

Get Started with Intel® VTune™ Profiler



Contents

Chapter 1: Get Started with Intel® VTune™ Profiler

Get Started with Intel® VTune™ Profiler for Windows* OS	5
Example: Profile an OpenMP* Application on Windows*	8
Example: Profile a SYCL* Application on Windows*	9
Get Started with Intel® VTune™ Profiler for Linux* OS	11
Example: Profile an OpenMP Application on Linux*	14
Example: Profile a SYCL* Application on Linux*	16
Get Started with Intel® VTune™ Profiler for macOS*	18
Learn More	21
Notices and Disclaimers.....	21



Get Started with Intel® VTune™ Profiler

1

Use Intel VTune Profiler to analyze local and remote target systems from Windows*, macOS*, and Linux* hosts. Improve application and system performance through these operations:

- Analyze algorithm choices.
- Find serial and parallel code bottlenecks.
- Understand where and how your application can benefit from available hardware resources.
- Speed up the execution of your application.

Download Intel VTune Profiler on your system through one of these ways:

- Download the [Standalone](#) version.
- Get Intel VTune Profiler as part of the [Intel® oneAPI Base Toolkit](#).

See the [VTune Profiler training page](#) for videos, webinars, and more material to help you get started.

NOTE

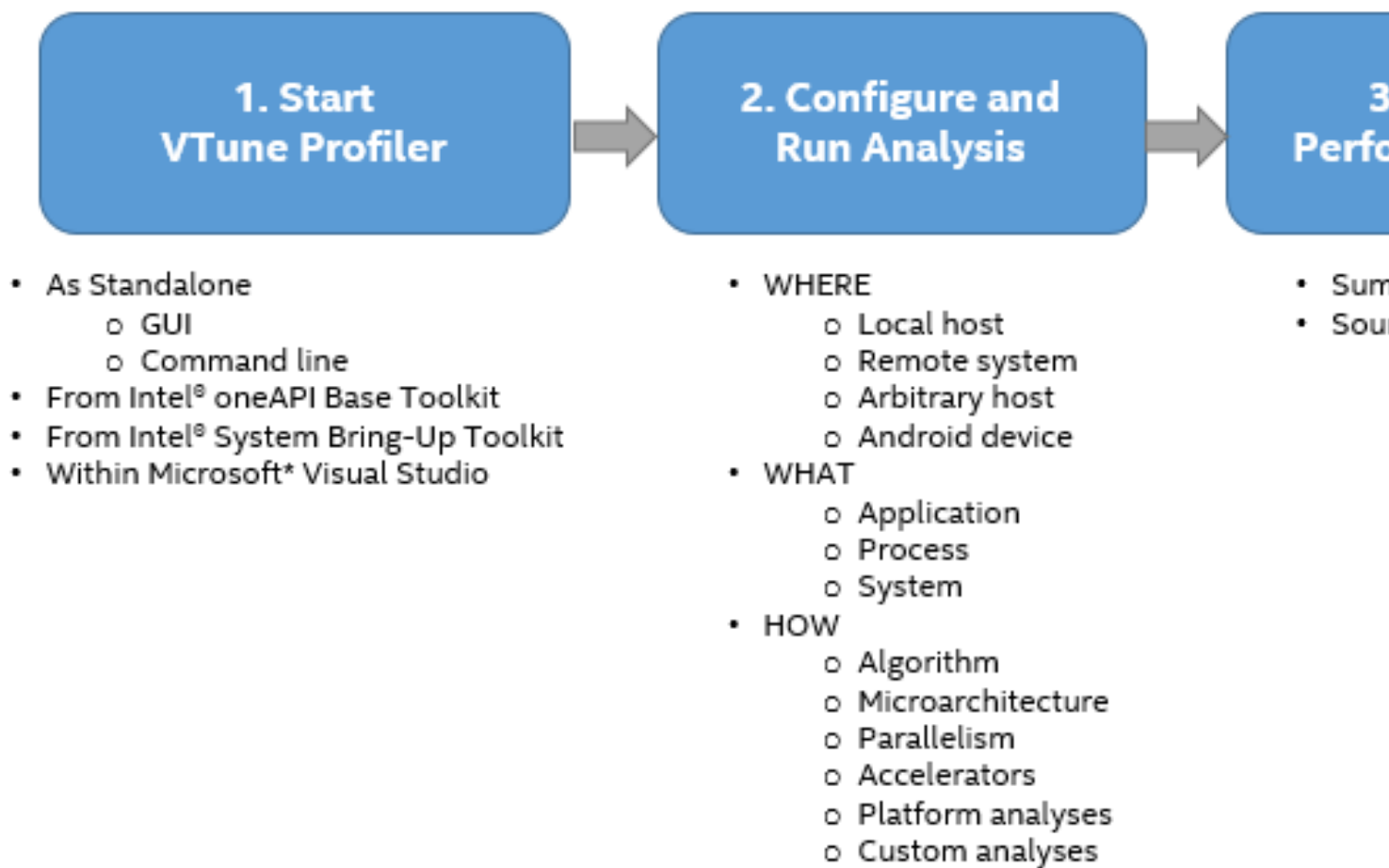
Documentation for versions of Intel® VTune™ Profiler prior to the 2021 release are available for download only. For a list of available documentation downloads by product version, see these pages:

- [Download Documentation for Intel Parallel Studio XE](#)
 - [Download Documentation for Intel System Studio](#)
-

Understand the Workflow

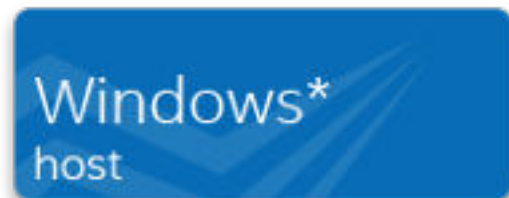
Use Intel VTune Profiler to profile an application and analyze results for performance improvements. The general workflow contains these steps:





Select Your Host System to Get Started

Learn more about system-specific workflows for [Windows*](#), [Linux*](#), or [macOS*](#).



Get Started with Intel® VTune™ Profiler for Windows* OS

Before You Begin

1. Install Intel® VTune™ Profiler on your Windows* system.
2. Build your application with symbol information and in **Release** mode with all optimizations enabled. For detailed information on compiler settings, see the [VTune Profiler online user guide](#).

You can also use the `matrix` sample application available in `<Documents>\VTune\Samples\matrix`. You can see corresponding sample results in `<Documents>\VTune\Projects\sample (matrix)`.

3. Set up the environment variables: Run the `<install-dir>\setvars.bat` script.

By default, the `<install-dir>` for oneAPI components is `Program Files (x86)\Intel\oneAPI`.

NOTE You do not need to run `setvars.bat` when using Intel® VTune™ Profiler within Microsoft* Visual Studio*.

Step 1: Start Intel® VTune™ Profiler

Start Intel VTune Profiler through one of these ways and set up a project. A project is a container for the application you want to analyze, the type of analysis, and data collection results.

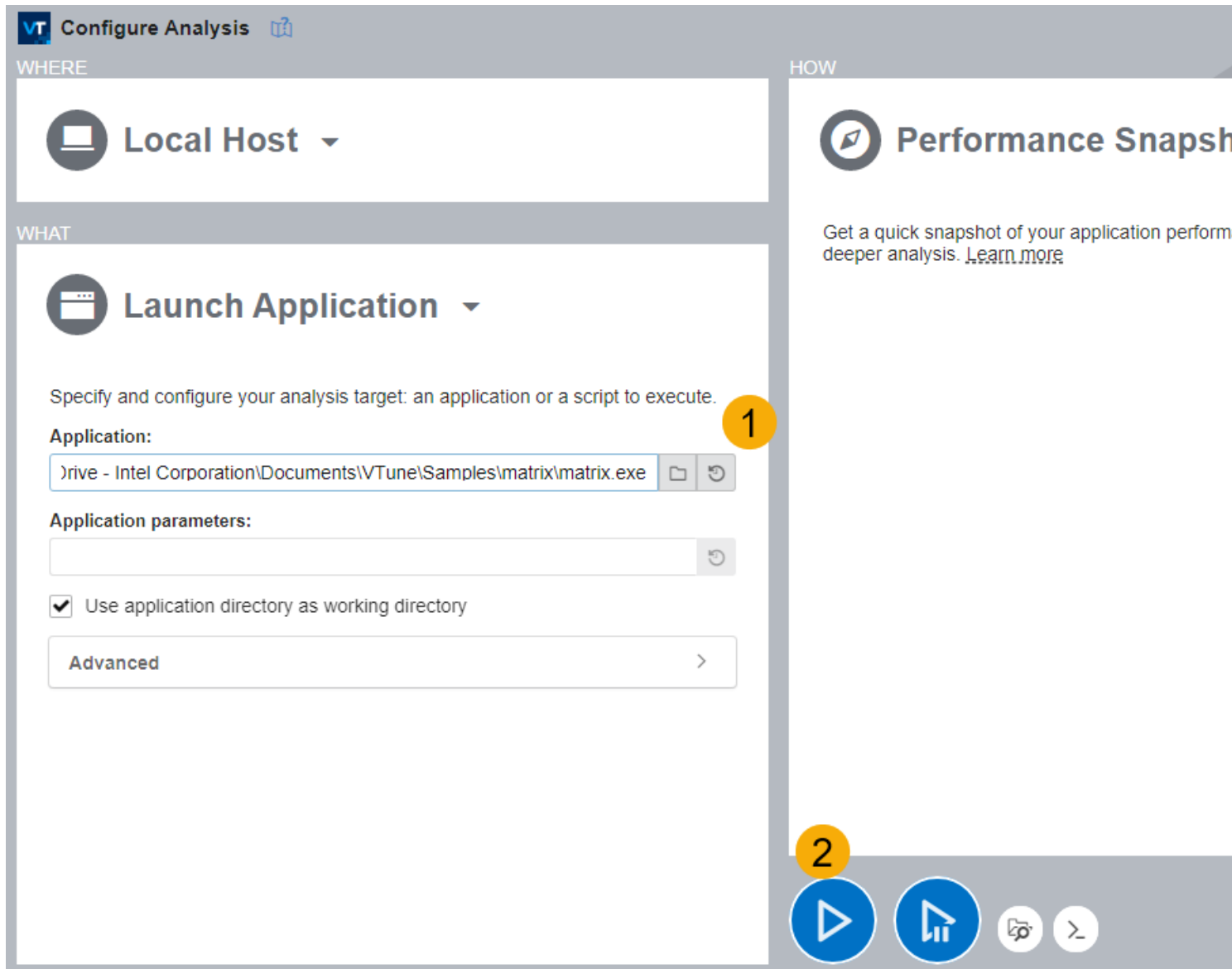
Source	Start VTune Profiler
Standalone (GUI)	1. Run the <code>vtune-gui</code> command or run Intel® VTune™ Profiler from the Start menu. 2. When the GUI opens, click  New Project... in the Welcome screen. 3. In the Create Project dialog box, specify the project name and location. 4. Click Create Project.
Standalone (Command line)	Run the <code>vtune</code> command.
Microsoft* Visual Studio* IDE	Open your solution in Visual Studio. The VTune Profiler toolbar is automatically enabled and your Visual Studio project is set as an analysis target.

NOTE You do not need to create a project when running Intel® VTune™ Profiler from the command line or within Microsoft* Visual Studio.

Step 2: Configure and Run Analysis

After creating a new project, the **Configure Analysis** window opens with these default values:





1. In the **Launch Application** section, browse to the location of your application executable file.
2. Click **Start** to run [Performance Snapshot](#) on your application. This analysis presents a general overview of issues affecting the performance of your application on the target system.

Step 3: View and Analyze Performance Data

When data collection completes, VTune Profiler displays analysis results in the **Summary** window. Here, you see a performance overview of your application.

The overview typically includes several metrics along with their descriptions.



Performance Snapshot Performance Snapshot ▼ ⓘ ⓘ

Analysis Configuration Collection Log Summary

Choose your next analysis type

Select a highlighted recommendation based on your performance snapshot.

ALGORITHM

- Hotspots
- Anomaly Detection (preview)

PARALLELISM

- Threading
- HPC Performance Characterization

ACCELERATORS

- GPU Offload (preview)
- GPU Compute/Media Hotspots (preview)
- CPU/FPGA Interaction (preview)

MICROARCHITECTURE

- Microarchitecture Exploration 20.9%
- Memory Access 47.2%

I/O

- Input and Output (preview)

PLATFORM ANALYSIS

- System Overview
- Throttling (preview)
- Platform Profiler

Elapsed Time ⓘ

- IPC ⓘ
- SP GFLOPS ⓘ
- DP GFLOPS ⓘ
- x87 GFLOPS ⓘ
- Average CPU Frequency ⓘ

Effective Logical Core Utilization ⓘ

97.7% (7.815 out of 8 cores)

Microarchitecture of Pipeline Slots ⓘ

The metric value is high. This can indicate that the significant fraction of execution pipeline slots could be stalled due to demand memory load and stores. Use Memory Access analysis to have the metric breakdown by memory hierarchy, memory bandwidth information, correlation by memory objects.

Vectorization ⓘ

of Packed FP Operations

- Instruction Mix:
 - SP FLOPs ⓘ
 - Packed ⓘ:
 - 128-bit ⓘ
 - 256-bit ⓘ

A

Expand each metric for detailed information about contributing factors.

B

A flagged metric indicates a value outside acceptable/normal operating range. Use tool tips to understand how to improve a flagged metric.

C

See guidance on other analyses you should consider running next. The **Analysis Tree** highlights these recommendations.

Next Steps

Performance Snapshot is a good starting point to get an overall assessment of application performance with VTune Profiler. Next, check if your algorithm requires tuning.

1. Follow a tutorial to analyze common performance bottlenecks.

2. Once your algorithm is well-tuned, run Performance Snapshot again to calibrate results and identify potential performance improvements in other areas.

See Also

[Microarchitecture Exploration](#)

[VTune Profiler Help Tour](#)

Example: Profile an OpenMP* Application on Windows*

Use Intel VTune Profiler on a Windows machine to profile a sample `iso3dfd_omp_offload` OpenMP application offloaded onto an Intel GPU. Learn how to run a GPU analysis and examine results.

Prerequisites

- Make sure your system is running Microsoft* Windows 10 or a newer version.
- Use one of these versions of Intel Processor Graphics:
 - Gen 8
 - Gen 9
 - Gen 11
- Your system should be running on one of these Intel processors:
 - 7th Generation Intel® Core™ i7 Processors (code name Kaby Lake)
 - 8th Generation Intel® Core™ i7 Processors (code name Coffee Lake)
 - 10th Generation Intel® Core™ i7 Processors (code name Ice Lake)
- Install Intel VTune Profiler from one of these sources:
 - [Standalone product download](#)
 - [Intel® oneAPI Base Toolkit](#)
 - [Intel® System Bring-up Toolkit](#)
- Download the [Intel® oneAPI HPC Toolkit](#) which contains the Intel® oneAPI DPC++/C++ Compiler(`icx/iccpx`) that you need to profile OpenMP applications.
- Set up environment variables. Execute the `vars.bat` script located in the `<vtune-install-dir>\env` directory.
- [Set up your system for GPU analysis.](#)

NOTE To install Intel VTune Profiler in the Microsoft* Visual Studio environment, see the [VTune Profiler User Guide](#).

Build and Compile the OpenMP Offload Application

1. Download the [iso3dfd_omp_offload](#) OpenMP Offload sample.
2. Open to the sample directory.

```
cd <sample_dir>/DirectProgramming/C++/StructuredGrids/iso3dfd_omp_offload
```

3. Compile the OpenMP Offload application.

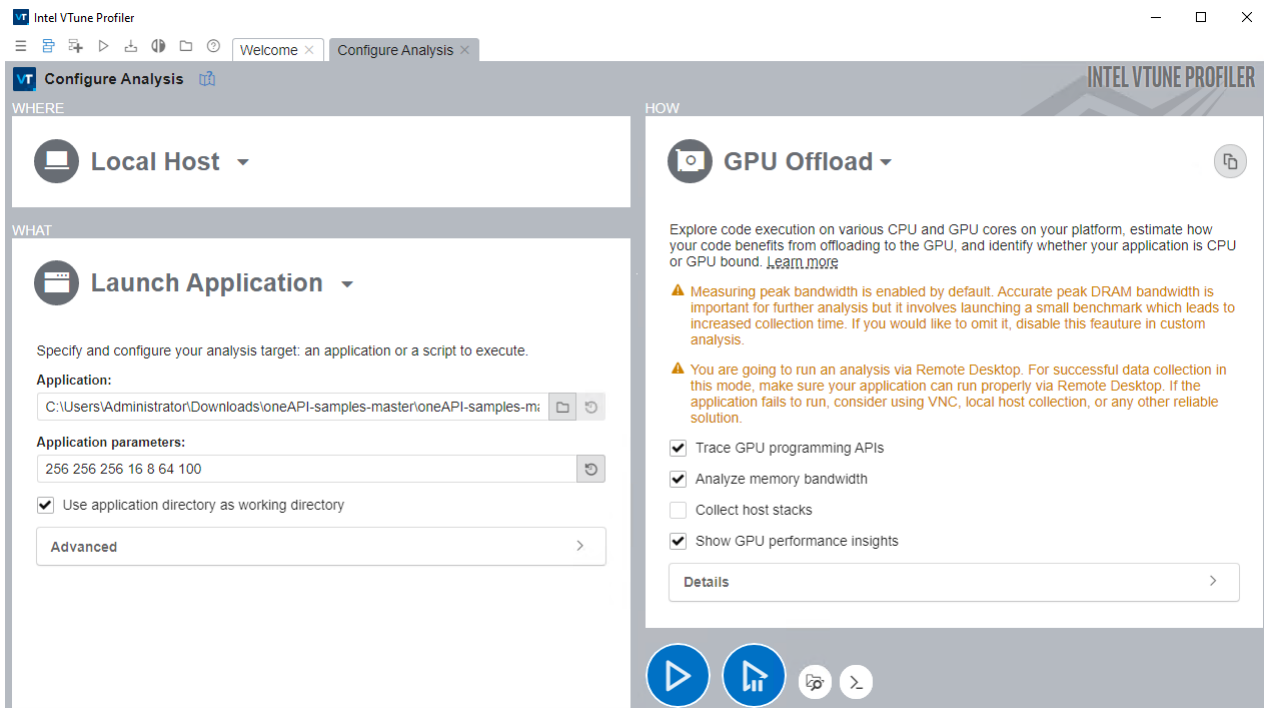
```
mkdir build
cd build
icx /std:c++17 /EHsc /Qopenmp /I../include\ /Qopenmp-targets:
spir64 /DUSE_BASELINE /DEBUG ../src/iso3dfd.cpp ../src/iso3dfd_verify.cpp ../src/utlis.cpp
```

Run a GPU Analysis on the OpenMP Offload Application

You are now ready to run the GPU Offload Analysis on the OpenMP application you compiled.



1. Open VTune Profiler and click on **New Project** to create a project.
2. On the welcome page, click on **Configure Analysis** to set up your analysis.
3. Select these settings for your analysis.
 - In the **WHERE** pane, select **Local Host**.
 - In the **WHAT** pane, select **Launch Application** and specify the `iso3dfd_omp_offload` binary as the application to profile.
 - In the **HOW** pane, select the **GPU Offload** analysis type from the **Accelerators** group in the Analysis Tree.



4. Click the **Start** button to run the analysis.

VTune Profiler collects data and displays analysis results in the **GPU Offload** viewpoint.

- In the **Summary** window, see statistics on CPU and GPU resource usage. Use this data to determine if your application is:
 - GPU-bound
 - CPU-bound
 - Utilizing the compute resources of your system inefficiently
- Use the information in the **Platform** window to see basic CPU and GPU metrics.
- Investigate specific computing tasks in the **Graphics** window.

For a deeper analysis, [see a related recipe](#) in the VTune Profiler Performance Analysis Cookbook. You can also continue your profiling with the [GPU Compute/Media Hotspots analysis](#).

Example: Profile a SYCL* Application on Windows*

Profile a sample `matrix_multiply` SYCL application with Intel® VTune™ Profiler. Get familiar with the product and understand the statistics collected for GPU-bound applications.

Prerequisites

- Make sure you have Microsoft* Visual Studio (v2017 or newer) installed on your system.

- Install Intel VTune Profiler from the [Intel® oneAPI Base Toolkit](#) or the [Intel® System Bring-up Toolkit](#). These toolkits contain the [Intel® oneAPI DPC++/C++ Compiler](#)(`icpx -fsycl`) compiler required for the profiling process.
- Set up environment variables. Execute the `vars.bat` script located in the `<vtune-install-dir>\env` directory.
- Ensure that the Intel oneAPI DPC++ Compiler (installed with the Intel oneAPI Base toolkit) is integrated into Microsoft Visual Studio.
- Compile the code using the `-gline-tables-only` and `-fdebug-info-for-profiling` options for Intel oneAPI DPC++ Compiler.
- [Set up your system for GPU analysis.](#)

For information on installing Intel VTune Profiler in the Microsoft* Visual Studio environment, see [VTune Profiler User Guide](#).

Build the Matrix App

Download the [matrix_multiply_vtune code sample package for Intel oneAPI toolkits](#). This contains the sample which you can use to build and profile a SYCL application.

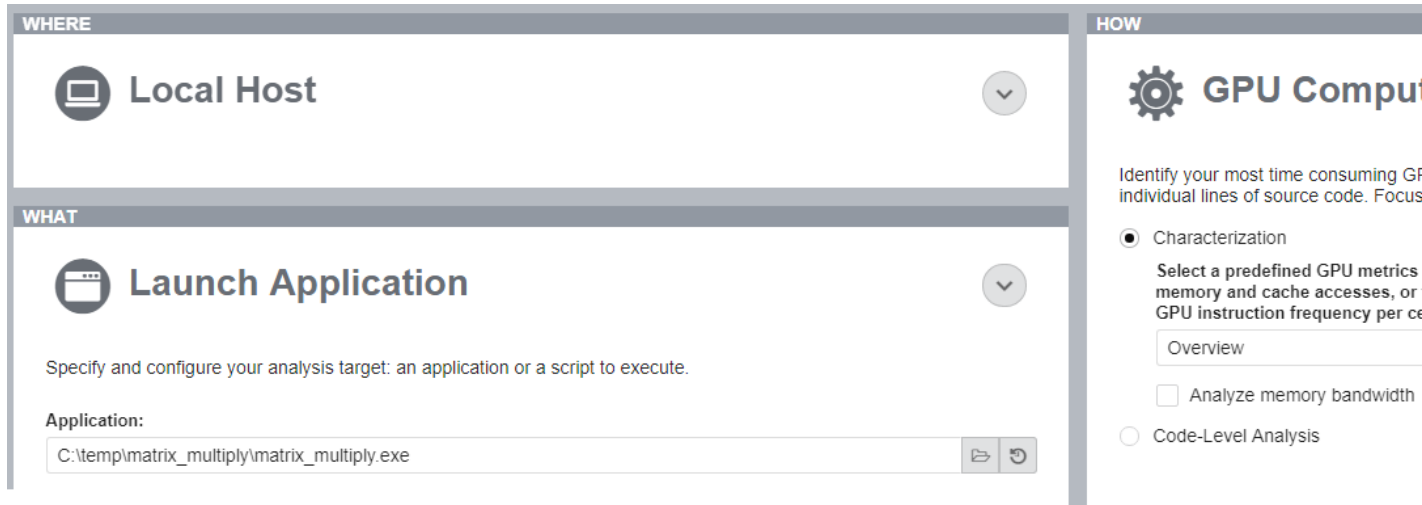
1. Open Microsoft* Visual Studio.
2. Click **File > Open > Project/Solution**. Find the `matrix_multiply_vtune` folder and select `matrix_multiply.sln`.
3. Build this configuration (**Project > Build**).
4. Run the program (**Debug > Start Without Debugging**).
5. To choose a DPC++ or threaded version of the sample, use preprocessor definitions.
 - a. Go to **Project Properties > DPC++ > Preprocessor > Preprocessor Definition**.
 - b. Define `icpx -fsycl` or `USE_THR`.

Run GPU Analysis

Run a GPU analysis on the Matrix sample.

1. From the Visual Studio toolbar, click the  **Configure Analysis** button.
The **Configure Analysis** window opens. By default, it inherits your VS project settings and specifies the `matrix_multiply.exe` as an application to profile.
2. 
In the **Configure Analysis** window, click the  Browse button in the **HOW** pane.
3. Select the **GPU Compute/Media Hotspots** analysis type from the **Accelerators** group in the Analysis Tree.





4. Click the **Start** button to launch the analysis with the predefined options.

Run GPU Analysis from Command Line:

1. Open the sample directory:

```
<sample_dir>\VtuneProfiler\matrix_multiply_vtune
```

2. In this directory, open a Visual Studio* project file named `matrix_multiply.sln`
3. The `multiply.cpp` file contains several versions of matrix multiplication. Select a version by editing the corresponding `#define MULTIPLY` line in `multiply.hpp`
4. Build the entire project with a Release configuration.
This generates an executable called `matrix_multiply.exe`.
5. Prepare the system to run a GPU analysis. See [Set Up System for GPU Analysis](#).
6. Set VTune Profiler environment variables by running the batch file:

```
export <install_dir>\env\vars.bat
```

7. Run the analysis command:

```
vtune.exe -collect gpu-offload -- matrix_multiply.exe
```

VTune Profiler collects data and displays analysis results in the **GPU Compute/Media Hotspots** viewpoint. In the **Summary** window, see statistics on CPU and GPU resource usage to understand if your application is GPU-bound. Switch to the **Graphics** window to see basic CPU and GPU metrics representing code execution over time.

Get Started with Intel® VTune™ Profiler for Linux* OS

Before You Begin

1. [Install Intel® VTune™ Profiler on your Linux* system.](#)
2. Build your application with symbol information and in **Release** mode with all optimizations enabled. For detailed information on compiler settings, see the [VTune Profiler online user guide](#).

You can also use the `matrix` sample application available in `<install_directory>\sample\matrix`. You can see sample results in `<install-dir>\sample (matrix)`.

3. Set up the environment variables:

```
source <install-dir>/setvars.sh
```

By default, the `<install-dir>` is:



- \$HOME/intel/oneapi/ when installed with user permissions;
- /opt/intel/oneapi/ when installed with root permissions.

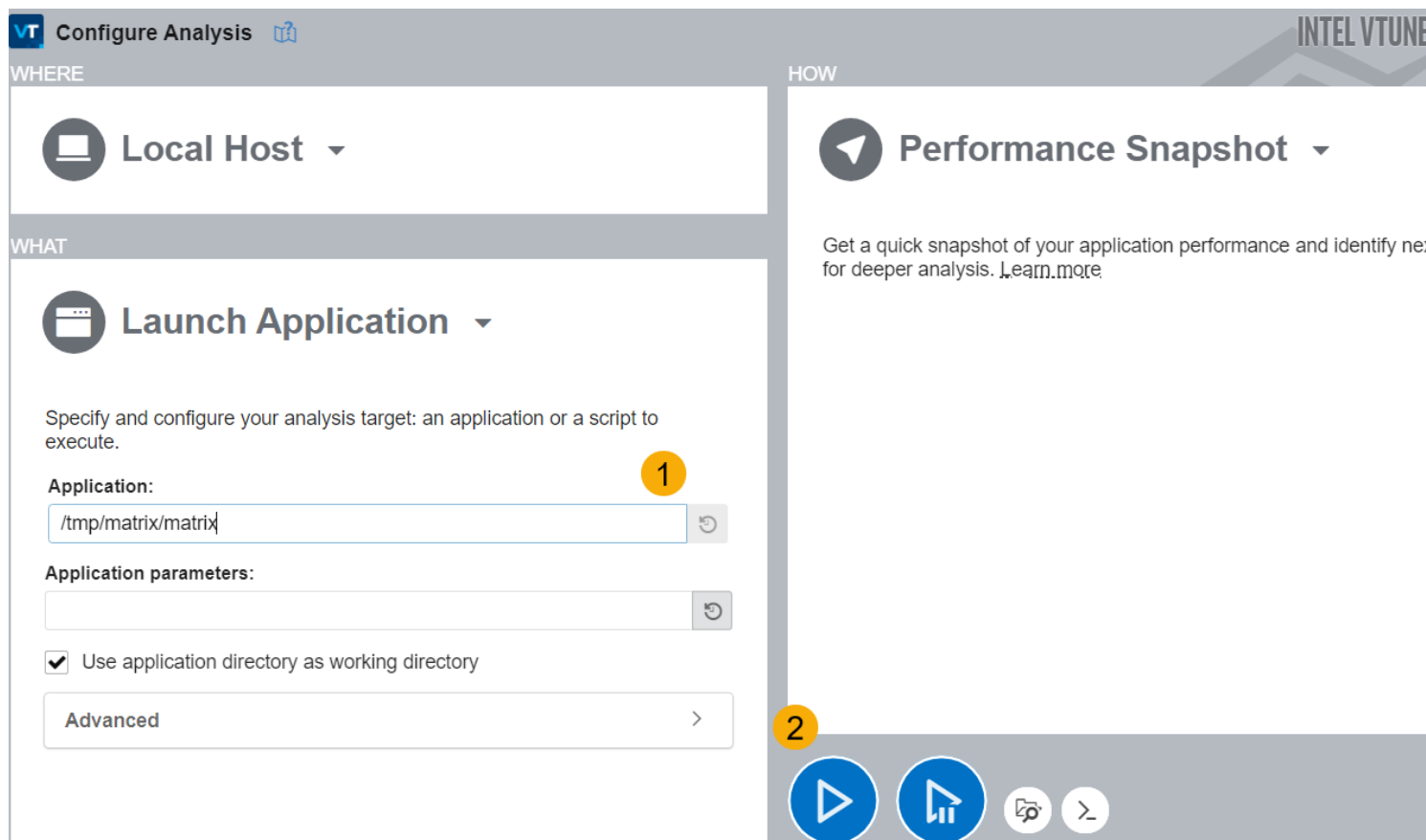
Step 1: Start VTune Profiler

Start VTune Profiler through one of these ways:

Source	Start VTune Profiler
Standalone/IDE (GUI)	1. Run the <code>vtune-gui</code> command. To start VTune Profiler from the Intel System Studio IDE, select Tools > VTune Profiler > Launch VTune Profiler. This sets all appropriate environment variables and launches a standalone interface of the product. 2. When the GUI opens, click  New Project... in the Welcome screen. 3. In the Create Project dialog box, specify the project name and location. 4. Click Create Project.
Standalone (Command line)	Run the <code>vtune</code> command.

Step 2: Configure and Run Analysis

After creating a new project, the **Configure Analysis** window opens with these default values:



1. In the **Launch Application** section, browse to the location of your application.
2. Click the **Start** to run **Performance Snapshot** on your application. This analysis presents a general overview of issues affecting the performance of your application on the target system.



Step 3: View and Analyze Performance Data

When data collection completes, VTune Profiler displays analysis results in the **Summary** window. Here, you see a performance overview of your application.

The overview typically includes several metrics along with their descriptions.

Performance Snapshot Performance Snapshot

Analysis Configuration Collection Log **Summary**

Choose your next analysis type

Select a highlighted recommendation based on your performance snapshot.

ALGORITHM

- Hotspots
- Anomaly Detection (preview)

MICROARCHITECTURE

- Microarchitecture Exploration 20.9%
- Memory Access 47.2%

PARALLELISM

- Threading
- HPC Performance Characterization

I/O

- Input and Output (preview)

ACCELERATORS

- GPU Offload (preview)
- GPU Compute/Media Hotspots (preview)
- CPU/FPGA Interaction (preview)

PLATFORM ANALYSIS

- System Overview
- Throttling (preview)
- Platform Profiler

Elapsed Time

- IPC
- SP GFLOPS
- DP GFLOPS
- x87 GFLOPS
- Average CPU Frequency

Effective Logical

97.7% (7.815 out)

Microarchitecture of Pipeline Slots

The metric value is high. This can indicate that the significant fraction of execution pipeline slots could be stalled due to demand memory load and stores. Use Memory Access analysis to have the metric breakdown by memory hierarchy, memory bandwidth information, correlation by memory objects.

Vectorization of Packed FP Ops

- Instruction Mix
- SP FLOPs
- Packed
- 128-bit
- 256-bit

A

Expand each metric for detailed information about contributing factors.

B

A flagged metric indicates a value outside acceptable/normal operating range. Use tool tips to understand how to improve a flagged metric.

C

See guidance on other analyses you should consider running next. The **Analysis Tree** highlights these recommendations.

Next Steps

Performance Snapshot is a good starting point to get an overall assessment of application performance with VTune Profiler. Next, check if your algorithm requires tuning.

1. [Follow a tutorial to analyze common performance bottlenecks.](#)
2. Once your algorithm is well-tuned, run Performance Snapshot again to calibrate results and identify potential performance improvements in other areas.

See Also

[Microarchitecture Exploration](#)

[VTune Profiler Help Tour](#)

Example: Profile an OpenMP Application on Linux*

Use Intel VTune Profiler on a Linux machine to profile a sample `iso3dfd_omp_offload` OpenMP application offloaded onto an Intel GPU. Learn how to run a GPU analysis and examine results.

Prerequisites

- Make sure your system is running Linux* OS kernel 4.14 or a newer version.
- Use one of these versions of Intel Processor Graphics:
 - Gen 8
 - Gen 9
 - Gen 11
- Your system should be running on one of these Intel processors:
 - 7th Generation Intel® Core™ i7 Processors (code name Kaby Lake)
 - 8th Generation Intel® Core™ i7 Processors (code name Coffee Lake)
 - 10th Generation Intel® Core™ i7 Processors (code name Ice Lake)
- For the Linux GUI, use:
 - GTK+ version 2.10 or newer (2.18 and newer versions are recommended)
 - Pango version 1.14 or newer
 - X.Org version 1.0 or newer (1.7 and newer versions are recommended)
- Install Intel VTune Profiler from one of these sources:
 - [Standalone product download](#)
 - [Intel® oneAPI Base Toolkit](#)
 - [Intel® System Bring-up Toolkit](#)
- Download the [Intel® oneAPI HPC Toolkit](#) which contains the Intel® oneAPI DPC++/C++ Compiler(`icx/icpx`) that you need to profile OpenMP applications.
- Set up environment variables. Execute the `vars.sh` script.
- [Set up your system for GPU analysis.](#)

Build and Compile the OpenMP Offload Application

1. Download the [iso3dfd_omp_offload OpenMP Offload sample](#).
2. Open to the sample directory.

```
cd <sample_dir>/DirectProgramming/C++/StructuredGrids/iso3dfd_omp_offload
```

3. Compile the OpenMP Offload application.

```
mkdir build;
cmake -DVERIFY_RESULTS=0 ..
make -j
```



This generates a `src/iso3dfd` executable.

To delete the program, type:

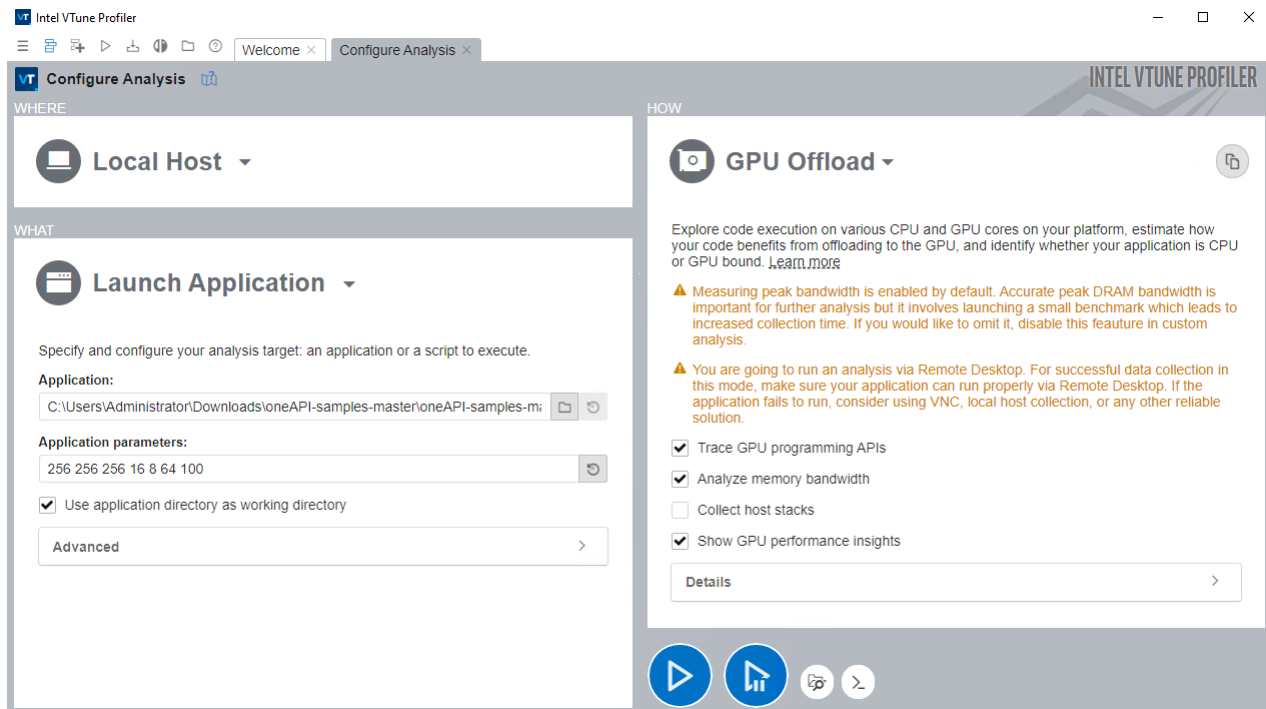
```
make clean
```

This removes the executable and object files that you created with the `make` command.

Run a GPU Analysis on the OpenMP Offload Application

You are now ready to run the GPU Offload Analysis on the OpenMP application you compiled.

1. Open VTune Profiler and click on **New Project** to create a project.
2. On the welcome page, click on **Configure Analysis** to set up your analysis.
3. Select these settings for your analysis.
 - In the **WHERE** pane, select **Local Host**.
 - In the **WHAT** pane, select **Launch Application** and specify the `iso3dfd_omp_offload` binary as the application to profile.
 - In the **HOW** pane, select the **GPU Offload** analysis type from the **Accelerators** group in the Analysis Tree.



4. Click the **Start** button to run the analysis.

VTune Profiler collects data and displays analysis results in the **GPU Offload** viewpoint.

- In the **Summary** window, see statistics on CPU and GPU resource usage. Use this data to determine if your application is:
 - GPU-bound
 - CPU-bound
 - Utilizing the compute resources of your system inefficiently
- Use the information in the **Platform** window to see basic CPU and GPU metrics.
- Investigate specific computing tasks in the **Graphics** window.

For a deeper analysis, [see a related recipe](#) in the [VTune Profiler Performance Analysis Cookbook](#). You can also continue your profiling with the [GPU Compute/Media Hotspots analysis](#).

Example: Profile a SYCL* Application on Linux*

Use VTune Profiler with a sample `matrix_multiply` SYCL application to quickly get familiar with the product and statistics collected for GPU-bound applications.

Prerequisites

- Install VTune Profiler and [Intel® oneAPI DPC++/C++ Compiler](#) from the [Intel® oneAPI Base Toolkit](#) or the [Intel® System Bring-up Toolkit](#).
- Set up environment variables by executing the `vars.sh` script.
- [Set up your system for GPU analysis](#).

Build the Matrix Application

Download the [matrix_multiply_vtune code sample package for Intel oneAPI toolkits](#). This contains the sample which you can use to build and profile a SYCL application.

To profile a SYCL application, make sure to compile the code using the `-gline-tables-only` and `-fdebug-info-for-profiling` Intel oneAPI DPC++ Compiler options.

To compile this sample application, do the following:

1. Go to the sample directory.

```
cd <sample_dir/VtuneProfiler/matrix_multiply>
```

2. The `multiply.cpp` file in the `src` folder contains several versions of matrix multiplication. Select a version by editing the corresponding `#define MULTIPLY` line in `multiply.h`.
3. Build the app using the existing Makefile:

```
cmake .
make
```

This should generate a `matrix.icpx -fsycl` executable.

To delete the program, type:

```
make clean
```

This removes the executable and object files that were created by the `make` command.

Run GPU Analysis

Run a GPU analysis on the Matrix sample.

1. Launch VTune Profiler with the `vtune-gui` command.
2. Click **New Project** from the Welcome page.
3. Specify a name and location for your sample project and click **Create Project**.
4. In the **WHAT** pane, browse to the `matrix.icpx -fsycl` file.
- 5.



In the **HOW** pane, click the  Browse button and select **GPU Compute/Media Hotspots** analysis from the **Accelerators** group in the Analysis Tree.



 Configure Analysis 

WHERE

 Local Host 

WHAT

 Launch Application 

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

Application:

Application parameters:



☒ Use application directory as working directory

Advanced ▶

- Click the **Start** button at the bottom to launch the analysis with the pre-selected options.

Run GPU Analysis from Command Line:

- Prepare the system to run a GPU analysis. See [Set Up System for GPU Analysis](#).
- Set up environment variables for Intel software tools:

```
source $ONEAPI_ROOT/setvars.sh
```

- Run the GPU Compute/Media Hotspots analysis:

```
vtune -collect gpu-hotspots -r ./result_gpu-hotspots -- ./matrix.icpx -fsycl
```

To see the summary report, type:

```
vtune -report summary -r ./result_gpu-hotspots
```

VTune Profiler collects data and displays analysis results in the **GPU Compute/Media Hotspots** viewpoint. In the **Summary** window, see statistics on CPU and GPU resource usage to understand if your application is GPU-bound. Switch to the **Graphics** window to see basic CPU and GPU metrics representing code execution over time.

Get Started with Intel® VTune™ Profiler for macOS*

Use VTune Profiler on a macOS system to perform remote target analysis on a non-macOS system (Linux* or Android* only) .

You cannot use VTune Profiler in a macOS environment for these purposes:

- Profile the macOS system on which it is installed.
- Collect data on a remote macOS system.

To analyze performance of a remote Linux* or Android* target from the macOS host, do one of these steps:

- Run a VTune Profiler analysis on the macOS system with a remote system specified as the target. When analysis begins, VTune Profiler connects to the remote system to collect data, then brings the results back to the macOS host for viewing.
- Run an analysis on the target system locally and copy the results to a macOS system for viewing in VTune Profiler.

The steps in this document assume a remote Linux target system and collect performance data using SSH access from VTune Profiler on a macOS host system.

Before You Begin

1. [Install Intel® VTune™ Profiler on your macOS* system.](#)
2. Build your Linux application with symbol information and in **Release** mode with all optimizations enabled. For detailed information, see the [compiler settings](#) in the VTune Profiler help.
3. [Set up SSH access](#) from the host macOS system to the target Linux system to work in the password-less mode.

Step 1: Start VTune Profiler

1. Launch VTune Profiler with the `vtune-gui` command.

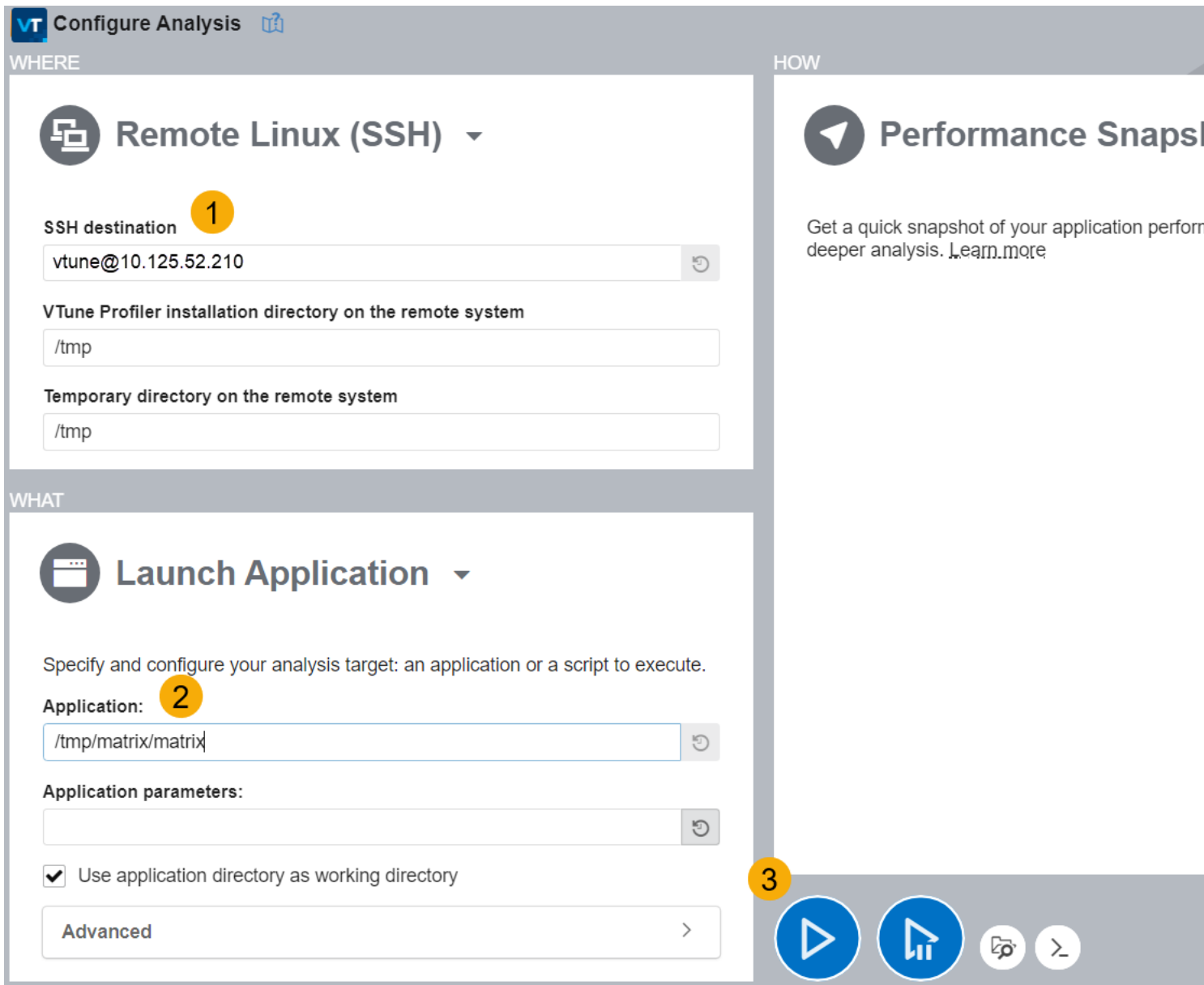
By default, the `<install_dir>` is `/opt/intel/oneapi/`.

2. When the GUI opens, click  **New Project...** in the **Welcome** screen.
3. In the **Create Project** dialog box, specify the project name and location.
4. Click **Create Project**.



Step 2: Configure and Run Analysis

After you create a new project, the **Configure Analysis** window opens with the [Performance Snapshot](#) analysis type. This analysis presents an overview of issues that affect the performance of your application on the target system.



1. In the **WHERE** pane, select **Remote Linux (SSH)** and specify the target Linux system using `username@hostname[:port]`.
VTune Profiler connects to the Linux system and installs the target package.
2. In the **WHAT** pane, provide the path to your application on the target Linux system.
3. Click the **Start** button to run [Performance Snapshot](#) on the application.

Step 3: View and Analyze Performance Data

When data collection completes, VTune Profiler displays analysis results on the macOS system. Start your analysis in the **Summary** window. Here, you see a performance overview of your application.



The overview typically includes several metrics along with their descriptions.

Performance Snapshot Performance Snapshot ▾ ⓘ 📖

Analysis Configuration Collection Log **Summary**

Choose your next analysis type

Select a highlighted recommendation based on your performance snapshot.

ALGORITHM

- Hotspots
- Anomaly Detection (preview)
- Memory Consumption

PARALLELISM

- Threading 13.6%
- HPC Performance Characterization

ACCELERATORS

- GPU Offload (preview)
- GPU Compute/Media Hotspots

MICROARCHITECTURE

- Microarchitecture Exploration 10.3%
- Memory Access 74.8%

I/O

- Input and Output

PLATFORM ANALYSES

- System Overview
- Throttling (preview)

Elapsed Time ⓘ

A IPC ⓘ:
SP GFLOPS ⓘ:
DP GFLOPS ⓘ:
x87 GFLOPS ⓘ:
Average CPU Frequency ⓘ

Effective Logical

13.6% (15.220 on

Microarchitecture of Pipeline Slots

The metric value is high. This can indicate that the significant fraction of execution pipeline slots could be stalled due to demand memory load and stores. Use Memory Access analysis to have the metric breakdown by memory hierarchy, memory bandwidth information, correlation by memory objects.

Vectorization ⓘ:
of Packed FP Op

A Expand each metric for detailed information about contributing factors.

B A flagged metric indicates a value outside acceptable/normal operating range. Use tool tips to understand how to improve a flagged metric.

C See guidance on other analyses you should consider running next. The **Analysis Tree** highlights these recommendations.

Next Steps

Performance Snapshot is a good starting point to get an overall assessment of application performance with VTune Profiler. Next, check if your algorithm requires tuning.

1. Run [Hotspots Analysis](#) on your application.
2. [Follow a Hotspots tutorial](#). Learn techniques to get the most out of your Hotspots analysis.



- Once your algorithm is well-tuned, run Performance Snapshot again to calibrate results and identify potential performance improvements in other areas.

See Also

[Microarchitecture Exploration](#)

[VTune Profiler Help Tour](#)

Learn More

Document	Description
User Guide	The User Guide is the primary documentation for VTune Profiler. NOTE You can also download an offline version of the VTune Profiler documentation.
Online Training	The online training site is an excellent resource to learn the basics of VTune Profiler with Getting Started guides, videos, tutorials, webinars, and technical articles.
Cookbook	Performance analysis cookbook that contains recipes to identify and solve popular performance problems using analysis types in VTune Profiler.
Installation Guide for Windows Linux macOS hosts	The Installation Guide contains basic installation instructions for VTune Profiler and post-installation configuration instructions for the various drivers and collectors.
Tutorials	VTune Profiler tutorials guide a new user through basic features with a short sample application.
Release Notes	Find information about the latest version of VTune Profiler, including a comprehensive description of new features, system requirements, and technical issues that were resolved. For the standalone and toolkit versions of VTune Profiler, understand the current System Requirements .

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Intel, the Intel logo, Intel Atom, Intel Core, Intel Xeon Phi, VTune and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.



This software and the related documents are Intel copyrighted materials, and your use of them is governed by the express license under which they were provided to you (**License**). Unless the License provides otherwise, you may not use, modify, copy, publish, distribute, disclose or transmit this software or the related documents without Intel's prior written permission.

This software and the related documents are provided as is, with no express or implied warranties, other than those that are expressly stated in the License.

© Intel Corporation.

