HPC.NRW

# PROFILING WITH INTEL VTUNE

Introduction

August 9, 2023 | Dr. Martin Errenst

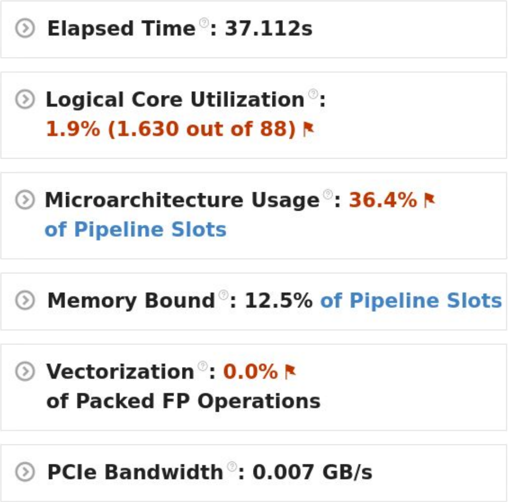
THE COMPETENCE NETWORK FOR HIGH-PERFORMANCE COMPUTING IN NRW.

# PROFILING WITH INTEL VTUNE

- *Getting started* with Intel VTune profiler
- What it is and when to use it
- Be able to profile an application and start interpreting results

# WHAT DOES VTUNE DO?

- Provides insight into application, for example
  - Find *hotspots*
  - Parallel performance
  - Find cache misses



Elapsed Time: 37.112s

Logical Core Utilization: 1.9% (1.630 out of 88)

Microarchitecture Usage: 36.4% of Pipeline Slots

Memory Bound: 12.5% of Pipeline Slots

Vectorization: 0.0% of Packed FP Operations

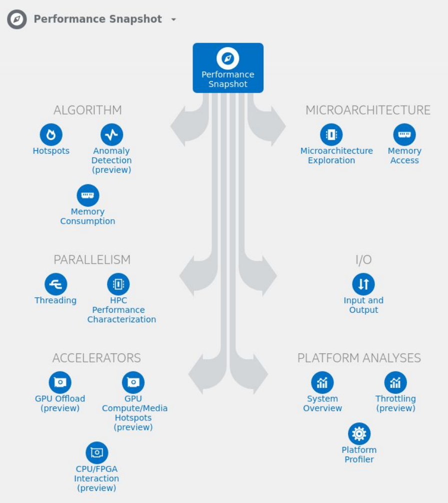
PCIe Bandwidth: 0.007 GB/s

# WHAT IS VTUNE?

- Profiler for single- and multithreaded applications
  - Mostly “Node-level” profiling
  - CPUs, GPUs, FPGAs
  - Linux, Windows, Android, FreeBSD

# ANALYSIS TYPES

- Multiple analysis types with different focus
  - Collect different data
  - Results focus on specific topic
  - May require specific kernel module and root
  - Varying execution/memory overhead
- Introduced overhead in single digit % range



# HOW DOES IT WORK?

- Event based sampling profiler
  - Regular sampling of performance counters during program execution
  - Attributing measurements to active program section
  - Statistical interpretation

What are *Hardware Events*?

- Hardware counters in CPU
- Occurrence operations and conditions, e.g. cycles with cache misses
- Events are combined to summarizing *metrics*, e.g. “Memory Bound”

# WHEN TO USE IT?

- Start with an observation:
  - “My application is too slow on this machine”
  - “My applications behavior is strange sometimes”
    (dependent on #threads, memory usage, execution time)

# PREPARE MEASUREMENTS

- Find a suitable test case
  - As small as possible, as large as necessary
    ⇒ Short iteration time in *optimization cycle*
  - Large examples can produce too much data
  - You can always sample small section in large examples

# OPTIMIZATION CYCLE

1. Observe issue (e.g. execution time too long)
2. Define test case
3. Profile test case & compare to previous result
4. Identify problematic section (**difficult part**!)
   - e.g. inefficient computing, throughput issues
5. Implement fix and go to 3.
   - Or break when happy

**VTune helps here**

# CREATING A FIRST PROFILE

Dr. Martin Errenst

# PROFILING WITH INTEL VTUNE

# INSTALLATION

- Assuming working installation of oneAPI, or the predecessor Parallel Studio
- Consult the oneAPI installation guide
- Look up in your cluster documentation or ask your administrator
- Usually sourcing a setup script or loading an environment module

# USING THE KERNEL MODULE

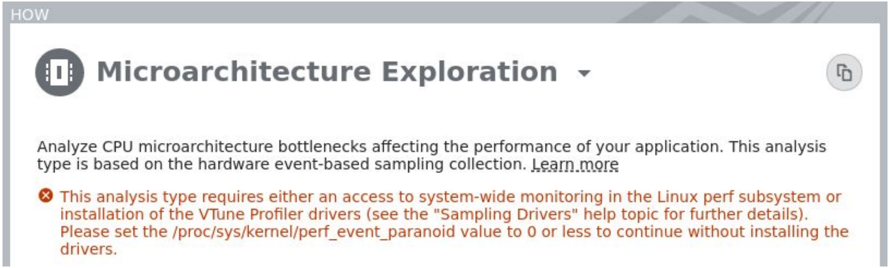
## More detailed profiles

- Driverless profiling is possible, if Linux perf available
- Otherwise sampling driver required, e.g. for
  - Disabled perf collection by administrator
  - Too new hardware
  - Very old kernels
  - Non linux-systems
- Requires root permissions during installation

# USING THE KERNEL MODULE

## Warnings in GUI

VTune will tell you if it is not available!



Microarchitecture Exploration

Analyze CPU microarchitecture bottlenecks affecting the performance of your application. This analysis type is based on the hardware event-based sampling collection. Learn more

This analysis type requires either an access to system-wide monitoring in the Linux perf subsystem or installation of the VTune Profiler drivers (see the "Sampling Drivers" help topic for further details). Please set the /proc/sys/kernel/perf_event_paranoid value to 0 or less to continue without installing the drivers.

# USING THE KERNEL MODULE

## Self checker script

**Check if it works**: `<install-dir>/bin64/vtune-self-checker.sh`

```
[...]
The system is ready for the following analyses:
* Performance Snapshot
* Hotspots and Threading with user-mode sampling
* Hotspots with HW event-based sampling, HPC Performance Characterization, etc.
* Microarchitecture Exploration
* Memory Access
* Hotspots with HW event-based sampling and call stacks
* Threading with HW event-based sampling

The following analyses have failed on the system:
* GPU Compute/Media Hotspots (characterization mode)
* GPU Compute/Media Hotspots (source analysis mode)
```

# COMPILING THE CODE

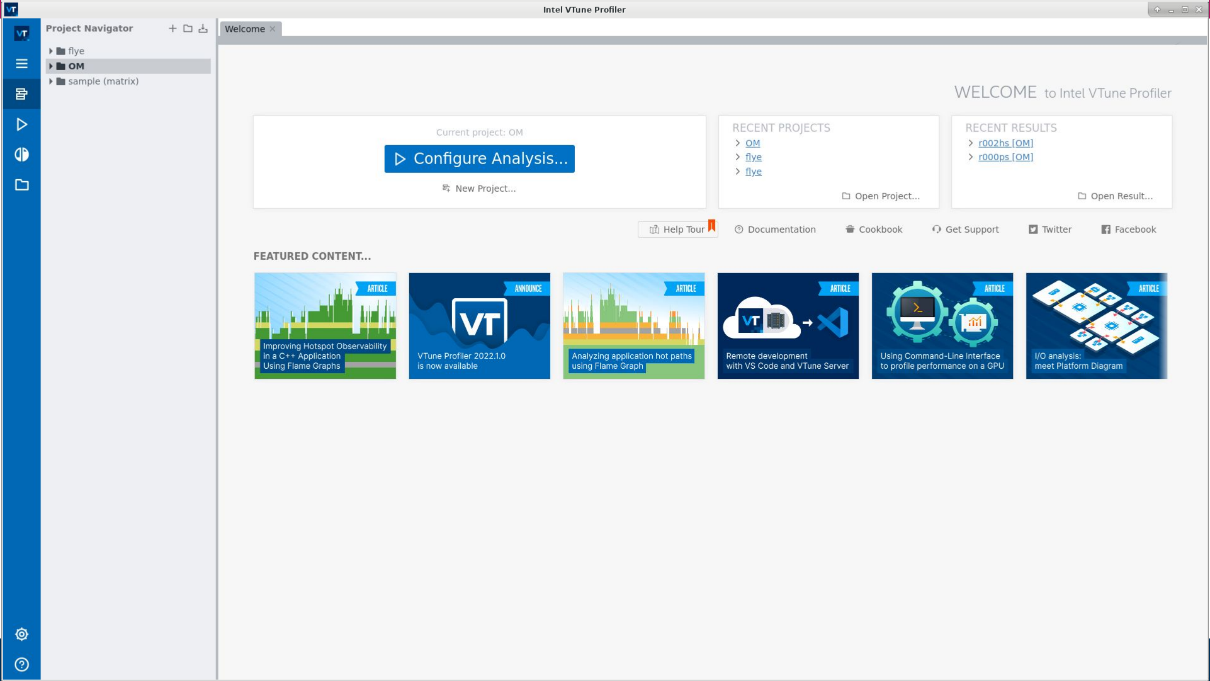
- Use `-g` to generate debug information
  - Allows for association between metrics and source code
- Release build to measure the real thing
  - E.g. `RelWithDebInfo` in CMake projects
- Build with Intel instructions

# OUR FIRST PROFILE

- Start VTune with `vtune-gui`
- Setup new project
- Select application and analysis type

```
// Repetitions for larger workload
// #pragma omp parallel for
for(size_t j = 0; j < repetitions; j++){
   std::vector<float> v3(vsize), v4(vsize);

   // add and multiply random vectors
   //#pragma omp simd
   for(size_t i = 0; i < vsize; i++){
      v3[i] = v1[i] + v2[i];
      v4[i] = v1[i] * v2[i];
   }
}
```



Intel VTune Profiler

Project Navigator

- flye
- OM
- sample (matrix)

Welcome

WELCOME to Intel VTune Profiler

Current project: OM

▷ Configure Analysis...
New Project...

RECENT PROJECTS

- OM
- flye
- flye

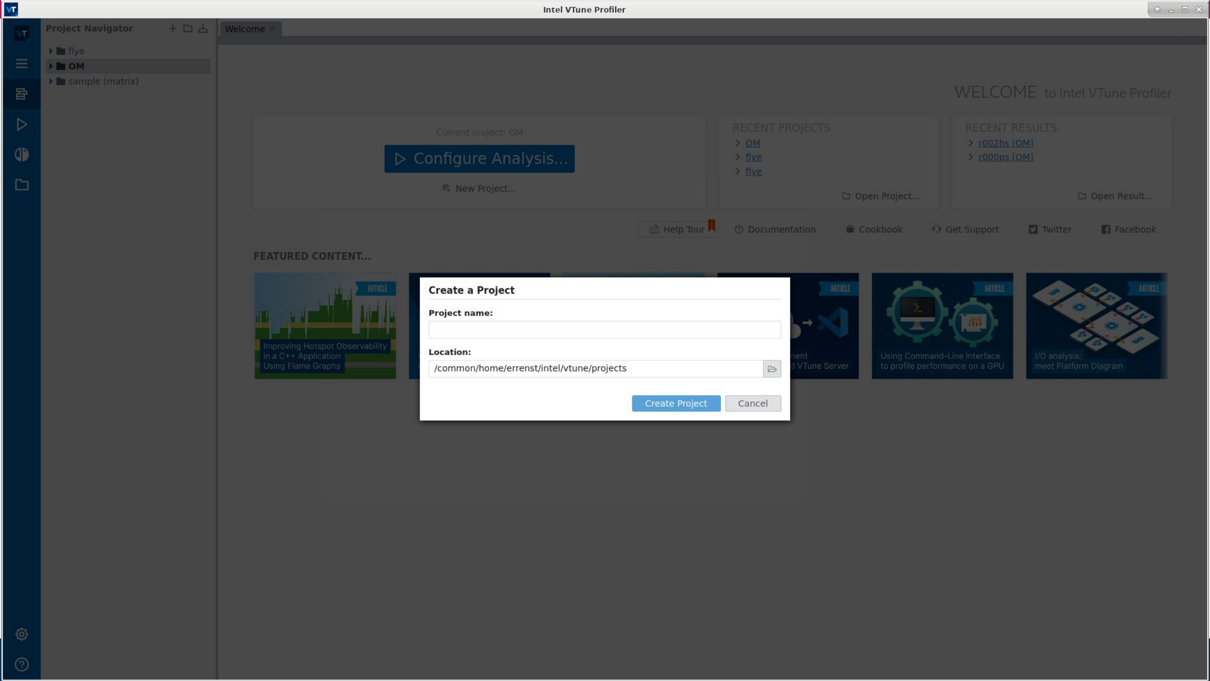
RECENT RESULTS

- r002hs [OM]
- r000ps [OM]

Open Result...
Help Tour
Documentation
Cookbook
Get Support
Twitter
Facebook

FEATURED CONTENT...

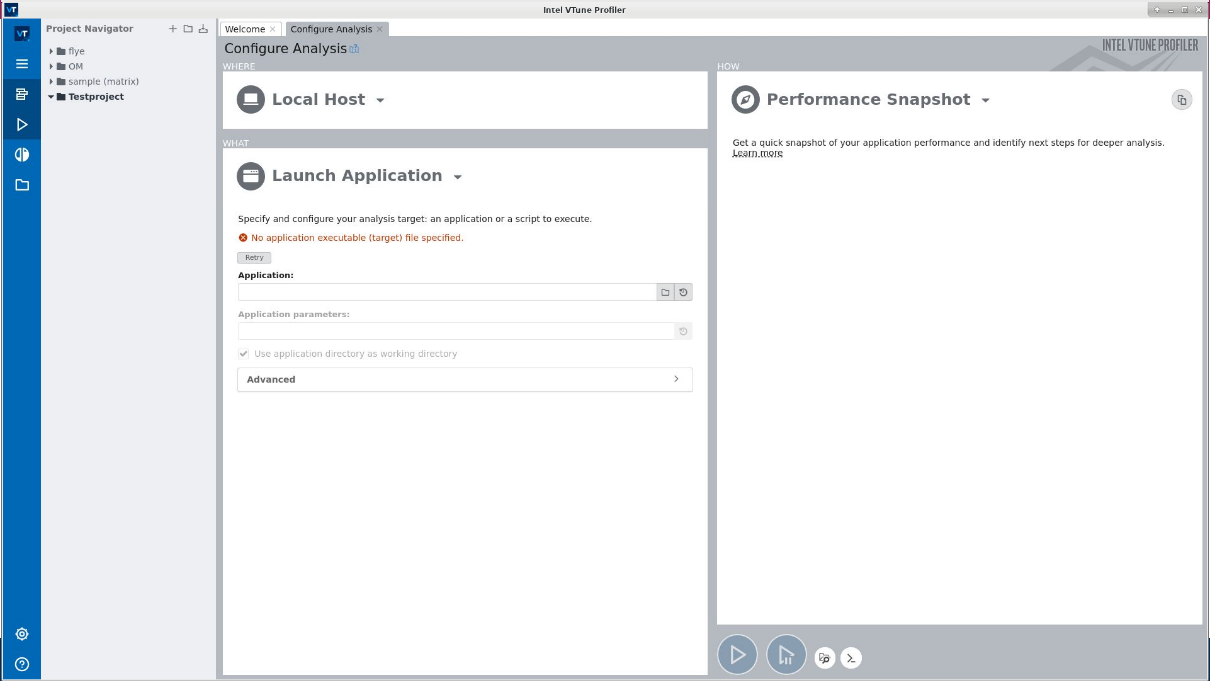
ARTICLE
Improving Hotspot Observability in a C++ Application Using Flame Graphs
ANNOUNCE
VTune Profiler 2022.1.0 is now available
ARTICLE
Analyzing application hot paths using Flame Graph
ARTICLE
Remote development with VS Code and VTune Server
ARTICLE
Using Command-Line Interface to profile performance on a GPU
ARTICLE
I/O analysis: meet Platform Diagram



**Create a Project**

**Project name:**

**Location:** /common/home/errenst/intel/vtune/projects



Configure Analysis

WHERE

Local Host

WHAT

Launch Application

Specify and configure your analysis target: an application or a script to execute.

No application executable (target) file specified.

Application:

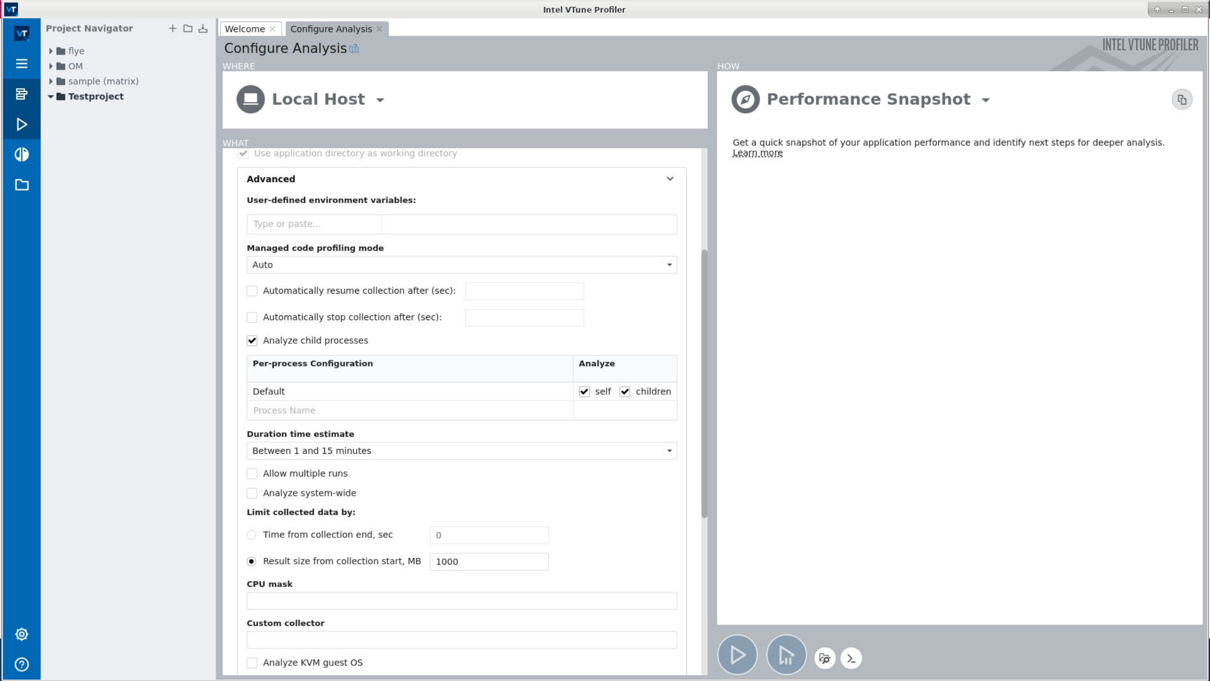
Application parameters:

☑ Use application directory as working directory

HOW

Performance Snapshot

Get a quick snapshot of your application performance and identify next steps for deeper analysis. Learn more



Configure Analysis

Local Host

Performance Snapshot

Get a quick snapshot of your application performance and identify next steps for deeper analysis. Learn more

☑ Use application directory as working directory

**Advanced**

**User-defined environment variables:**

Type or paste...

**Managed code profiling mode**

Auto

□ Automatically resume collection after (sec):

□ Automatically stop collection after (sec):

☑ Analyze child processes

| Per-process Configuration | Analyze |
|---|---|
| Default | ☑ self ☑ children |
| Process Name | |

**Duration time estimate**

Between 1 and 15 minutes

□ Allow multiple runs

□ Analyze system-wide

**Limit collected data by:**

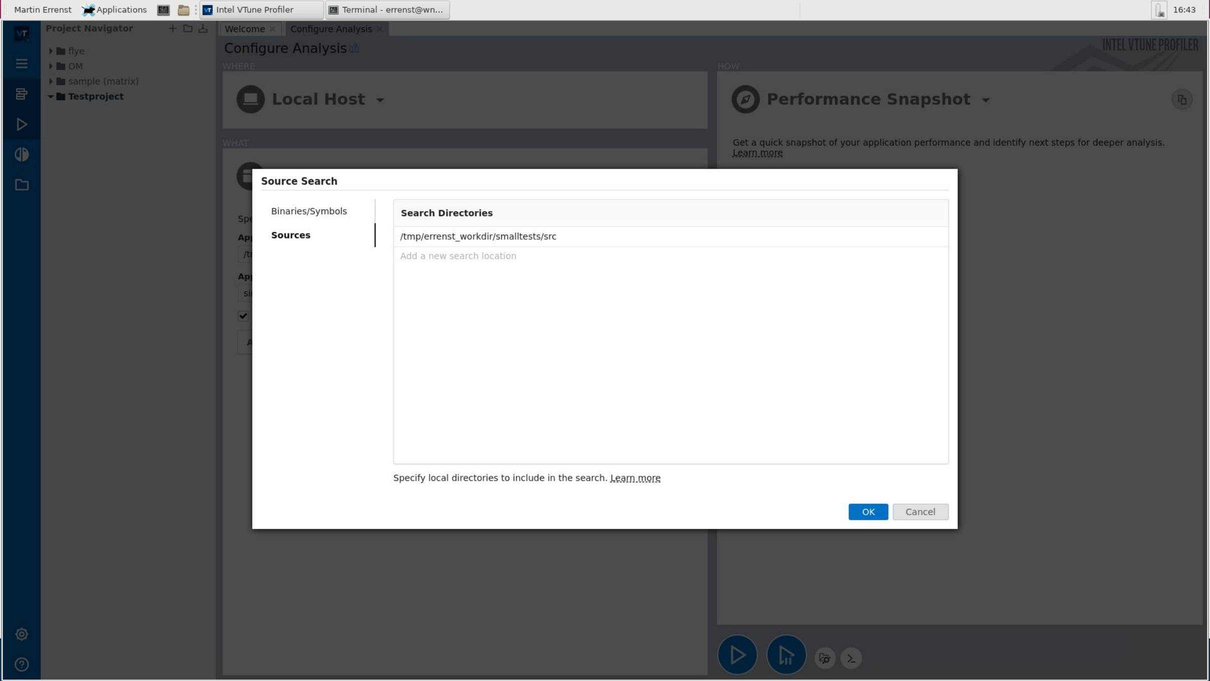
○ Time from collection end, sec 0

● Result size from collection start, MB 1000

**CPU mask**

**Custom collector**

□ Analyze KVM guest OS



Configure Analysis

Local Host

Performance Snapshot

Get a quick snapshot of your application performance and identify next steps for deeper analysis. Learn more

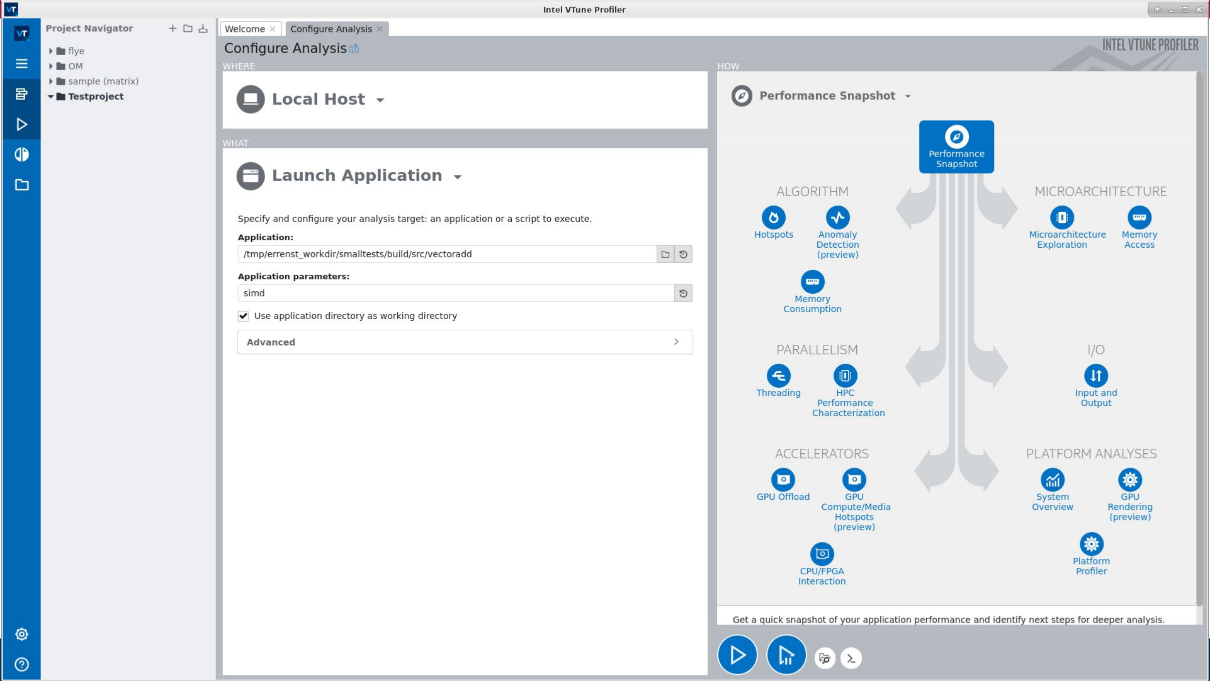
Source Search

Search Directories

/tmp/errenst_workdir/smalltests/src

Add a new search location

Specify local directories to include in the search. Learn more



Configure Analysis

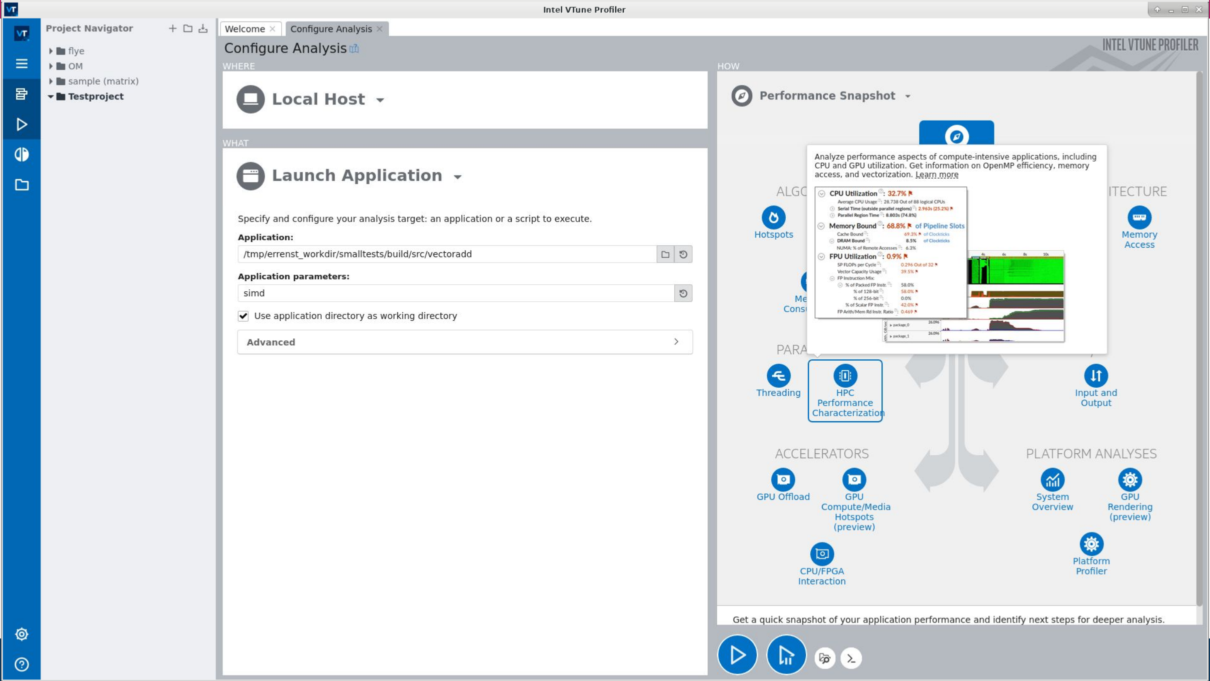
Specify and configure your analysis target: an application or a script to execute.

**Application:** /tmp/errenst_workdir/smalltests/build/src/vectoradd

**Application parameters:** simd

☑ Use application directory as working directory

Get a quick snapshot of your application performance and identify next steps for deeper analysis.



# Configure Analysis

Specify and configure your analysis target: an application or a script to execute.

**Application:** /tmp/errenst_workdir/smalltests/build/src/vectoradd

**Application parameters:** simd

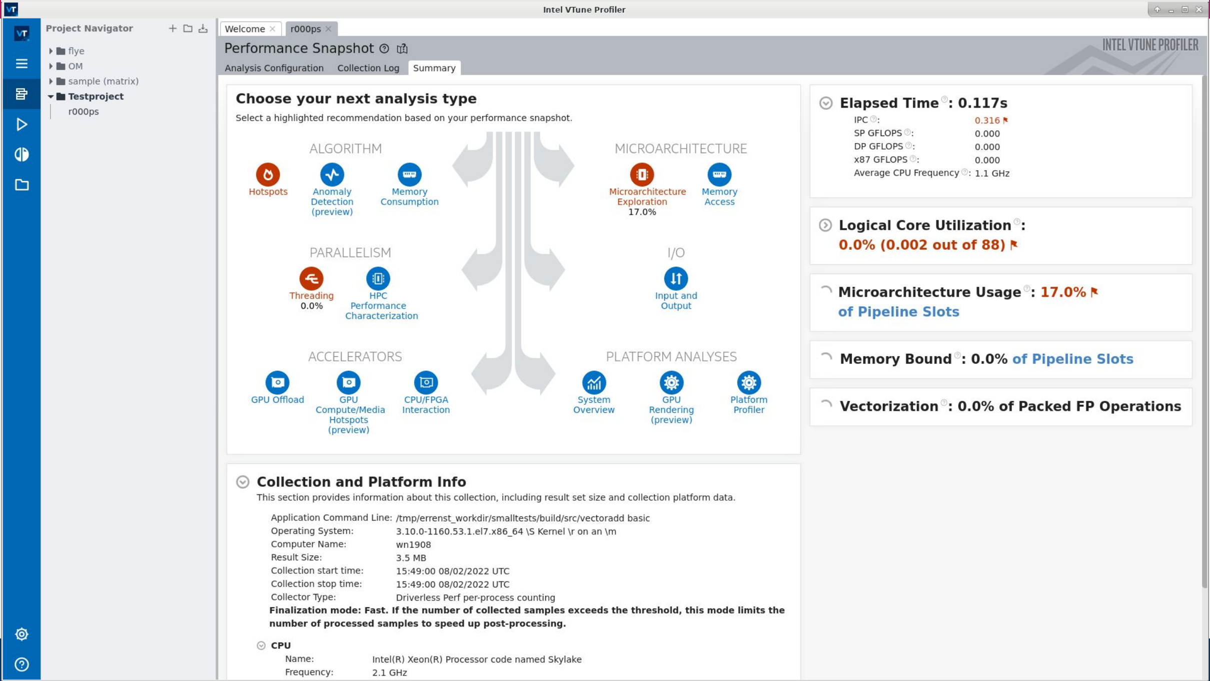
☑ Use application directory as working directory

**Advanced** ›

Analyze performance aspects of compute-intensive applications, including CPU and GPU utilization. Get information on OpenMP efficiency, memory access, and vectorization. Learn more

```
CPU Utilization: 32.7%
    Average CPU Usage: 28.738 Out of 88 logical CPUs
    Serial Time (outside parallel regions): 2.963s (25.2%)
    Parallel Region Time: 8.803s (74.8%)
Memory Bound: 68.8% of Pipeline Slots
    Cache Bound:                        69.3%   of Clockticks
    DRAM Bound:                         8.5%    of Clockticks
    NUMA: % of Remote Accesses:         6.3%
FPU Utilization: 0.9%
    SP FLOPs per Cycle:                 0.296 Out of 32
    Vector Capacity Usage:              39.5%
    FP Instruction Mix:
      % of Packed FP Instr.:            58.0%
        % of 128-bit:                   58.0%
        % of 256-bit:                   0.0%
      % of Scalar FP Instr.:            42.0%
    FP Arith/Mem Rd Instr. Ratio:       0.469
```

Get a quick snapshot of your application performance and identify next steps for deeper analysis.



Performance Snapshot

## Choose your next analysis type

Select a highlighted recommendation based on your performance snapshot.

## Elapsed Time: 0.117s

```
IPC:                        0.316
SP GFLOPS:                  0.000
DP GFLOPS:                  0.000
x87 GFLOPS:                 0.000
Average CPU Frequency:      1.1 GHz
```

## Logical Core Utilization: 0.0% (0.002 out of 88)

## Microarchitecture Usage: 17.0% of Pipeline Slots

## Memory Bound: 0.0% of Pipeline Slots

## Vectorization: 0.0% of Packed FP Operations

## Collection and Platform Info

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

```
Application Command Line:  /tmp/errenst_workdir/smalltests/build/src/vectoradd basic
Operating System:          3.10.0-1160.53.1.el7.x86_64 \S Kernel \r on an \m
Computer Name:             wn1908
Result Size:               3.5 MB
Collection start time:     15:49:00 08/02/2022 UTC
Collection stop time:      15:49:00 08/02/2022 UTC
Collector Type:            Driverless Perf per-process counting
```

**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 Skylake
Frequency:    2.1 GHz
```

# SUMMARY

**What we have learned:**

- Introduction to Intel VTune
- How it works
- When to use it
- How to set up the environment
- Existence of sampling driver
- Producing a first profile

**Next steps:**

- Look at next VTune tutorials
- Apply profiling in real life
- Understand results
- Improve our applications