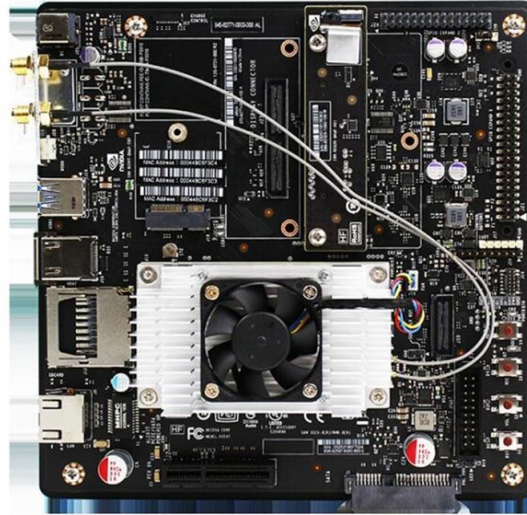


ORB SLAM2 CUDA Project



M.Sc Student: Eran Moshe

Supervision By Yehonatan Mendel

[Autonomous Drones Lab](#)

Faculty of Electrical Engineering, Tel Aviv University

Project Description

Build CUDA Infrastructure for Parallel Programming on Nvidia Jetson TX2

In many applications in embedded devices there are more and more interactions between GPU and CPU to accelerate processes, **CUDA** allows to create efficiently and quickly **Parallel Programs on GPU's** (device side) which controlled by CPU (Host side). As the field of AI grows, there is a growing demand for autonomous surveillance by drones. **Low latency** is very important requirement for drone detection and tracking which can be reached by parallelism of operations.

The main task in the project is to build lab Cuda infrastructure to execute parallel code on **Nvidia Jetson TX2**. Jetson TX2 is the fastest, most power-efficient embedded AI computing device. It's built around an **NVIDIA Pascal™-family GPU** and loaded with 8GB of memory and 59.7GB/s of memory bandwidth.



Project Stages



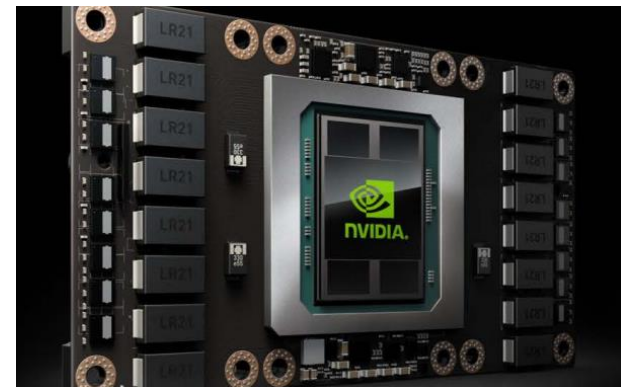
- Internet Research - Overview.
- Learning Architecture and System.
- Learning Cuda Programming.
- Ramp up Jetson TX2 Environment for Running ORB SLAM Application in Linux Environment.
- Research about Nvidia Profiling Tools to Debug Cuda Applications.

Jetson TX2 - Hardware

Jetson TX2 Hardware Specifications

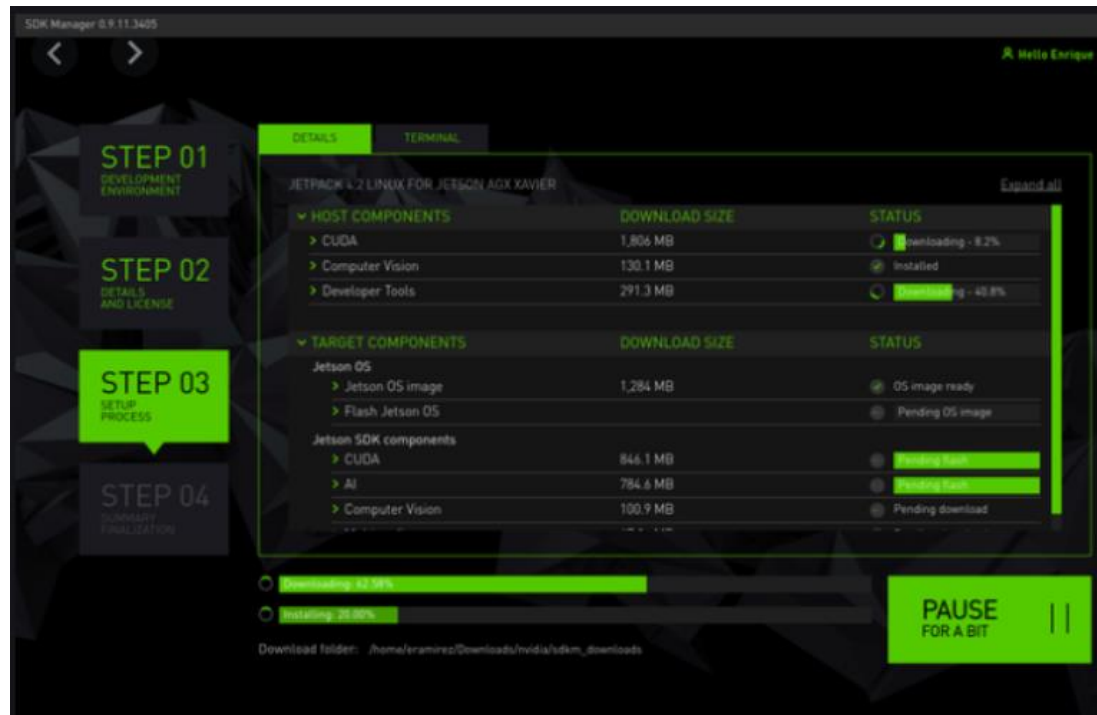
Jetson TX2 is a 7.5-watt supercomputer on a module that brings true AI computing at the edge. It's built around an NVIDIA Pascal™-family GPU and loaded with 8 GB of memory and 59.7 GB/s of memory bandwidth.

- GPU - **NVIDIA Pascal™** architecture with 256 NVIDIA CUDA cores 1.3 TFLOPS (FP16).
- CPU - Dual-core Denver 2 64-bit CPU and quad-core ARM A57 complex
- Memory - 8 GB 128-bit LPDDR4 1866MHz - 59.7 GB/s
- Storage - 32 GB eMMC 5.1
- Connectivity - 10/100/1000 BASE-T Ethernet



Jetson TX2 - SoftWare

NVIDIA JetPack SDK is a comprehensive resource for building AI applications. It includes Linux Driver Package (L4T) which provides the Linux kernel, bootloader, NVIDIA drivers, flashing utilities, sample filesystem, and more for the Jetson platform tools together with accelerated software libraries, APIs, sample applications, developer tools, and documentation.



ORB SLAM Application

ORB-SLAM2 is a real-time SLAM (Simultaneous Localization And Mapping) library for **Monocular**, **Stereo** and **RGB-D** cameras that computes the camera path and a sparse 3D reconstruction.

ORB AND FAST Algorithms

Fast Algorithm is a corner detection method, which could be used to extract feature points.

ORB performs the task of **feature detection**. ORB builds on the well-known FAST keypoint detector and then by BRIEF to create the descriptors.

These features points used to track and map objects in many computer vision tasks.

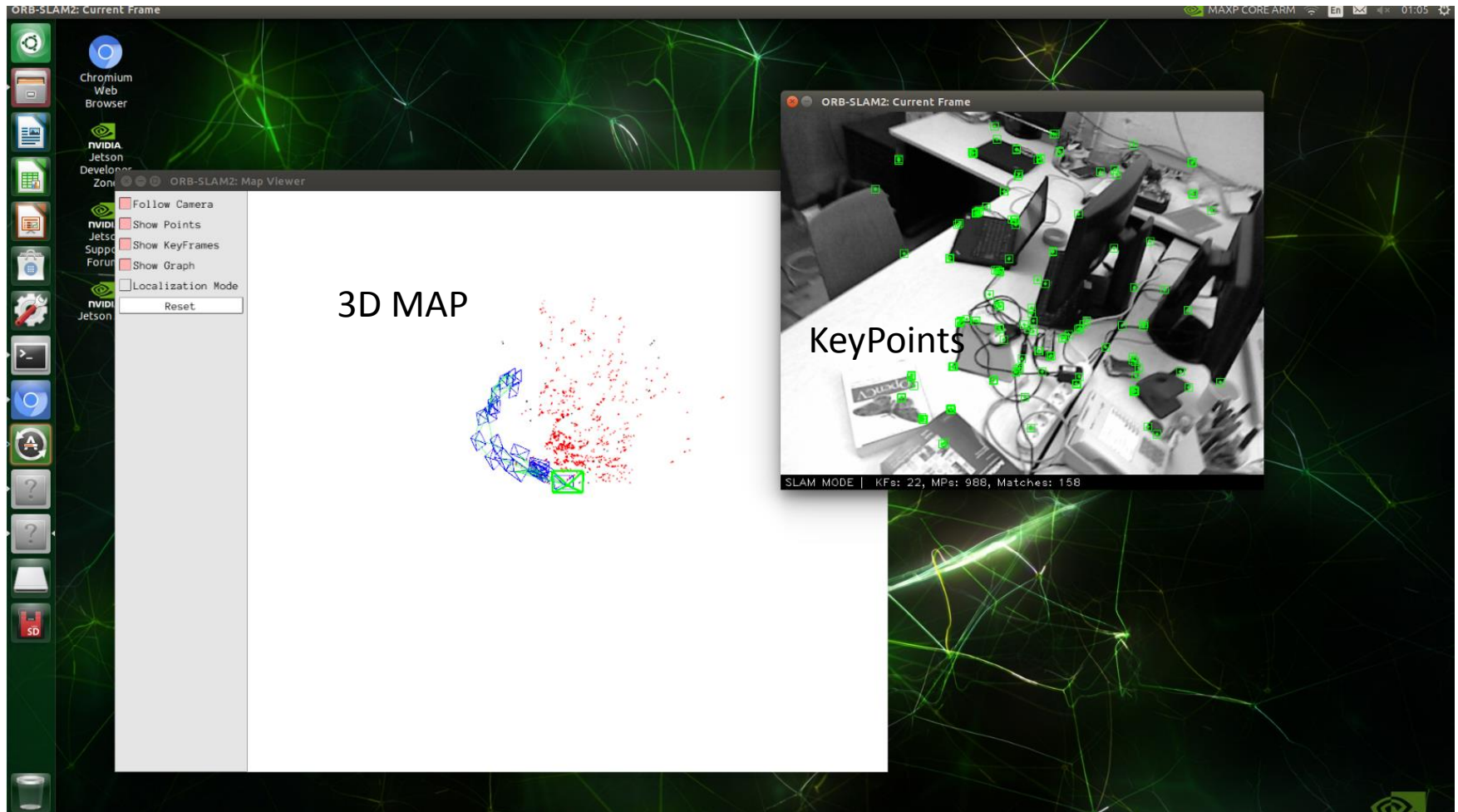
These techniques are attractive because of their good performance and low cost.

References:

<https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf>
<https://medium.com/data-breach/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6>
<https://www.youtube.com/watch?v=Lel67zjfpQ>
<https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65>

ORB SLAM Application Environment

Configuration and Build



Cuda

CUDA[®] is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads. These cores have shared resources including a register file and a shared memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

Cuda CPU VS GPU

- **CPU**

- Optimised for low-latency access to cached data sets
- Control logic for out-of-order and speculative execution

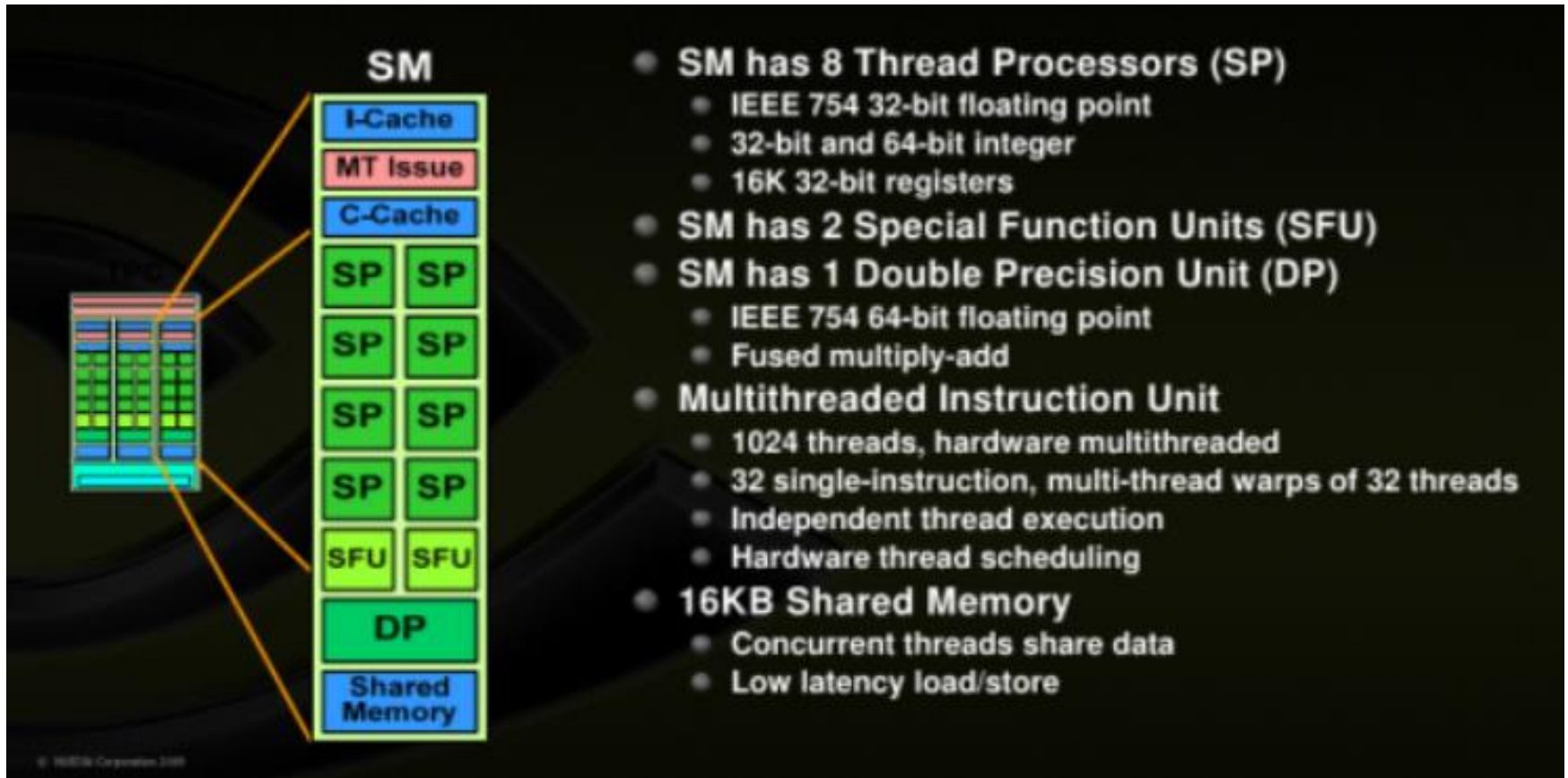
- **GPU**

- Optimised for data-parallel, throughput computation
- Architecture tolerant of memory latency
- More transistors dedicated to computation



Reference: <https://www.slideshare.net/maheshkha/cuda-tutorial>

Cuda GPU Example

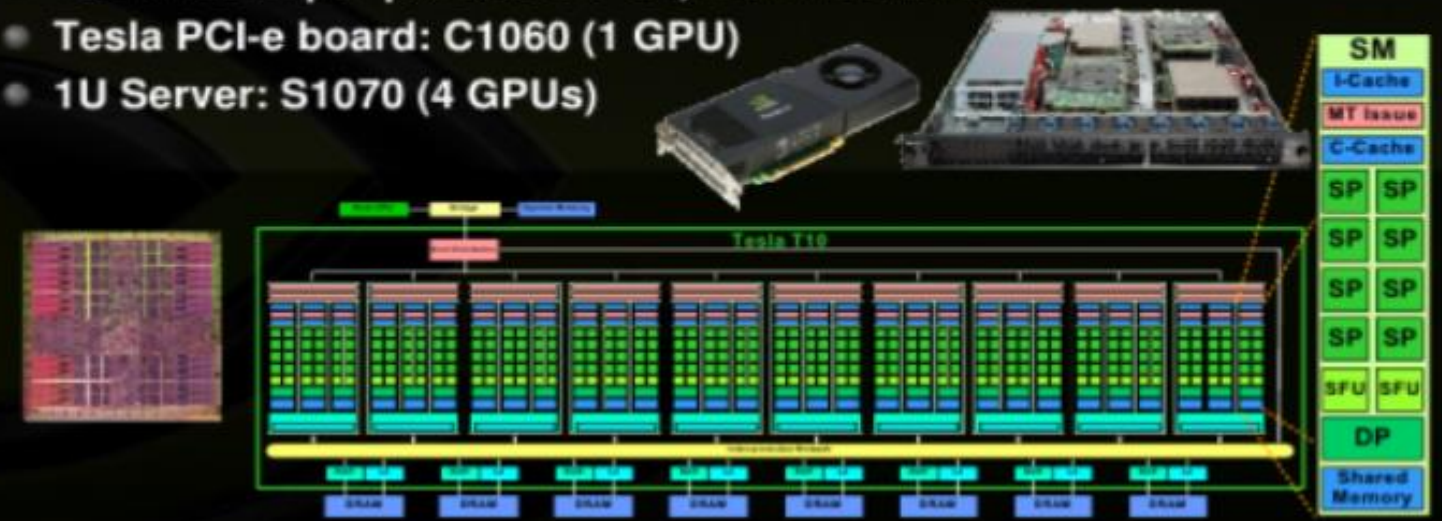


Reference: <https://www.slideshare.net/maheshkha/cuda-tutorial>

Cuda GPU Example – Cont.

Computing with Tesla

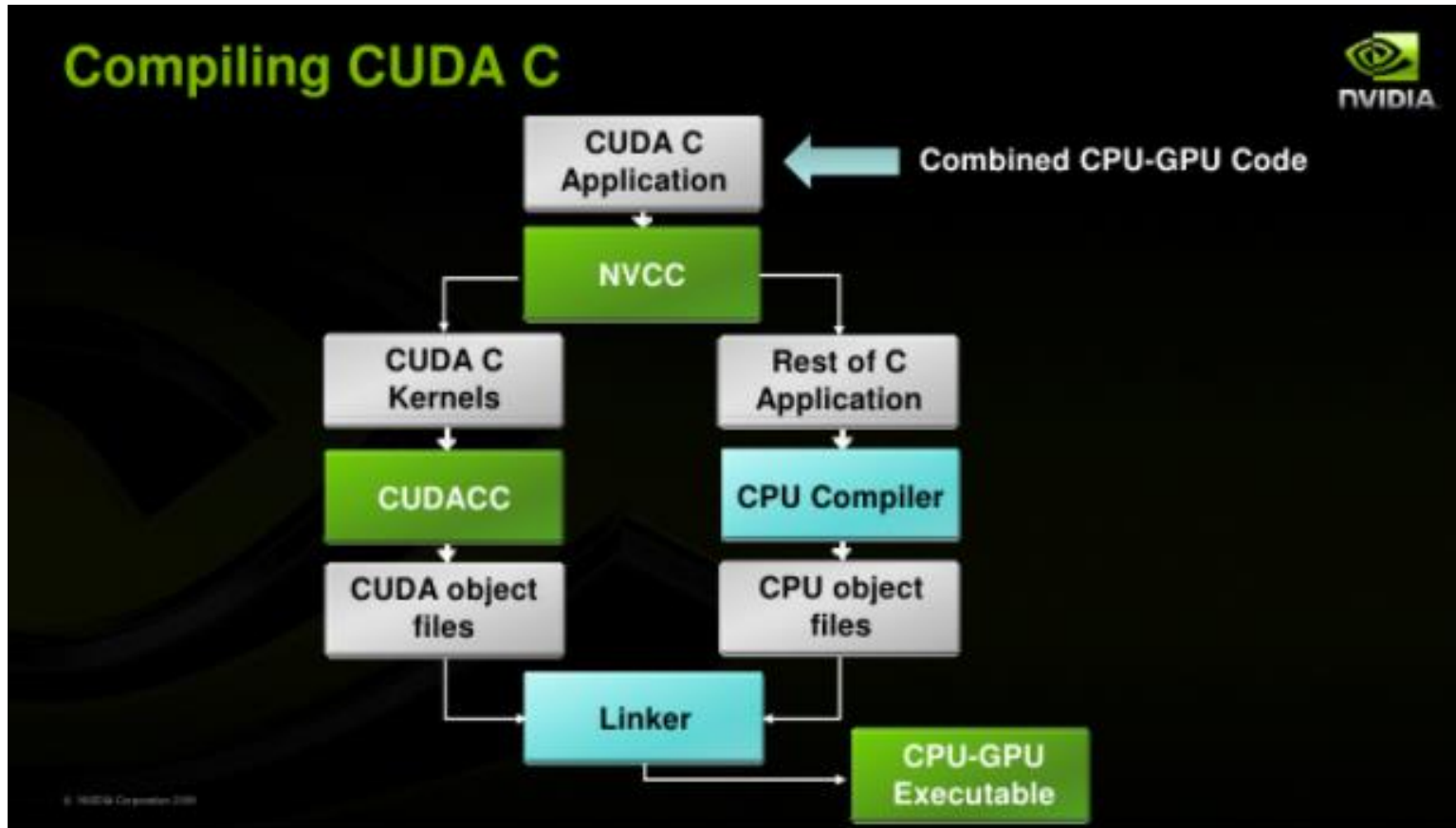
- 240 SP processors at 1.5 GHz: 1 TFLOPS peak
- 128 threads per processor: 30,720 threads total
- Tesla PCI-e board: C1060 (1 GPU)
- 1U Server: S1070 (4 GPUs)



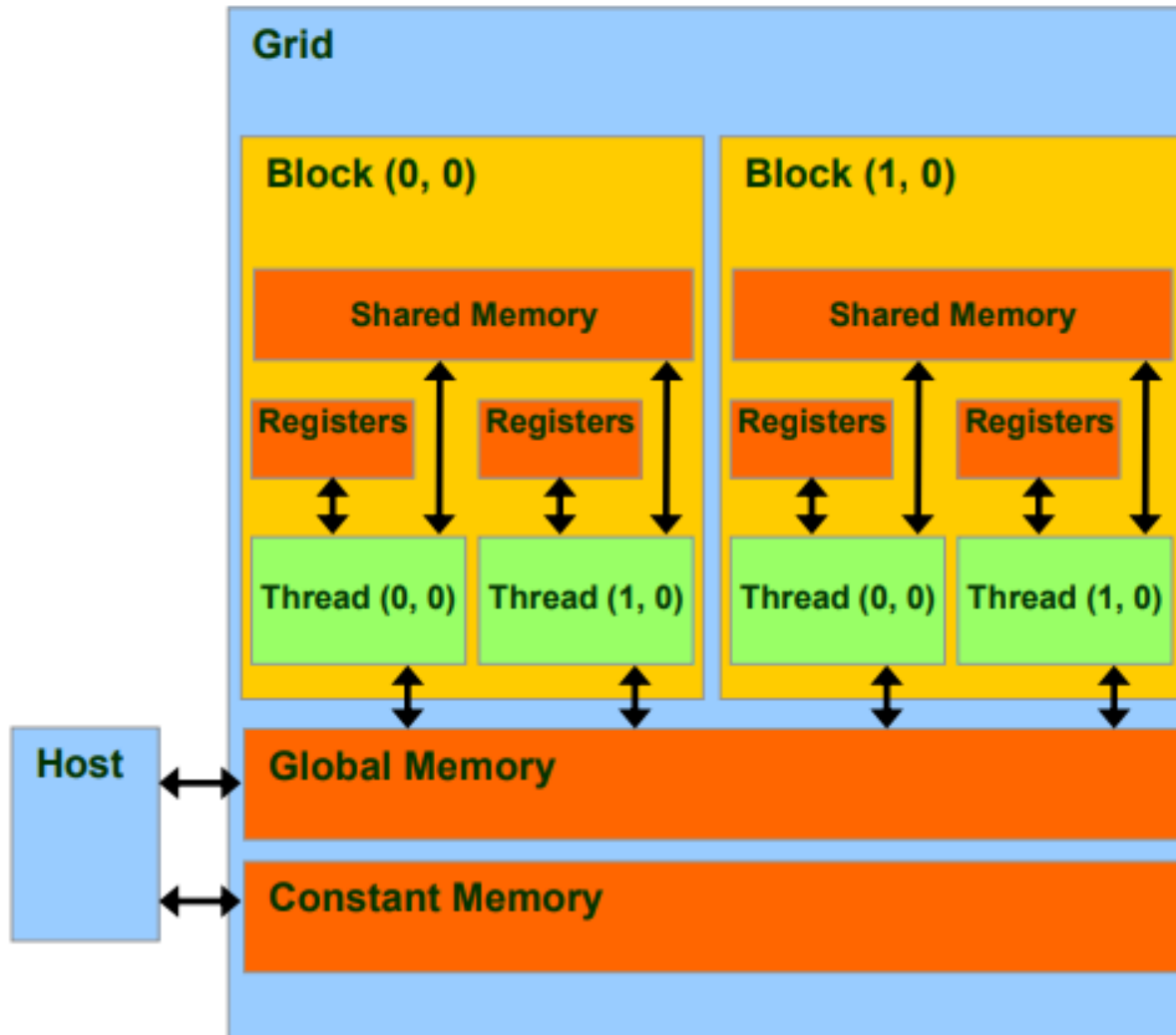
The diagram illustrates the architecture of the Tesla T10 GPU. It shows a top-down view of the GPU chip, a physical PCI-e card, and a server rack. Below these is a schematic of the Tesla T10 architecture. The schematic shows a central 'Tesla T10' block containing multiple 'SM' (Streaming Multiprocessor) units. Each SM unit is composed of several components: 'L-Cache', 'MT Issue', 'C-Cache', 'SP' (Scalar Processor) units, 'SFU' (Special Function Unit), 'DP' (Double Precision), and 'Shared Memory'. The SM units are connected to a 'Memory Controller' and 'Memory' blocks. The diagram also shows a 'PCI-e' interface and a 'System Bus'.

Reference: <https://www.slideshare.net/maheshkha/cuda-tutorial>

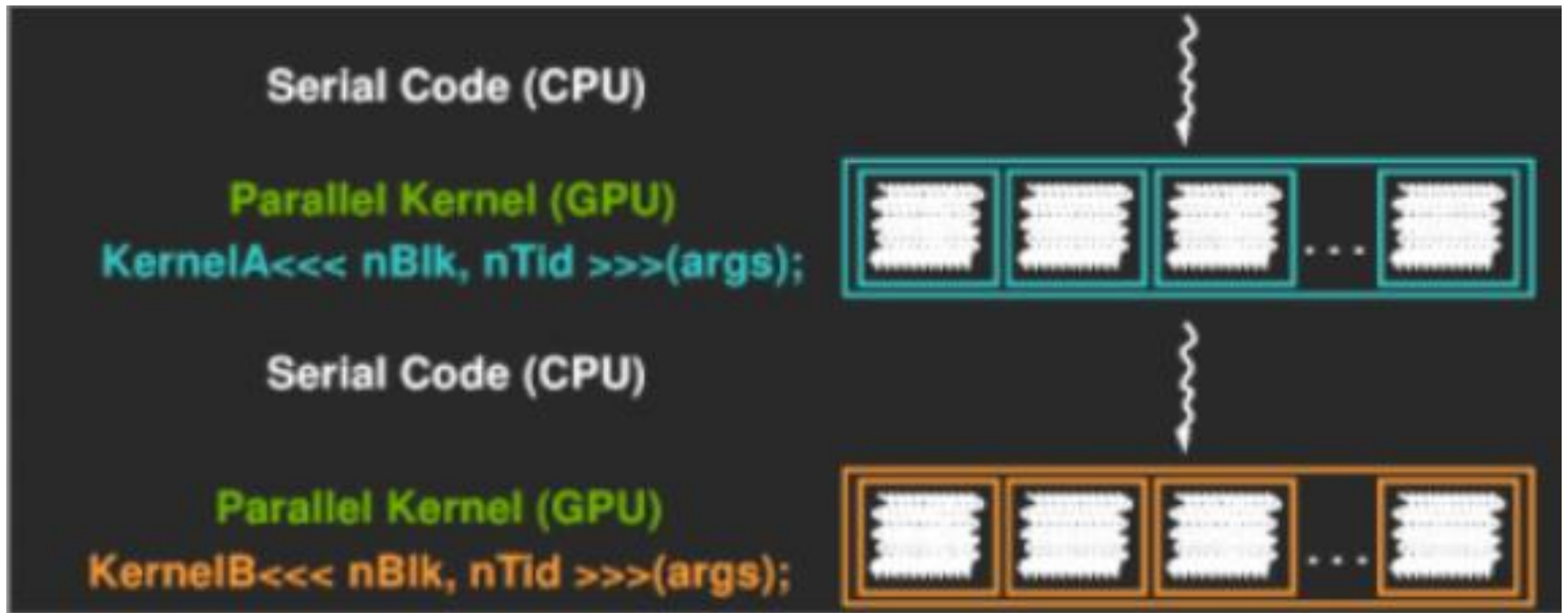
Cuda – Compilation



Cuda – Memory View

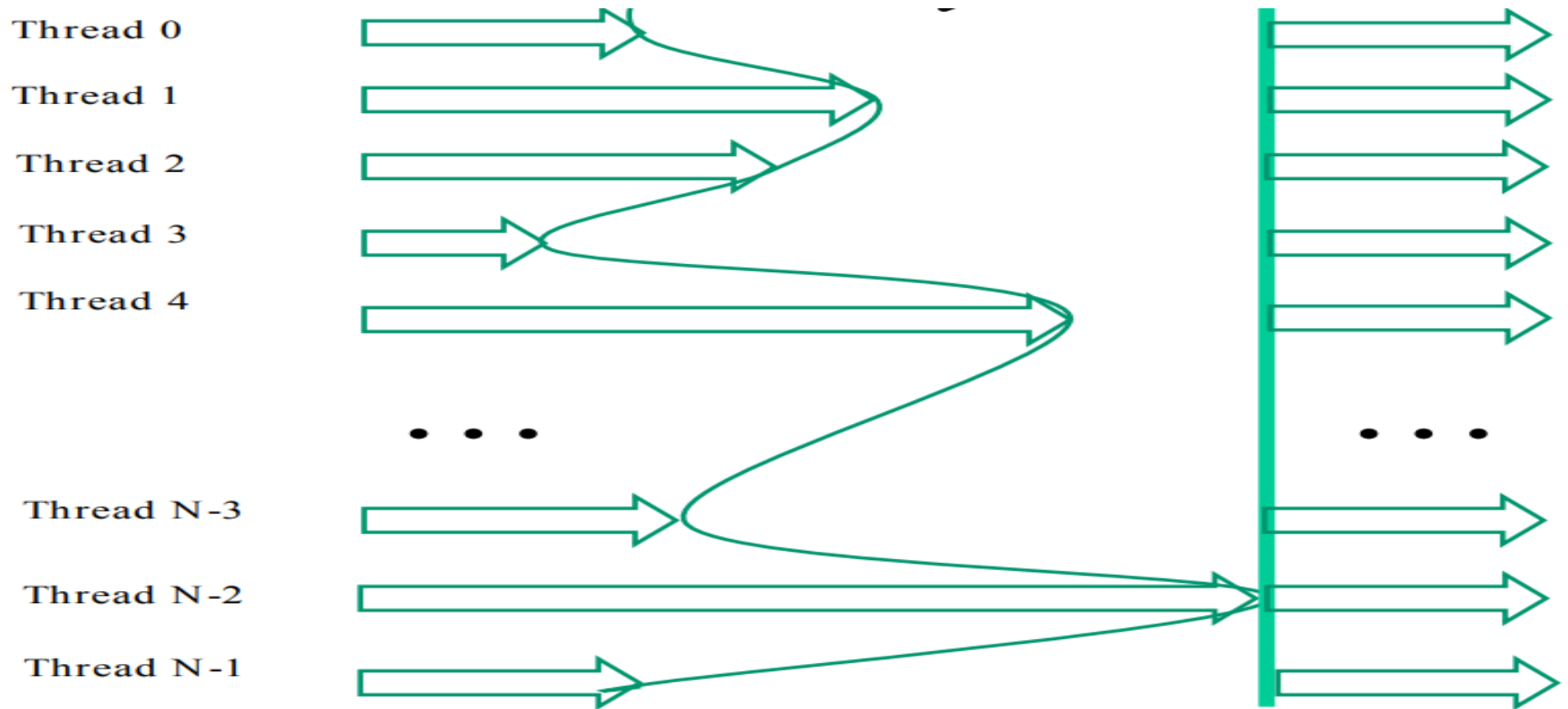


Cuda – CPU-GPU Interaction

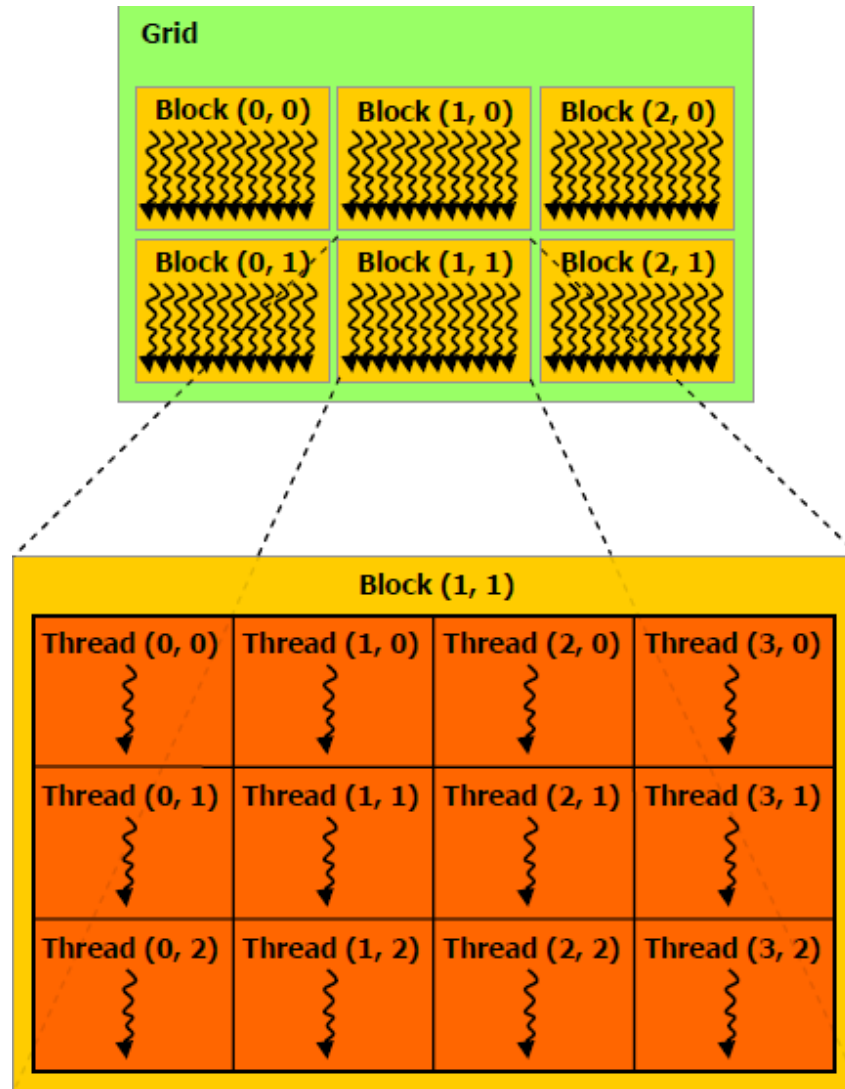


Reference: <https://www.slideshare.net/maheshkha/cuda-tutorial>

Cuda – Run with Barrier.



Cuda – Grid\Block\Threads



Profiling Tools

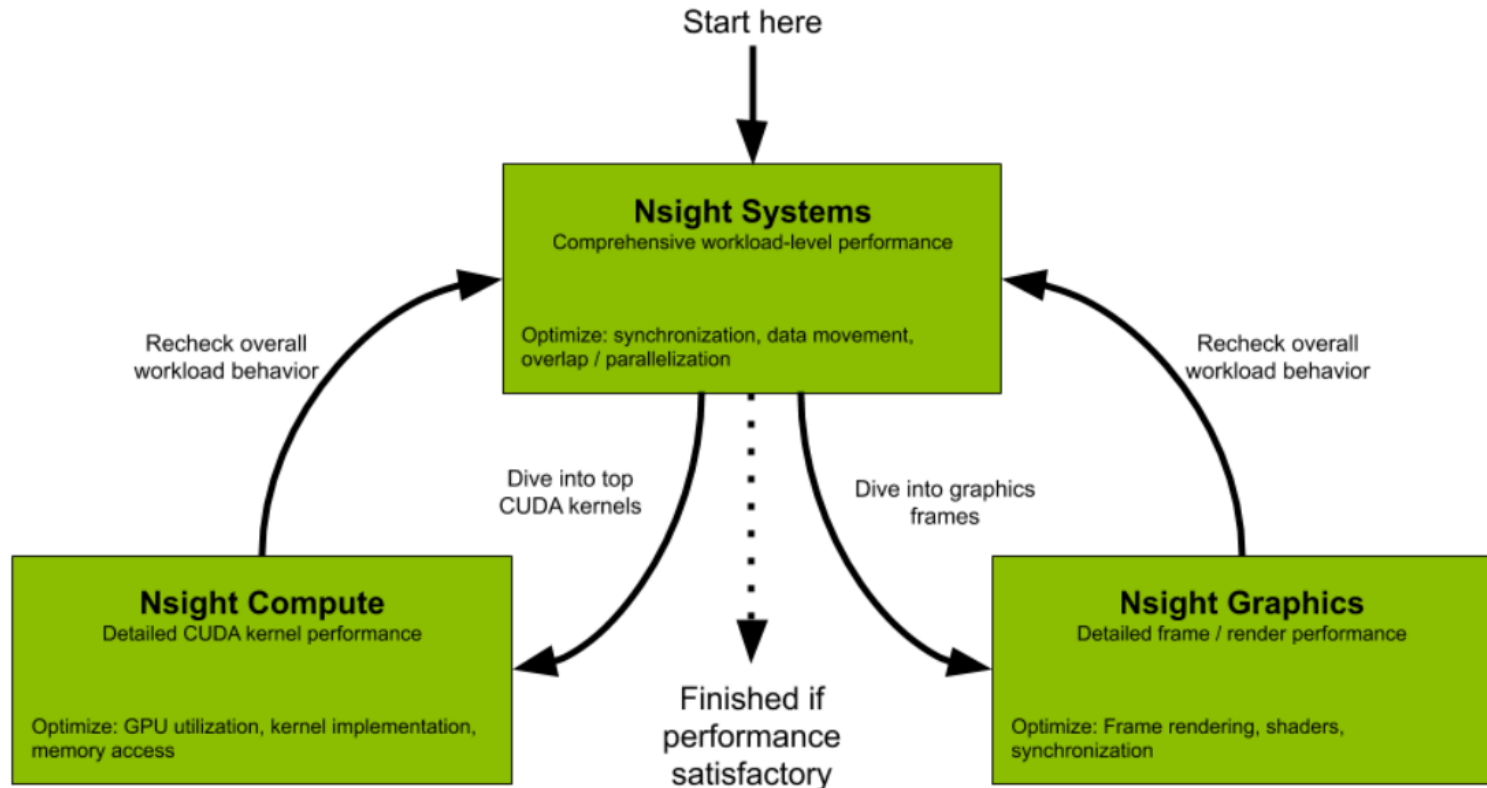
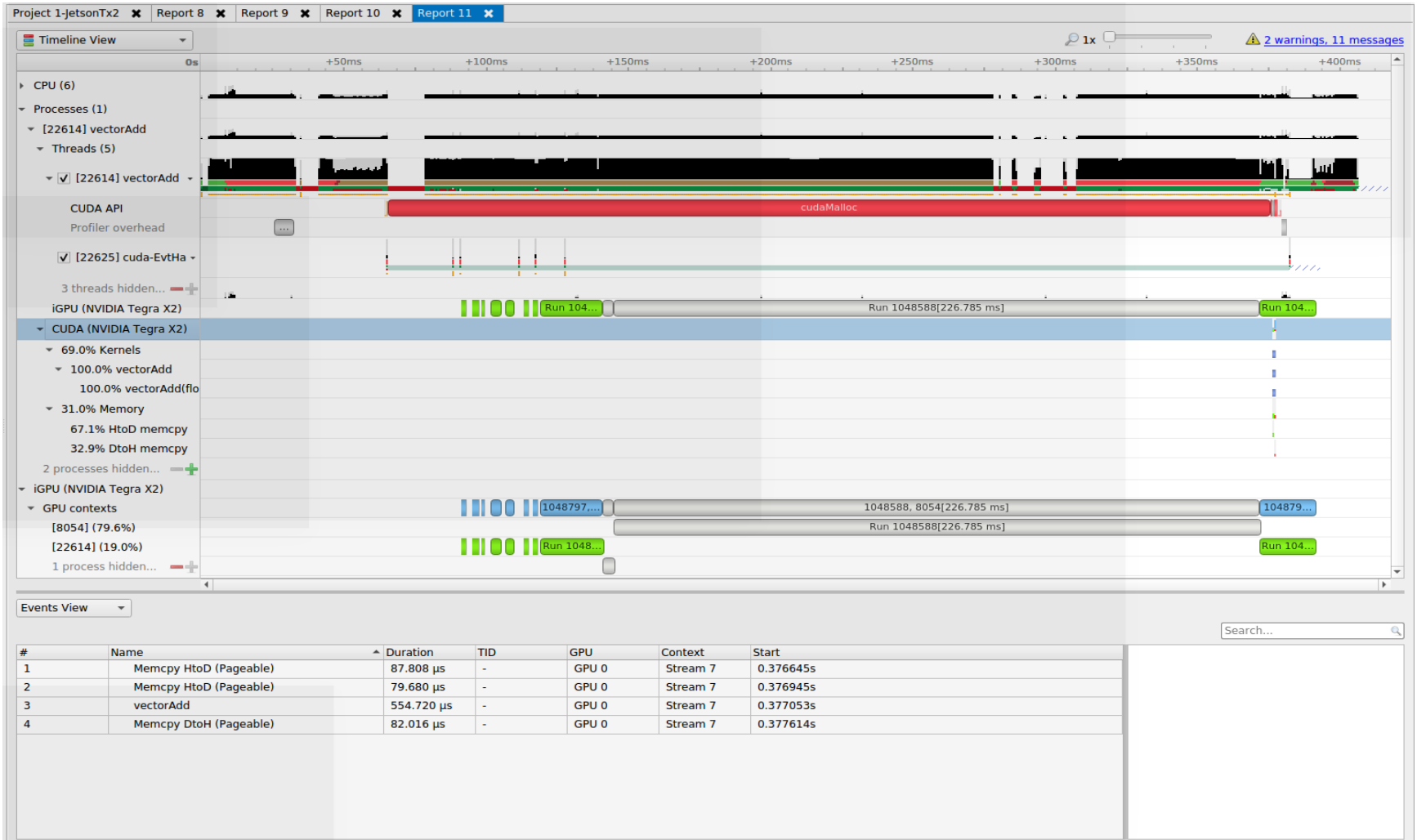
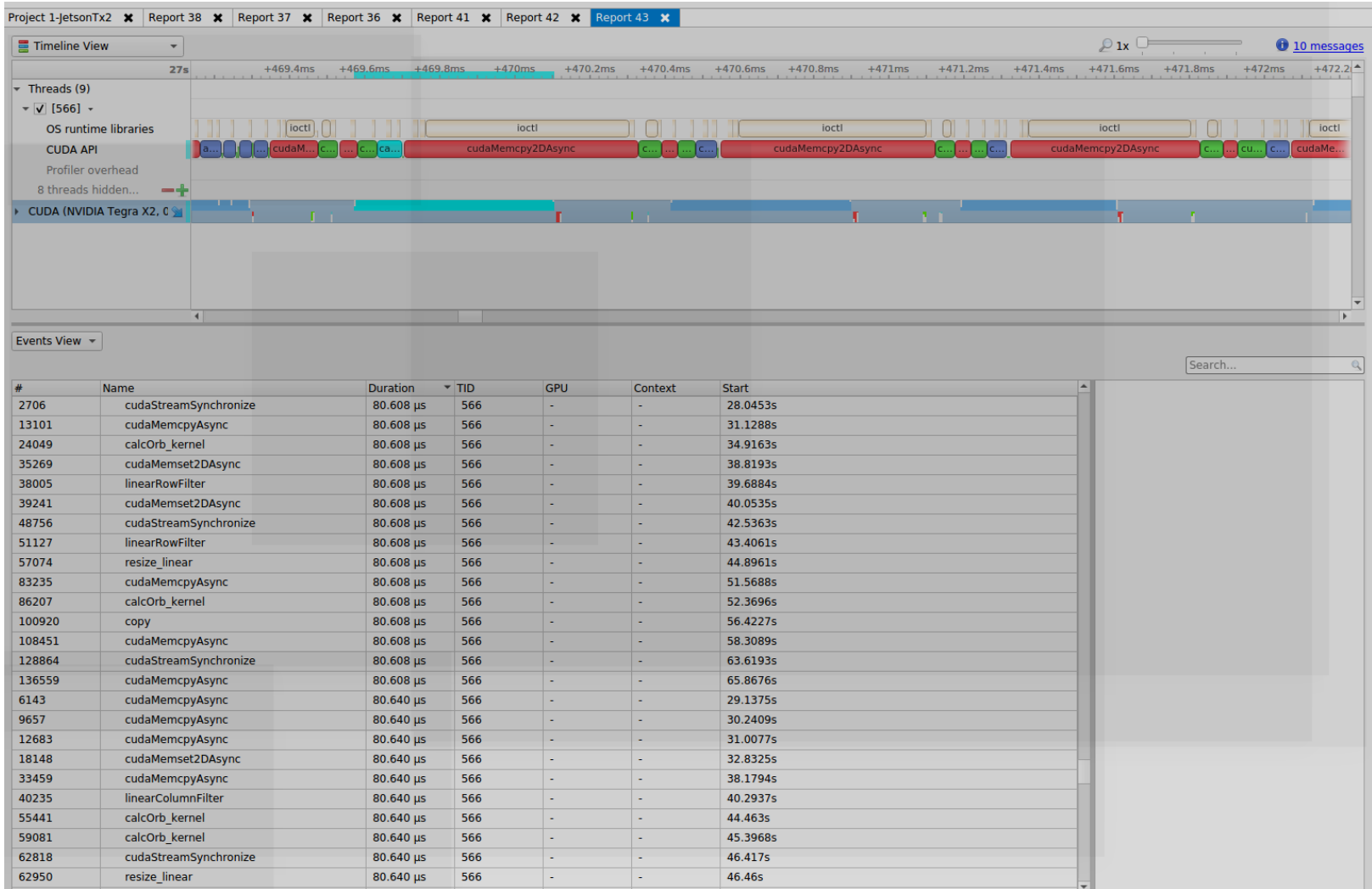


Figure 1. Flowchart describing working with new NVIDIA Nsight tools for performance optimization

Nsight Systems



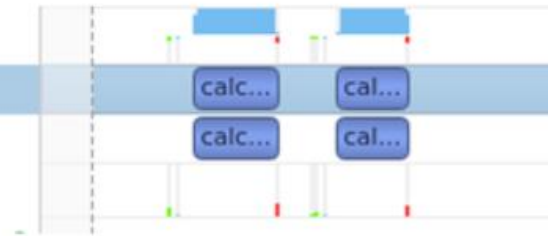
Nsight Systems – Cont.



Nsight Systems – Cont.

CalcOrb_kernel()

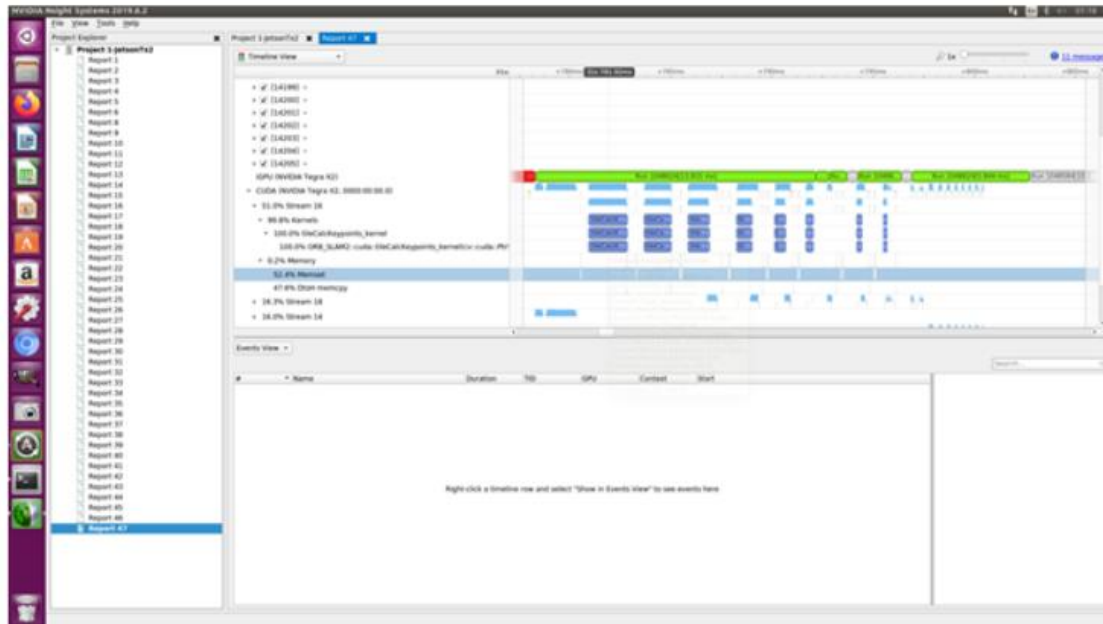
- ▼ 1.3% Stream 27
 - ▼ 97.8% Kernels
 - ▶ 100.0% calcOrb_kernel
 - ▶ 2.2% Memory



- ▼ 7.9% Stream 20
 - ▼ 97.6% Kernels
 - ▶ 100.0% calcOrb_kernel
 - ▼ 2.4% Memory
 - 13.7% Memset
 - 36.7% HtoD memcpy
 - 49.5% DtoH memcpy



Nsight Systems – Cont.



Nsight Compute

The screenshot displays the NVIDIA Nsight Compute application interface, which is used for profiling and analyzing GPU applications. The interface is divided into several panels:

- API Statistics:** A table showing the performance of various API calls. The columns are Name, Number of Calls, Total Duration, Average Duration, and Minimum Duration.
- API Call Log:** A detailed log of API calls, including the API Name, Details, Func Return, and Func Parameter.
- Summary:** A summary of the current session, including the launch ID, device name, and various performance metrics.
- Output Messages:** A panel for viewing output messages and logs.

API Statistics Table:

Name	Number of Calls	Total Duration	Average Duration	Minimum Duration
cudaStreamSynchronize	1780	193.160us	110.810us	55.48us
cudaStreamDestroy	8	335.980us	55.994us	48.342us
cudaStreamCreate	14	113.429us	8.24220us	78.847us
cudaStreamAttachMemAsync	4	423.990us	105.999us	85.423us
cudaMemcpyAsync	544	86.519us	122.276us	183.711us
cudaMemcpy2DAsync	1888	194.437us	178.711us	136.382us
cudaMemcpyToSymbol	4	2.27260us	568.17us	115.479us
cudaMemcpyAsync	2176	788.711us	358.783us	187.113us
cudaMemcpy2DAsync	544	239.599us	440.44us	134.287us
cudaMemcpy2D	78	24.3684us	348.137us	216.398us
cudaMallocPitch	72	18.5622us	271.975us	289.68us
cudaMallocManaged	12	6.39917us	532.264us	293.187us
cudaFree	8	1.54667us	183.334us	78.864us

API Call Log Table:

ID	Time	API Call ID	Function Name	Demangled Name	Device Name
0	2020-Jan-28 21:55:...	4998	copy	void copy@cuda...	NVIDIA Tegr...
1	2020-Jan-28 21:55:...	5156	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
2	2020-Jan-28 21:55:...	5148	copy	void copy@cuda...	NVIDIA Tegr...
3	2020-Jan-28 21:55:...	5144	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
4	2020-Jan-28 21:55:...	5148	copy	void copy@cuda...	NVIDIA Tegr...
5	2020-Jan-28 21:55:...	5152	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
6	2020-Jan-28 21:55:...	5156	copy	void copy@cuda...	NVIDIA Tegr...
7	2020-Jan-28 21:55:...	5160	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
8	2020-Jan-28 21:55:...	5164	copy	void copy@cuda...	NVIDIA Tegr...
9	2020-Jan-28 21:55:...	5168	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
10	2020-Jan-28 21:55:...	5172	copy	void copy@cuda...	NVIDIA Tegr...
11	2020-Jan-28 21:55:...	5176	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
12	2020-Jan-28 21:55:...	5180	copy	void copy@cuda...	NVIDIA Tegr...
13	2020-Jan-28 21:55:...	5184	resize_linear	void resize_linear@cuda...	NVIDIA Tegr...
14	2020-Jan-28 21:55:...	5188	copy	void copy@cuda...	NVIDIA Tegr...
15	2020-Jan-28 21:55:...	6288	tileCalcKernel	0x8 SLAM2:cod...	NVIDIA Tegr...
16	2020-Jan-28 21:55:...	6212	tileCalcKernel	0x8 SLAM2:cod...	NVIDIA Tegr...
17	2020-Jan-28 21:55:...	6224	tileCalcKernel	0x8 SLAM2:cod...	NVIDIA Tegr...
18	2020-Jan-28 21:55:...	6236	addBorder_kern...	0x8 SLAM2:cod...	NVIDIA Tegr...
19	2020-Jan-28 21:55:...	6234	IC Angle kernel	0x8 SLAM2:cod...	NVIDIA Tegr...
20	2020-Jan-28 21:55:...	11184	linearFlowFilter	void row_filt...	NVIDIA Tegr...

Nsight Compute – Cont.

The screenshot displays the NVIDIA Nsight Compute application window. At the top, the 'Page' is set to 'Session' and the 'Launch' dropdown shows '6200 - tileCalcKeypoints_kernel'. Below this, the 'Current' session details are shown: '6200 - tileCalcKeypoints_kernel (0, 0, 0)' with 'Time: n/a', 'Cycles: n/a', 'Regs: n/a', 'GPU: NVIDIA Tegra X2', 'SM Frequency: n/a', and 'CC: 6.2'.

The 'Launch Settings' section lists the following parameters:

- Executable: ./build/mono_tum
- Working Directory: /home/slamgpu/ORB_SLAM2_CUDA
- Command Line Arguments: Vocabulary/ORBvoc.txt Examples/Monocular/TUM1.yaml /media/slamgpu/64GB/rgbd_dataset_freiburg1_desk true
- Environment: DISPLAY=:1
- Activity Type: Interactive Profile
- Enable NVTX Support: Yes
- Disable Profiling Start/Stop: Yes
- Enable Profiling From Start: Yes

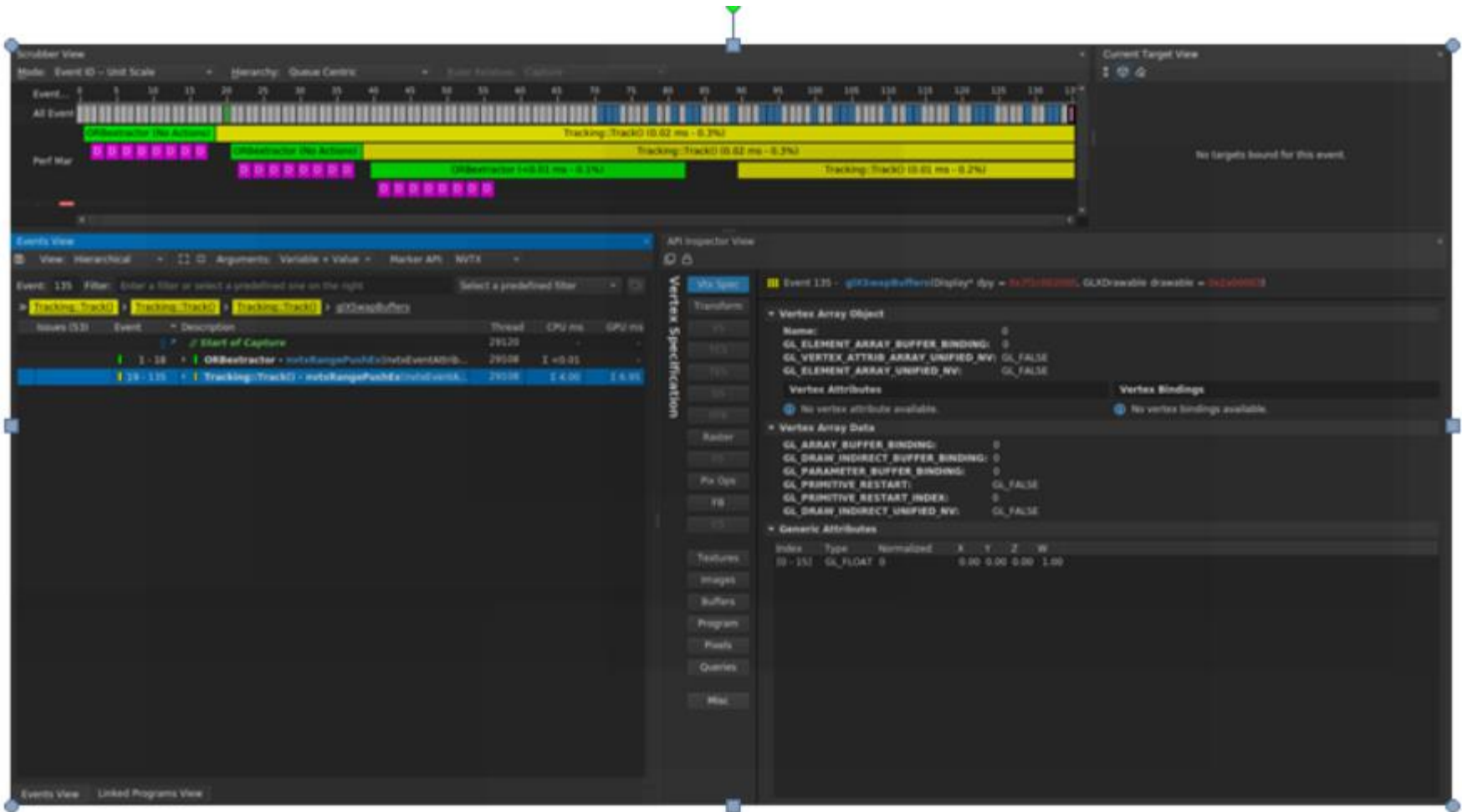
The 'Session Info' section provides details about the session and host:

- Created: 2020-Jan-28 21:55:28
- Host Name: yoni3-GB-BRi7-H-8550
- Host OS: Linux - #81~16.04.1-Ubuntu SMP Tue Nov 26 16:34:21 UTC 2019
- Host Architecture: x86_64
- Host Processor: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
- Target Name: slamgpu-desktop
- Target OS: Linux - #1 SMP PREEMPT Mon Aug 12 21:29:52 PDT 2019
- Target Architecture: aarch64
- CUDA Version: 10.0
- NVIDIA Nsight Compute: 1.0 (Build 25942742)

The 'Device Attributes' section shows a table of attributes for the 'NVIDIA Tegra X2 (0)' device:

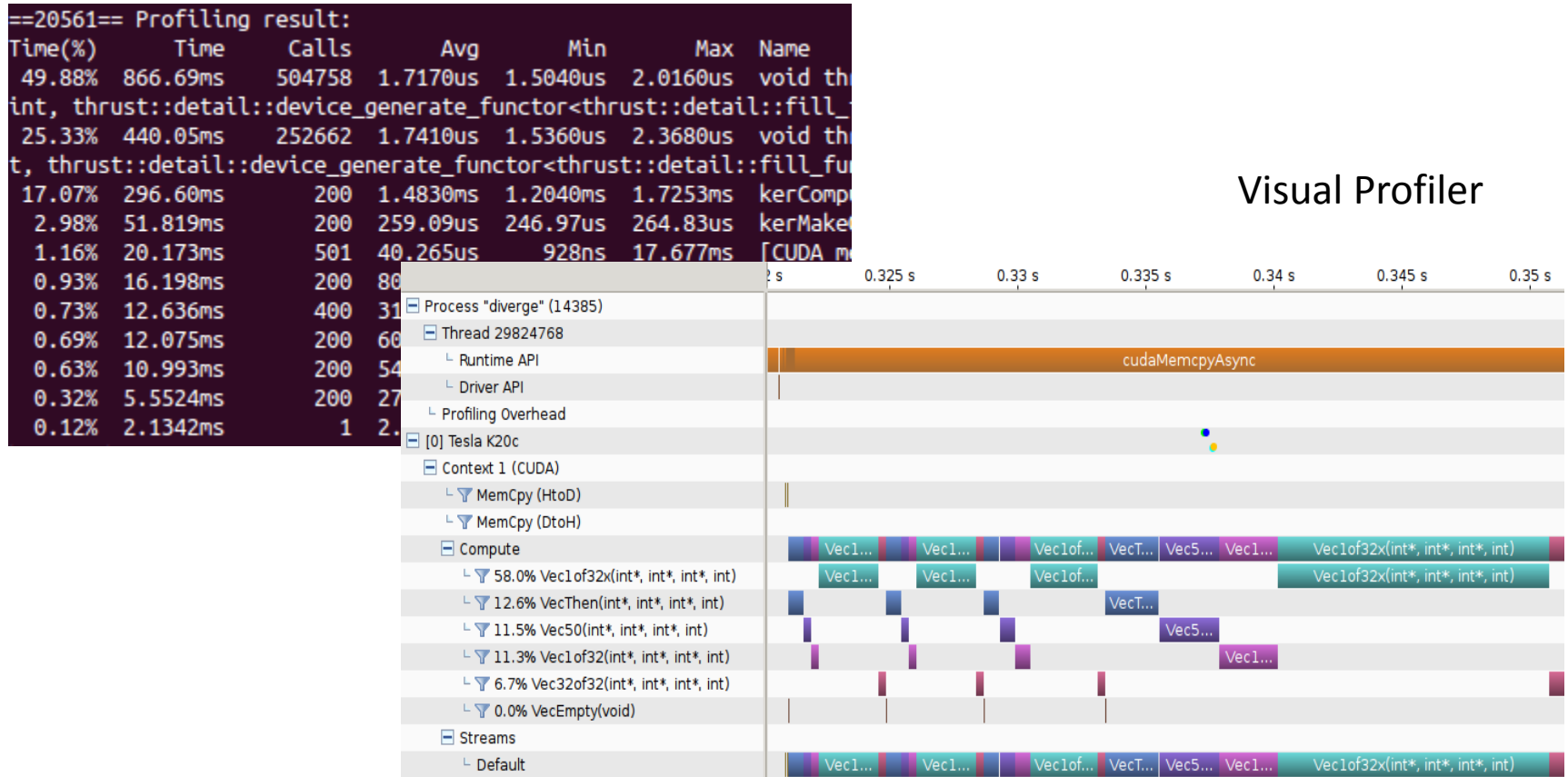
Attribute	NVIDIA Tegra X2 (0)
architecture	304
async_engine_count	1
can_flush_remote_writes	0
can_map_host_memory	1
can_tex2d_gather	1
can_use_64_bit_stream_mem_ops	0
can_use_host_pointer_for_registered_mem	0
can_use_stream_mem_ops	0
can_use_stream_wait_value_nor	0
chip	315

Nsight Graphics

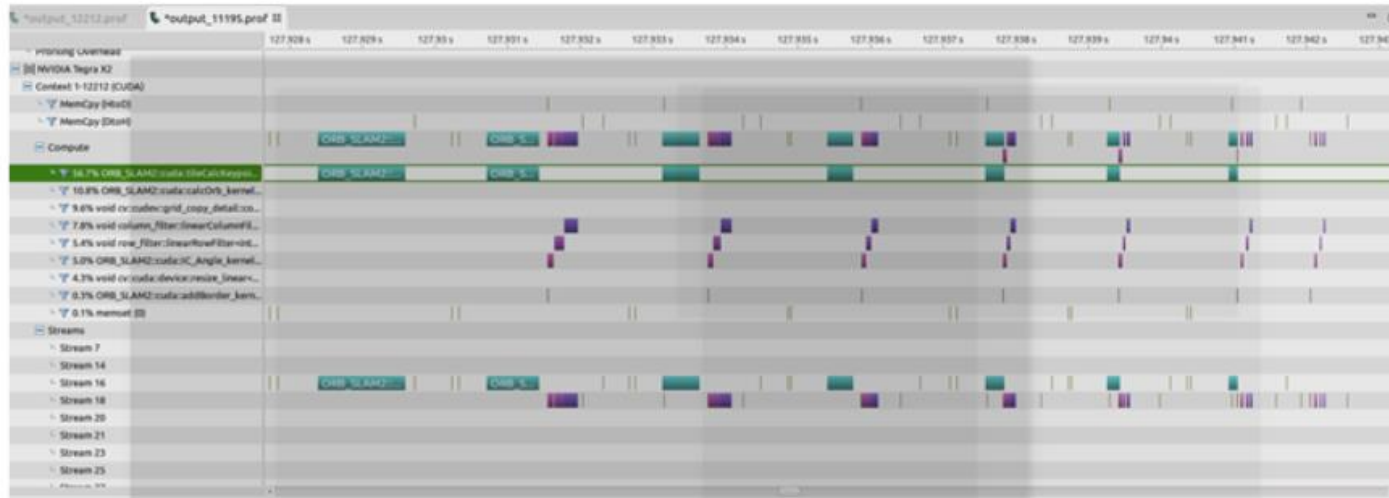


Profiling Tools - Legacy

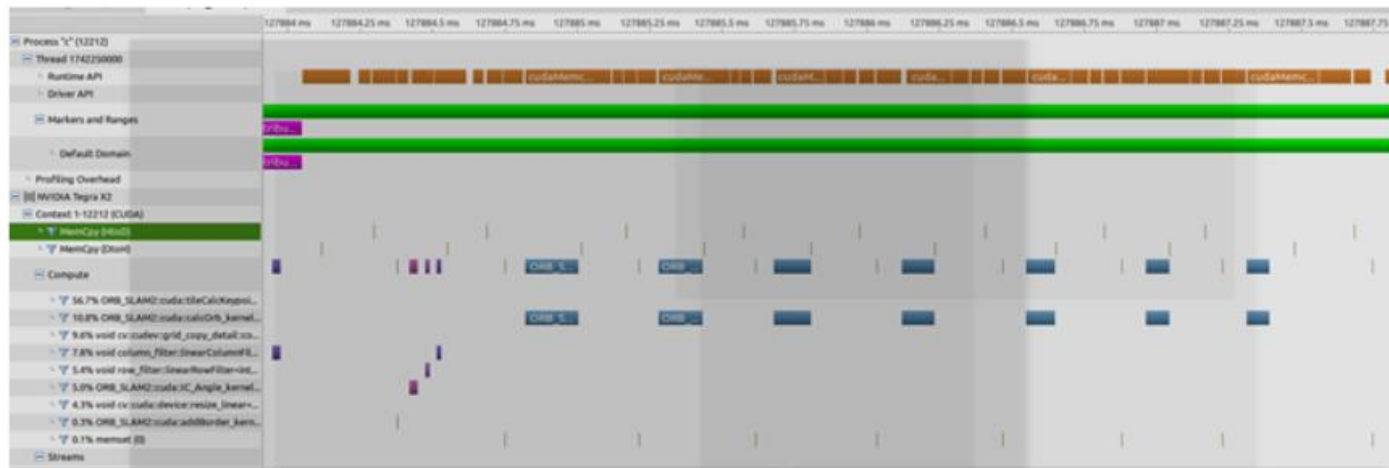
NVPROF



Profiling Tools – Legacy – Cont.



You can see the times for transfer data between host to device and vice versa.



Remote Profiling Configuration

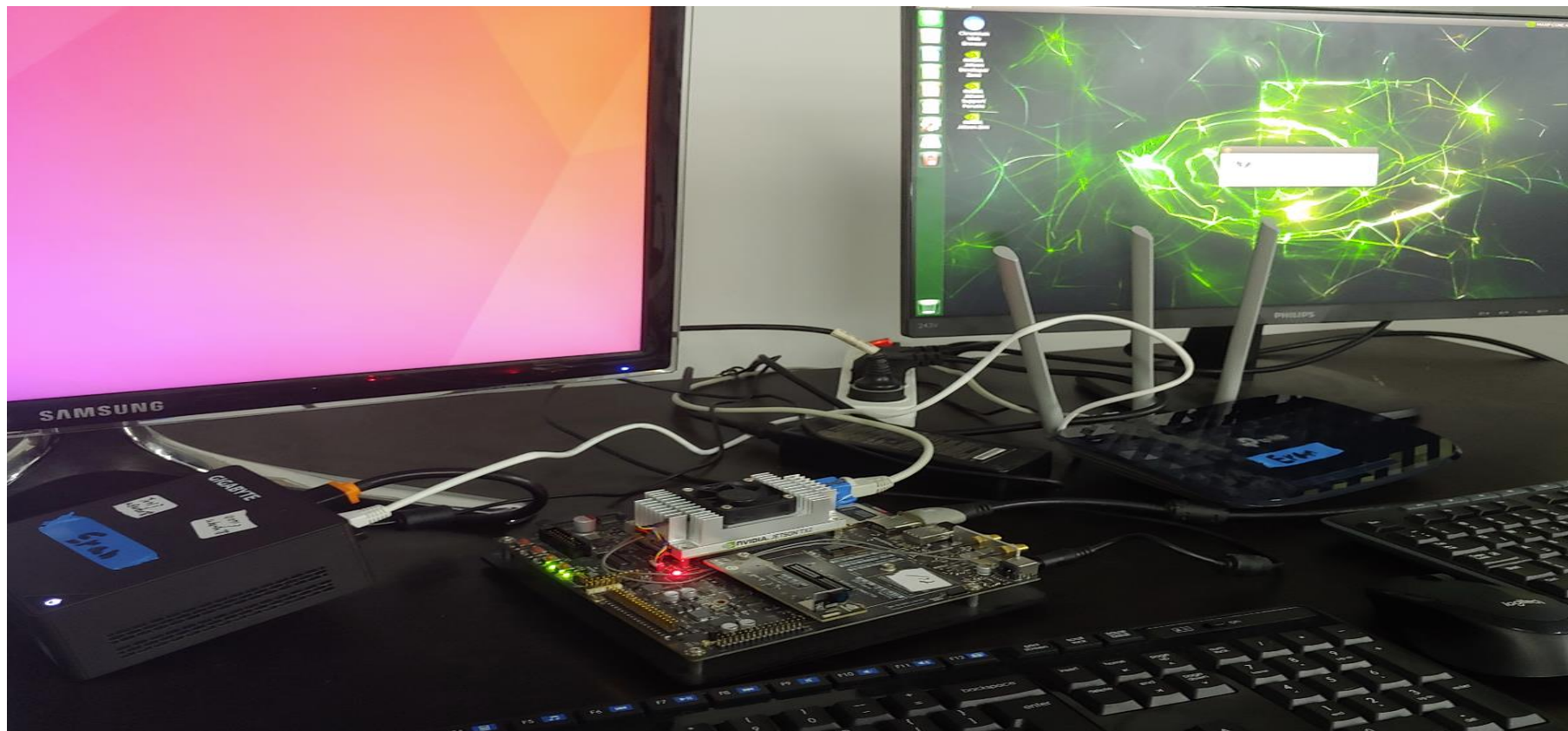
Jetson TX2



Switch/Router



Remote Computer



Future Research

This project will help in the future to students and people which interesting in **Parallel Programming** study **Cuda programming** in easy and fast way then could implement, improve and optimize parallel applications. By using Nvidia profiling tools students and people could understand how to **debug** in better way application which run on CPU and especially on GPU, understand more **deeply** about the **Parallel concepts** and how to improve the **Performance**.

Suggestion For Next Research Project

In the next project, the student can learn about the Cuda programming then deep into the computer vision algorithms (Specially FAST, ORB algorithms) then run the ORB SLAM application and change the kernels (GPU main function calls) parameters like grid size (blocks count) and threads per block and try to optimize the application and use the profiling tools to archive performance and latency benefits. Optimize the code and add more functionalities can be useful for the future.

Questions ???



Thanks

