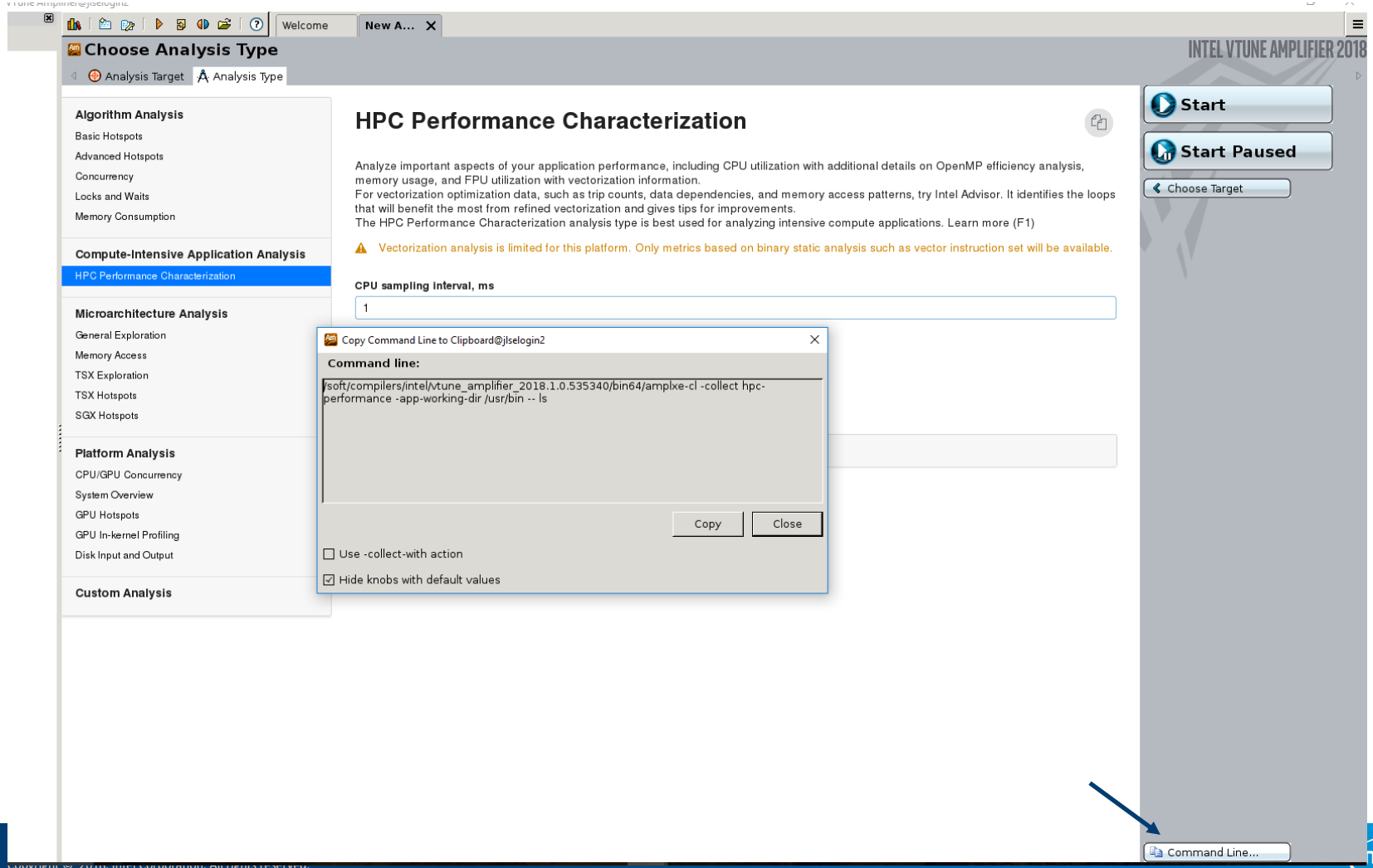
- Copy and customize the script from /soft/perftools/intel/vtune/amplxe.qsub
- All-in-one script for profiling
  - Job size - ranks, threads, hyperthreads, affinity
  - **Attach to a single, multiple or all ranks**
  - Binary as arg#1, input as arg#2
    - qsub amplxe.qsub ./your_exe arg1 arg2 …
  - Binary and source search directory locations
  - Timestamp + binary name as result directory
  - Save cobalt job files to result directory



Viewing the result

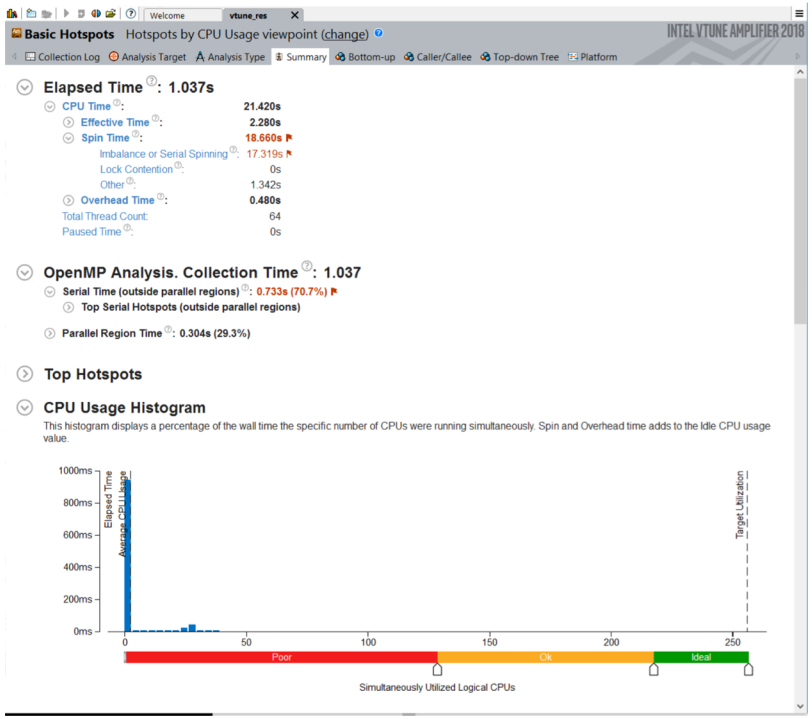
- Text file reports:
  - amplxe-cl -help report — How do I create a text report?
  - amplxe-cl -help report hotspots — What can I change
  - amplxe-cl -R hotspots -r ./res_dir -column=? Which columns are available?
  - Ex: Report top 5% of loops, Total time and L2 Cache hit rates
  - amplxe-cl -R hotspots -loops-only \

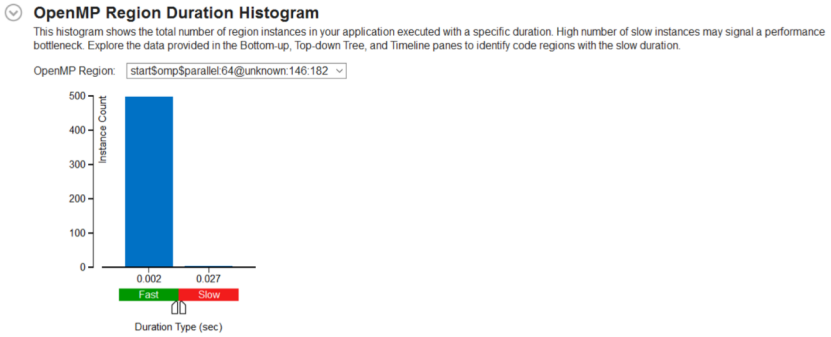
    -limit=5 -column="L2_CACHE_HIT, Time Self (%)"
- Vtune GUI
  - amplxe-gui



Hotspots analysis for nbody demo

- qsub amplxe.qsub ./your_exe ./inputs/inp



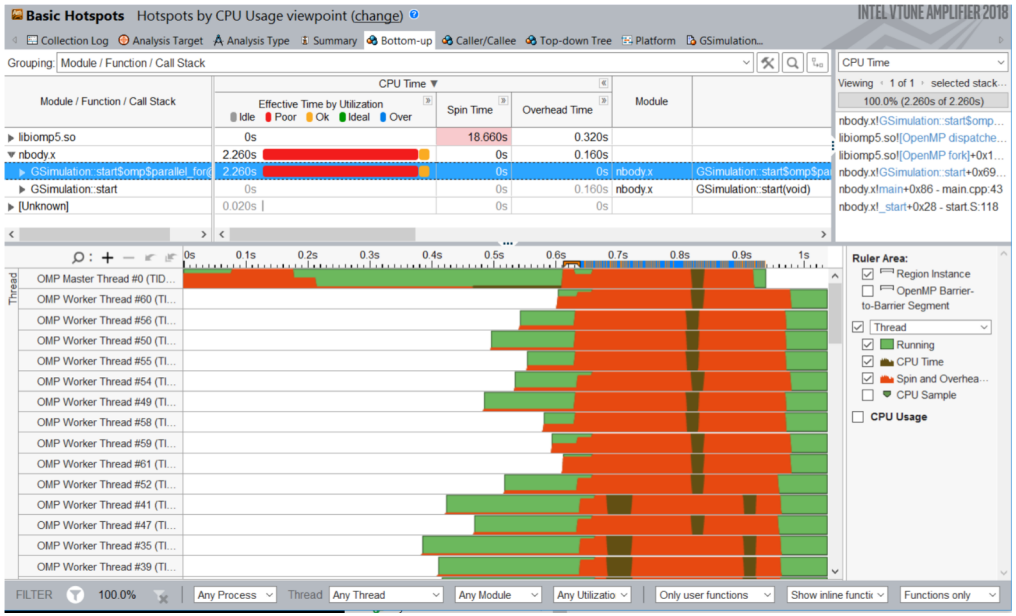


Lots of spin time indicate issues with load balance and synchronization

Given the short OpenMP region duration it is likely we do not have sufficient work per thread

Let's look a the timeline for each thread to understand things better…

Bottom-up Hotspots view

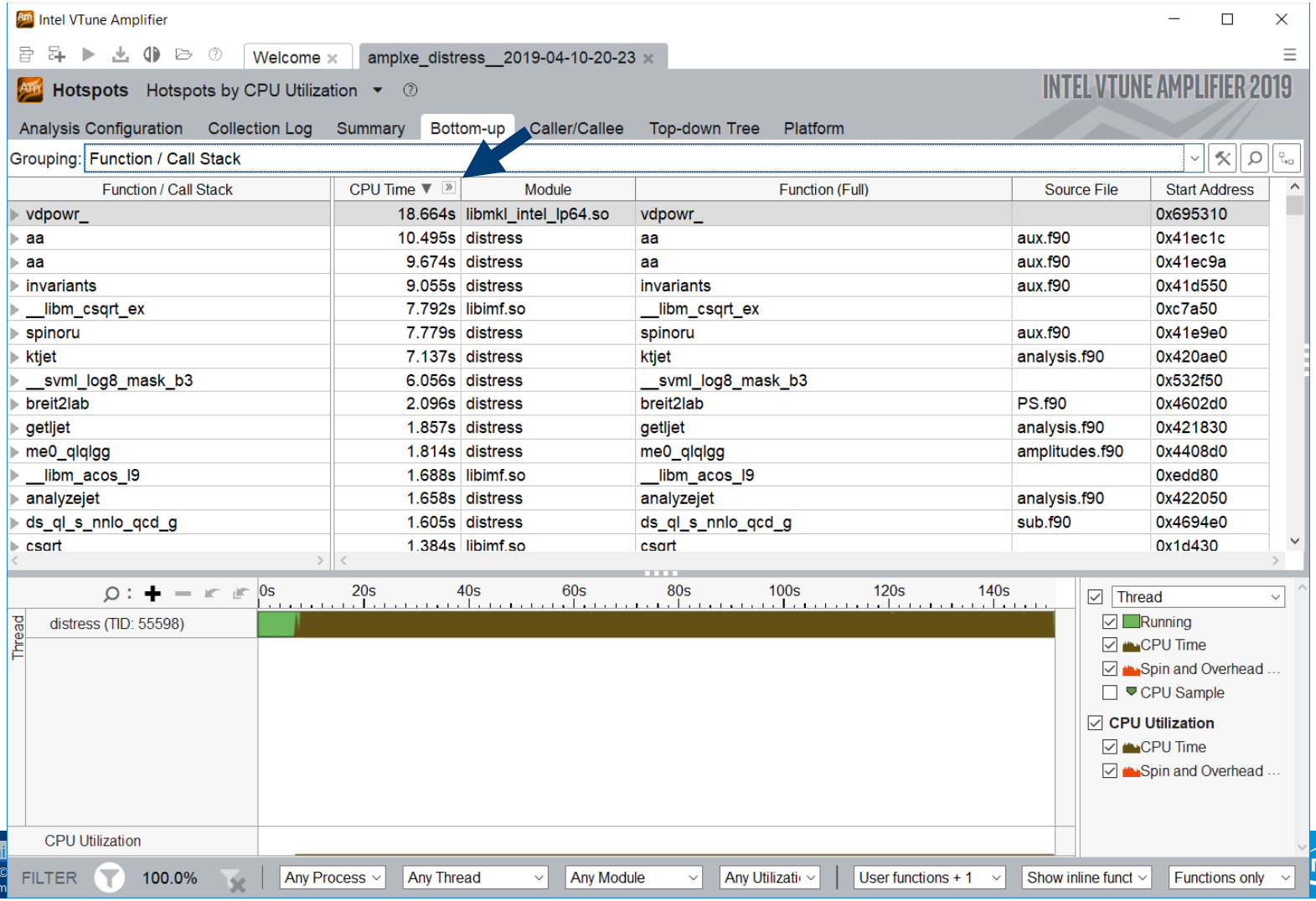


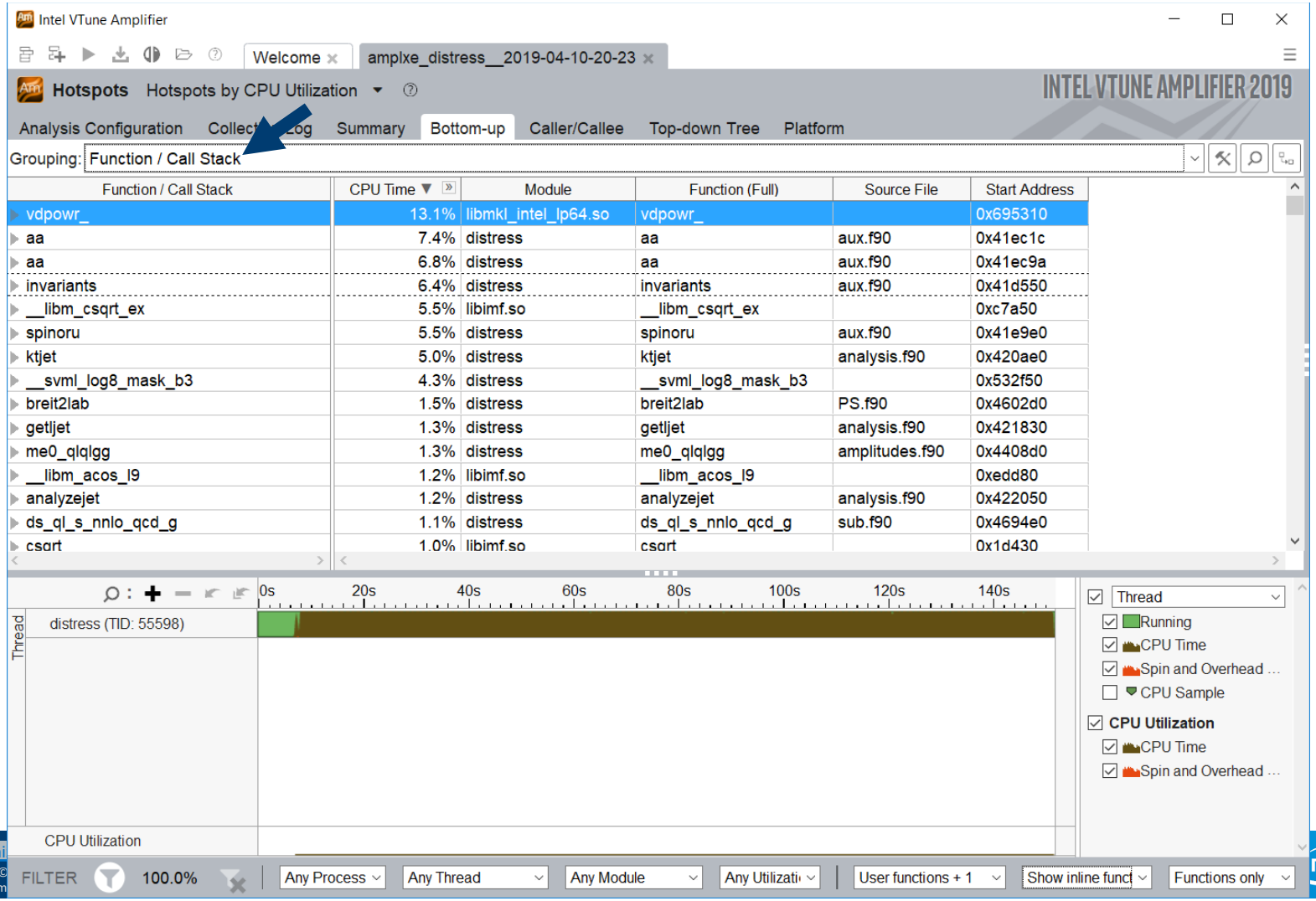
There is not enough work per thread in this particular example.

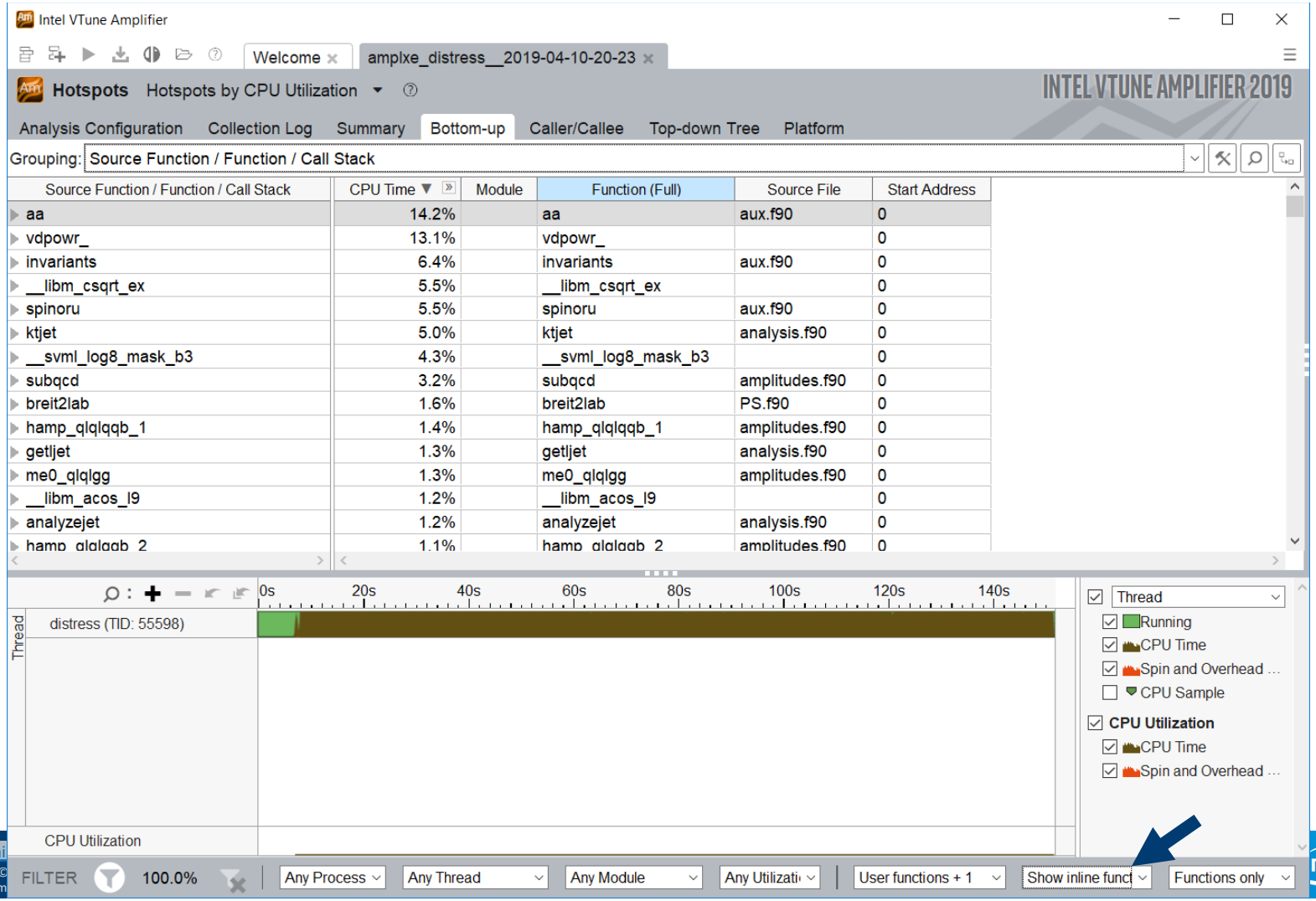
Double click on line to access source and assembly.

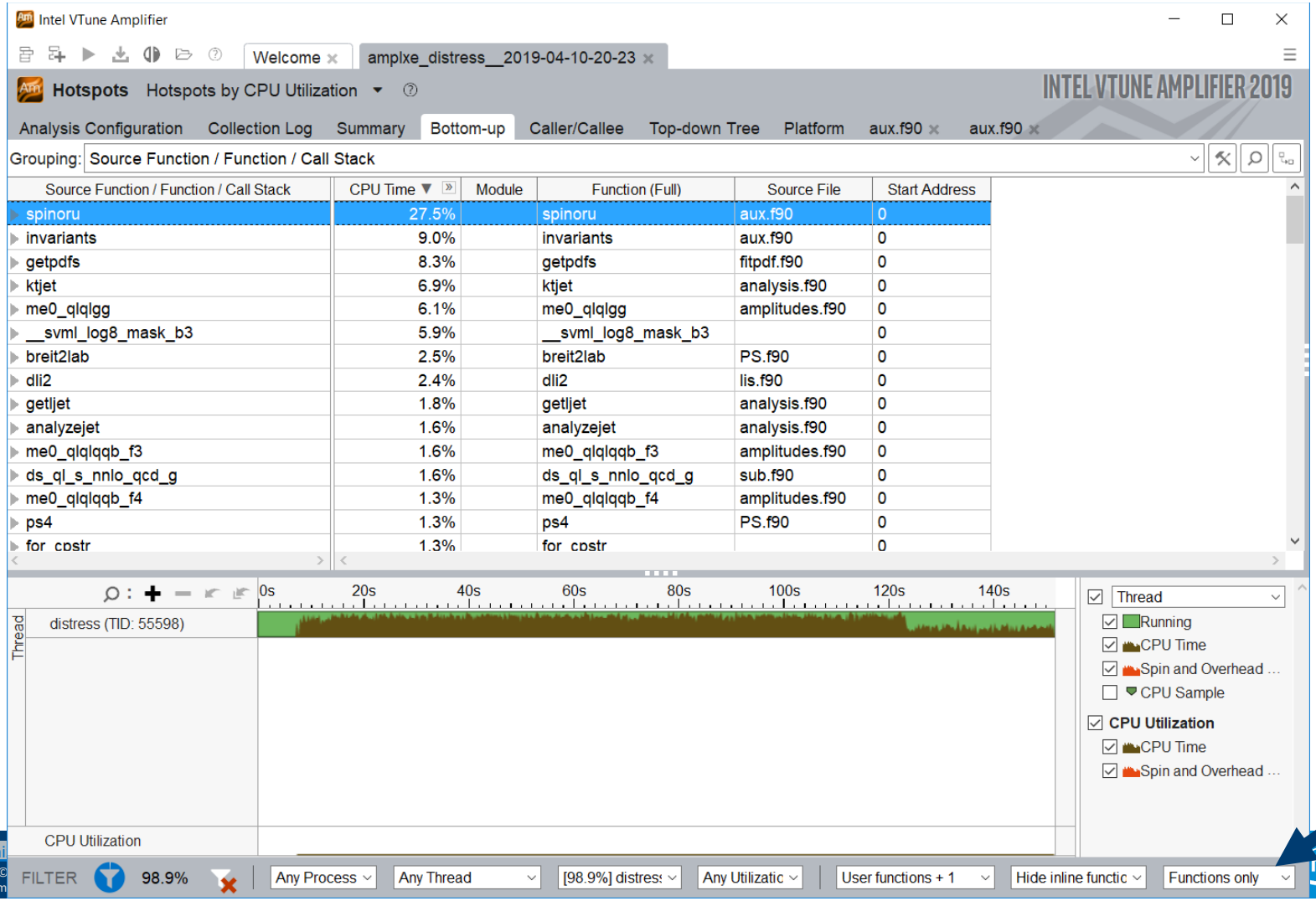
Notice the filtering options at the bottom, which allow customization of this view.

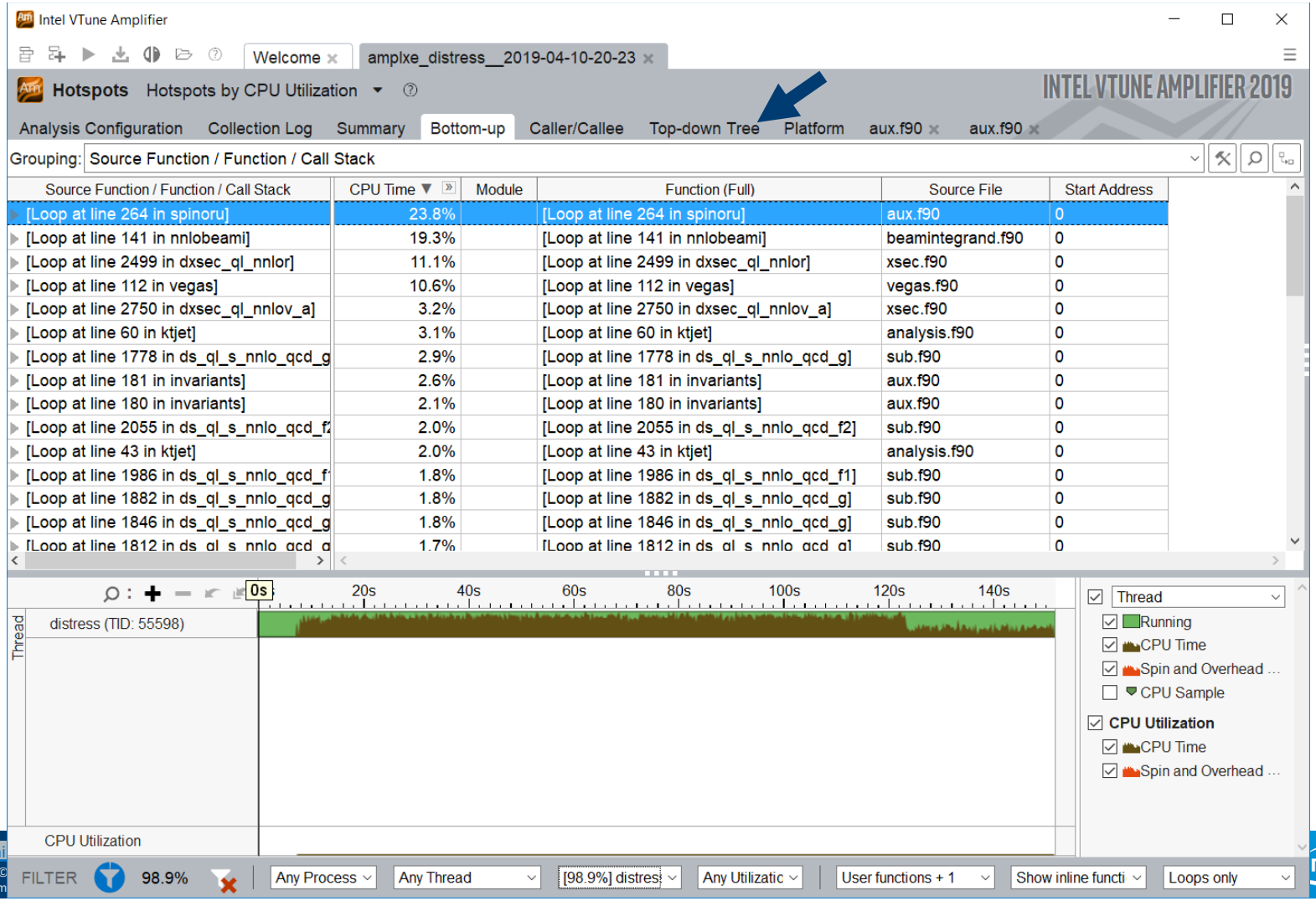
Next steps would include additional analysis to continue the optimization process.

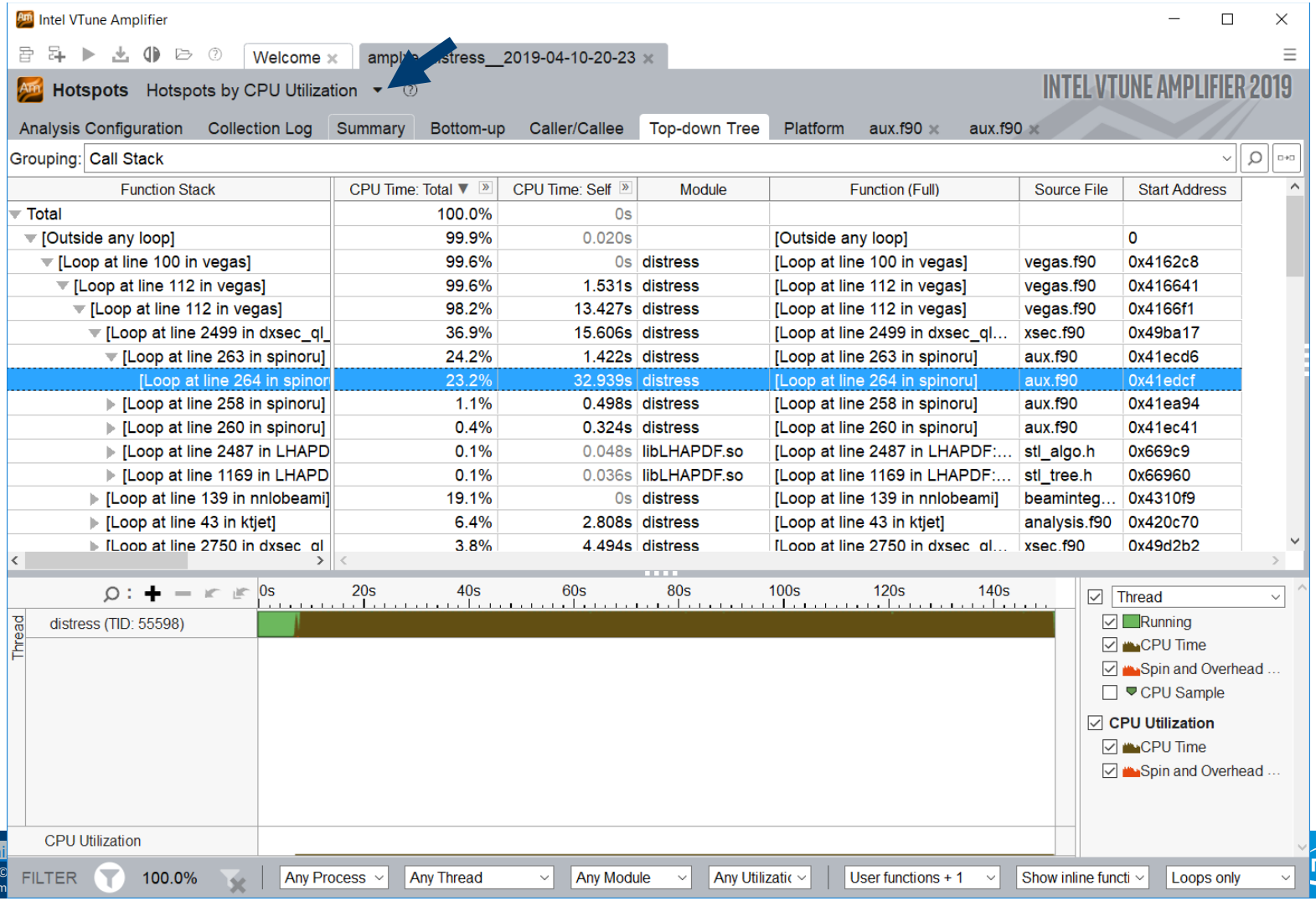

Intel VTune Amplifier
Welcome
amplxe_distress__2019-04-10-20-23
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Grouping: Function / Call Stack
Function / Call Stack | CPU Time | Module | Function (Full) | Source File | Start Address
vdpowr_ | 18.664s | libmkl_intel_lp64.so | vdpowr_ | | 0x695310
aa | 10.495s | distress | aa | aux.f90 | 0x41ec1c
aa | 9.674s | distress | aa | aux.f90 | 0x41ec9a
invariants | 9.055s | distress | invariants | aux.f90 | 0x41d550
__libm_csqrt_ex | 7.792s | libimf.so | __libm_csqrt_ex | | 0xc7a50
spinoru | 7.779s | distress | spinoru | aux.f90 | 0x41e9e0
ktjet | 7.137s | distress | ktjet | analysis.f90 | 0x420ae0
__svml_log8_mask_b3 | 6.056s | distress | __svml_log8_mask_b3 | | 0x532f50
breit2lab | 2.096s | distress | breit2lab | PS.f90 | 0x4602d0
getljet | 1.857s | distress | getljet | analysis.f90 | 0x421830
me0_qlqlgg | 1.814s | distress | me0_qlqlgg | amplitudes.f90 | 0x4408d0
__libm_acos_l9 | 1.688s | libimf.so | __libm_acos_l9 | | 0xedd80
analyzejet | 1.658s | distress | analyzejet | analysis.f90 | 0x422050
ds_ql_s_nnlo_qcd_g | 1.605s | distress | ds_ql_s_nnlo_qcd_g | sub.f90 | 0x4694e0
csqrt | 1.384s | libimf.so | csqrt | | 0x1d430
0s 20s 40s 60s 80s 100s 120s 140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER
100.0%
Any Process
Any Thread
Any Module
Any Utilizati
User functions + 1
Show inline funct
Functions only

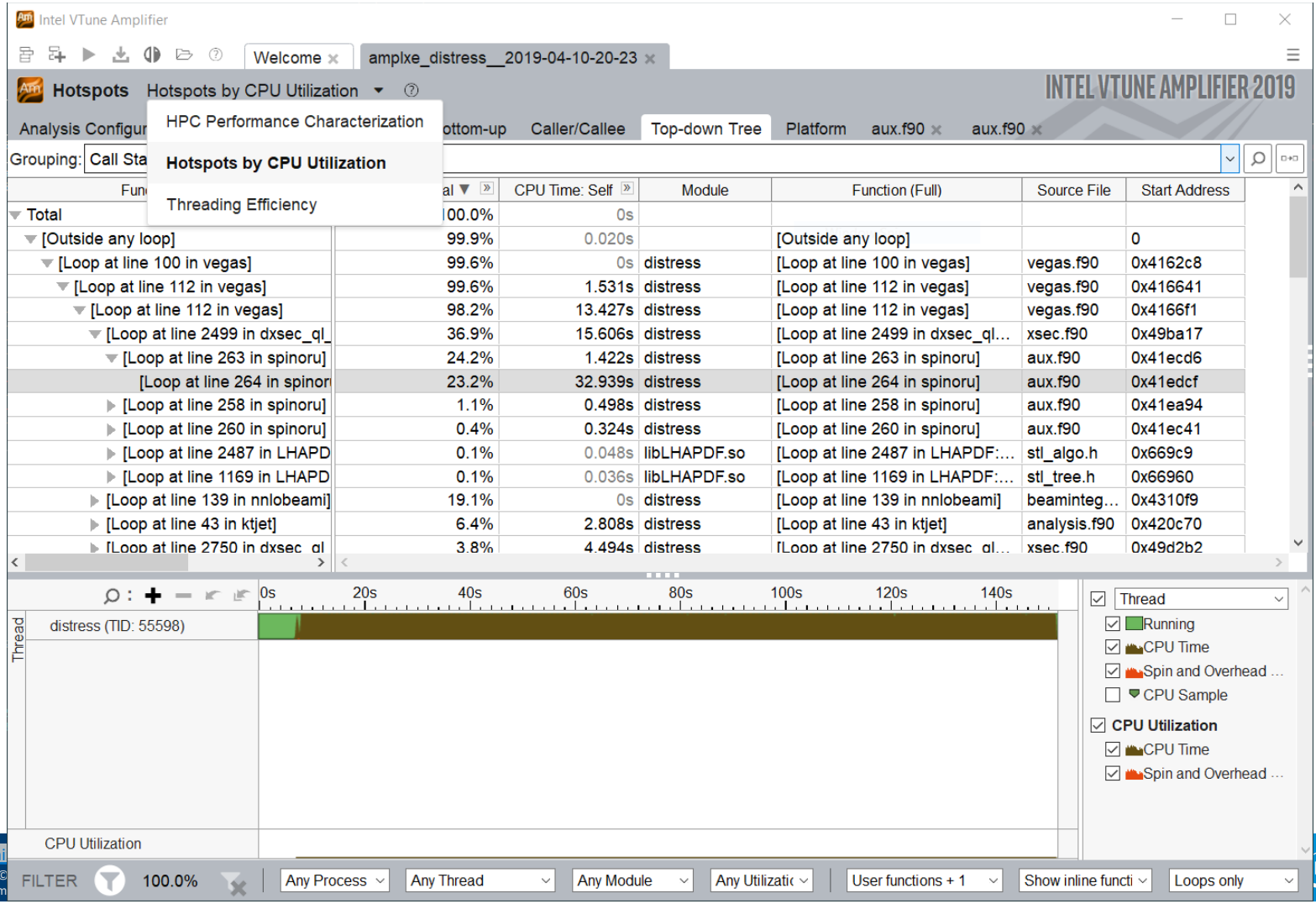

Intel VTune Amplifier
Welcome
amplxe_distress__2019-04-10-20-23
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collect
Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Grouping: Function / Call Stack
Function / Call Stack | CPU Time | Module | Function (Full) | Source File | Start Address
vdpowr_ | 13.1% | libmkl_intel_lp64.so | vdpowr_ | | 0x695310
aa | 7.4% | distress | aa | aux.f90 | 0x41ec1c
aa | 6.8% | distress | aa | aux.f90 | 0x41ec9a
invariants | 6.4% | distress | invariants | aux.f90 | 0x41d550
__libm_csqrt_ex | 5.5% | libimf.so | __libm_csqrt_ex | | 0xc7a50
spinoru | 5.5% | distress | spinoru | aux.f90 | 0x41e9e0
ktjet | 5.0% | distress | ktjet | analysis.f90 | 0x420ae0
__svml_log8_mask_b3 | 4.3% | distress | __svml_log8_mask_b3 | | 0x532f50
breit2lab | 1.5% | distress | breit2lab | PS.f90 | 0x4602d0
getljet | 1.3% | distress | getljet | analysis.f90 | 0x421830
me0_qlqlgg | 1.3% | distress | me0_qlqlgg | amplitudes.f90 | 0x4408d0
__libm_acos_l9 | 1.2% | libimf.so | __libm_acos_l9 | | 0xedd80
analyzejet | 1.2% | distress | analyzejet | analysis.f90 | 0x422050
ds_ql_s_nnlo_qcd_g | 1.1% | distress | ds_ql_s_nnlo_qcd_g | sub.f90 | 0x4694e0
csqrt | 1.0% | libimf.so | csqrt | | 0x1d430
0s 20s 40s 60s 80s 100s 120s 140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER
100.0%
Any Process
Any Thread
Any Module
Any Utilizati
User functions + 1
Show inline funct
Functions only

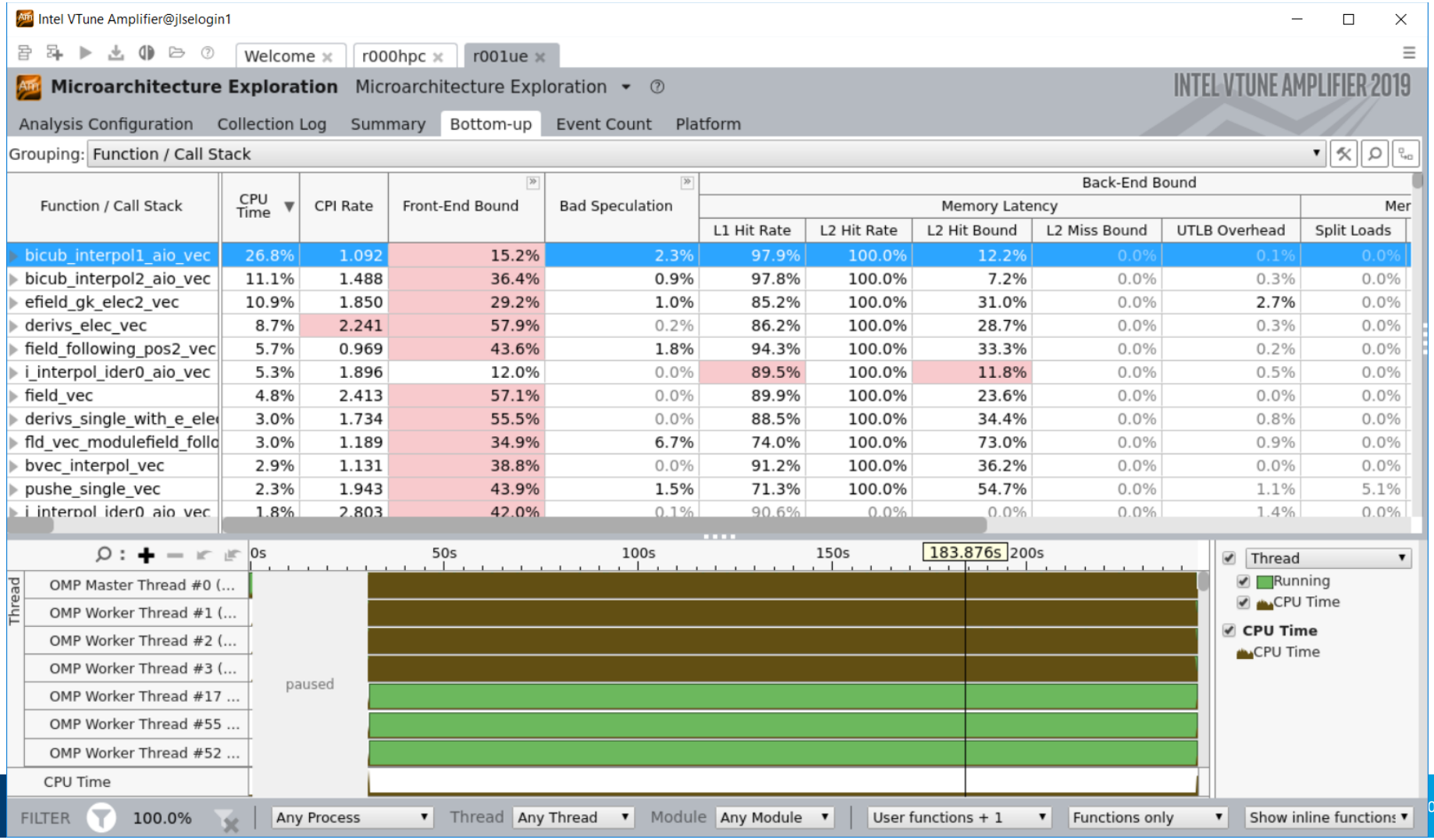

Intel VTune Amplifier
Welcome
amplxe_distress__2019-04-10-20-23
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
Grouping: Source Function / Function / Call Stack
Source Function / Function / Call Stack
CPU Time
Module
Function (Full)
Source File
Start Address
aa 14.2% aa aux.f90 0
vdpowr_ 13.1% vdpowr_ 0
invariants 6.4% invariants aux.f90 0
__libm_csqrt_ex 5.5% __libm_csqrt_ex 0
spinoru 5.5% spinoru aux.f90 0
ktjet 5.0% ktjet analysis.f90 0
__svml_log8_mask_b3 4.3% __svml_log8_mask_b3 0
subqcd 3.2% subqcd amplitudes.f90 0
breit2lab 1.6% breit2lab PS.f90 0
hamp_qlqlqqb_1 1.4% hamp_qlqlqqb_1 amplitudes.f90 0
getljet 1.3% getljet analysis.f90 0
me0_qlqlgg 1.3% me0_qlqlgg amplitudes.f90 0
__libm_acos_l9 1.2% __libm_acos_l9 0
analyzejet 1.2% analyzejet analysis.f90 0
0s 20s 40s 60s 80s 100s 120s 140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER
100.0%
Any Process
Any Thread
Any Module
Any Utilizati
User functions + 1
Show inline func
Functions only


Intel VTune Amplifier
Welcome
amplxe_distress__2019-04-10-20-23
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
aux.f90
aux.f90
Grouping: Source Function / Function / Call Stack
Source Function / Function / Call Stack
CPU Time
Module
Function (Full)
Source File
Start Address
spinoru 27.5% spinoru aux.f90 0
invariants 9.0% invariants aux.f90 0
getpdfs 8.3% getpdfs fitpdf.f90 0
ktjet 6.9% ktjet analysis.f90 0
me0_qlqlgg 6.1% me0_qlqlgg amplitudes.f90 0
__svml_log8_mask_b3 5.9% __svml_log8_mask_b3 0
breit2lab 2.5% breit2lab PS.f90 0
dli2 2.4% dli2 lis.f90 0
getljet 1.8% getljet analysis.f90 0
analyzejet 1.6% analyzejet analysis.f90 0
me0_qlqlqqb_f3 1.6% me0_qlqlqqb_f3 amplitudes.f90 0
ds_ql_s_nnlo_qcd_g 1.6% ds_ql_s_nnlo_qcd_g sub.f90 0
me0_qlqlqqb_f4 1.3% me0_qlqlqqb_f4 amplitudes.f90 0
ps4 1.3% ps4 PS.f90 0
for_cpstr 1.3% for_cpstr 0
0s 20s 40s 60s 80s 100s 120s 140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER
98.9%
Any Process
Any Thread
[98.9%] distres
Any Utilizatic
User functions + 1
Hide inline functic
Functions only


Intel VTune Amplifier
Welcome
amplxe_distress__2019-04-10-20-23
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
aux.f90
aux.f90
Grouping: Source Function / Function / Call Stack
Source Function / Function / Call Stack
CPU Time
Module
Function (Full)
Source File
Start Address
[Loop at line 264 in spinoru] 23.8% [Loop at line 264 in spinoru] aux.f90 0
[Loop at line 141 in nnlobeami] 19.3% [Loop at line 141 in nnlobeami] beamintegrand.f90 0
[Loop at line 2499 in dxsec_ql_nnlor] 11.1% [Loop at line 2499 in dxsec_ql_nnlor] xsec.f90 0
[Loop at line 112 in vegas] 10.6% [Loop at line 112 in vegas] vegas.f90 0
[Loop at line 2750 in dxsec_ql_nnlov_a] 3.2% [Loop at line 2750 in dxsec_ql_nnlov_a] xsec.f90 0
[Loop at line 60 in ktjet] 3.1% [Loop at line 60 in ktjet] analysis.f90 0
[Loop at line 1778 in ds_ql_s_nnlo_qcd_g 2.9% [Loop at line 1778 in ds_ql_s_nnlo_qcd_g] sub.f90 0
[Loop at line 181 in invariants] 2.6% [Loop at line 181 in invariants] aux.f90 0
[Loop at line 180 in invariants] 2.1% [Loop at line 180 in invariants] aux.f90 0
[Loop at line 2055 in ds_ql_s_nnlo_qcd_f2 2.0% [Loop at line 2055 in ds_ql_s_nnlo_qcd_f2] sub.f90 0
[Loop at line 43 in ktjet] 2.0% [Loop at line 43 in ktjet] analysis.f90 0
[Loop at line 1986 in ds_ql_s_nnlo_qcd_f 1.8% [Loop at line 1986 in ds_ql_s_nnlo_qcd_f1] sub.f90 0
[Loop at line 1882 in ds_ql_s_nnlo_qcd_g 1.8% [Loop at line 1882 in ds_ql_s_nnlo_qcd_g] sub.f90 0
[Loop at line 1846 in ds_ql_s_nnlo_qcd_g 1.8% [Loop at line 1846 in ds_ql_s_nnlo_qcd_g] sub.f90 0
[Loop at line 1812 in ds_ql_s_nnlo_qcd_g 1.7% [Loop at line 1812 in ds_ql_s_nnlo_qcd_g] sub.f90 0
0s
20s
40s
60s
80s
100s
120s
140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER
98.9%
Any Process
Any Thread
[98.9%] distres
Any Utilizatic
User functions + 1
Show inline functi
Loops only


Intel VTune Amplifier
Welcome
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collection Log
Summary
Bottom-up
Caller/Callee
Top-down Tree
Platform
aux.f90
aux.f90
Grouping: Call Stack
Function Stack
CPU Time: Total
CPU Time: Self
Module
Function (Full)
Source File
Start Address
Total 100.0% 0s
[Outside any loop] 99.9% 0.020s [Outside any loop] 0
[Loop at line 100 in vegas] 99.6% 0s distress [Loop at line 100 in vegas] vegas.f90 0x4162c8
[Loop at line 112 in vegas] 99.6% 1.531s distress [Loop at line 112 in vegas] vegas.f90 0x416641
[Loop at line 112 in vegas] 98.2% 13.427s distress [Loop at line 112 in vegas] vegas.f90 0x4166f1
[Loop at line 2499 in dxsec_ql_ 36.9% 15.606s distress [Loop at line 2499 in dxsec_ql... xsec.f90 0x49ba17
[Loop at line 263 in spinoru] 24.2% 1.422s distress [Loop at line 263 in spinoru] aux.f90 0x41ecd6
[Loop at line 264 in spinor 23.2% 32.939s distress [Loop at line 264 in spinoru] aux.f90 0x41edcf
[Loop at line 258 in spinoru] 1.1% 0.498s distress [Loop at line 258 in spinoru] aux.f90 0x41ea94
[Loop at line 260 in spinoru] 0.4% 0.324s distress [Loop at line 260 in spinoru] aux.f90 0x41ec41
[Loop at line 2487 in LHAPD 0.1% 0.048s libLHAPDF.so [Loop at line 2487 in LHAPDF:... stl_algo.h 0x669c9
[Loop at line 1169 in LHAPD 0.1% 0.036s libLHAPDF.so [Loop at line 1169 in LHAPDF:... stl_tree.h 0x66960
[Loop at line 139 in nnlobeami] 19.1% 0s distress [Loop at line 139 in nnlobeami] beaminteg... 0x4310f9
[Loop at line 43 in ktjet] 6.4% 2.808s distress [Loop at line 43 in ktjet] analysis.f90 0x420c70
[Loop at line 2750 in dxsec_ql 3.8% 4.494s distress [Loop at line 2750 in dxsec_ql... xsec.f90 0x49d2b2
0s 20s 40s 60s 80s 100s 120s 140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER 100.0%
Any Process
Any Thread
Any Module
Any Utilizatic
User functions + 1
Show inline functi
Loops only


Intel VTune Amplifier
Welcome
amplxe_distress__2019-04-10-20-23
Hotspots
Hotspots by CPU Utilization
INTEL VTUNE AMPLIFIER 2019
HPC Performance Characterization
Hotspots by CPU Utilization
Threading Efficiency
Analysis Configur
ottom-up
Caller/Callee
Top-down Tree
Platform
aux.f90
aux.f90
Grouping: Call Sta
Fun
al
CPU Time: Self
Module
Function (Full)
Source File
Start Address
Total 100.0% 0s
[Outside any loop] 99.9% 0.020s [Outside any loop] 0
[Loop at line 100 in vegas] 99.6% 0s distress [Loop at line 100 in vegas] vegas.f90 0x4162c8
[Loop at line 112 in vegas] 99.6% 1.531s distress [Loop at line 112 in vegas] vegas.f90 0x416641
[Loop at line 112 in vegas] 98.2% 13.427s distress [Loop at line 112 in vegas] vegas.f90 0x4166f1
[Loop at line 2499 in dxsec_ql_ 36.9% 15.606s distress [Loop at line 2499 in dxsec_ql... xsec.f90 0x49ba17
[Loop at line 263 in spinoru] 24.2% 1.422s distress [Loop at line 263 in spinoru] aux.f90 0x41ecd6
[Loop at line 264 in spinor 23.2% 32.939s distress [Loop at line 264 in spinoru] aux.f90 0x41edcf
[Loop at line 258 in spinoru] 1.1% 0.498s distress [Loop at line 258 in spinoru] aux.f90 0x41ea94
[Loop at line 260 in spinoru] 0.4% 0.324s distress [Loop at line 260 in spinoru] aux.f90 0x41ec41
[Loop at line 2487 in LHAPD 0.1% 0.048s libLHAPDF.so [Loop at line 2487 in LHAPDF:... stl_algo.h 0x669c9
[Loop at line 1169 in LHAPD 0.1% 0.036s libLHAPDF.so [Loop at line 1169 in LHAPDF:... stl_tree.h 0x66960
[Loop at line 139 in nnlobeami] 19.1% 0s distress [Loop at line 139 in nnlobeami] beaminteg... 0x4310f9
[Loop at line 43 in ktjet] 6.4% 2.808s distress [Loop at line 43 in ktjet] analysis.f90 0x420c70
[Loop at line 2750 in dxsec_ql 3.8% 4.494s distress [Loop at line 2750 in dxsec_ql... xsec.f90 0x49d2b2
0s
20s
40s
60s
80s
100s
120s
140s
Thread
distress (TID: 55598)
CPU Utilization
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
CPU Time
Spin and Overhead ...
FILTER
100.0%
Any Process
Any Thread
Any Module
Any Utilizatic
User functions + 1
Show inline functi
Loops only


Intel VTune Amplifier@jlselogin1
Welcome
r000hpc
r001ue
Microarchitecture Exploration
Microarchitecture Exploration
INTEL VTUNE AMPLIFIER 2019
Analysis Configuration
Collection Log
Summary
Bottom-up
Event Count
Platform
Grouping: Function / Call Stack
Function / Call Stack
CPU Time
CPI Rate
Front-End Bound
Bad Speculation
Back-End Bound
Memory Latency
L1 Hit Rate
L2 Hit Rate
L2 Hit Bound
L2 Miss Bound
UTLB Overhead
Split Loads
bicub_interpol1_aio_vec 26.8% 1.092 15.2% 2.3% 97.9% 100.0% 12.2% 0.0% 0.1% 0.0%
bicub_interpol2_aio_vec 11.1% 1.488 36.4% 0.9% 97.8% 100.0% 7.2% 0.0% 0.3% 0.0%
efield_gk_elec2_vec 10.9% 1.850 29.2% 1.0% 85.2% 100.0% 31.0% 0.0% 2.7% 0.0%
derivs_elec_vec 8.7% 2.241 57.9% 0.2% 86.2% 100.0% 28.7% 0.0% 0.3% 0.0%
field_following_pos2_vec 5.7% 0.969 43.6% 1.8% 94.3% 100.0% 33.3% 0.0% 0.2% 0.0%
i_interpol_ider0_aio_vec 5.3% 1.896 12.0% 0.0% 89.5% 100.0% 11.8% 0.0% 0.5% 0.0%
field_vec 4.8% 2.413 57.1% 0.0% 89.9% 100.0% 23.6% 0.0% 0.0% 0.0%
derivs_single_with_e_ele 3.0% 1.734 55.5% 0.0% 88.5% 100.0% 34.4% 0.0% 0.8% 0.0%
fld_vec_modulefield_follo 3.0% 1.189 34.9% 6.7% 74.0% 100.0% 73.0% 0.0% 0.9% 0.0%
bvec_interpol_vec 2.9% 1.131 38.8% 0.0% 91.2% 100.0% 36.2% 0.0% 0.0% 0.0%
pushe_single_vec 2.3% 1.943 43.9% 1.5% 71.3% 100.0% 54.7% 0.0% 1.1% 5.1%
i_interpol_ider0_aio_vec 1.8% 2.803 42.0% 0.1% 90.6% 0.0% 0.0% 0.0% 1.4% 0.0%
0s
50s
100s
150s
183.876s
200s
Thread
OMP Master Thread #0 (...
OMP Worker Thread #1 (...
OMP Worker Thread #2 (...
OMP Worker Thread #3 (...
OMP Worker Thread #17 ...
OMP Worker Thread #55 ...
OMP Worker Thread #52 ...
CPU Time
paused
Thread
Running
CPU Time
CPU Time
CPU Time
FILTER
100.0%
Any Process
Thread
Any Thread
Module
Any Module
User functions + 1
Functions only
Show inline functions

Profiling Python

Python

Profiling Python is straightforward in VTune™ Amplifier, as long as one does the following:

- The “application” should be the full path to the python interpreter used
- The python code should be passed as “arguments” to the “application”

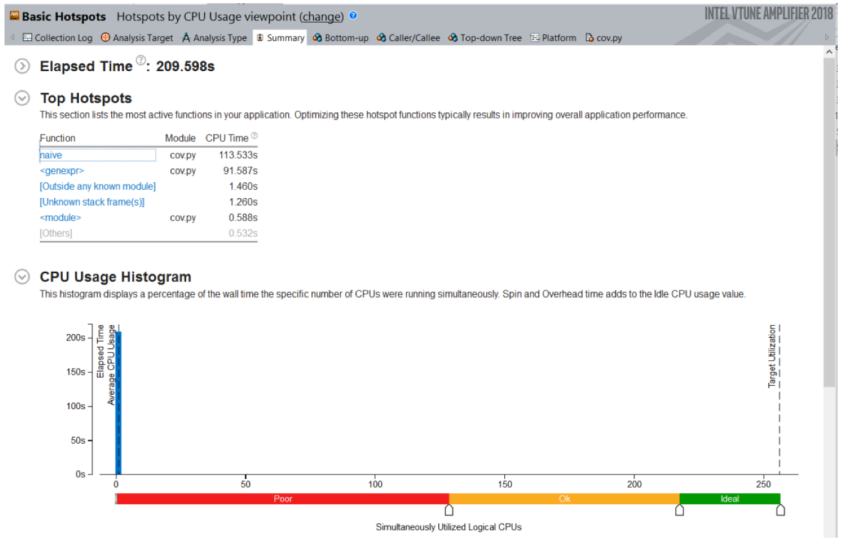
In Theta this would look like this:

```
aprun -n 1 -N 1 amplxe-cl -c hotspots -r res_dir \
                -- /usr/bin/python3 mycode.py myarguments
```

Or use amplxe.qsub script as a starting point

Simple Python Example on Theta

```
aprun -n 1 -N 1 amplxe-cl -c hotspots -r vt_pytest \
                          -- /usr/bin/python ./cov.py naive 100 1000
```



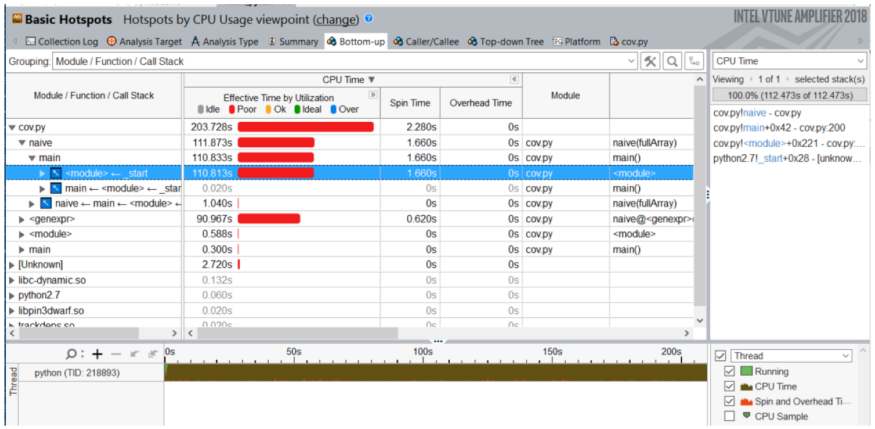
Naïve implementation of the calculation of a covariance matrix

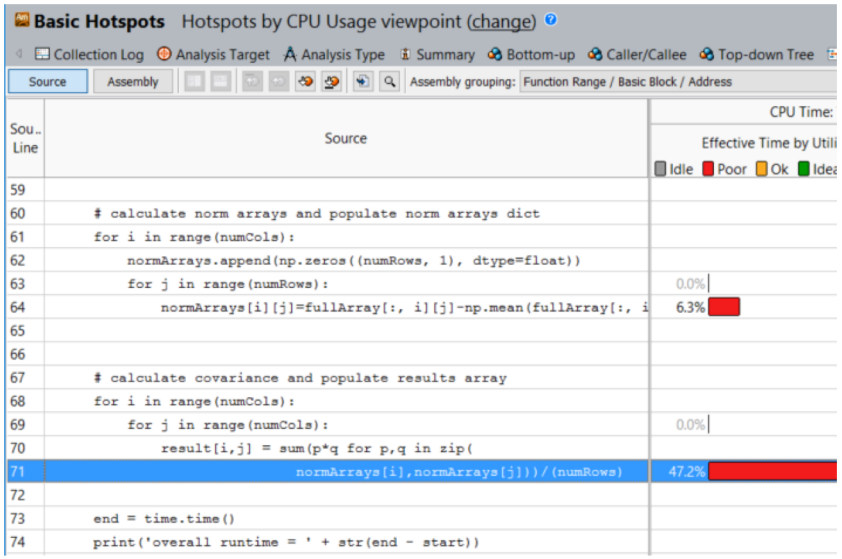
Summary shows:

- Single thread execution
- Top function is “naive”

Click on top function to go to Bottom-up view

Bottom-up View and Source Code





Inefficient array multiplication found quickly
We could use numpy to improve on this

Note that for mixed Python/C code a Top-Down view can often be helpful to drill down into the C kernels

Python & ML/DL

Vtune overheads scale with number of threads

First, ask yourself what is exactly that you want to find out?

- MPI call cost distribution can be acquired running aps at scale
- MKL call information will be given if you do MKL_VERBOSE=1 python job.py

Managing overheads

ITT_NOTIFY

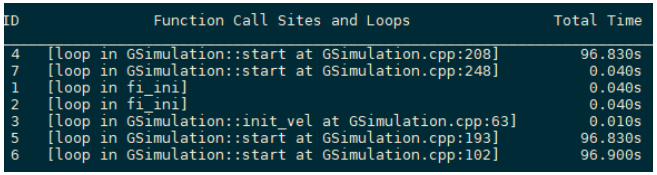
- Instrument your code with itt_pause() and itt_resume()
  - Start, stop, detach and finalize your collection
  - Great when
    - Interested in specific region of code
    - Difficult to adjust overall runtime of your application
- Available in Fortran, C/C++, and Python
- Works with APS, Vtune, and Advisor

APS

- aps is meant to scale so has least amount of built-in knobs
  - Does work with itt_notify
- E.x: Profile the training portion of ML app, excluding setup and finalization
  - Insert itt_resume() before training step
  - Insert itt_detach() after training step
  - aprun aps --start-paused python ./app --arg1 --arg2
    - Run paused, resume before training, detach and finalize after training
      - Job can be killed after itt_detach() is called

Advisor

- Survey has to be run always
  - Overheads minimal
- Use mark-up-list tripcounts/map/dependencies only for regions of interest
  - advixe-cl -R survey --project-dir ./res …
  - advixe-cl -c tripcounts --mark-up-list=4 …
- --interval=100
- --stop-after=1000 (1000 seconds)



ID	Function Call Sites and Loops	Total Time
4	[loop in GSimulation::start at GSimulation.cpp:208]	96.830s
7	[loop in GSimulation::start at GSimulation.cpp:248]	0.040s
1	[loop in fi_ini]	0.040s
2	[loop in fi_ini]	0.040s
3	[loop in GSimulation::init_vel at GSimulation.cpp:63]	0.010s
5	[loop in GSimulation::start at GSimulation.cpp:193]	96.830s
6	[loop in GSimulation::start at GSimulation.cpp:102]	96.900s

Finalization

This step is inherently serial, takes a long time on KNL

SRC="--search-dir src:=/abs/path/to/source"

BIN="--search-dir bin:=/abs/path/to/binary"

- Advisor
  - aprun advixe-cl -c survey --project-dir test -no-auto-finalize
  - advixe-cl -R survey --project-dir test $SRC $BIN
- Vtune
  - aprun amplxe-cl -c hotspots -r test -no-auto-finalize
  - amplxe-cl -R summary -r test $SRC $BIN

Common Issues

- I don’t see my functions
  - Debug symbols
    - Do a text report, look for “Cannot find” warnings
    - Add those to --search-dir
    - Re-finalize

Common Issues

- My python job hangs
  - Detach collection before your job finishes (--duration, --stop-after, or itt_detach)
  - Reduce number of threads

Remember

Compile with -g and -dynamic

Profile 1 rank and small number of threads - amplxe.qsub/advixe.qsub

Advisor for big picture

Vtune for details

Resources

Product Pages

- https://software.intel.com/sites/products/snapshots/application-snapshot
- https://software.intel.com/en-us/advisor
- https://software.intel.com/en-us/intel-vtune-amplifier-xe

Detailed Articles

- https://software.intel.com/en-us/articles/intel-advisor-on-cray-systems
- https://software.intel.com/en-us/articles/using-intel-advisor-and-vtune-amplifier-with-mpi
- https://software.intel.com/en-us/articles/profiling-python-with-intel-vtune-amplifier-a-covariance-demonstration

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel
Software

When do I use Vtune vs Advisor?

Vtune

- What's my cache hit ratio?
- Which loop/function is consuming most time overall? (bottom-up)
- Am I stalling often? IPC?
- Am I keeping all the threads busy?
- Am I hitting remote NUMA?
- When do I maximize my BW?

Advisor

- Which vector ISA am I using?
- Flow of execution (callstacks)
- What is my vectorization efficiency?
- Can I safely force vectorization?
- Inlining? Data type conversions?
- Roofline

VTune Cheat Sheet

```
Compile with -g -dynamic

amplxe-cl -c hpc-performance -flags -- ./executable
```

- `--result-dir=./vtune_output_dir`
- `--search-dir src:=../src --search-dir bin:=./`
- `-knob enable-stack-collection=true -knob collect-memory-bandwidth=false`
- `-knob analyze-openmp=true`
- `-finalization-mode=deferred if finalization is taking too long on KNL`
- `-data-limit=125 ← in mb`
- `-trace-mpi for MPI metrics on Theta`
- `amplxe-cl -help collect survey`

https://software.intel.com/en-us/vtune-amplifier-help-amplxe-cl-command-syntax

Advisor Cheat Sheet

```
Compile with -g -dynamic

advixe-cl -c roofline/depencies/map -flags -- ./executable
```

- `--project-dir=./advixe_output_dir`
- `--search-dir src:=../src --search-dir bin:=./`
- `-no-auto-finalize if finalization is taking too long on KNL`
- `--interval 1 (sample at 1ms interval, helps for profiling short runs)`
- `-data-limit=125 ← in mb`
- `advixe-cl -help`

https://software.intel.com/en-us/advisor-help-lin-command-line-interface-reference

nbody - ver5_mpi

```
[pvelesko@jlselogin1 ver5_mpi]$ mpirun -n 2 advixe-cl -c survey --project-dir SDL -- ./nbody.x
Intel(R) Advisor Command Line Tool
Copyright (C) 2009-2019 Intel Corporation. All rights reserved.
Intel(R) Advisor Command Line Tool
Copyright (C) 2009-2019 Intel Corporation. All rights reserved.

advixe: Collection started.
advixe: Collection started.
==============================
 Initialize Gravity Simulation
 nPart = 2000; nSteps = 500; dt = 0.1
------------------------------------------------
 s       dt      kenergy     time (s)    GFlops
------------------------------------------------
 50      5       0.1432      1.0434      5.5606
 100     10      2.4341      1.0409      5.5738
 150     15      8.1256      1.0419      5.5685
 200     20      17.877      1.0423      5.5665
 250     25      32.966      1.0401      5.578
 300     30      55.786      1.0409      5.5738
 350     35      91.132      1.0408      5.5747
 400     40      150.12      1.0428      5.5638
 450     45      264.78      1.0413      5.572
 500     50      571.53      1.0405      5.5762

# Number Ranks       : 2
# Number Threads     : 1
# Total Time (s)     : 10.415
# Average Perfomance : 5.5717 +- 0.0046592
==============================
advixe: Collection stopped.
advixe: Collection stopped.
advixe: Opening result 21 % Resolving information for `libc.so.6'
advixe: Warning: Cannot locate debugging information for file `/lib64/libc.so.6'.

advixe: Warning: Cannot locate debugging information for file `/lib64/libc.so.6'.
advixe: Opening result 100 % done
advixe: Preparing frequently used data  0 % done
advixe: Preparing frequently used data 100 % done

Elapsed Time: 10.43s
Total CPU time: 10.42

advixe: Opening result 99 % done
advixe: Preparing frequently used data  0 % done
advixe: Preparing frequently used data 100 % done

Elapsed Time: 10.43s
Total CPU time: 10.42

[pvelesko@jlselogin1 ver5_mpi]$
[pvelesko@jlselogin1 ver5_mpi]$
```