C-DAC ACTS

HIGH PERFORMANCE COMPUTING IN APPLICATION PROGRAMMING

Project on
Profiling Tool Development

Project : Team

- Abhijeet Shinde
- Aditya Bhalavi
- Dipali Kothimbire
- Poonam Waghmare
- Sudhanshu Kamble

Project Head

Om Jadhav Sir

Project Guide

Himanshu Sharma Sir

Objective

- To Develop an advanced High-Performance Computing (HPC) profiling tool that enables comprehensive analysis and optimization of parallel and distributed applications, facilitating improved program performance and resource utilization.

- The tool goes beyond basic profiling capabilities and offers comprehensive analysis features for developers in identifying performance bottlenecks, optimizing parallel execution, and making efficient use of computing resources.

- The ultimate aim is to enhance the overall performance of HPC applications, leading to faster computations and better resource allocation.

Profiling

What is Profiling ?

Profiling in the context of High-Performance Computing (HPC) refers to the practice of gathering and analyzing data about the performance of software applications or hardware systems to identify bottlenecks, optimize performance, and improve efficiency. Profiling techniques in HPC are used to gain insights into the behavior and performance characteristics of parallel and distributed applications running on large-scale computing systems.

Profiling used for :

- Performance Analysis
- Resource Utilization
- Parallelism and Load Balancing
- Code Optimization
- Scalability Analysis

Literature Survey

Intel Advisor
The tool is designed to help programmers identify performance bottlenecks and provide insights into how to improve the efficiency of their applications.

HPCToolkit
HPC Toolkit is a tool for measurement and analysis of program performance on computers ranging from multicore desktop systems to the largest GPU-accelerated supercomputers.

Intel VTune
It empowers developers to enhance parallelism, and improve vectorization, resulting in faster and more efficient code execution.

TAU
TAU(Tuning and Analysis Utilities) toolkit is a comprehensive profiling and tracing toolkit.

LIKWID
LIKWID provides a set of tools and libraries which optimizes the performance of application.

In this project, we will be creating automation of 5 profiling tools HPCToolkit, TAU, LIKWID, Intel VTune and Intel Offload Advisor by creating shell scripts. We will be working on development of profiling tool on the basis of above tools. Final tool developed will be used to analyze the behaviour of any user application and find out its resource consumption.

1. Intel Advisior
2. HPCToolkit
3. Intel Vtune
4. TAU
5. LIKWID

Intel Advisior

- Intel Offload Advisor is a powerful tool specifically tailored for the high-performance computing (HPC) domain.
- Intel Offload Advisor is to assist developers in optimizing their applications for heterogeneous computing platforms.
- In computing, "heterogeneous" describes a system where various components, such as processors, accelerator, and co-processors, come together to perform different tasks based on their specialized capabilities.
- By offloading these tasks, developers can achieve higher performance, improved energy efficiency, and scalability.

Workflow

CPU-to-GPU Modeling:
CPU-to-GPU modeling refers to the process of offloading computations from a central processing unit (CPU) to a graphics processing unit (GPU) to enhance performance and efficiency.

Workflow:
The CPU identifies computationally intensive tasks that can be performed in parallel. It then transfers these tasks to the GPU, which processes them concurrently, resulting in faster execution times.

GPU-to-GPU Modeling:
GPU-to-GPU modeling involves utilizing multiple GPUs within a system to collaborate on a single computation, enabling even more parallelism and performance improvement.

Workflow: The problem is divided into smaller parts, and each part is assigned to a different GPU. The GPUs process their portions simultaneously, and the results are combined for the final output.

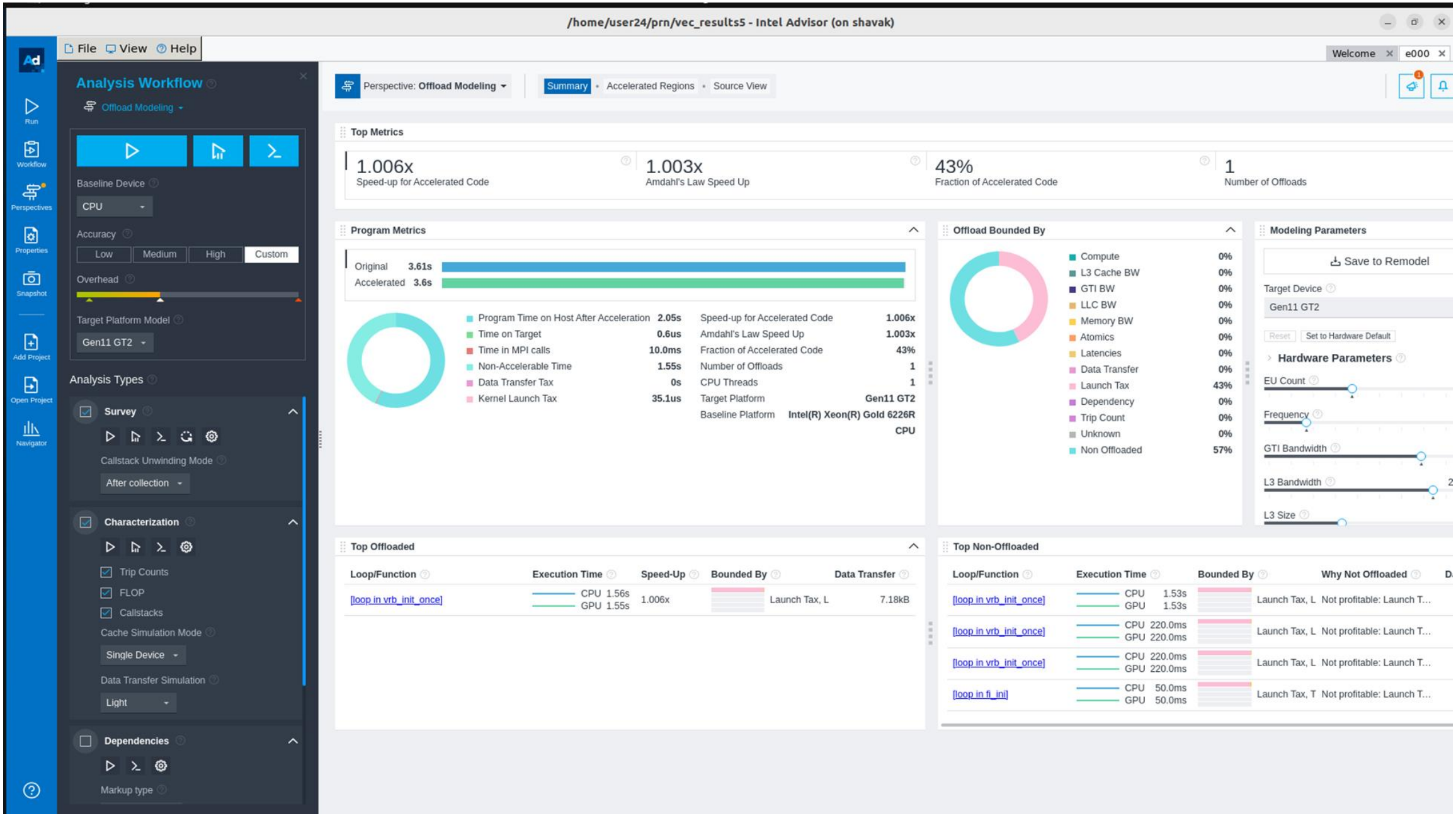
Intel Advisor Installation Script

```
^V#!/bin/bash

ADVISOR_PACKAGE="Intel offload advisor"

if command -v $ADVISOR_PACKAGE &>/dev/null; then
    echo "Advisor is already installed."

else

          cd /home/mobaxterm/proj16/Installation/

    DOWNLOAD_COMMAND="wget https://registrationcenter-download.intel.com/akdlm/IRC_NAS/992857b9-624c-45de-9701-f6445d845359/l_BaseKit_p_2
023.2.0.49397.sh"
     $ DOWNLOAD_COMMAND

    chmod +x  l_BaseKit_p_2023.2.0.49397.sh
    PATH="/home/mobaxterm/proj16/Installation/l_BaseKit_p_2023.2.0.49397.sh"
      echo "Installing advisor..."

     sh ./PATH


  if [ $? -eq 0 ]; then
        echo "Advisor installation completed successfully."
    else
        echo "Error: Advisor installation failed. Please check your package manager and try again."
        exit 1
    fi

fi

~
```

Intel Advisor Shell Script

```
#!/bin/bash


module load advisor


echo "Enter configuration:"
read config

echo "Enter accuracy:low,high,medium:"
read accuracy

echo "Enter the project directory:"
read project_dir


echo "Enter the path to the binary file:"
read binary_path

echo "Enter name for gerated report:"
read vector_report

if [ ! -f "$binary_path" ]; then
    echo "Binary path not found: $binary_path"
    exit 1
fi

advisor --collect=offload --accuracy="$accuracy" --config="$config" --project-dir="$project_dir" -- "$binary_path"

echo "Opening Intel advisor GUI...."

advixe-gui "$project_dir"

echo "profiling done!........."

~
```

Output


/home/user24/prn/vec_results5 - Intel Advisor (on shavak)
File View Help
Welcome e000
Analysis Workflow
Offload Modeling
Perspective: Offload Modeling
Summary • Accelerated Regions • Source View
Run
Workflow
Perspectives
Properties
Snapshot
Add Project
Open Project
Navigator
Baseline Device
CPU
Accuracy
Low Medium High Custom
Overhead
Target Platform Model
Gen11 GT2
Analysis Types
Survey
Callstack Unwinding Mode
After collection
Characterization
Trip Counts
FLOP
Callstacks
Cache Simulation Mode
Single Device
Data Transfer Simulation
Light
Dependencies
Markup type
Top Metrics
1.006x
Speed-up for Accelerated Code
1.003x
Amdahl's Law Speed Up
43%
Fraction of Accelerated Code
1
Number of Offloads
Program Metrics
Original 3.61s
Accelerated 3.6s
Program Time on Host After Acceleration 2.05s
Time on Target 0.6us
Time in MPI calls 10.0ms
Non-Accelerable Time 1.55s
Data Transfer Tax 0s
Kernel Launch Tax 35.1us
Speed-up for Accelerated Code 1.006x
Amdahl's Law Speed Up 1.003x
Fraction of Accelerated Code 43%
Number of Offloads 1
CPU Threads 1
Target Platform Gen11 GT2
Baseline Platform Intel(R) Xeon(R) Gold 6226R CPU
Offload Bounded By
Compute 0%
L3 Cache BW 0%
GTI BW 0%
LLC BW 0%
Memory BW 0%
Atomics 0%
Latencies 0%
Data Transfer 0%
Launch Tax 43%
Dependency 0%
Trip Count 0%
Unknown 0%
Non Offloaded 57%
Modeling Parameters
Save to Remodel
Target Device
Gen11 GT2
Reset
Set to Hardware Default
Hardware Parameters
EU Count
Frequency
GTI Bandwidth
L3 Bandwidth
L3 Size
Top Offloaded
Loop/Function
Execution Time
Speed-Up
Bounded By
Data Transfer
[loop in vrb_init_once]
CPU 1.56s
GPU 1.55s
1.006x
Launch Tax, L
7.18kB
Top Non-Offloaded
Loop/Function
Execution Time
Bounded By
Why Not Offloaded
[loop in vrb_init_once]
CPU 1.53s
GPU 1.53s
Launch Tax, L
Not profitable: Launch T...
[loop in vrb_init_once]
CPU 220.0ms
GPU 220.0ms
Launch Tax, L
Not profitable: Launch T...
[loop in vrb_init_once]
CPU 220.0ms
GPU 220.0ms
Launch Tax, L
Not profitable: Launch T...
[loop in fi_ini]
CPU 50.0ms
GPU 50.0ms
Launch Tax, T
Not profitable: Launch T...

HPCToolkit

HPCToolkit is a performance analysis tool suite designed for profiling and analyzing the performance of high-performance computing (HPC) applications. It is widely used by developers and researchers to understand the behavior of their applications, identify performance bottlenecks, and optimize code for better performance.

HPCToolkit supports various programming languages, including C, C++, and Fortran. It's commonly used in the HPC community to optimize applications for performance on supercomputers and clusters. By identifying performance bottlenecks and areas of inefficiency, developers can make targeted optimizations that lead to more efficient and faster-running applications.

The HPCToolkit suite includes various command that help in different aspects of performance analysis:

hpcrun: This is a binary instrumentation tool that collects data about the execution of an application. It inserts lightweight instrumentation code into the application's binary, allowing it to track various performance metrics such as time spent in functions, call paths, and memory usage.

hpcstruct: It helps in associating the performance data collected with the source code, enabling a more user-friendly representation of performance metrics in the context of the code.

hpcprof: This tool combines the instrumented binary, performance data, and mapping information to generate a profile that shows how much time is spent in various functions and call paths.

hpcviewer: This tool is used to visualize and explore the collected performance data. It provides an interactive graphical interface that allows users to analyze the application's behavior, identify performance hotspots, and understand the call path hierarchy.

The shell script of hpctoolkit

```
#!/bin/bash

set -e
source /home/apps/spack/share/spack/setup-env.sh

spack load hpctoolkit

mkdir -p /home/shavak/hpctoolkit/output

echo "Enter executable path"
read -p "executable_path: " executable_path

echo "Choose an event: REALTIME, CPUTIME, MEMLEAK"
read -p "event_name: " event_name

executable_name=$(basename "$executable_path")
output_dir="hpctoolkit-${executable_name}-measurement"

hpcrun -e "$event_name" -o "$output_dir" -t "$executable_path"

hpcstruct "$output_dir"

#database_dir="/home/shavak/hpctoolkit/output"

hpcprof -S "$output_dir" -o "$output_dir" "$output_dir"

hpcviewer "$output_dir"

echo "HPCToolkit analysis completed."
~
~
~
~
~
```

Output

```
[shavak@shavak script]$ bash hpctoolkit_script.sh
Enter executable path
executable_path: /home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpctoolkit-2021.10.15-xwzccbhlwuknxd3d2oiiok6dmnss3eiv/output/a.out
Choose an event: REALTIME, CPUTIME, MEMLEAK
event_name: REALTIME

 Execution time is = 1.728377 seconds

Program exit!
msg: begin concurrent analysis of libnss_dns-2.17.so (size = 31344, using 1 of 16 threads)
msg: begin concurrent analysis of libverbs-1.12-fi.so (size = 463248, using 1 of 16 threads)
msg: begin concurrent analysis of libpthread-2.17.so (size = 142144, using 1 of 16 threads)
msg: begin concurrent analysis of libgcc_s.so.1 (size = 440344, using 1 of 16 threads)
msg: begin concurrent analysis of libtcp-fi.so (size = 365036, using 1 of 16 threads)
msg: begin concurrent analysis of libuuid.so.1.3.0 (size = 20064, using 1 of 16 threads)
msg: begin concurrent analysis of 2d5225ed21d409a45a0bbd31eba5ebc1.[vdso] (size = 4928, using 1 of 16 threads)
msg: begin concurrent analysis of liblzma.so.5.2.5 (size = 174952, using 1 of 16 threads)
msg: begin concurrent analysis of libnl-route-3.so.200.23.0 (size = 444816, using 1 of 16 threads)
msg: begin concurrent analysis of libshm-fi.so (size = 435079, using 1 of 16 threads)
msg: begin concurrent analysis of libresolv-2.17.so (size = 109976, using 1 of 16 threads)
msg: begin concurrent analysis of libelf-0.186.so (size = 936856, using 1 of 16 threads)
msg: begin concurrent analysis of librxm-fi.so (size = 434838, using 1 of 16 threads)
msg: begin concurrent analysis of libmpifort.so.12.0.0 (size = 4954906, using 1 of 16 threads)
msg: begin concurrent analysis of libbz2.so.1.0.8 (size = 201368, using 1 of 16 threads)
msg: begin concurrent analysis of libmlx-fi.so (size = 346685, using 1 of 16 threads)
msg: end concurrent analysis of 2d5225ed21d409a45a0bbd31eba5ebc1.[vdso]
msg: end concurrent analysis of libnss_dns-2.17.so
msg: end concurrent analysis of libuuid.so.1.3.0
msg: begin concurrent analysis of libfabric.so.1 (size = 383905, using 1 of 16 threads)
msg: begin concurrent analysis of libnl-3.so.200.23.0 (size = 139016, using 1 of 16 threads)
msg: begin concurrent analysis of librt-2.17.so (size = 43712, using 1 of 16 threads)
msg: end concurrent analysis of libpthread-2.17.so
msg: end concurrent analysis of libbz2.so.1.0.8
msg: begin concurrent analysis of libcap.so.2.22 (size = 20048, using 1 of 16 threads)
msg: begin concurrent analysis of libdl-2.17.so (size = 19248, using 1 of 16 threads)
msg: end concurrent analysis of libresolv-2.17.so
msg: end concurrent analysis of librt-2.17.so
msg: begin concurrent analysis of librdmacm.so.1.1.22.4 (size = 92016, using 1 of 16 threads)
msg: begin concurrent analysis of ld-2.17.so (size = 163312, using 1 of 16 threads)
msg: end concurrent analysis of liblzma.so.5.2.5
msg: end concurrent analysis of libdl-2.17.so
msg: end concurrent analysis of libcap.so.2.22
msg: begin concurrent analysis of libdw-0.186.so (size = 3869144, using 1 of 16 threads)
msg: begin concurrent analysis of libmpi.so.12.0.0 (size = 15674270, using 1 of 16 threads)
msg: end concurrent analysis of libnl-3.so.200.23.0
msg: begin concurrent analysis of libnss_myhostname.so.2 (size = 86464, using 1 of 16 threads)
msg: begin concurrent analysis of libverbs-1.1-fi.so (size = 449818, using 1 of 16 threads)
msg: end concurrent analysis of libnl-route-3.so.200.23.0
msg: end concurrent analysis of libgcc_s.so.1
msg: begin concurrent analysis of a.out (size = 12496, using 1 of 16 threads)
msg: end concurrent analysis of libelf-0.186.so
```

```
msg: begin concurrent analysis of libatomic.so.1.2.0 (size = 145872, using 1 of 16 threads)
msg: end concurrent analysis of librdmacm.so.1.1.22.4
msg: end concurrent analysis of a.out
msg: begin concurrent analysis of libpapi.so.6.0.0.1 (size = 1032840, using 1 of 16 threads)
msg: end concurrent analysis of libnss_myhostname.so.2
msg: begin concurrent analysis of libsockets-fi.so (size = 468152, using 1 of 16 threads)
msg: begin concurrent analysis of libz.so.1.2.11 (size = 101248, using 1 of 16 threads)
msg: end concurrent analysis of libtcp-fi.so
msg: end concurrent analysis of libmlx-fi.so
msg: begin concurrent analysis of libnuma.so.1.0.0 (size = 151368, using 1 of 16 threads)
msg: end concurrent analysis of libatomic.so.1.2.0
msg: end concurrent analysis of libshm-fi.so
msg: begin concurrent analysis of libibverbs.so.1.5.22.4 (size = 105704, using 1 of 16 threads)
msg: begin concurrent analysis of libm-2.17.so (size = 1136944, using 1 of 16 threads)
msg: begin concurrent analysis of libefa-fi.so (size = 634703, using 1 of 16 threads)
msg: end concurrent analysis of ld-2.17.so
msg: begin concurrent analysis of libattr.so.1.1.0 (size = 19896, using 1 of 16 threads)
msg: end concurrent analysis of libverbs-1.12-fi.so
msg: end concurrent analysis of librxm-fi.so
msg: begin concurrent analysis of libpsm3-fi.so (size = 1458432, using 1 of 16 threads)
msg: begin concurrent analysis of libpsmx2-fi.so (size = 641663, using 1 of 16 threads)
msg: end concurrent analysis of libfabric.so.1
msg: end concurrent analysis of libattr.so.1.1.0
msg: end concurrent analysis of libnuma.so.1.0.0
msg: end concurrent analysis of libz.so.1.2.11
msg: begin concurrent analysis of libnss_files-2.17.so (size = 61560, using 1 of 16 threads)
msg: end concurrent analysis of libibverbs.so.1.5.22.4
msg: end concurrent analysis of libpsm3-fi.so
msg: end concurrent analysis of libnss_files-2.17.so
msg: end concurrent analysis of libpapi.so.6.0.0.1
msg: end concurrent analysis of libverbs-1.1-fi.so
msg: end concurrent analysis of libsockets-fi.so
msg: end concurrent analysis of libm-2.17.so
msg: end concurrent analysis of libefa-fi.so
msg: end concurrent analysis of libpsmx2-fi.so
msg: end concurrent analysis of libdw-0.186.so
msg: end concurrent analysis of libmpifort.so.12.0.0
msg: end concurrent analysis of libc-2.17.so
msg: end concurrent analysis of libmpi.so.12.0.0
msg: Directory 'hpctoolkit-a.out-measurement' already exists. Trying 'hpctoolkit-a.out-measurement-143943'
msg: Created directory: hpctoolkit-a.out-measurement-143943
WARNING: Unable to read Document:STRUCTURE file: 'hpctoolkit-a.out-measurement': Is a directory
msg: STRUCTURE: /home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpctoolkit-2021.10.15-xwzccbhlwuknxd3d2oiiok6dmnss3eiv/output/a.out
msg: Line map : /home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpctoolkit-2021.10.15-xwzccbhlwuknxd3d2oiiok6dmnss3eiv/lib/hpctoolkit/libhpcrun.so
msg: Line map : /home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpctoolkit-2021.10.15-xwzccbhlwuknxd3d2oiiok6dmnss3eiv/lib/hpctoolkit/ext-libs/libmonitor.so.0.0.0
msg: STRUCTURE: /opt/intel/oneapi/mpi/2021.8.0/lib/libmpifort.so.12.0.0
msg: STRUCTURE: /opt/intel/oneapi/mpi/2021.8.0/lib/release/libmpi.so.12.0.0
msg: STRUCTURE: /usr/lib64/libc-2.17.so
msg: STRUCTURE: /opt/intel/oneapi/mpi/2021.8.0/libfabric/lib/prov/librxm-fi.so
msg: Populating Experiment database: /home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpctoolkit-2021.10.15-xwzccbhlwuknxd3d2oiiok6dmnss3eiv/script/hpctoolkit-a.out-measurement-143943
/home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpcviewer-2021.10-ayxvrngmsvijpus54kuv3gsuab7crd6y/bin/hpcviewer: error: DISPLAY variable is not set
[shavak@shavak script]$
```

Intel Vtune

Intel VTune Profiler is a performance analysis tool developed by Intel to help developers optimize software applications for better performance on various hardware platforms.

Intel VTune Profiler is a powerful and versatile performance analysis tool with features that make it stand out from other profiling tools. Its comprehensive analysis capabilities, support for rich visualization, and integration with development tools make it an invaluable asset for developers to create high-performance applications.

VTune Tool Purposes:

- Collect performance analysis data for your target application using your specified analysis type and other options.
- Generate reports from analysis results.
- Import data files collected remotely.
- Compare performance before and after optimization.

Vtune Installation Script

user24@shavak:~/project/vtune_tool

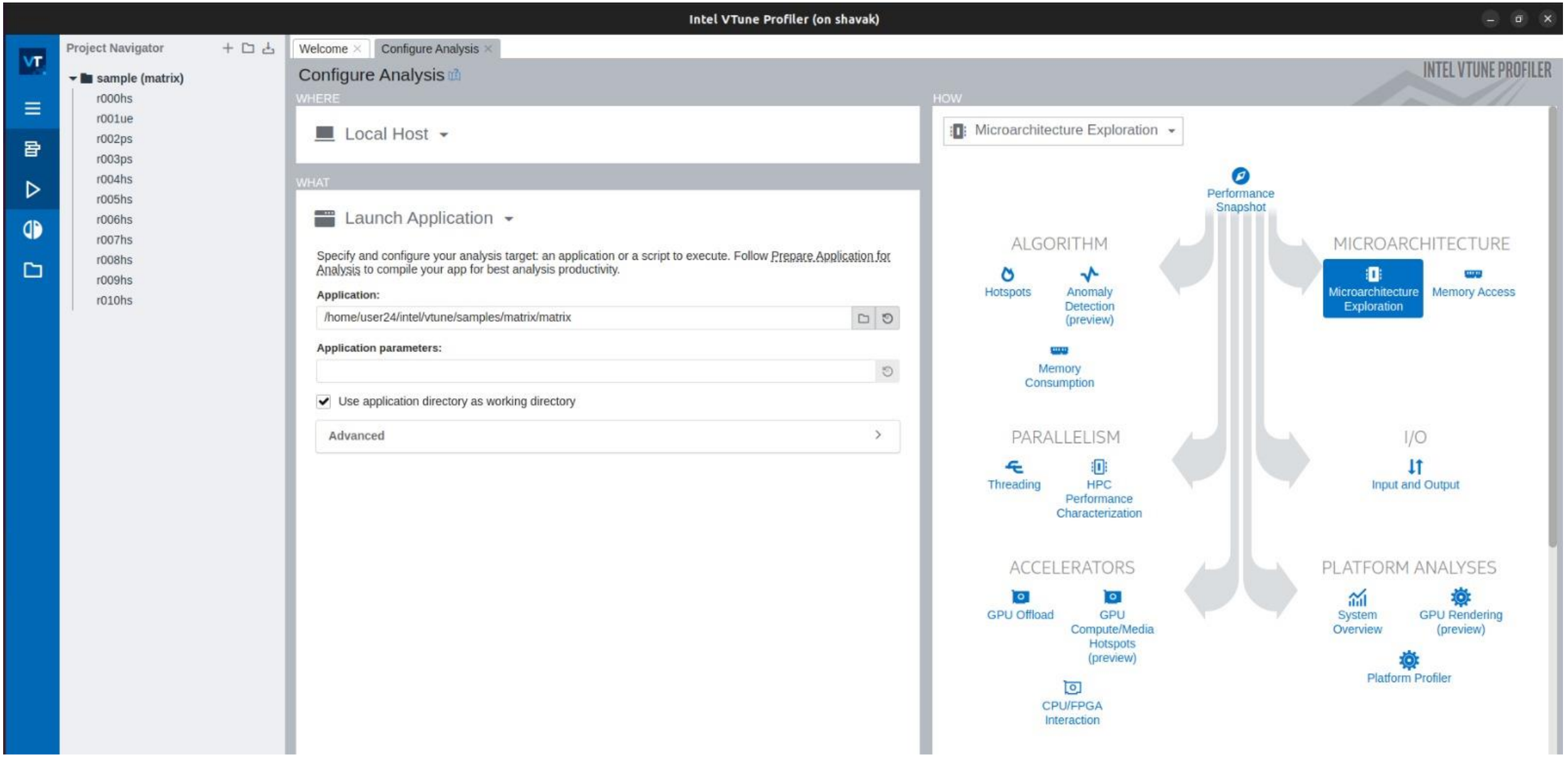
```
#!/bin/bash

wget = "https://registrationcenter-download.intel.com/akdlm/IRC_NAS/dfae6f23-6c90-4b9f-80e2-fa2a5037fe36/l_oneapi_vtune_p_2023.2.0.49485.sh"

chmod +x l_oneapi_vtune_p_2023.2.0.49485.sh

./l_oneapi_vtune_p_2023.2.0.49485.sh

module load vtune

echo "VTUNE VERSION"

which vtune

echo "Intel VTune Profiler installation completed."
```

The shell script of Intel VTune

user24@shavak:~/project/vtune_tool

```
#!/bin/bash

module load vtune/2023.0.0

echo "enter the events you want to profile
      (hotspots, memory-consumption, hpc-performance):"
read events

echo "enter the path to the application you want to profile:"
read app_path

echo "enter the path to the initial result directory:"
read result_dir

if [ ! -z "$(ls -A $result_dir)" ]; then

    echo "Initial result directory is not empty."

    count=1
    while true; do
        new_result_dir="${result_dir}_${count}"
        if [ ! -d "$new_result_dir" ]; then
            result_dir="$new_result_dir"
            mkdir -p "$result_dir"
            break
        fi
        ((count++))
    done
fi

vtune_cmd="vtune -collect $events -result-dir $result_dir -- $app_path"

echo "$vtune_cmd"

echo "profiling results are available in $result_dir"

res= $result_dir

vtune-gui $res
```

Output


Intel VTune Profiler (on shavak)
Project Navigator
sample (matrix)
r000hs
r001ue
r002ps
r003ps
r004hs
r005hs
r006hs
r007hs
r008hs
r009hs
r010hs
Welcome
Configure Analysis
Configure Analysis
INTEL VTUNE PROFILER
WHERE
Local Host
WHAT
Launch Application
Specify and configure your analysis target: an application or a script to execute. Follow Prepare Application for Analysis to compile your app for best analysis productivity.
Application:
/home/user24/intel/vtune/samples/matrix/matrix
Application parameters:
Use application directory as working directory
Advanced
HOW
Microarchitecture Exploration
Performance Snapshot
ALGORITHM
Hotspots
Anomaly Detection (preview)
Memory Consumption
MICROARCHITECTURE
Microarchitecture Exploration
Memory Access
PARALLELISM
Threading
HPC Performance Characterization
I/O
Input and Output
ACCELERATORS
GPU Offload
GPU Compute/Media Hotspots (preview)
CPU/FPGA Interaction
PLATFORM ANALYSES
System Overview
GPU Rendering (preview)
Platform Profiler

Intel VTune Profiler (on shavak)

Project Navigator

- sample (matrix)
  - r000hs
  - r001ue
  - r002ps
  - r003ps
  - r004hs
  - r005hs
  - r006hs
  - r007hs
  - r008hs
  - r009hs
  - r010hs
  - r011hs

Welcome | r011hs

Hotspots

INTEL VTUNE PROFILER

Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Flame Graph | Platform

Elapsed Time: 7.668s

- CPU Time: 107.959s
- Total Thread Count: 17
- Paused 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	% of CPU Time
multiply1	matrix	107.939s	100.0%
pthread_create	libpthread.so.0	0.010s	0.0%
init_arr	matrix	0.010s	0.0%

*N/A is applied to non-summable metrics.

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the Bottom-up view for in-depth analysis per function. Otherwise, use the Caller/Callee or the Flame Graph view to track critical paths for these hotspots.

Explore Additional Insights

Parallelism : 44.0%
Use Threading to explore more opportunities to increase parallelism in your application.

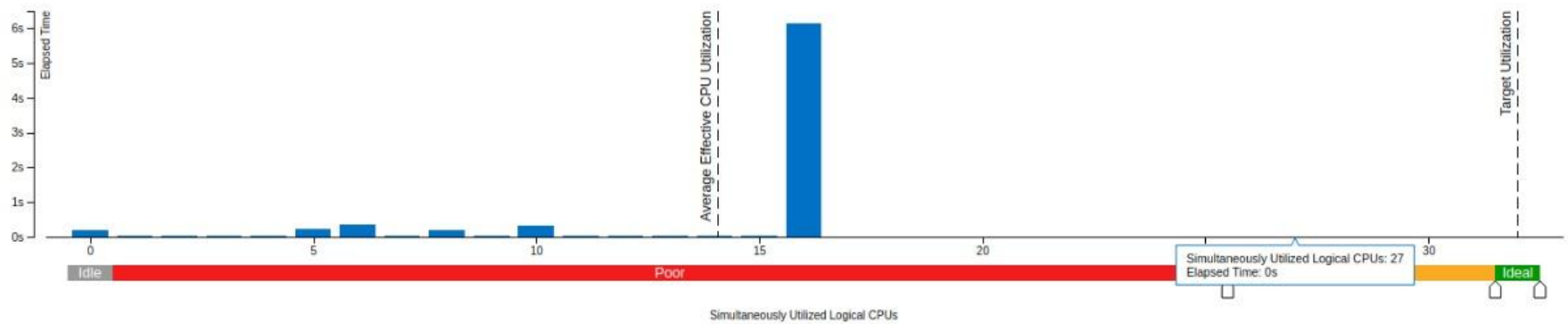
Microarchitecture Usage : 2.6%
Use Microarchitecture Exploration to explore how efficiently your application runs on the used hardware.

Vectorization : 0.1%
Use HPC Performance Characterization to learn more on vectorization efficiency of your application. A significant fraction of floating point arithmetic instructions are scalar. Use Intel Advisor to see possible reasons why the code was not vectorized.

INSIGHTS

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.



Intel VTune Profiler (on shavak)

Project Navigator

- sample (matrix)
  - r000hs
  - r001ue
  - r002ps
  - r003ps
  - r004hs
  - r005hs
  - r006hs
  - r007hs
  - r008hs
  - r009hs
  - r010hs
  - r011hs

Welcome | r011hs

Hotspots

INTEL VTUNE PROFILER

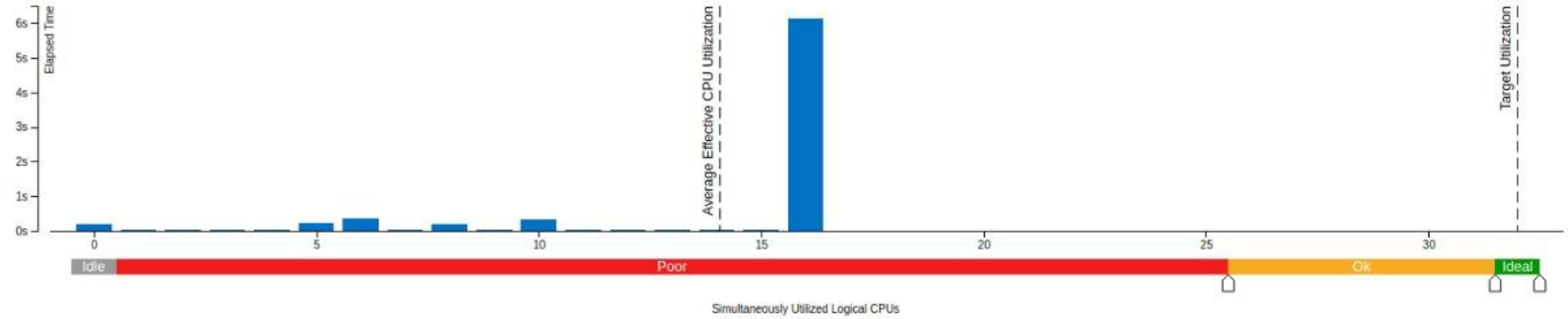
Analysis Configuration | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Flame Graph | Platform

*N/A is applied to non-summable metrics.

possible reasons why the code was not vectorized.

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.



Collection and Platform Info

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

Application Command Line: /home/user24/intel/vtune/samples/matrix/matrix
Operating System: 3.10.0-1160.el7.x86_64 \S Kernel \r on an \m
Computer Name: shavak
Result Size: 5.6 MB
Collection start time: 04:14:18 30/08/2023 UTC
Collection stop time: 04:14:26 30/08/2023 UTC
Collector Type: Driverless Perf per-process counting,User-mode sampling and tracing

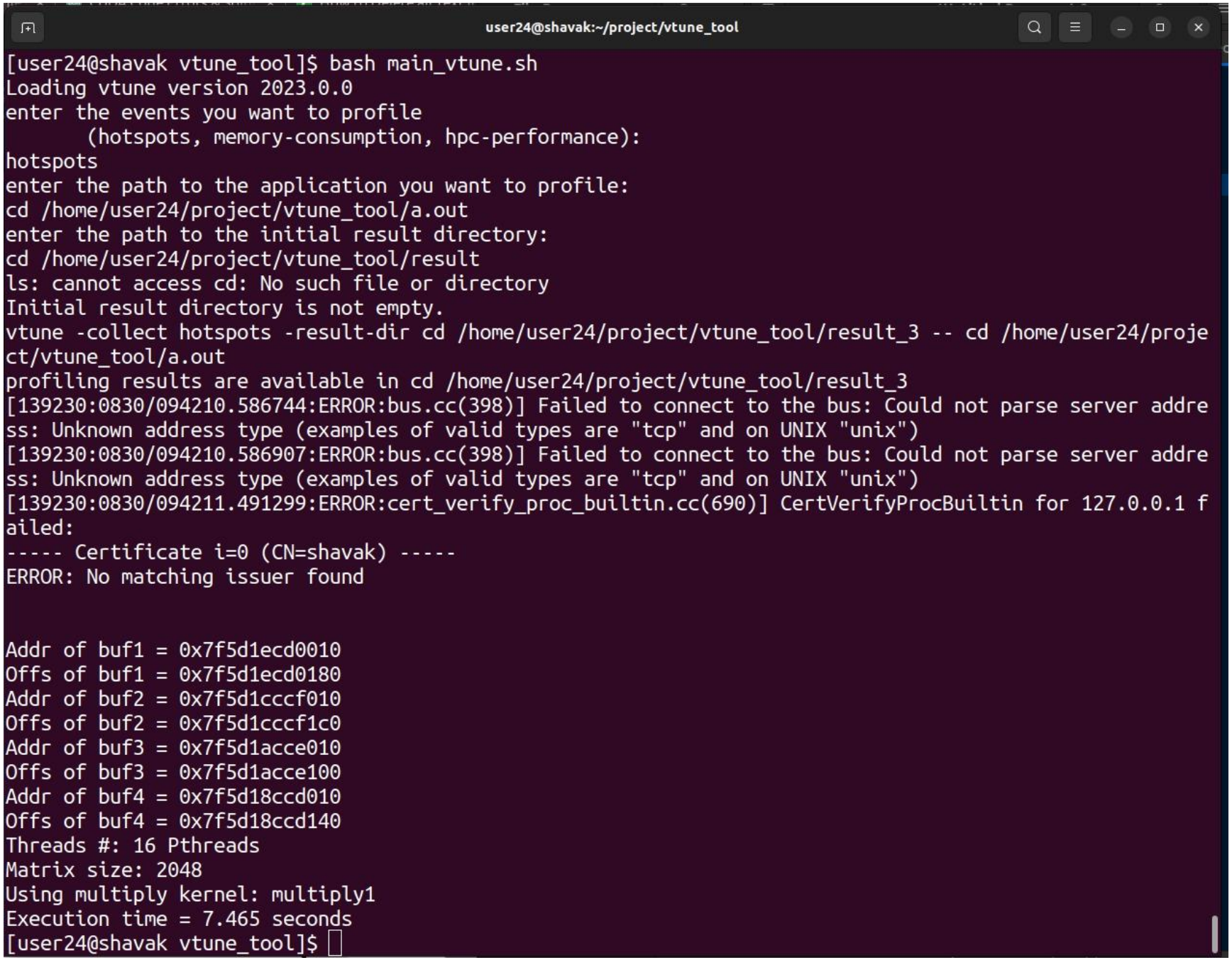
Finalization mode: Fast. If the number of collected samples exceeds the threshold, this mode limits the number of processed samples to speed up post-processing.

CPU

Name: Intel(R) Xeon(R) Processor code named Cascadelake
Frequency: 2.9 GHz
Logical CPU Count: 32
LLC size: 23.1 MB

Cache Allocation Technology

Level 2 capability: not detected
Level 3 capability: available


user24@shavak:~/project/vtune_tool
[user24@shavak vtune_tool]$ bash main_vtune.sh
Loading vtune version 2023.0.0
enter the events you want to profile
(hotspots, memory-consumption, hpc-performance):
hotspots
enter the path to the application you want to profile:
cd /home/user24/project/vtune_tool/a.out
enter the path to the initial result directory:
cd /home/user24/project/vtune_tool/result
ls: cannot access cd: No such file or directory
Initial result directory is not empty.
vtune -collect hotspots -result-dir cd /home/user24/project/vtune_tool/result_3 -- cd /home/user24/proje
ct/vtune_tool/a.out
profiling results are available in cd /home/user24/project/vtune_tool/result_3
[139230:0830/094210.586744:ERROR:bus.cc(398)] Failed to connect to the bus: Could not parse server addre
ss: Unknown address type (examples of valid types are "tcp" and on UNIX "unix")
[139230:0830/094210.586907:ERROR:bus.cc(398)] Failed to connect to the bus: Could not parse server addre
ss: Unknown address type (examples of valid types are "tcp" and on UNIX "unix")
[139230:0830/094211.491299:ERROR:cert_verify_proc_builtin.cc(690)] CertVerifyProcBuiltin for 127.0.0.1 f
ailed:
----- Certificate i=0 (CN=shavak) -----
ERROR: No matching issuer found
Addr of buf1 = 0x7f5d1ecd0010
Offs of buf1 = 0x7f5d1ecd0180
Addr of buf2 = 0x7f5d1cccf010
Offs of buf2 = 0x7f5d1cccf1c0
Addr of buf3 = 0x7f5d1acce010
Offs of buf3 = 0x7f5d1acce100
Addr of buf4 = 0x7f5d18ccd010
Offs of buf4 = 0x7f5d18ccd140
Threads #: 16 Pthreads
Matrix size: 2048
Using multiply kernel: multiply1
Execution time = 7.465 seconds
[user24@shavak vtune_tool]$

TAU

What is TAU?

- TAU(Tuning and Analysis Utilities) is a comprehensive profiling and tracing toolkit.
- TAU Performance System is a portable profiling and tracing toolkit for performance analysis of parallel programs written in Fortran, C, C++, UPC(Universal Product Code), Java, Python.
- TAU can automatically instrument your source code using a package called PDT for routines, loops, I/O, memory, phases, etc.
- TAU runs on most High Performance Computing platforms and it is free
- TAU provides a comprehensive set of profiling and analysis capabilities that enable developers to gain deep insights into their code execution and identify areas for improvement. By utilizing TAU, developers can optimize their code, reduce execution time, and make the most efficient use of available resources.

TAU Architecture and Workflow

Instrumentation: Add probes to perform measurements

• Source code instrumentation using pre-processors and compiler scripts

• Wrapping external libraries (I/O, MPI, Memory, CUDA, OpenCL, pthread)

• Rewriting the binary executable

Measurement: Profiling or tracing using various metrics

• Direct instrumentation (Interval events measure exclusive or inclusive duration)

• Indirect instrumentation (Sampling measures statement level contribution)

• Throttling and runtime control of low-level events that execute frequently

• Per-thread storage of performance data

• Interface with external packages (e.g. PAPI how performance counter library)

Analysis: Visualization of profiles and traces

• 3D visualization of profile data in paraprof tool

• Trace conversion & display in external visualizers (Vampir, Jumpshot, ParaVer)

Key Features:

- The TAU (Tuning and Analysis Utilities) toolkit offers several key features and functionalities that make it a powerful tool for profiling and analyzing the performance of parallel and distributed applications. Some of its features are:
- Support for Multiple Programming Models: TAU supports various programming models and environments commonly used in HPC, including MPI, OpenMP, CUDA, UPC, SHMEM, and more.
- Instrumentation Flexibility: TAU provides flexible instrumentation options. It supports both source code instrumentation and binary instrumentation techniques.
- Comprehensive Performance Data Collection: TAU collects a wide range of performance data during application execution. It captures metrics such as CPU time, memory usage, I/O operations, synchronization overheads, communication patterns, and more

TAU Shell Script

```
#!/bin/bash

#source /home/user24/project/spack/share/spack/setup-env.sh

#source /path/to/your/spack/share/spack/setup-env.sh

#spack load tau
#export PATH=/home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/tau-2.30.2-n2jpt6v25fkzhf53u5gfs2oes2v52kay:$PATH

export PATH=/home/user24/project/spack/opt/spack/linux-centos7-haswell/gcc-4.8.5/tau-2.32-3glvqcdlatqmxlpoyey6zmtd7qkbszgb/bin:$PATH

export PATH=/home/user24/project/spack/opt/spack/linux-centos7-haswell/gcc-4.8.5/tau-2.32-3glvqcdlatqmxlpoyey6zmtd7qkbszgb/lib:$PATH

export PATH=/home/user24/project/spack/opt/spack/linux-centos7-haswell/gcc-4.8.5/tau-2.32-3glvqcdlatqmxlpoyey6zmtd7qkbszgb/include:$PATH

echo "Select an option:"
echo "1. Run pprof"
echo "2. Launch paraprof"
echo "3. Manage taudb"
echo "4. Launch taudb_gui"
read -p "Enter the number of your choice: " choice

case $choice in
    1)
        echo "Enter a path: "
        read user_input
        pprof "$user_input"
        ;;
    2)
        paraprof &
        ;;
    3)
        taudb_manage
        ;;
    4)
        taudb_gui &
        ;;
    *)
        echo "Invalid choice"
        exit 1
        ;;
esac

~
~
~
~
~
~
~
~
~
~
~
"Tau_script_1.sh" 43L, 1150C                                                    27,9          All
```

LIKWID

- Likwid is a widely used tool in the field of High-Performance Computing (HPC) that helps in performance analysis and optimization of parallel applications.

- Likwid stands for "Like I Knew What I'm Doing" and it provides a set of command-line utilities and a programming API for accessing hardware performance counters and other low-level performance metrics.

- The Likwid tool is primarily focused on analyzing the performance of x86-based processors, especially those with support for Performance Monitoring Units (PMUs).

- It allows users to gather detailed information about the performance characteristics of their applications by measuring metrics such as cache utilization, instruction counts, floating-point operations, memory bandwidth, and many others.

Some key features and functionalities of Likwid include

Hardware Performance Counters: Likwid enables users to access and utilize the hardware performance counters available on modern processors. These counters can provide insights into low-level performance events and bottlenecks.

CPU Profiling: The tool allows for profiling the performance of individual processor cores, including metrics like instruction and cycle counts, cache misses, and pipeline stalls.

Memory Profiling: Likwid can analyze the memory behavior of an application, providing information on memory bandwidth usage, cache utilization, and cache misses.

Thread Analysis: It offers features for profiling and analyzing the behavior of multithreaded applications, such as measuring thread synchronization overhead and identifying load imbalances.

Energy Measurement: Likwid can also measure the energy consumption of an application, providing insights into power efficiency and optimization opportunities.

The LIKWID suite includes various command that help in different aspects of performance analysis:

likwid-topology : Display the thread and cache topology on multicore/multisocket computers

likwid-perfctr : Count hardware performance events. It can be used as wrapper application, which does not require modification of the code to be analyzed

likwid-pin : Pin the threads of an application without changing the code. Works for OpenMP, C++11 threads, pthreads, etc.

likwid-bench : A benchmarking framework that allows rapid prototyping of threaded assembly kernels.

likwid-powermeter : Access RAPL counters (for energy measurements) .

likwid-setFrequencies : Set the clock frequency of CPU cores.

```
#!/bin/bash
spack load
module load likwid
source /home/user24/spack/opt/spack/linux-centos7-haswell/gcc-4.8.5/likwid-5.2.2-woqqbjbz6itml3tre3a7id7q5nqsu6z6
source /home/user24/spack/share/spack/setup-env.sh

export PATH=/home/user24/spack/opt/spack/linux-centos7-haswell/gcc-4.8.5/likwid-5.2.2-woqqbjbz6itml3tre3a7id7q5nqsu6z6/bin:$PATH

# Load LIKWID environment
module load likwid

# Function to print available LIKWID tools
while true; do
print_likwid_tools() {


    echo "Available LIKWID tools:"
    echo "1. likwid-topology"
    echo "2. likwid-pin"
    echo "3. likwid-perfctr"
    echo "4. likwid-bench"
    echo "5. Exit"

}

print_likwid_tools
read -p " Choose the Option from above: " tool_choice

read -p "Enter the executable file name: " fname


case $tool_choice in

    1)
        likwid-topology -G
continue;;
    2)
        likwid-pin -c 0,2,4-6 ./$fname
         #likwid-pin -c 3,4,5,6 -s 0x1 ./$fname

continue;;
    3)
        likwid-perfctr  -C S0:1  -g BRANCH  ./$fname
continue;;
    4)
        likwid-bench  -t copy -w S1:100kB
continue;;
    5)
        echo "Exiting...."
break;;
    *)
        echo "Invalid Choice"

esac
                                                                    29,1          Top
```

Conclusion

- In conclusion, the development of our profiling tool was an enlightening experience that allows us to develop one profiling toolkit that combines the features of five different toolkits we have used.
- The choice of a profiling tool depends on the specific goals, application domain, and hardware platform. Developers seeking to optimize performance should consider the features of these tools and choose the one that aligns best with their requirements.
- The project is not only expanded our understanding of profiling techniques but also the importance of selecting the right tool for the task.
- The ongoing evolution of these tools ensures that software developers have robust toolkit to analyze, optimize, and enhance the performance of their applications in an increasingly demanding computing landscape.

Main .sh

user24@shavak:~/prn

```
#!/bin/bash

set -e

select_profiling_Tool_Choice() {
    echo "Select a profiling Tool option:"

    echo "1. Intel Offload advisor"
    echo "2. Intel VTune"
    echo "3. HPCToolkit"
    echo "4. TAU"
    echo "5. Likwid"
    echo "6. Exit"
    read -p "Enter your choice: " choice
}


while true; do
    select_profiling_Tool_Choice

    case $choice in
        1)
            /home/user24/prn/IntelAdvisor_offload_script.sh

            ;;
        2)
            /home/user24/project/vtune_tool/main_vtune.sh

            ;;
        3)
            /home/apps/spack/opt/spack/linux-centos7-cascadelake/gcc-11.2.0/hpctoolkit-2021.10.15-xwzccbhlwuknxd3d2oiiok6dmnss3eiv/script/hpctoolkit_script.sh
            ;;
        4)
            /home/user24/project/tau/tau_main.sh

            ;;
        5)
            /home/user24/likwid_Scripts/likwid_script_v3.sh
            ;;
        6)
            echo "Exiting."
            exit
            ;;
        *)
            echo "Invalid choice. Please select a valid option."
            ;;
    esac
done
~
~
~
~
~
~
"all_tool.sh" 48L, 1095C
```

Future Scope

Performance Portability: As software spans various hardware architectures, from traditional CPUs to GPUs, FPGAs, and accelerators, profiling tools will need to offer performance analysis across heterogeneous platforms.

Tools that can analyze code behavior and performance on different architectures will be essential for optimizing applications for various target systems.

Server less and Function-as-a-Service: These tools will help developers optimize server less functions for better execution times and resource utilization.

THANK YOU!