Legal Disclaimer & Optimization Notice

OVERVIEW OF ADVISOR AND VTUNE

Renzo Bustamante

Application Engineer

renzo.bustamante@intel.com

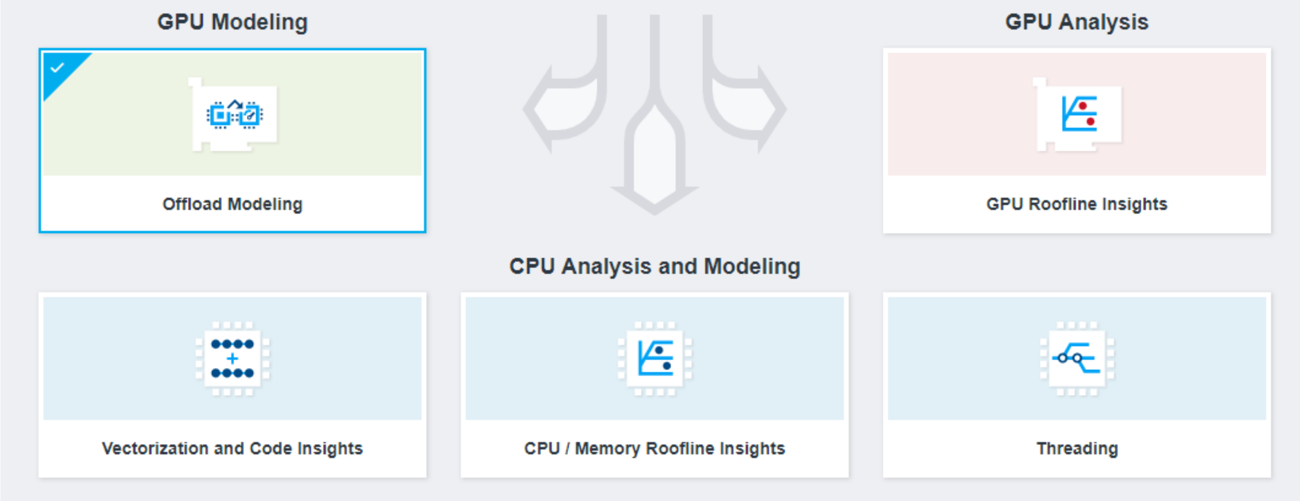
INTEL® ADVISOR

Vectorization and Static Analysis

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

What is advisor and what can it do

Advisor is a performance estimation tool for **CPU** and **GPUs** that helps us design and optimize high-performance code. It supports Fortan, C, C++, SYCL, OpenMP, OpenCL and Python code to realize full performance potential on modern computer architecture.



Where to download

For stand-alone installation:

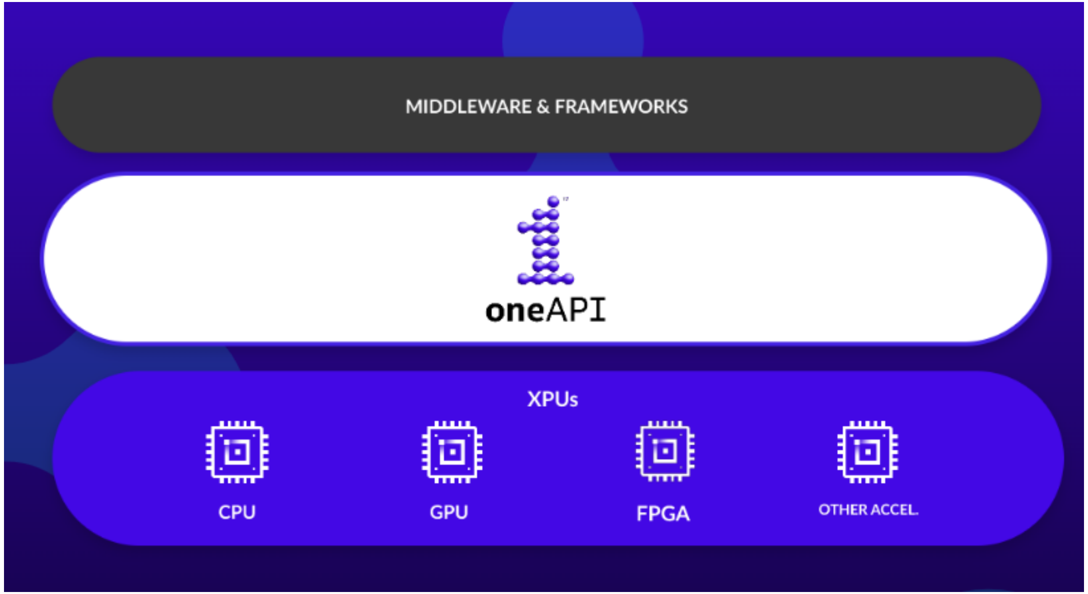
https://www.intel.com/content/www/us/en/developer/articles/tool/oneapi-standalone-components.html#advisor

As Part of Intel's OneAPI:

https://www.intel.com/content/www/us/en/developer/tools/oneapi/toolkits.html#base-kit

What is Intel oneAPI?

oneAPI is an **open**, cross-industry, standard-based, unified, multiarchitecture, multi-vendor programming model that delivers a common developer experience across accelerator architectures.



Compilers for:
C,C++, Fortran, Python
Supports the following programming models:
SYCL(C,C++) , DPCPP, (C++)
OpenMP (C,C++,Fortran)
And much more…
Works on Windows and Linux.

Intel oneAPI

DPC++
oneAPI Data Parallel C++

oneDPL
oneAPI Data Parallel C++ Library

oneDNN
oneAPI Deep Neural Network Library

oneCCL
oneAPI Collective Communications Library

Level Zero
oneAPI Level Zero

oneDAL
oneAPI Data Analytics Library

oneTBB
oneAPI Threading Building Blocks

oneVPL
oneAPI Video Processing Library

oneMKL
oneAPI Math Kernel Library

Ray Tracing
oneAPI Ray Tracing

Advisor Capabilities

Vectorization and Code Insights – Allows us to find unvectorized and under-vectorized loops and functions in our applications and get code-specific recommendations for how improving application performance and vectorization

CPU/Memory Roofline Insights – Produces Roofline chart for our application.

Offload Modeling. Allows us to identify where in our applications we could benefit by offloading it to a GPU.

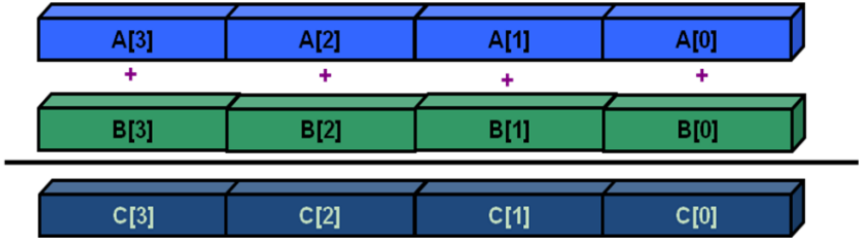
GPU Roofline Insight. Produces Roofline chart for our offloaded application (OpenMP ,DPC++,OpenCL)

Threading . Threading design options and project scaling on systems with larger core counts

Vectorization and code insights

With this tool we can analyze loops and functions that can benefit the most from parallelism , locate un-vectorized and under-vectorized time-consuming functions/loops and calculate estimated performance gain by vectorization.

Vectorization allows us to load more than one element of data in special vector registers and execute instructions on all those registers at the same time.



Vectorization code

For this demo we will use an n-body simulation kernel based on the work of Dr. Fabio Barufa

```
#ifndef _PARTICLE_HPP
#define _PARTICLE_HPP
#include <cmath>
#include "types.hpp"

struct Particle
{
  public:
    Particle() { init();}
    void init()
    {
      pos[0] = 0.; pos[1] = 0.; pos[2] = 0.;
      vel[0] = 0.; vel[1] = 0.; vel[2] = 0.;
      acc[0] = 0.; acc[1] = 0.; acc[2] = 0.;
      mass   = 0.;
    }
    real_type pos[3];
    real_type vel[3];
    real_type acc[3];
    real_type mass;
};

#endif
```

```
const double t0 = time.start();
for (int s=1; s<=get_nsteps(); ++s)
{
  ts0 += time.start();
  for (i = 0; i < n; i++)// update acceleration
  {
    for (j = 0; j < n; j++)
    {
        real_type dx, dy, dz;
        real_type distanceSqr = 0.0;
        real_type distanceInv = 0.0;

        dx = particles[j].pos[0] - particles[i].pos[0];      //1flop
        dy = particles[j].pos[1] - particles[i].pos[1];      //1flop
        dz = particles[j].pos[2] - particles[i].pos[2];      //1flop

        distanceSqr = dx*dx + dy*dy + dz*dz + softeningSquared;      //6fl
```

Vectorization and code insights

$git clone https://github.com/pvelesko/nbody-demo.git

On Theta:

$qsub -I -n 1 -t 59 -q comp_perf_workshop -A Comp_Perf_Workshop

$module load advisor

$export PMI_NO_FORK=1

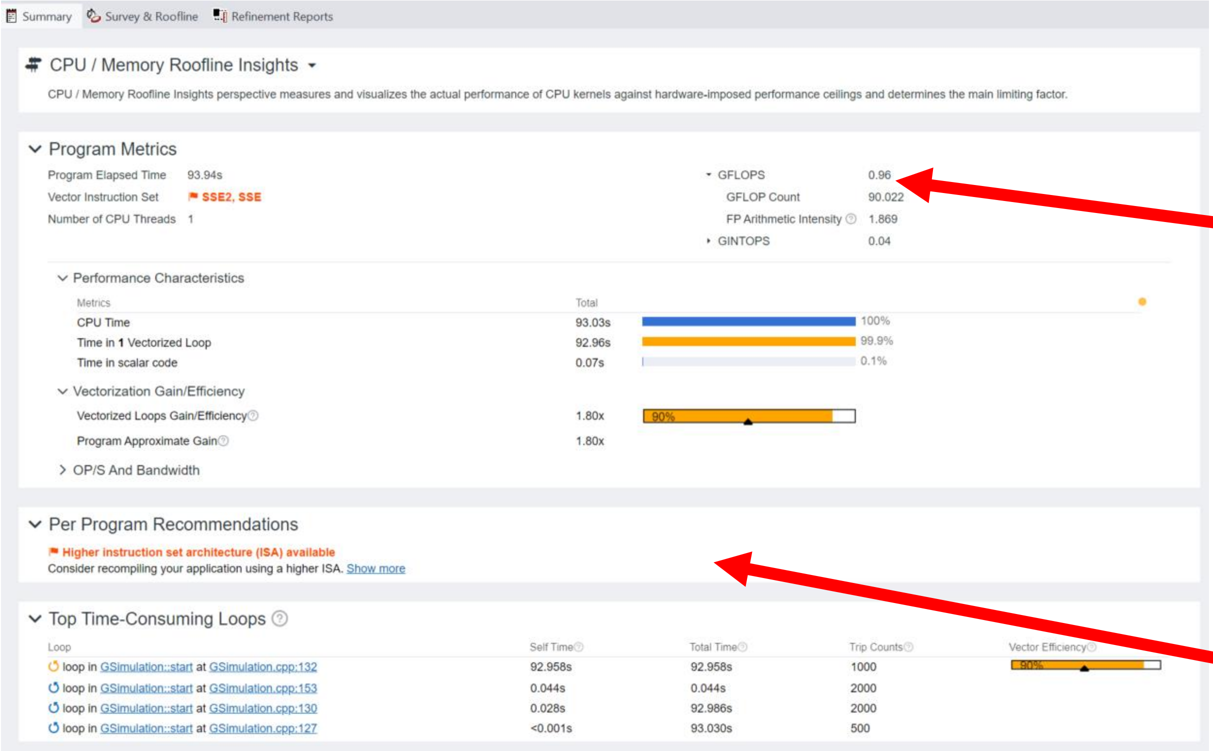
on /**projects** directory ,not /**home**

$cd var0 $make

$aprun -n 1 -N 1 advixe-cl --collect=survey --project-dir=results0Ver --search-dir src:r=/projects/intel/bustamante/nbody-demo/ver0/ -- ./nbody.x

$aprun -n 1 -N 1 advixe-cl --collect=tripcounts --flop --project-dir=results0Ver --search-dir src:r=/projects/intel/bustamante/nbody-demo/ver0/ -- ./nbody.x

Vectorization and code insights

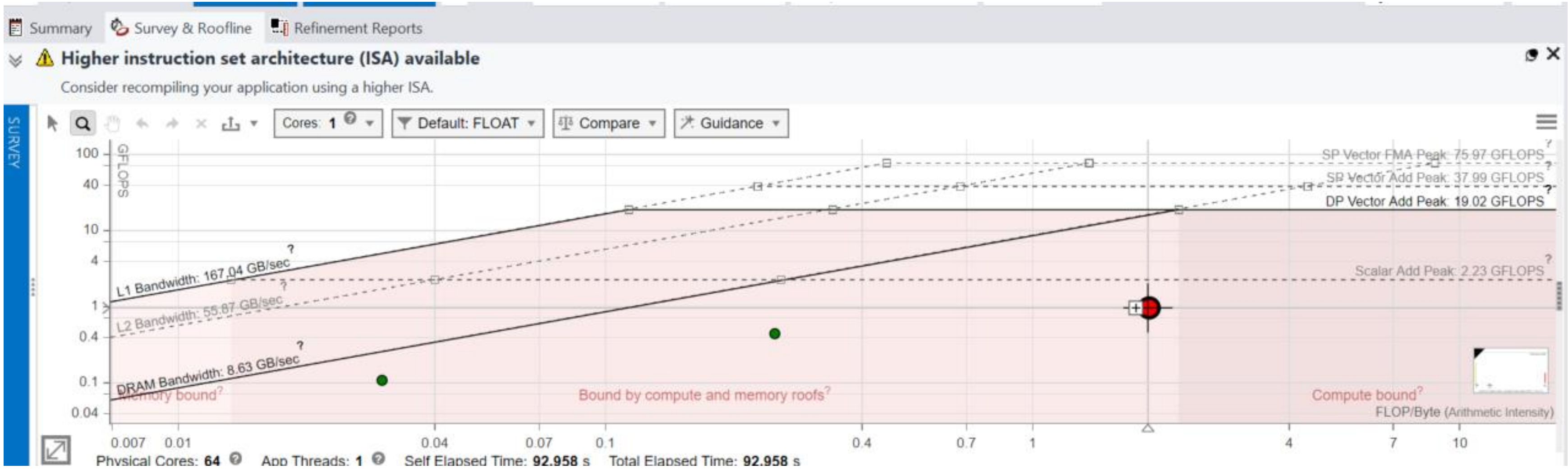


Compilation flags used in v0:

-g –std=c++11 –O2
0.96 GFLOPS

Indicates that our host architecture has hardware resources such as AVX-512 that could be used to increase performance

Vectorization and code insights Ver0



Cache-Aware Roofline

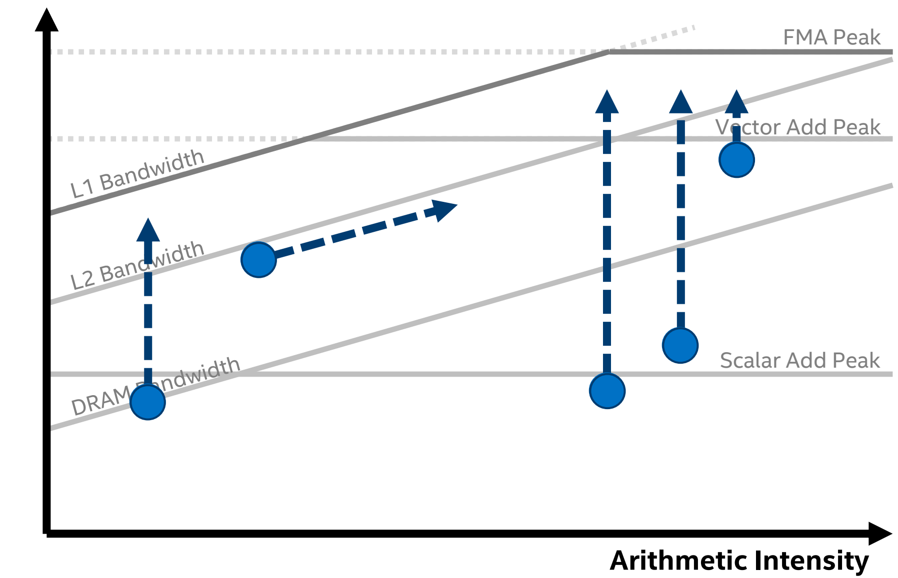
Next Steps

If under or near a memory roof...

- Try a MAP analysis. Make any appropriate **cache optimizations.**
- If cache optimization is impossible, try **reworking the algorithm to have a higher AI.**

If Under the Vector Add Peak

Check “Traits” in the Survey to see if FMAs are used. If not, try altering your code or compiler flags to **induce FMA usage.**



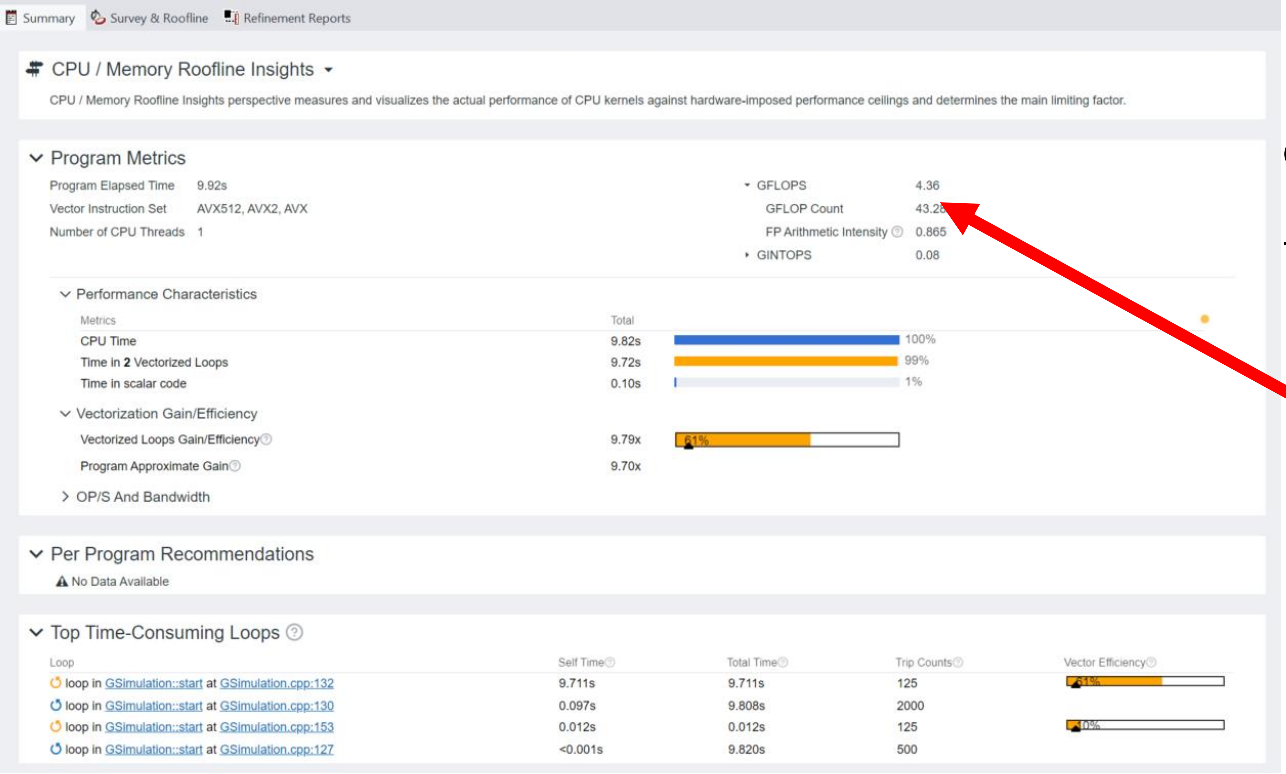
If just above the Scalar Add Peak

Check **vectorization efficiency** in the Survey. Follow the recommendations to improve it if it’s low.

If under the Scalar Add Peak...

Check the Survey Report to see if the loop vectorized. If not, try to **get it to vectorize** if possible. This may involve running Dependencies to see if it’s safe to force it.

Vectorization and code insights Ver2

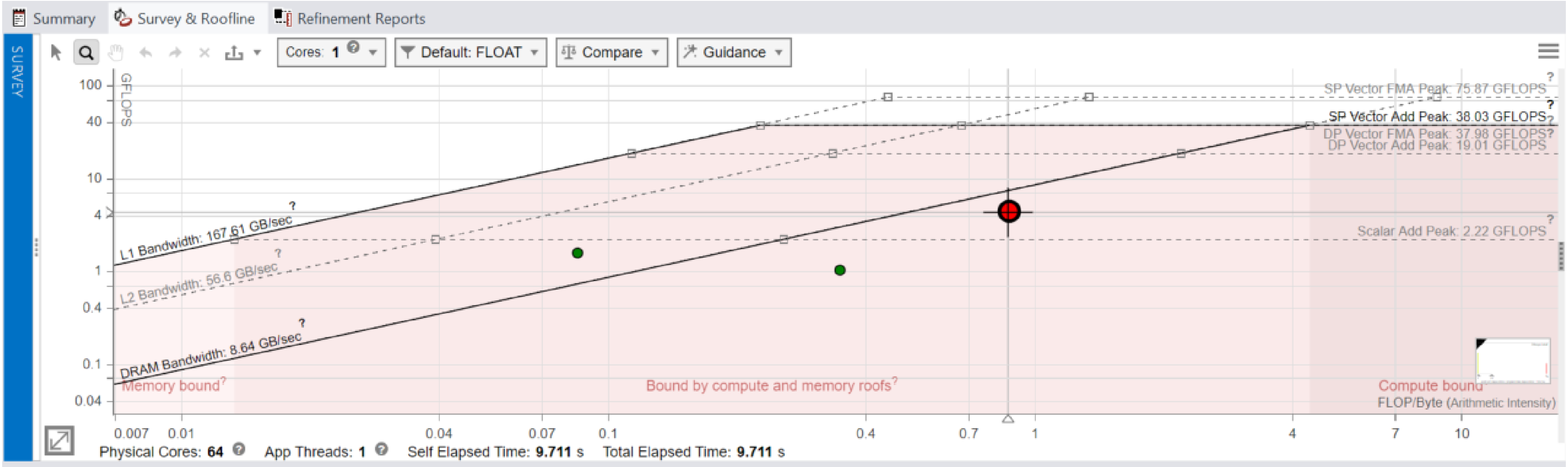


Compilation flags used in v0:

-g –std=c++11 –O2 –xMIC-AVX512

4.36
GFLOPS

Vectorization and code insights



If using mpi

$mpirun –n 1 advisor --collect=survey –project-dir=results --search-dir src:r=/source -- ./exe

INTEL® VTUNE™ AMPLIFIER

Core-level hardware metrics

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

What is Vtune

Intel® VTune™ Profiler optimizes application performance, system performance, and system configuration for HPC, cloud, IoT, media, storage, and more.

•CPU, GPU, and FPGA: Tune the entire application's performance—not just the accelerated portion.

•Multilingual: Profile SYCL*, C, C++, C#, Fortran, OpenCL™ code, Python*, Google Go* programming language, Java*, .NET, Assembly, or any combination of languages.

•System or Application: Get coarse-grained system data for an extended period or detailed results mapped to source code.

•Power: Optimize performance while avoiding power- and thermal-related throttling.

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
…	

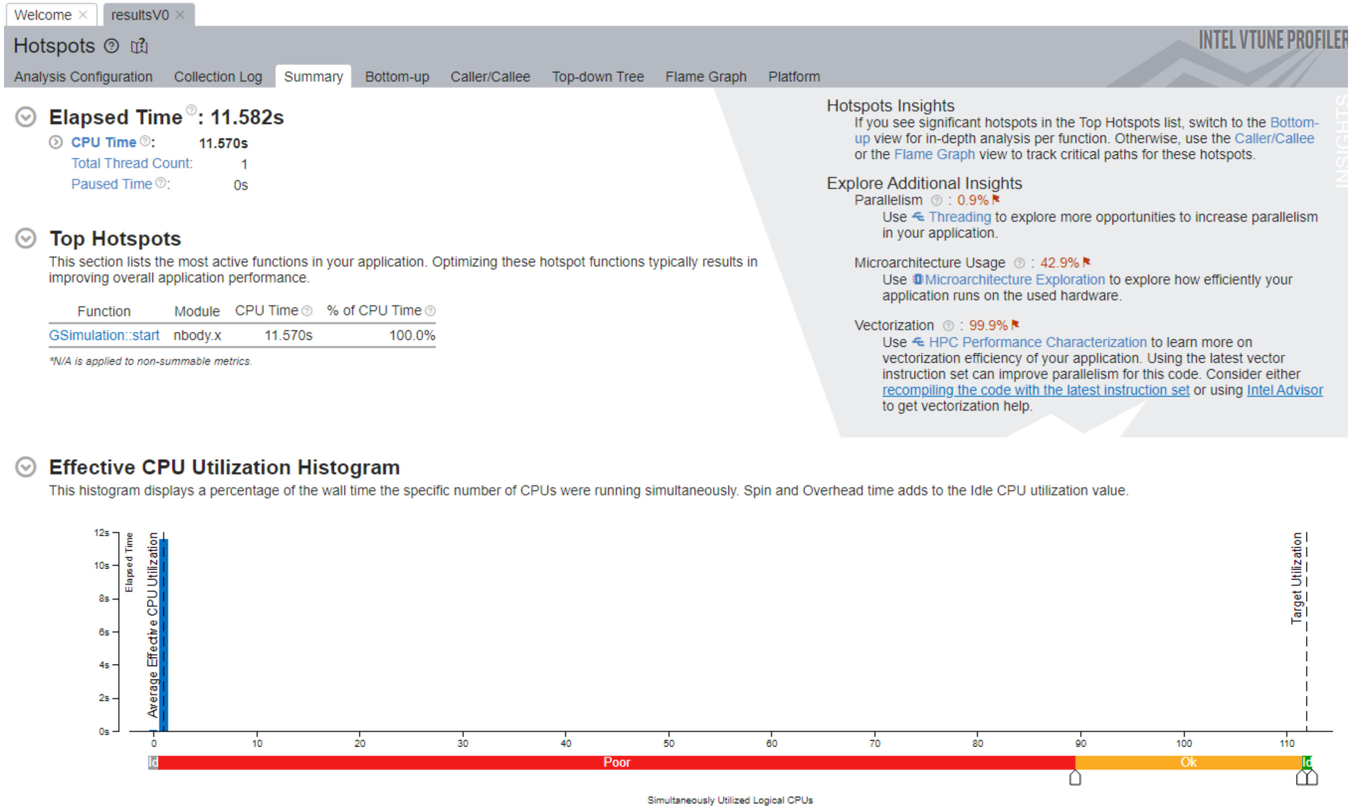
Python Support

Command line

```
$vtune -collect hotspots -r resultsV0 ./nbody.x
```

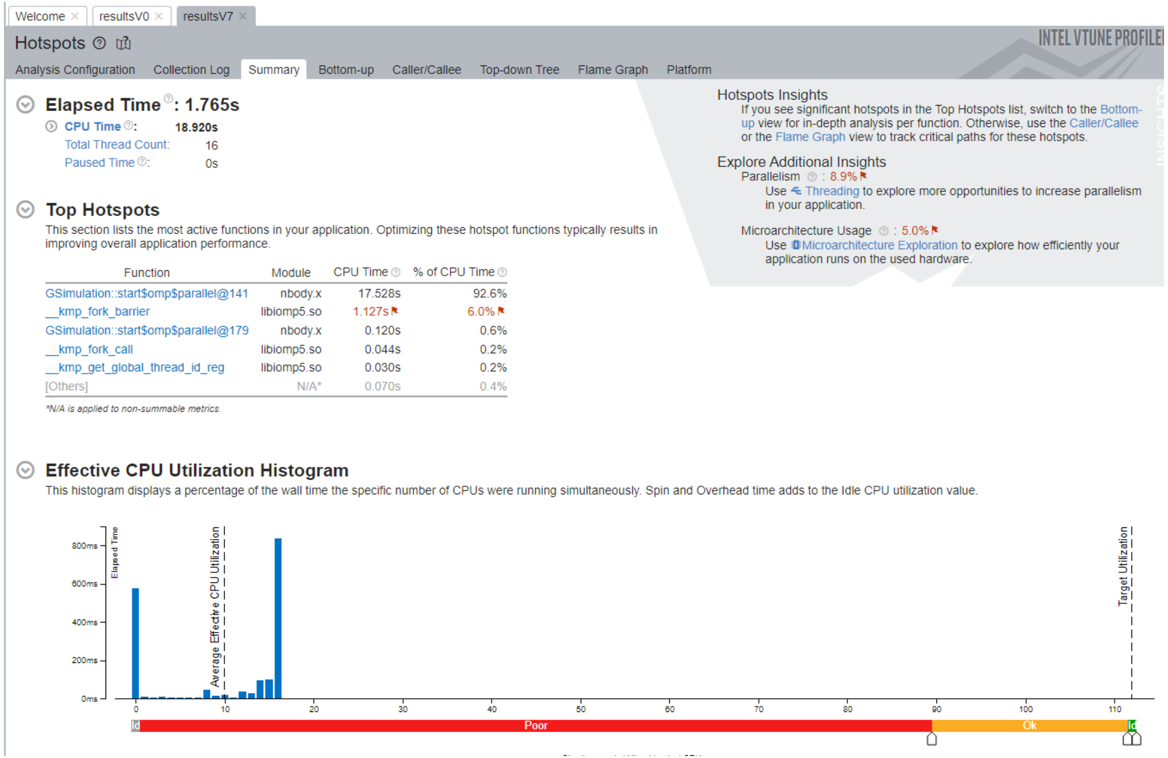
Copy the folder file to our local machine for further analysis.

Vtune GUI . Version 0 , not threaded

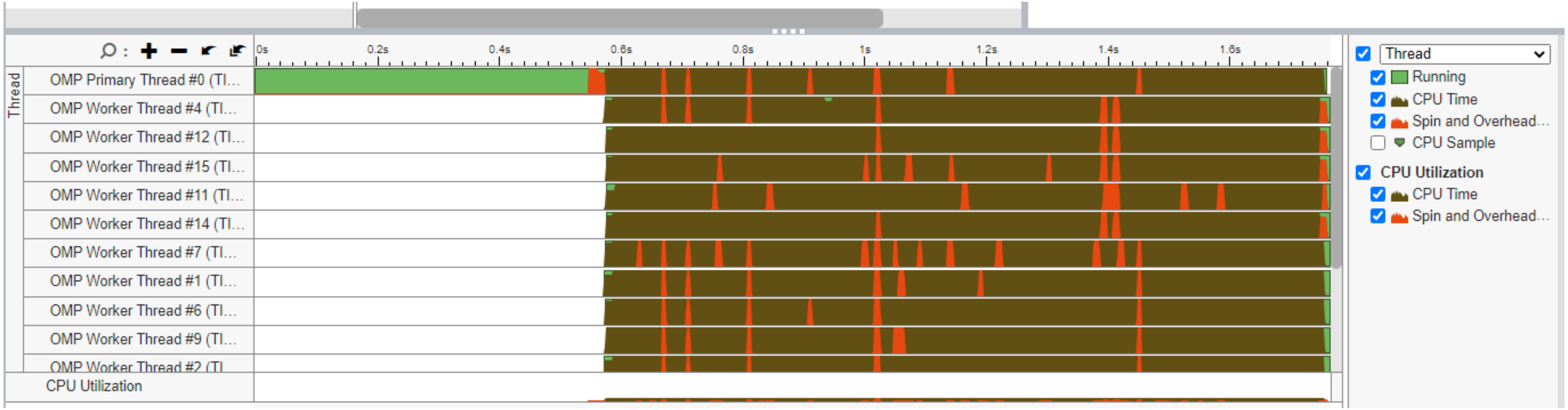


Not threaded, underutilizing Hardware resources

Vtune GUI. Version 7 , Threaded



Vtune GUI Version 7. Threaded



TIPS AND TRICKS

Managing overheads

Advisor Dependencies and MAP analyses can have huge overheads

If able, run on reduced problem size. Advisor just needs to figure out the execution flow.

Only analyze loops/functions of interest:

https://software.intel.com/en-us/advisor-user-guide-mark-up-loops

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

BACKUP

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

intel®
Software