Software

Profiling your PYTHON application with Intel® Vtune™ Amplifier

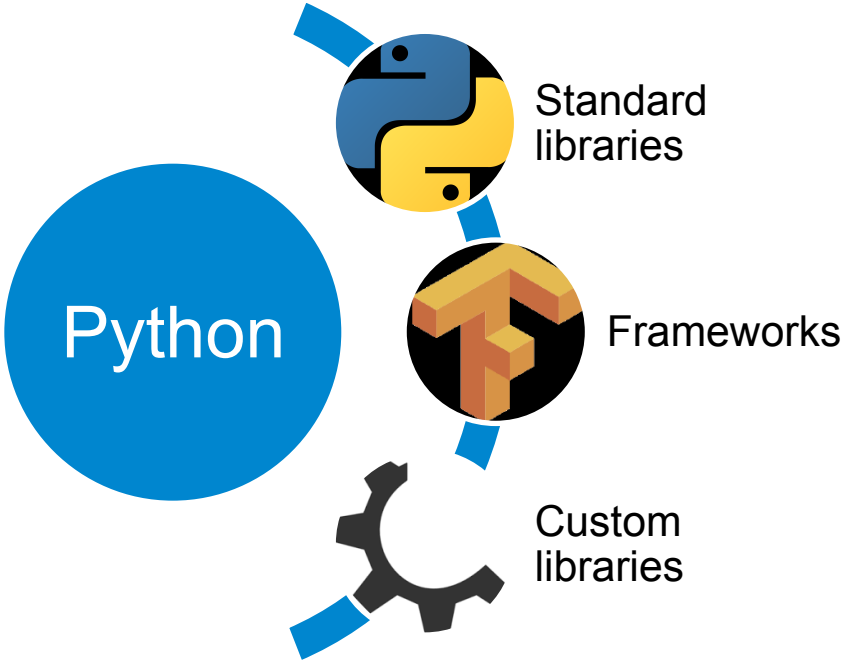
Paulius Velesko

Application Engineer, Intel Corporation

High Performance Python

- Rapid prototyping
- Strong user base
- Lots of libraries

3 Types of Python Workloads



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

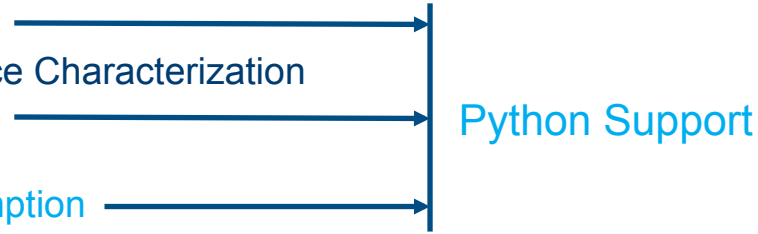
Analyzing code access to shared resources is critical to achieve good performance on multicore and manycore systems

VTune Amplifier takes over where Intel® Advisor left

Predefined Collections

Many available analysis types:

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



Vtune ™ Syntax

source /soft/compilers/intel/vtune_amplifier/amplxe-vars.sh <- sets up the PATH

amplxe-cl <action> <action options> -- <application>

amplxe-cl -c hotspots -- python foo.py

amplxe-cl -h collect

amplxe-cl -h collect hotspots

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

Running an analysis

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

```
$: amplxe-cl -c hotspots --
python mycode.py 10000

$: amplxe-gui &
```

mycode.py

```
import numpy as np
n = sys.argv[1]
arr1 = np.random.rand(n, n)
arr2 = np.random.rand(n, n)

arr3 = arr1 * arr2
arr3 = np.log(arr3)
arr3 = np.exp(arr3)
```

Summary

Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform | workload.py

Elapsed Time: 24.331s

CPU Time: 24.039s
Total Thread Count: 1
Paused Time: 0s

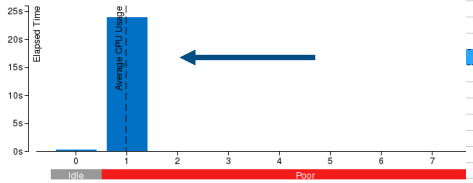
Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically

Function	Module	CPU Time
exp	libm.so.6	8.749s
__ieee754_log_avx	libm.so.6	7.780s
PyUFunc_d_d	umath.so	3.671s
rk_random	mtrand.so	1.150s
__pyx_f_6mtrand_cont0_array	mtrand.so	0.930s
[Others]		1.759s

CPU Usage Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultane



Collection and Platform Info

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

Application Command Line: python "./workload.py"
Operating System: 3.10.0-693.11.6.el7.x86_64 NAME="Red Hat Enterprise Linux Server" VERSION="7.4 (Maipo)" ID="rhel" ID_LIKE="fedora" VARIANT="Server" VARIANT_ID="server" VERSION_ID="7.4" PRETTY_NAME="Red Hat Enterprise Linux Server 7.4 (Maipo)" ANSI_COLOR="0;31" CPE_NAME="cpe:/o:redhat:enterprise_linux:7.4: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=7.4 REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux" REDHAT_SUPPORT_PRODUCT_VERSION="7.4"
Computer Name: jlselogin2
Result Size: 4 MB
Collection start time: 14:10:40 12/06/2018 UTC
Collection stop time: 14:11:04 12/06/2018 UTC
Collector Type: User-mode sampling and tracing

CPU
Name: Intel(R) Xeon(R) E5 Processor code named Jaketown
Frequency: 2.2 GHz
Logical CPU Count: 16

Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform | workload.py

Function	CPU Time: Total	CPU Time: Self	Module
_start	100.0%	0s	python2.7

Callers	CPU Time: Total	CPU Time: Self
__ieee754_log_avx	100.0%	7.780s

Basic Hotspots Hotspots by CPU Usage viewpoint (change)

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform | workload.py

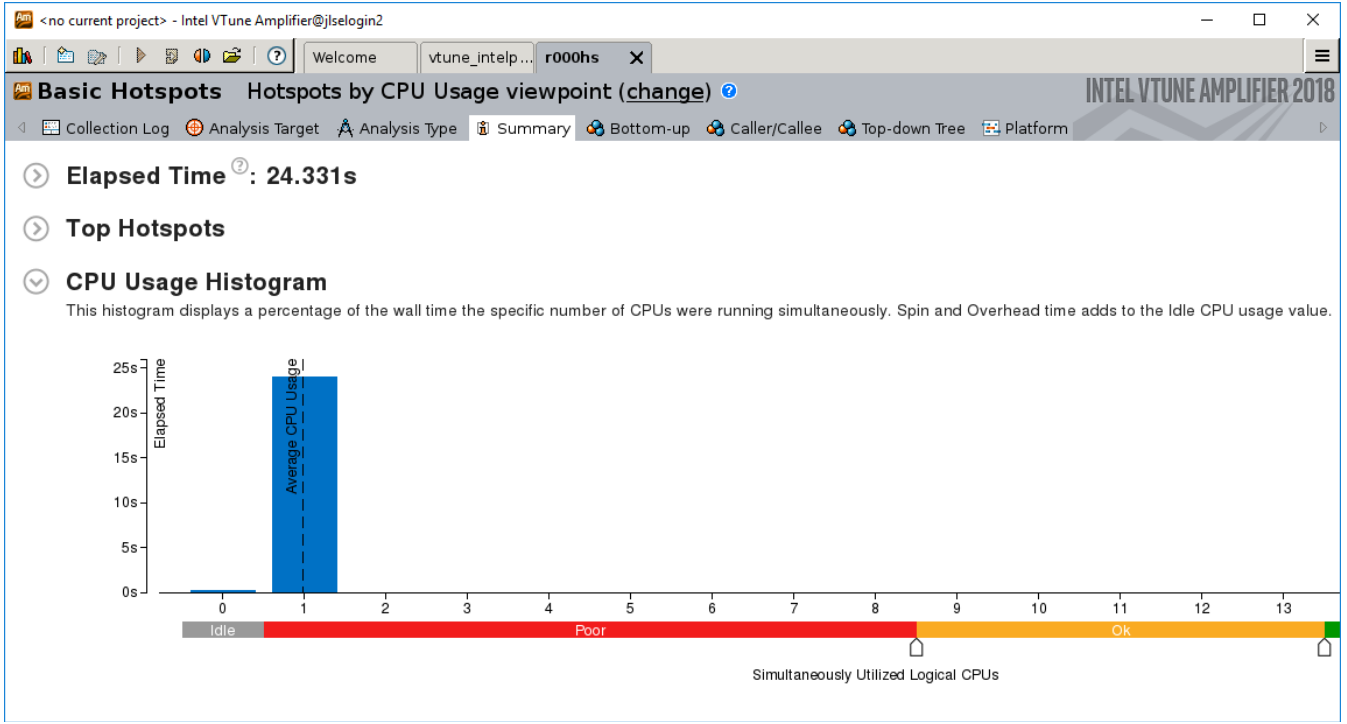
Grouping: Call Stack

Function Stack	CPU Time: Total	CPU Time: Self	Module	Source File	Function (Full)	Start Address
Total	100.0%	0s				
_start	100.0%	0s	python2.7		_start	0x856
__libc_start_main	100.0%	0.010s	libc.so.6		__libc_start_main	0x21b10
<module>	99.7%	0s	workload.py	workload.py	<module>	0x7f1858eb...
PyObject_Call	84.9%	0s	libpython2.7.so.1.0		PyObject_Call	0x48730
ufunc_generic_call	84.9%	0s	umath.so		ufunc_generic_call	0xe6fa0
PyUFunc_GenericFunction	84.9%	0s	umath.so		PyUFunc_GenericFunction	0xe52b0
trivial_two_operand_loop	84.8%	0s	umath.so		trivial_two_operand_loop	0xde960
PyUFunc_d_d	84.4%	3.671s	umath.so		PyUFunc_d_d	0x1b1a0
exp	36.4%	8.749s	libm.so.6		exp	0x25bf0
__ieee754_log_avx	32.4%	7.780s	libm.so.6		__ieee754_log_avx	0x5ba00
log	0.4%	0.100s	libm.so.6		log	0x26330
[Unknown stack frame(s)]	0.3%	0s			[Unknown stack frame(s)]	0
[Unknown stack frame(s)]	0.2%	0s			[Unknown stack frame(s)]	0
__pyx_pw_6mtrand_11RandomState_29rand	12.3%	0s	mtrand.so		__pyx_pw_6mtrand_11RandomStat...	0xf070
PyNumber_Multiply	1.7%	0s	libpython2.7.so.1.0		PyNumber_Multiply	0x46b10
PyEval_CallObjectWithKeywords	0.8%	0s	libpython2.7.so.1.0		PyEval_CallObjectWithKeywords	0xda690
[Unknown stack frame(s)]	0.2%	0s			[Unknown stack frame(s)]	0

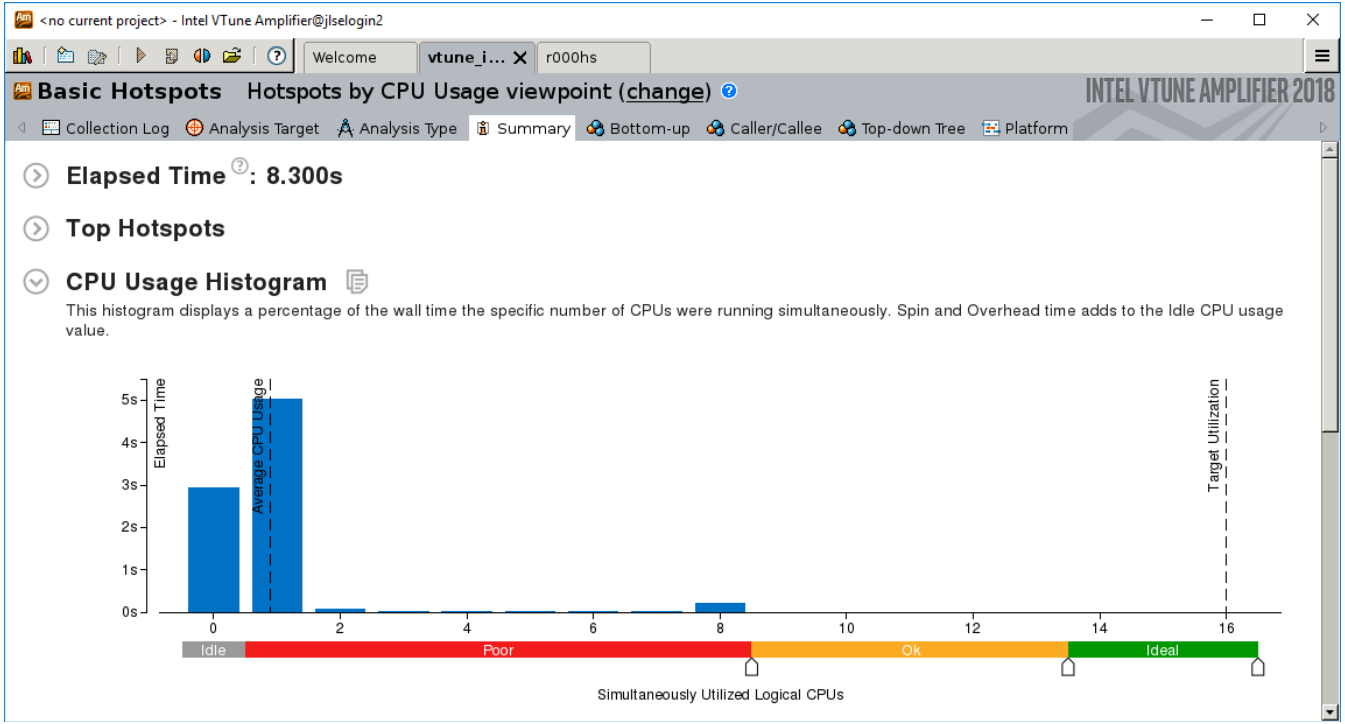
npy_log	0.5%	0.120s	umath.so	npy_log
log	0.4%	0.100s	libm.so.6	log
[Outside any known module	0.2%	0.059s		[Outside any known mod
<module>	0.2%	0s	__init__.py	<module>
memmove	0.1%	0.030s	libc-dynamic.so	memmove
memcpy	0.1%	0s	libc-dynamic.so	memcpy

Callees	CPU Time: Total	CPU Time: Self
__ieee754_log_avx	100.0%	7.780s

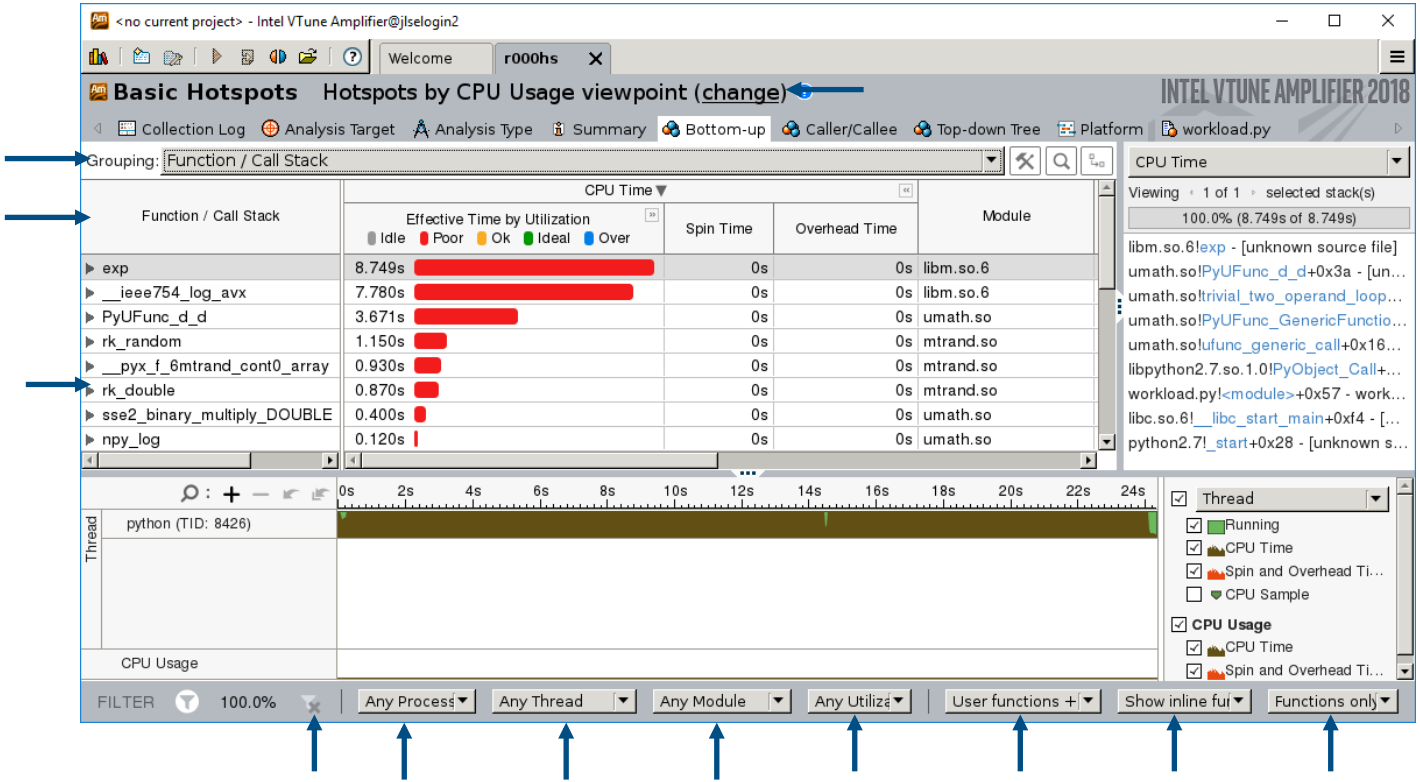
Summary of Anaconda Python



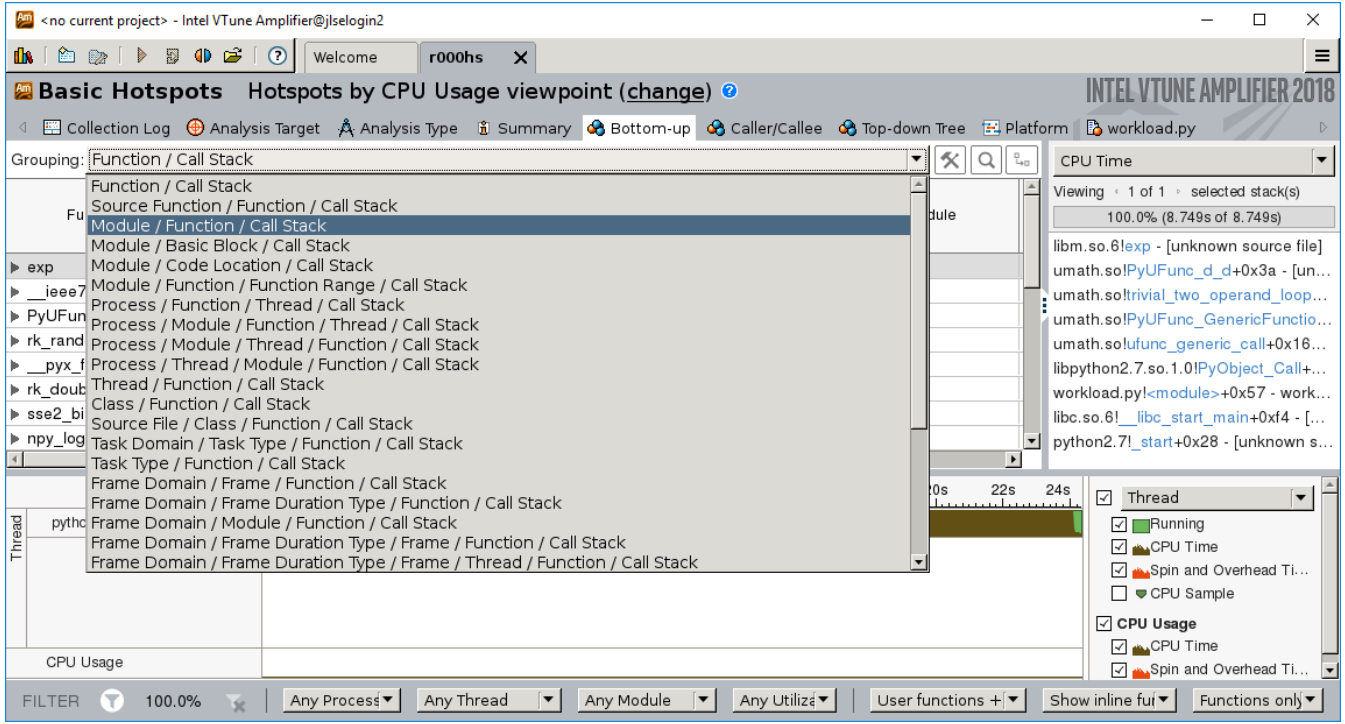
Summary of Intel® Python



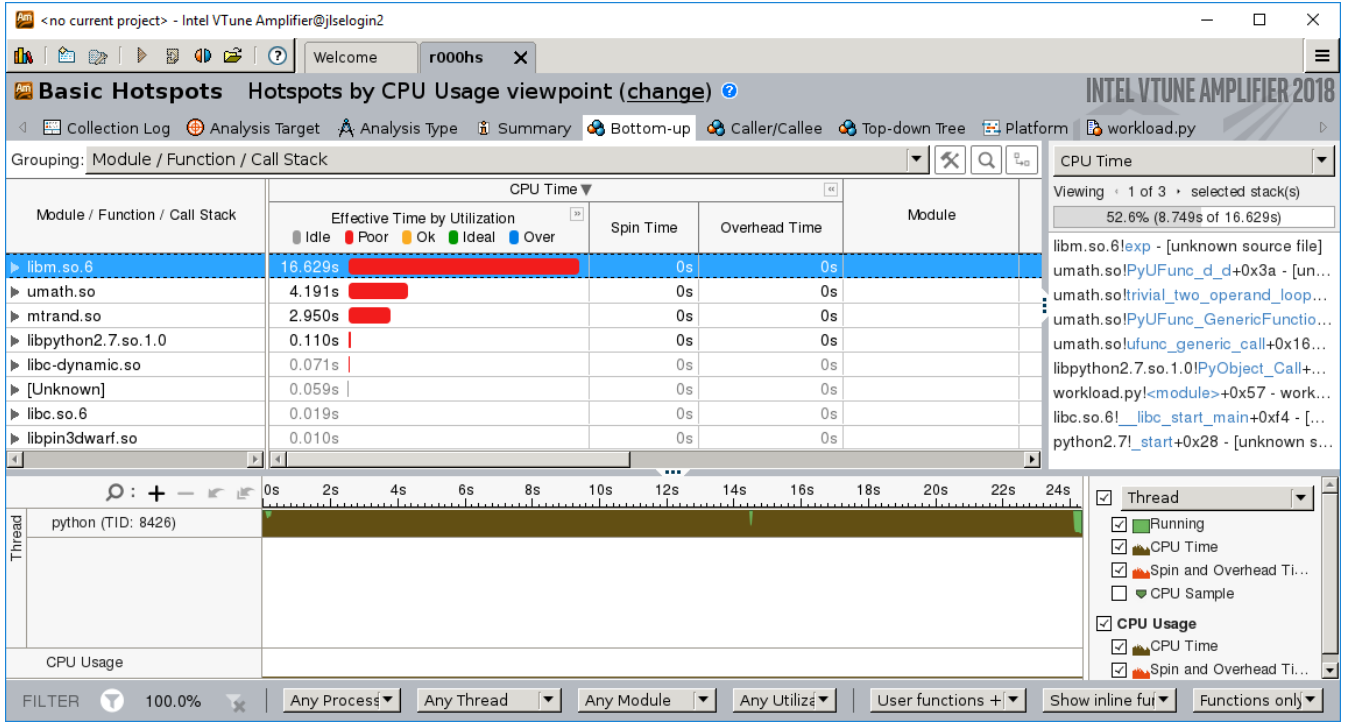
Bottom-up View



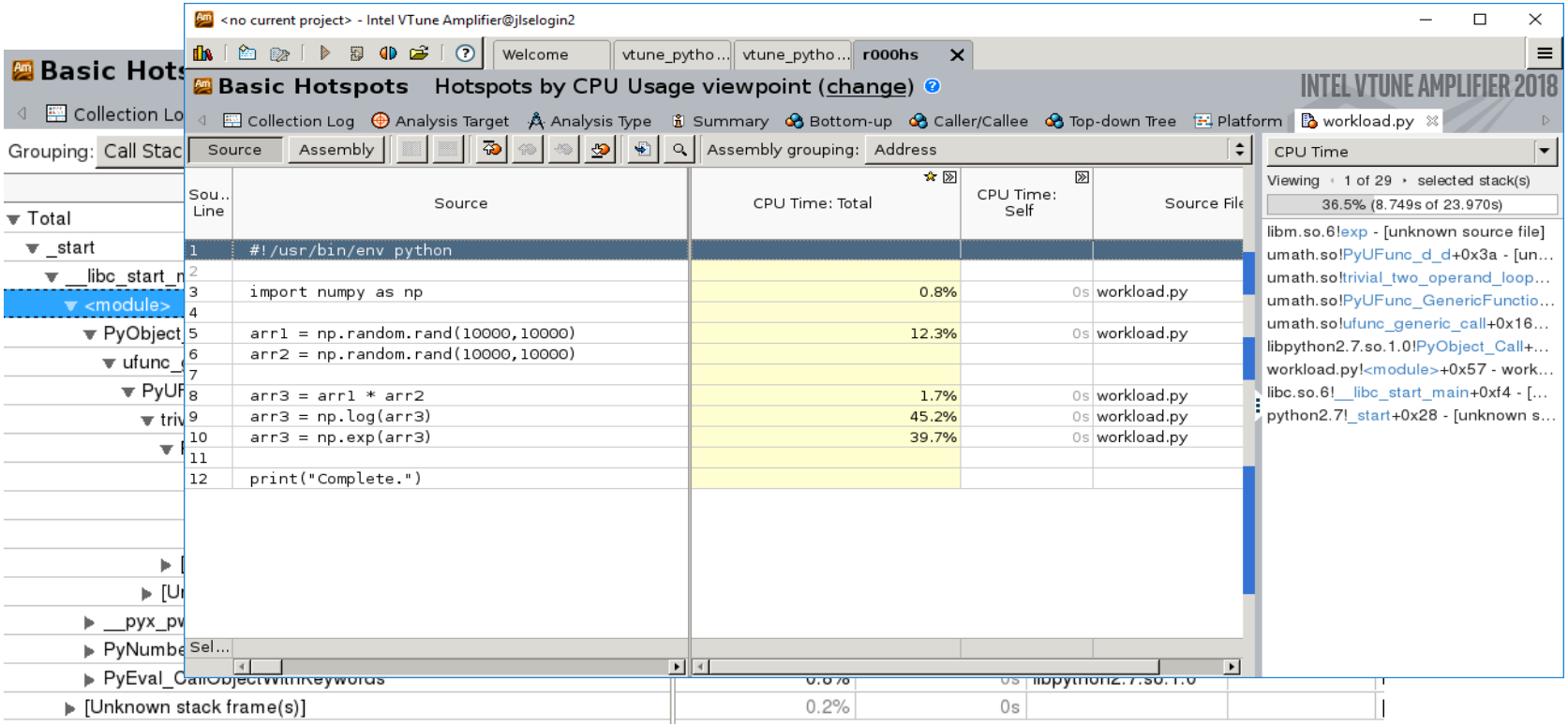
Bottom-up View



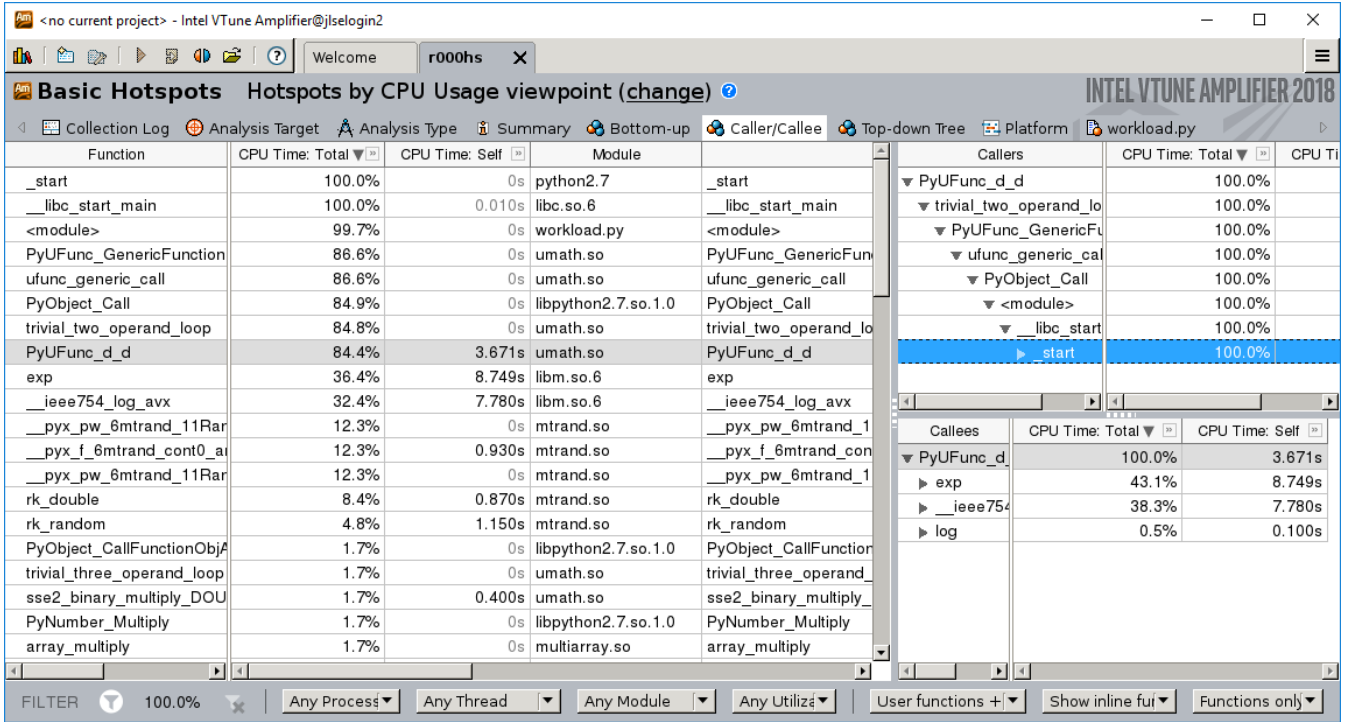
Bottom-up View



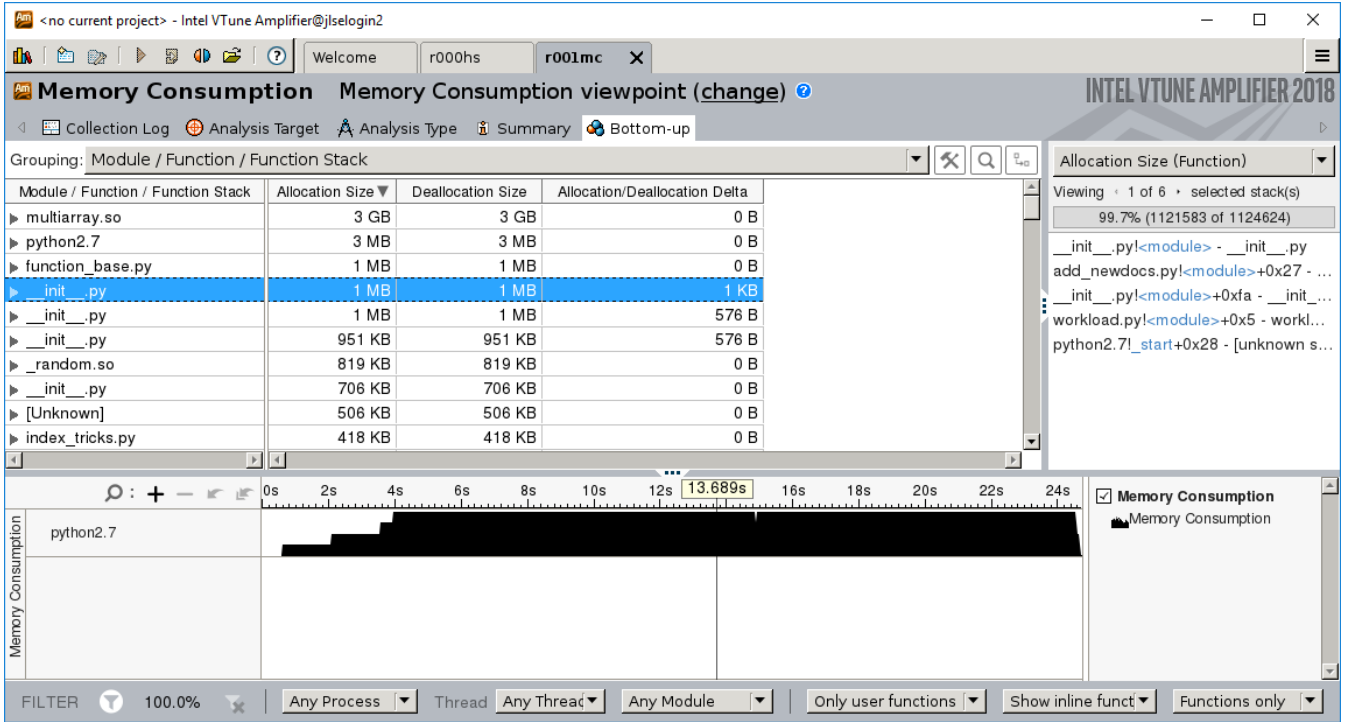
Top-down View



Caller/Callee



Memory Consumption Analysis



Profiling MPI4py

https://github.com/jbornschein/mpi4py-examples

Profiling python MPI jobs

- 2 options
  - Collect on every rank
    - Might be unable to launch enough amplxe-cl instances
  - Collect on select ranks
    - Smaller result

Profiling python MPI jobs - every rank

```
$: mpirun -n 2 \
amplxe-cl -c hotspots -r vtune_res \
-- ~/intel/intelpython2/bin/python 07-matrix-vector-product
```

Profiling python MPI jobs - Select rank

```
$: mpirun -n 1 \
amplxe-cl -c hotspots -r vtune_res \
-- ~/intel/intelpython2/bin/python 07-matrix-vector-product
\
: -n 1 ~/intel/intelpython2/bin/python 07-matrix-vector-
product
```

Profiling python MPI jobs - Select rank Cray

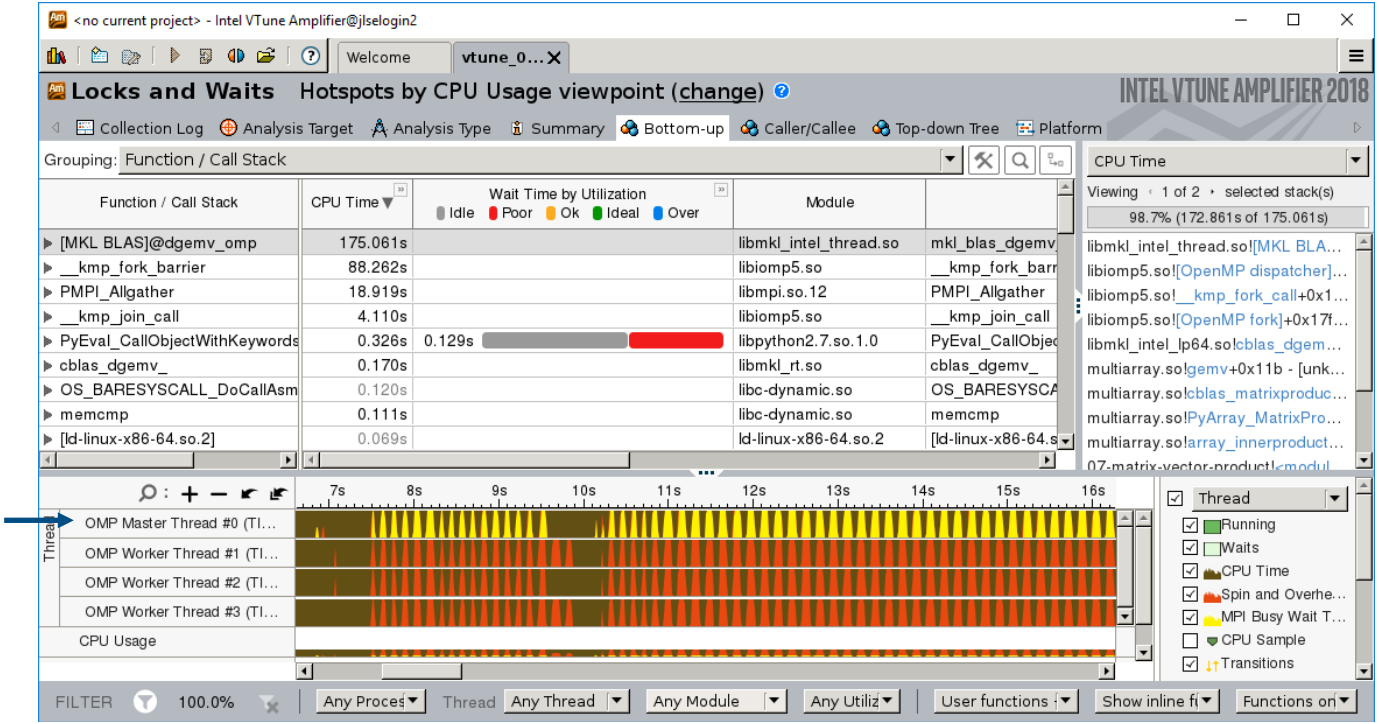
profile1.sh

```
 1 #!/bin/bash
 2 # source /opt/intel/parallel_studio_xe_2018/psxevars.sh intel64
 3 # export LD_LIBRARY_PATH=/opt/intel/advisor/lib64:$LD_LIBRARY_PATH
 4 # export LD_LIBRARY_PATH=/opt/intel/vtune_amplifier/lib64:$LD_LIBRARY_PATH
 5
 6 export PE_RANK=$ALPS_APP_PE
 7 export PMI_NO_FORK=1
 8 if [ "$PE_RANK" == 0 ];then
 9 $1 -- $2
10 else
11 $2
12 fi
```

Profiling python MPI jobs - Select rank Cray

```
aprun -n 2 ./profile1.sh \
"amplxe-cl -c hotspots -r vtune_res" \
"~/intel/intelpython2/bin/python 07-matrix-vector-product"
```

Bottom-up View – MPI



Profiling Libraries

Python Module for Quantum Chemistry

https://github.com/sunqm/pyscf

Prerequisites

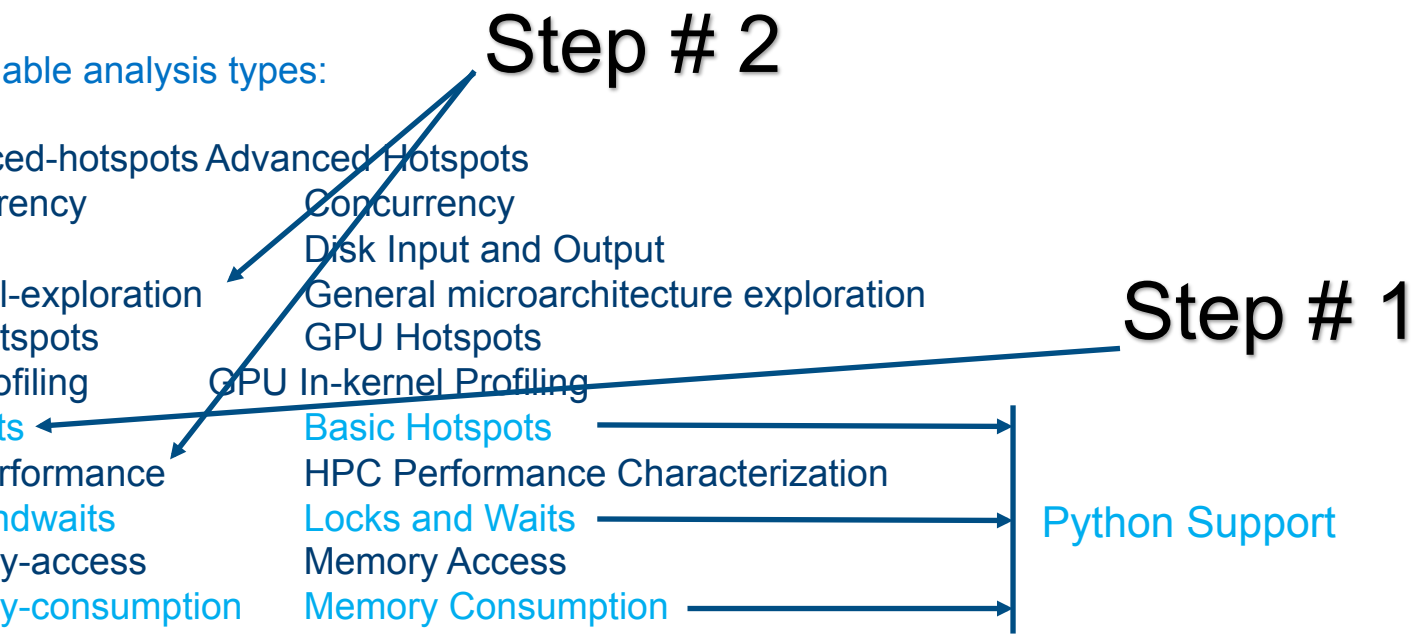
Build your libraries with -g to include debug symbols

Might have to add --search-dir src:=/path/to/library/source to your collection line

Two-Step Process

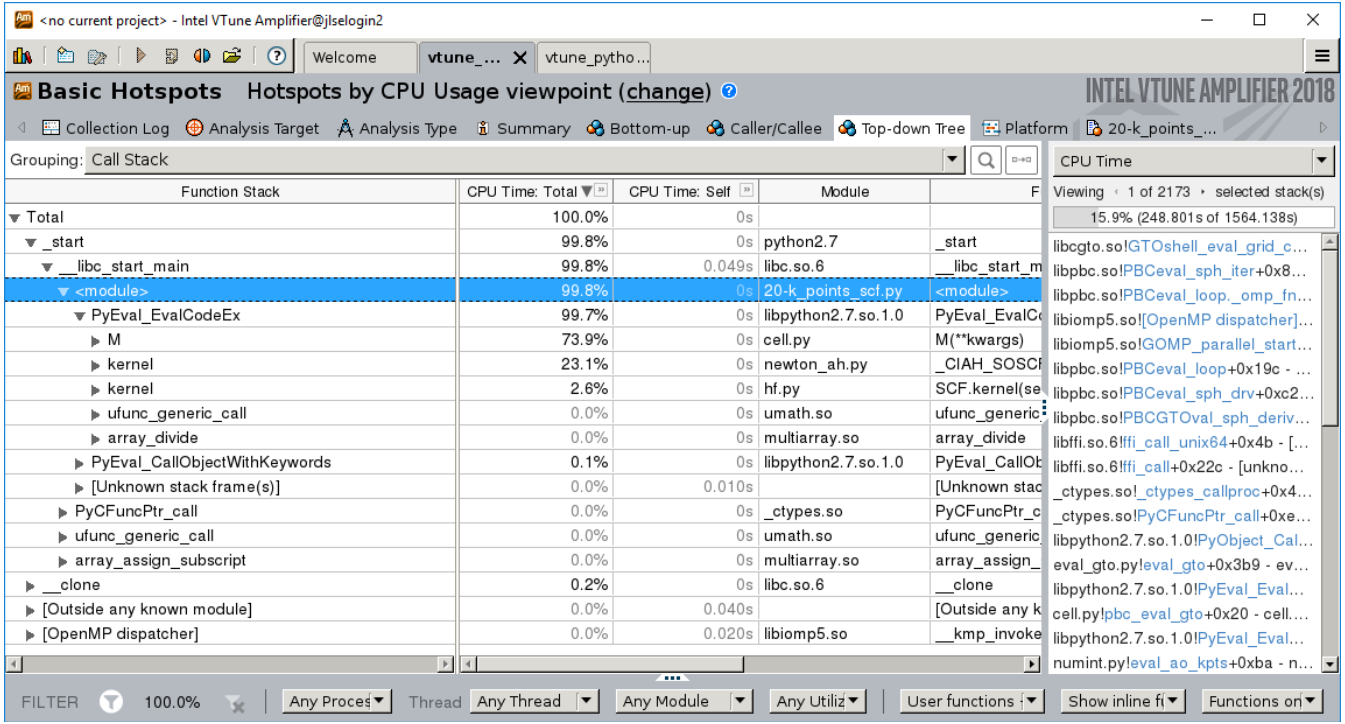
Many available analysis types:

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



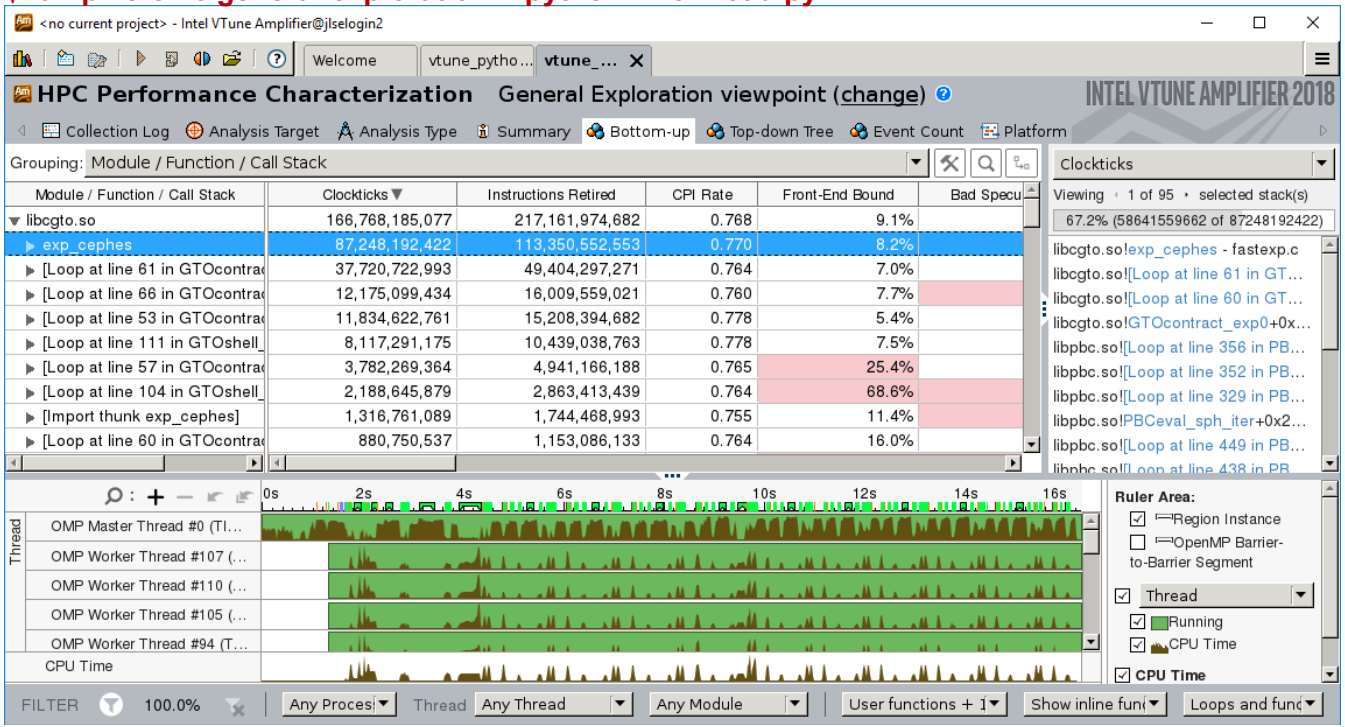
Step # 1

$: amplxe-cl -c hotspots -- python ./workload.py



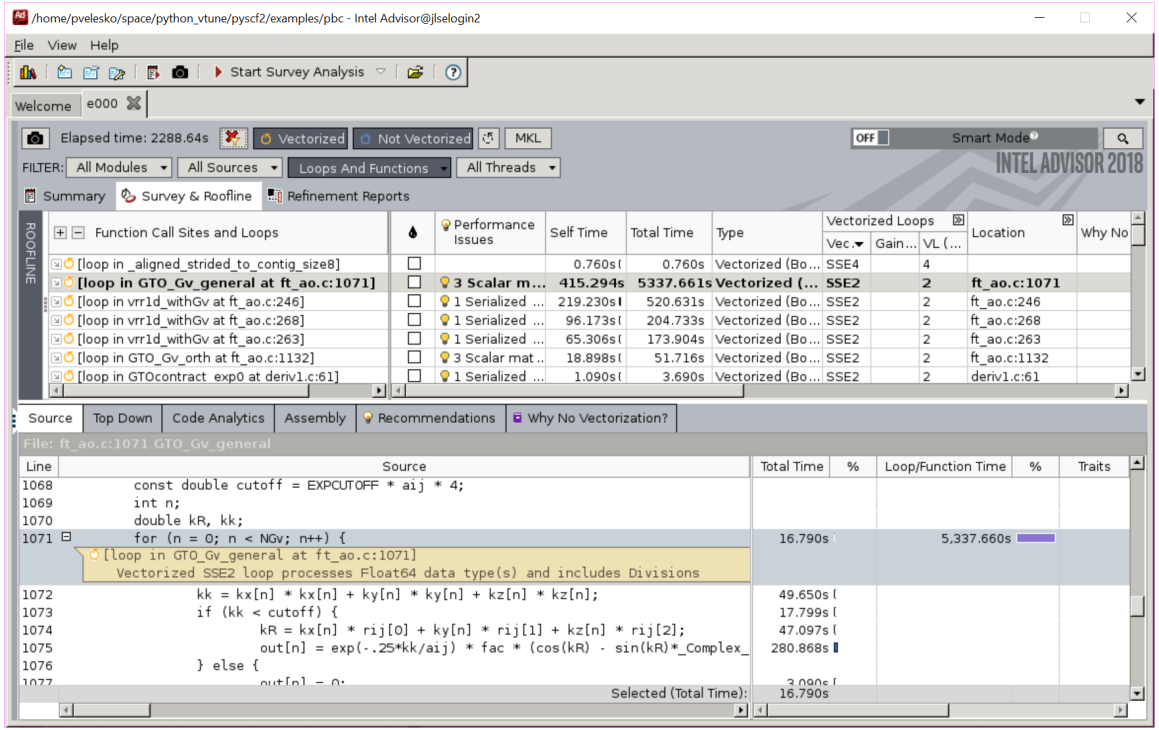
Step # 2- General Exploration

$: amplxe-cl -c general-exploration -- python ./workload.py



Step # 2b- Advisor

$: advixe-cl -c roofline -- python ./workload.py



Note on Collection/Finalization

- Your job is too short to collect sufficient information
  - Increase sampling freq --interval 0
- Finalization takes way too long
  - --finalization-mode=none
- Unknowns in your results
  - Libraries compiled with –g?
- Sources not being found
  - --search-dir src:=/path/to/source/dir

Legal Disclaimer & Optimization Notice

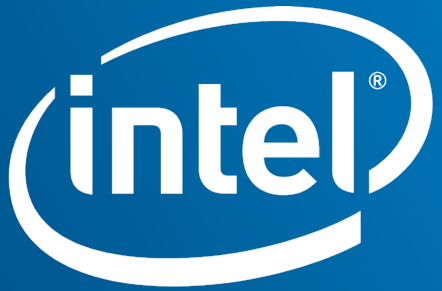
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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



Software