Using Intel Software Development Tools For Maximizing Deep Learning Performance

Alaa Eltablawy, Colfax International

Create an account on the Intel AI DevCloud:
https://colfaxresearch.com/aidevcon18
Passcode: **AG7WNN92**

Connect to the Intel® AI DevCloud

1. Account: colfaxresearch.com/aidevcon18
   Passcode: AG7WNN92
2. Follow the link in the invitation email
3. From the options on the top right, choose connect
4. Follow the link in Connect via Jupyter Hub section
5. From the New dropdown menu on the top right choose Terminal, and run command “devcon18”
6. Open “Human_Segmentation/Hands-on.ipynb”

Need a 30-day account? Go here after the workshop:

Agenda

- What you will learn
- Deep Learning on Intel Architecture
- Intel Parallel Studio XE tools
- Motivating DL example: Human segmentation
- Conclusion and Q&A

What You Will Learn

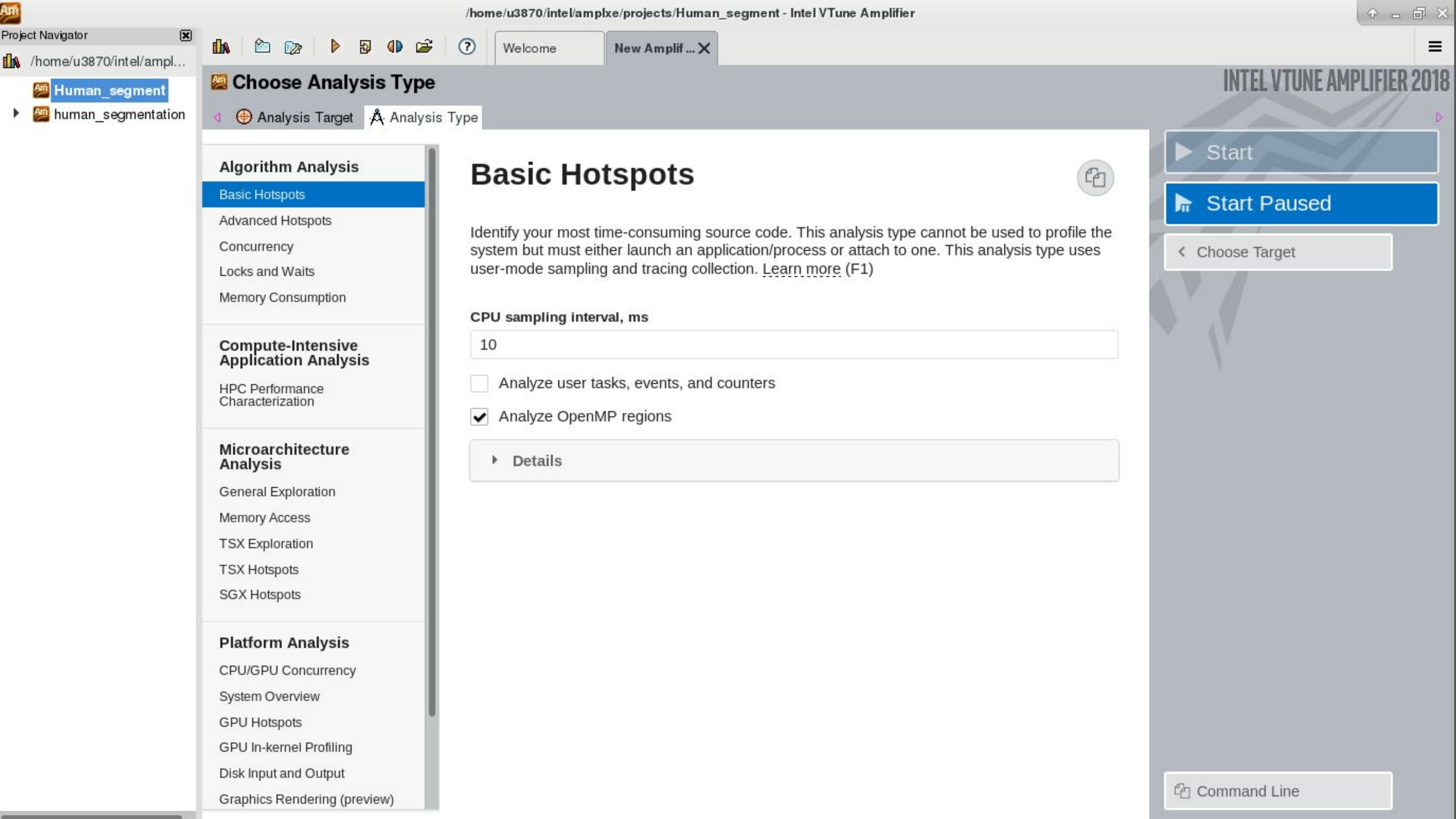
1. Why and how to use performance analysis tools like Intel VTune Amplifier
2. Measure the computational efficiency of your deep learning application on an Intel CPU
3. Use the guidance of the analysis tools to turn the appropriate “tuning knob”
4. Take advantage of optimizations in DL frameworks for Intel architecture

Deep Learning on Intel Architecture

- Deep learning (DL) applications on Intel Architecture
- Intel-optimized DL frameworks
  - DNN Primitives in Intel MKL
  - TensorFlow, Caffe, etc..
  - Optimizations for CPU

Intel VTune Amplifier

- Statistical performance analysis tool
  - Hardware events
  - User-mode sampling
- Results:
  - Hotspots
  - General issues (memory, cache, I/O, arithmetics,...)
- Getting started with VTune Amplifier


/home/u3870/intel/amplxe/projects/Human_segment - Intel VTune Amplifier
Project Navigator
/home/u3870/intel/ampl...
Human_segment
human_segmentation
Welcome
New Amplif ...
Choose Analysis Type
INTEL VTUNE AMPLIFIER 2018
Analysis Target
Analysis Type
Algorithm Analysis
Basic Hotspots
Advanced Hotspots
Concurrency
Locks and Waits
Memory Consumption
Compute-Intensive Application Analysis
HPC Performance Characterization
Microarchitecture Analysis
General Exploration
Memory Access
TSX Exploration
TSX Hotspots
SGX Hotspots
Platform Analysis
CPU/GPU Concurrency
System Overview
GPU Hotspots
GPU In-kernel Profiling
Disk Input and Output
Graphics Rendering (preview)
Basic Hotspots
Identify your most time-consuming source code. This analysis type cannot be used to profile the system but must either launch an application/process or attach to one. This analysis type uses user-mode sampling and tracing collection. Learn more (F1)
CPU sampling interval, ms
10
Analyze user tasks, events, and counters
Analyze OpenMP regions
Details
Start
Start Paused
Choose Target
Command Line

/home/u3870/intel/amplxe/projects/Human_segment - Intel VTune Amplifier

Project Navigator

/home/u3870/intel/ampl...

Human_segment

r000hs

human_segmentation

Welcome | r000hs

Basic Hotspots Hotspots by CPU Utilization viewpoint (change)

INTEL VTUNE AMPLIFIER 201

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

Elapsed Time: 64.186s

CPU Time: 432.872s

Total Thread Count: 321

Paused Time: 10.008s

OpenMP Analysis. Collection Time: 54.178

Serial Time (outside parallel regions): 2.472s (4.6%)

Parallel Region Time: 51.706s (95.4%)

Estimated Ideal Time: 51.170s (94.4%)

OpenMP Potential Gain: 0.536s (1.0%)

Top OpenMP Regions by Potential Gain

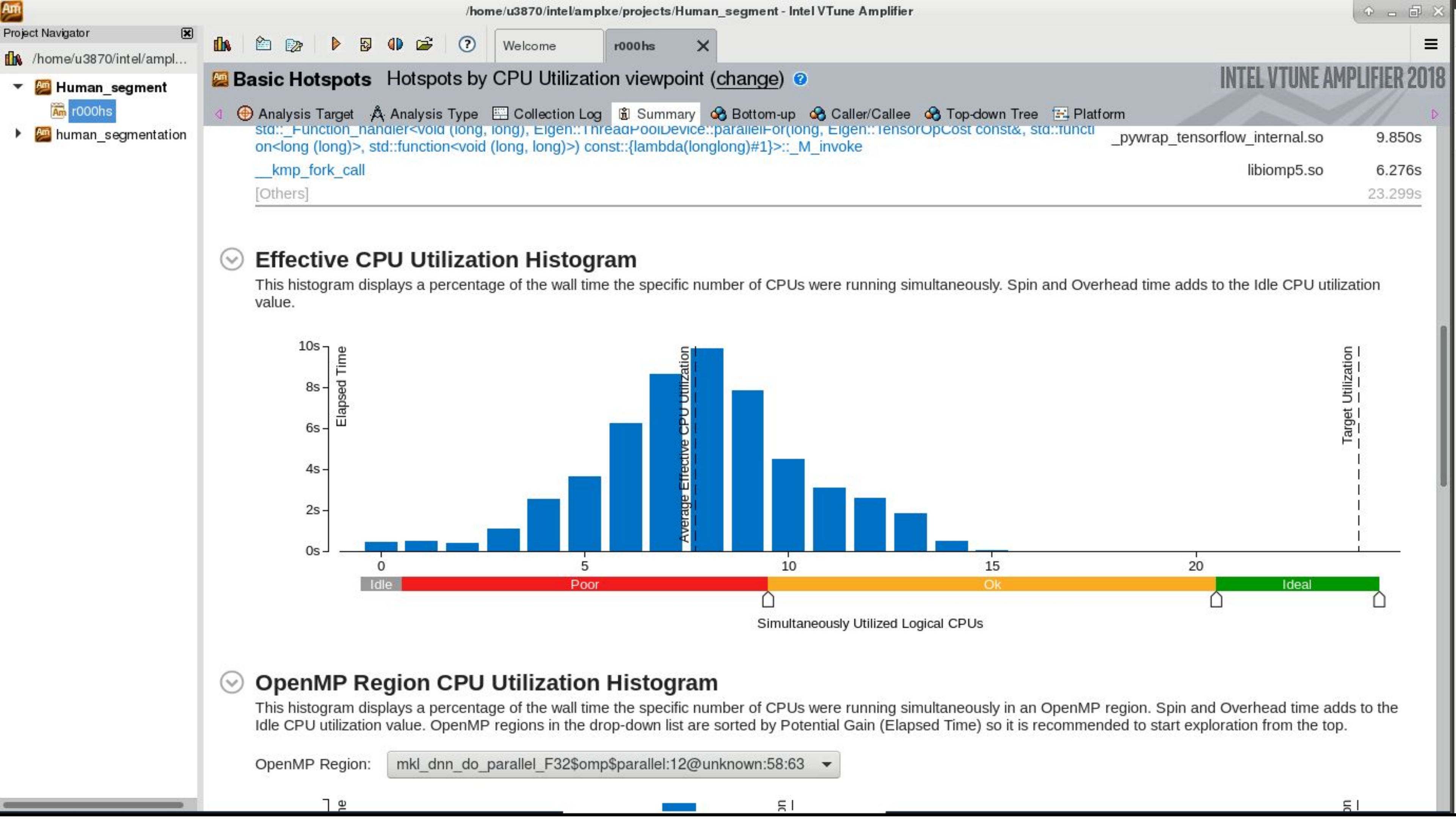
This section lists OpenMP regions with the highest potential for performance improvement. The Potential Gain metric shows the elapsed time that could be saved if the region was optimized to have no load imbalance assuming no runtime overhead.

OpenMP Region	OpenMP Potential Gain	(%)	OpenMP Region Time
mkl_dnn_do_parallel_F32ompparallel:12@unknown:58:63	19.175s	35.4%	48.199s
gemm_omp_driver_v2ompparallel:12@unknown:1750:1809	0.920s	1.7%	3.507s

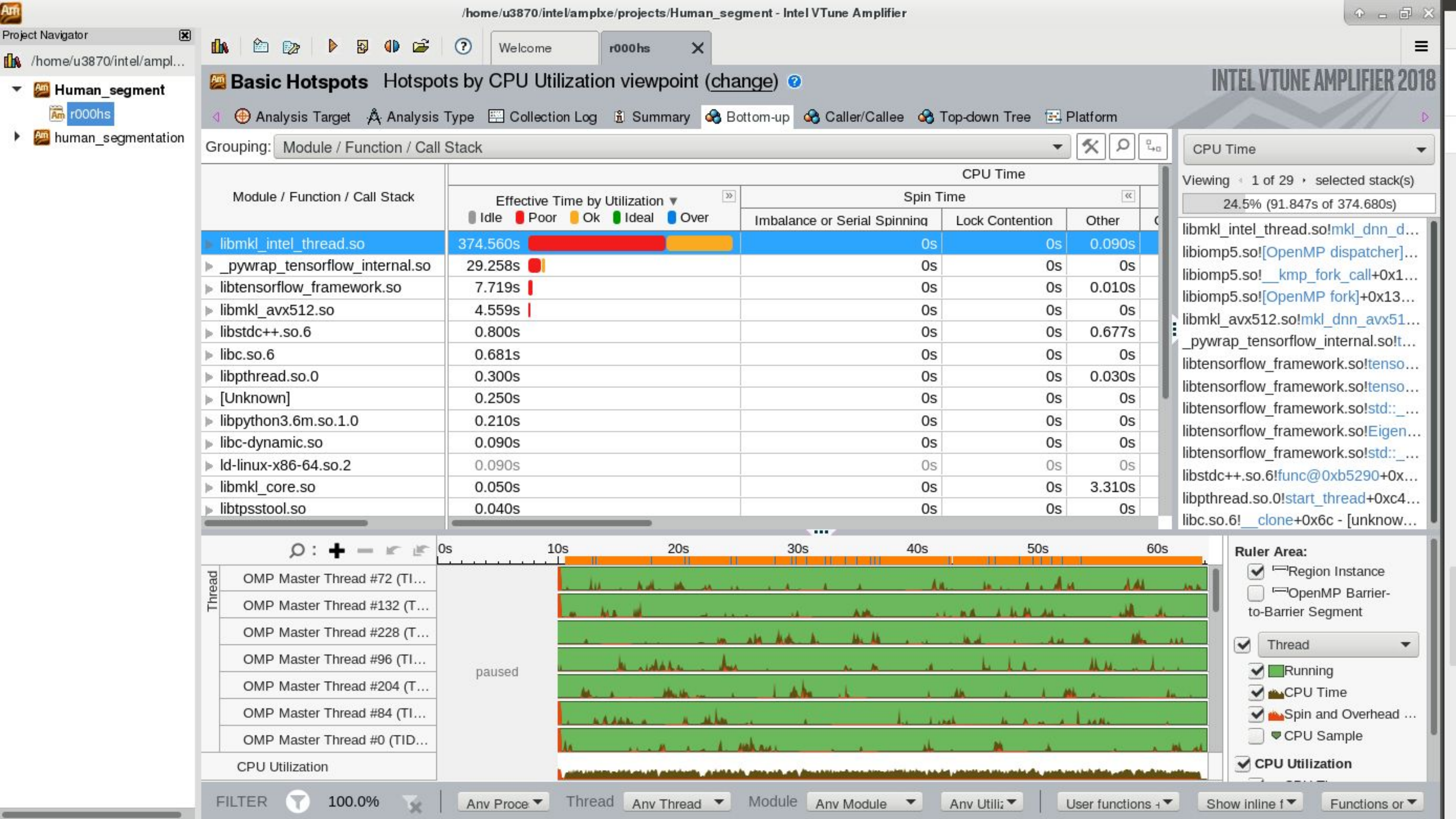
Top Hotspots

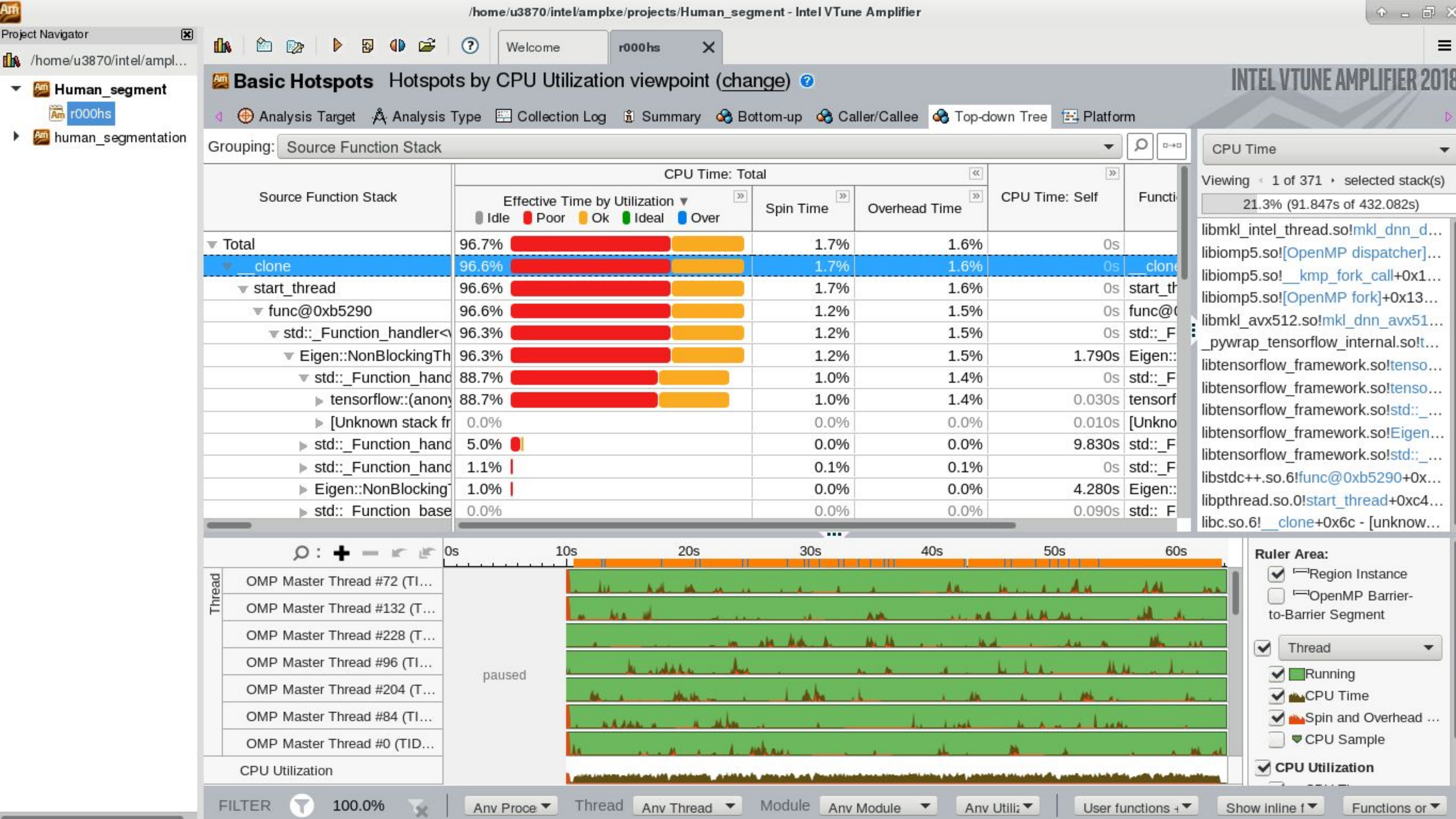
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
mkl_dnn_do_parallel_F32	libmkl_intel_thread.so	348.988s
gemm_omp_driver_v2	libmkl_intel_thread.so	25.692s
Eigen::internal::EvalRange<Eigen::TensorEvaluator<Eigen::TensorAssignOp<Eigen::TensorMap<Eigen::Tensor<float, (int)1, (int)1, long>, (int)16, Eigen::MakePointer>, Eigen::TensorConversionOp<float, Eigen::TensorReductionOp<Eigen::internal::SumReducer<float>, Eigen::IndexList<Eigen::type2index<(long)0>, <>> const, Eigen::TensorReshapingOp<Eigen::DSizes<int, (int)2> const, Eigen::TensorConversionOp<float, Eigen::TensorMap<Eigen::Tensor<float const, (int)1, (int)1, long>, (int)16, Eige	_pywrap_tensorflow_internal.so	18.768s



/home/u3870/intel/amplxe/projects/Human_segment - Intel VTune Amplifier
Project Navigator
/home/u3870/intel/ampl...
Human_segment
r000hs
human_segmentation
Welcome
r000hs
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
std::_Function_handler<void (long, long), Eigen::ThreadPoolDevice::parallelFor(long, Eigen::TensorOpCost const&, std::function<long (long)>, std::function<void (long, long)>) const::{lambda(longlong)#1}>::_M_invoke _pywrap_tensorflow_internal.so 9.850s
__kmp_fork_call libiomp5.so 6.276s
[Others] 23.299s
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
Idle
Poor
Ok
Ideal
Simultaneously Utilized Logical CPUs
OpenMP Region CPU Utilization Histogram
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously in an OpenMP region. Spin and Overhead time adds to the Idle CPU utilization value. OpenMP regions in the drop-down list are sorted by Potential Gain (Elapsed Time) so it is recommended to start exploration from the top.
OpenMP Region: mkl_dnn_do_parallel_F32ompparallel:12@unknown:58:63


/home/u3870/intel/amplxe/projects/Human_segment - Intel VTune Amplifier
Project Navigator
/home/u3870/intel/ampl...
Human_segment
r000hs
human_segmentation
Welcome
r000hs
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
Grouping: Module / Function / Call Stack
CPU Time
Module / Function / Call Stack
Effective Time by Utilization
Idle
Poor
Ok
Ideal
Over
Spin Time
Imbalance or Serial Spinning
Lock Contention
Other
libmkl_intel_thread.so 374.560s 0s 0s 0.090s
_pywrap_tensorflow_internal.so 29.258s 0s 0s 0s
libtensorflow_framework.so 7.719s 0s 0s 0.010s
libmkl_avx512.so 4.559s 0s 0s 0s
libstdc++.so.6 0.800s 0s 0s 0.677s
libc.so.6 0.681s 0s 0s 0s
libpthread.so.0 0.300s 0s 0s 0.030s
[Unknown] 0.250s 0s 0s 0s
libpython3.6m.so.1.0 0.210s 0s 0s 0s
libc-dynamic.so 0.090s 0s 0s 0s
ld-linux-x86-64.so.2 0.090s 0s 0s 0s
libmkl_core.so 0.050s 0s 0s 3.310s
libtpsstool.so 0.040s 0s 0s 0s
CPU Time
Viewing 1 of 29 selected stack(s)
24.5% (91.847s of 374.680s)
libmkl_intel_thread.so!mkl_dnn_d...
libiomp5.so![OpenMP dispatcher]...
libiomp5.so!__kmp_fork_call+0x1...
libiomp5.so![OpenMP fork]+0x13...
libmkl_avx512.so!mkl_dnn_avx51...
_pywrap_tensorflow_internal.so!t...
libtensorflow_framework.so!tenso...
libtensorflow_framework.so!tenso...
libtensorflow_framework.so!std::_...
libtensorflow_framework.so!Eigen...
libtensorflow_framework.so!std::_...
libstdc++.so.6!func@0xb5290+0x...
libpthread.so.0!start_thread+0xc4...
libc.so.6!__clone+0x6c - [unknow...
0s
10s
20s
30s
40s
50s
60s
Thread
OMP Master Thread #72 (TI...
OMP Master Thread #132 (T...
OMP Master Thread #228 (T...
OMP Master Thread #96 (TI...
OMP Master Thread #204 (T...
OMP Master Thread #84 (TI...
OMP Master Thread #0 (TID...
CPU Utilization
paused
Ruler Area:
Region Instance
OpenMP Barrier-to-Barrier Segment
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
FILTER
100.0%
Any Proce
Thread
Any Thread
Module
Any Module
Any Utiliz
User functions
Show inline f
Functions or


/home/u3870/intel/amplxe/projects/Human_segment - Intel VTune Amplifier
Project Navigator
/home/u3870/intel/ampl...
Human_segment
r000hs
human_segmentation
Welcome
r000hs
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
Grouping: Source Function Stack
CPU Time
Viewing 1 of 371 selected stack(s)
21.3% (91.847s of 432.082s)
Source Function Stack
CPU Time: Total
Effective Time by Utilization
Idle Poor Ok Ideal Over
Spin Time
Overhead Time
CPU Time: Self
Total 96.7% 1.7% 1.6% 0s
__clone 96.6% 1.7% 1.6% 0s
start_thread 96.6% 1.7% 1.6% 0s
func@0xb5290 96.6% 1.2% 1.5% 0s
std::_Function_handler 96.3% 1.2% 1.5% 0s
Eigen::NonBlockingTh 96.3% 1.2% 1.5% 1.790s
std::_Function_hand 88.7% 1.0% 1.4% 0s
tensorflow::(anony 88.7% 1.0% 1.4% 0.030s
[Unknown stack fr 0.0% 0.0% 0.0% 0.010s
std::_Function_hand 5.0% 0.0% 0.0% 9.830s
std::_Function_hand 1.1% 0.1% 0.1% 0s
Eigen::NonBlocking 1.0% 0.0% 0.0% 4.280s
std::_Function_base 0.0% 0.0% 0.0% 0.090s
libmkl_intel_thread.so!mkl_dnn_d...
libiomp5.so![OpenMP dispatcher]...
libiomp5.so!__kmp_fork_call+0x1...
libiomp5.so![OpenMP fork]+0x13...
libmkl_avx512.so!mkl_dnn_avx51...
_pywrap_tensorflow_internal.so!t...
libtensorflow_framework.so!tenso...
libtensorflow_framework.so!tenso...
libtensorflow_framework.so!std::_...
libtensorflow_framework.so!Eigen...
libtensorflow_framework.so!std::_...
libstdc++.so.6!func@0xb5290+0x...
libpthread.so.0!start_thread+0xc4...
libc.so.6!__clone+0x6c - [unknow...
0s 10s 20s 30s 40s 50s 60s
Thread
OMP Master Thread #72 (TI...
OMP Master Thread #132 (T...
OMP Master Thread #228 (T...
OMP Master Thread #96 (TI...
OMP Master Thread #204 (T...
OMP Master Thread #84 (TI...
OMP Master Thread #0 (TID...
CPU Utilization
paused
Ruler Area:
Region Instance
OpenMP Barrier-to-Barrier Segment
Thread
Running
CPU Time
Spin and Overhead ...
CPU Sample
CPU Utilization
FILTER 100.0%
Any Proce
Thread Any Thread
Module Any Module
Any Utiliz
User functions
Show inline f
Functions or

Application Performance Snapshot

- Part of VTune Amplifier
- Analyze CPU usage, OpenMP imbalance, Memory access efficiency, FPU usage
- Automatic guidance to recommended tools
- Getting started with Application Performance Snapshot

Intel® VTune™ Amplifier

Application Performance Snapshot

Application: *python3*
Report creation date: *2018-04-12 11:50:46*
HW Platform: *Intel(R) Xeon(R) Processor code named Skylake*
Logical Core Count per node: *24*
Collector type: *Event-based counting driver*

102.65s
Elapsed Time

287.57
SP GFLOPS

0.88
CPI

Your application might underutilize the available logical CPU cores

because of insufficient parallel work, blocking on synchronization, or too much I/O. Perform function or source line-level profiling with tools like Intel® VTune™ Amplifier to discover why the CPU is underutilized.

	Current run	Target	Delta
Physical Core Utilization	72.60%	>80%	
Memory Stalls	20.60%	<20%	
FPU Utilization	7.70%	>50%	
I/O Bound	0.01%	<10%	

Physical Core Utilization
72.60%

Average Physical Core Utilization
8.71 out of 12.00 physical cores

Memory Footprint
Resident total: 931.34 MB
Virtual total: 6656.16 MB

Memory Stalls
20.60% of pipeline slots

Cache Stalls
18.00% of cycles

DRAM Stalls
6.60% of cycles

Average DRAM Bandwidth
30.07 GB/s

NUMA
49.60% of remote accesses

FPU Utilization
7.70%

SP FLOPs per Cycle
4.94 Out of 64.00

Vector Capacity Usage
49.60%

FP Instruction Mix
% of Packed FP Instr.: 99.10%
% of 128-bit: 0.00%
% of 256-bit: 99.10%
% of 512-bit: 0.00%
% of Scalar FP Instr.: 0.90%

I/O Bound
0.01%
(AVG 0.01, PEAK 0.01)

Read
AVG 577.5 MB, MAX 577.5 MB

Write
AVG 55.1 KB, MAX 55.1 KB

Motivating example: Human segmentation

- Image semantic segmentation problem
- Applications using semantic segmentation:
  - Autonomous driving
  - Augmented and virtual reality
  - Indoor navigation

48x48x3

Conv 5x5 + Pool	Conv 5x5 + Pool	Conv 3x3	Conv 3x3	Conv 3x3	Conv 3x3	Conv 3x3	Conv 3x3 + Pool	FC1	FC2	FC3

48x48

Code based on Song et al. (2015) — http://ieeexplore.ieee.org/abstract/document/7486548/

Connect to the Intel® AI DevCloud

1. Account: colfaxresearch.com/aidevcon18
   Passcode: AG7WNN92
2. Follow the link in the invitation email
3. From the options on the top right, choose connect
4. Follow the link in Connect via Jupyter Hub section
5. From the New dropdown menu on the top right choose Terminal, and run command “devcon18”
6. Open “Human_Segmentation/Hands-on.ipynb”

Need a 30-day account? Go here after the workshop:

Tensorflow Optimizations for Intel architecture

TensorFlow* Optimizations for the Intel® Xeon® Scalable Processor

Intel Optimized TensorFlow* Installation Guide

TensorFlow* Optimizations on Modern Intel® Architecture

TensorFlow Performance Guide

Intel distribution for Python

Fast Deep Learning on Intel Architecture

- Intel Software development tools for fine tuning:
  - Application performance snapshot (APS) to diagnose global issues
  - Intel VTune Amplifier to detect hotspots and for platform analysis
- Programming practices for high performance:
  - Input data serialization
  - Environment control for efficient computing
  - Parallelism optimization (see Intel AI Academy)