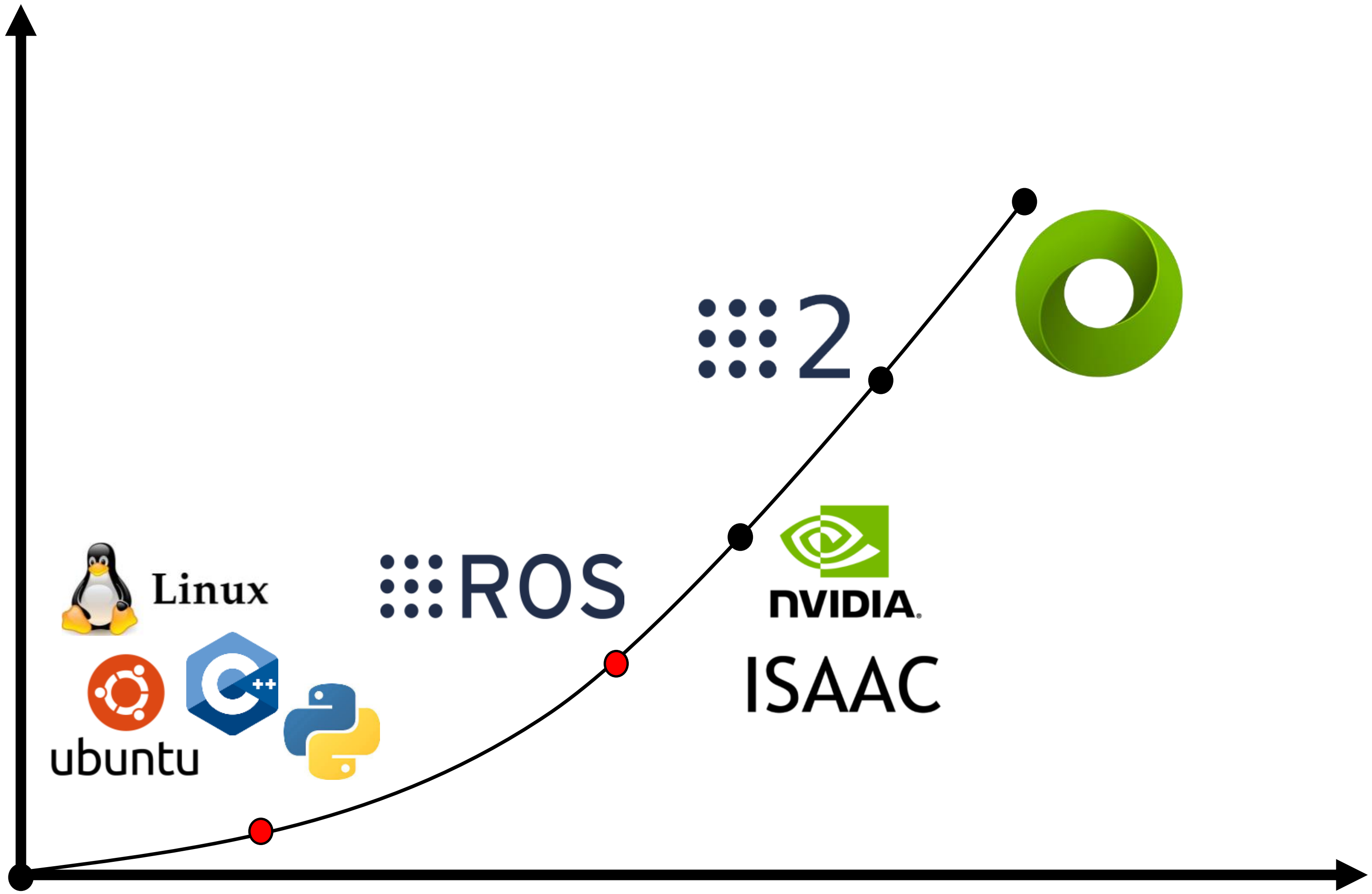
LEONARDO

LEONARDO TECHNICAL TRAINING

Robotic Software
Lezione 5

NVIDIA ISAAC SDK

Learning path



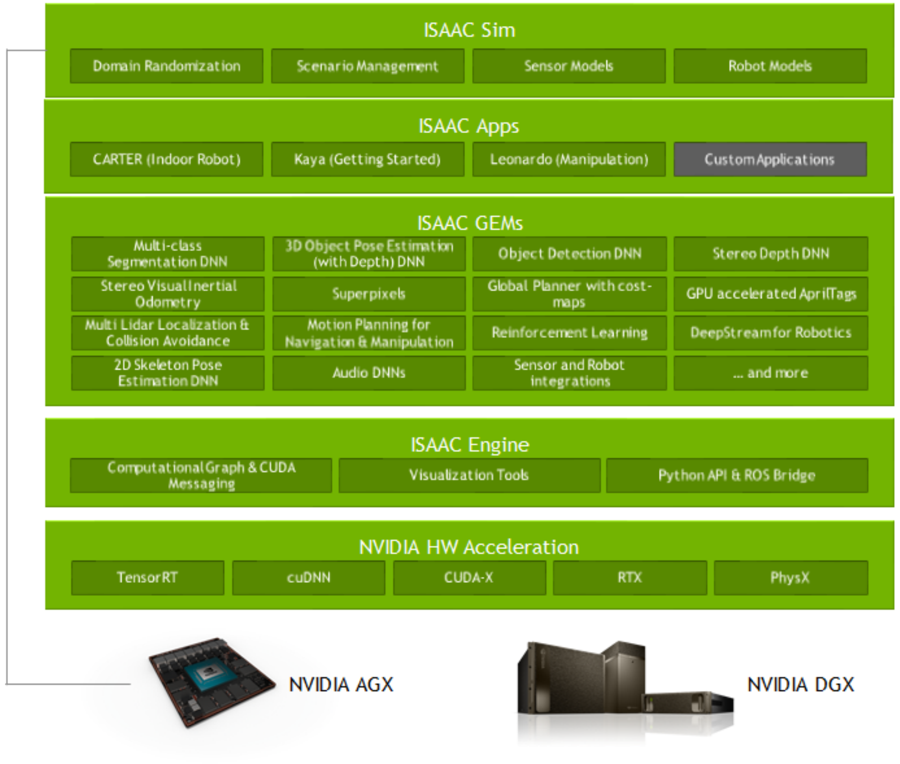
NVIDIA SDK

- Goal:
    - Build and deploy commercial-grade, AI-powered robots
    - The NVIDIA Isaac SDK is a toolkit that includes building blocks and tools that accelerate robot developments that require the increased perception and navigation features enabled by AI

- Artificial Intelligence: the SDK features GPU-accelerated algorithms and deep neural networks (DNNs) for perception and planning, and machine learning workflows for supervised and reinforcement learning
- Navigation: modular robotic algorithms provide sensing, planning, or actuation for both navigation use cases
- Simulation: training and continuous testing in high-fidelity physics and photorealistic simulation accelerates robot development and deployment (ISAAC-SIM)

NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework
  - Isaac GEMS: packages with high-performance robotics algorithms
    - Open-source
  - Isaac Apps: reference applications
  - NVIDIA Isaac Sim: a powerful simulation platform
  - These tools and APIs accelerate robot development by making it easier to add for perception and navigation



ISAAC Sim

Domain Randomization
Scenario Management
Sensor Models
Robot Models

ISAAC Apps

CARTER (Indoor Robot)
Kaya (Getting Started)
Leonardo (Manipulation)
Custom Applications

ISAAC GEMs

Multi-class Segmentation DNN
3D Object Pose Estimation (with Depth) DNN
Object Detection DNN
Stereo Depth DNN
Stereo Visual Inertial Odometry
Superpixels
Global Planner with cost-maps
GPU accelerated AprilTags
Multi Lidar Localization & Collision Avoidance
Motion Planning for Navigation & Manipulation
Reinforcement Learning
DeepStream for Robotics
2D Skeleton Pose Estimation DNN
Audio DNNs
Sensor and Robot integrations
... and more

ISAAC Engine

Computational Graph & CUDA Messaging
Visualization Tools
Python API & ROS Bridge

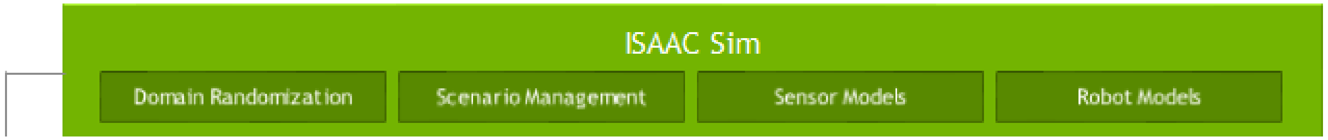
NVIDIA HW Acceleration

TensorRT
cuDNN
CUDA-X
RTX
PhysX

NVIDIA AGX
NVIDIA DGX

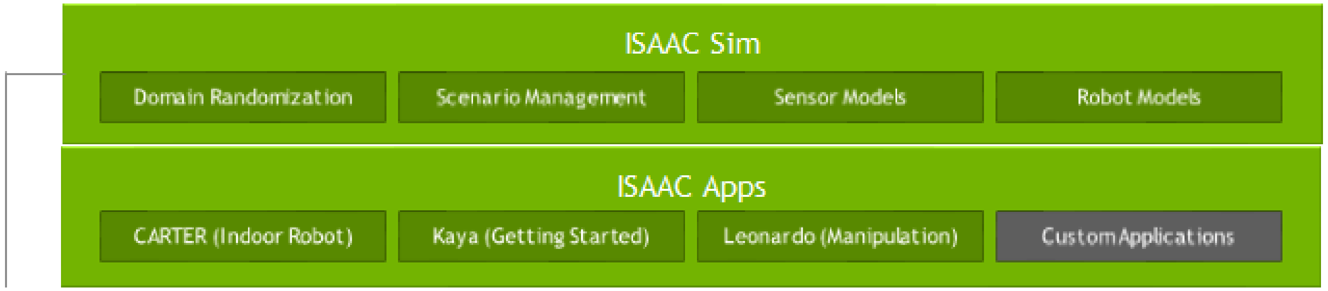
NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework



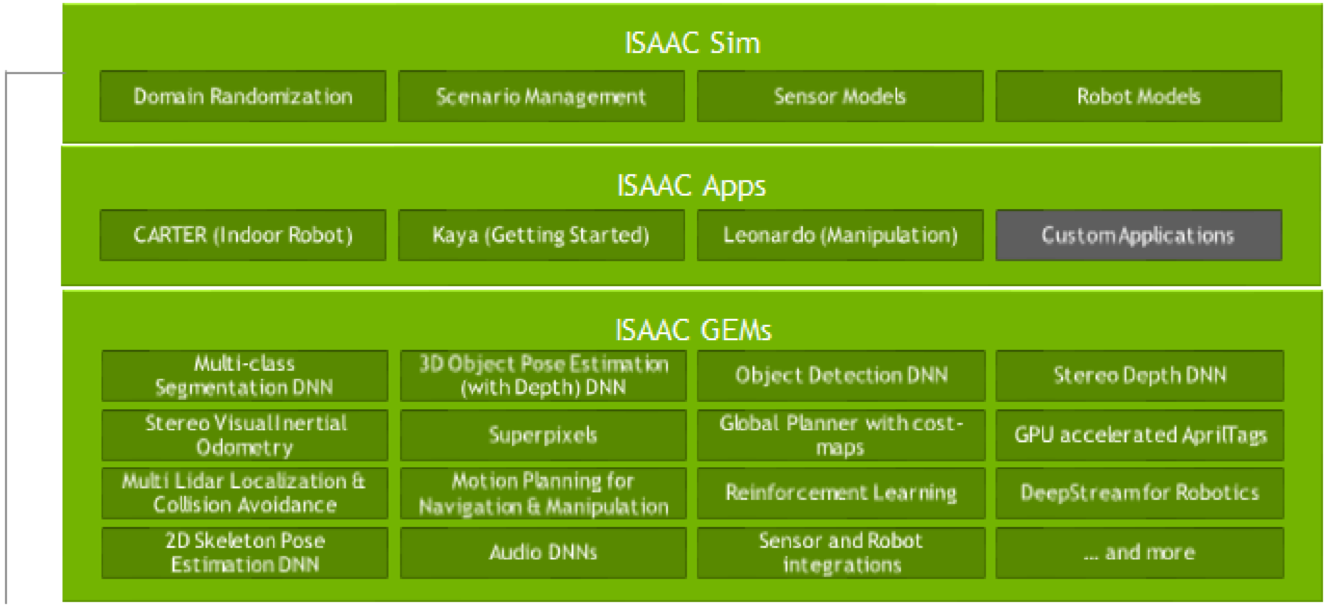
NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework



NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework



ISAAC Sim

Domain Randomization | Scenario Management | Sensor Models | Robot Models

ISAAC Apps

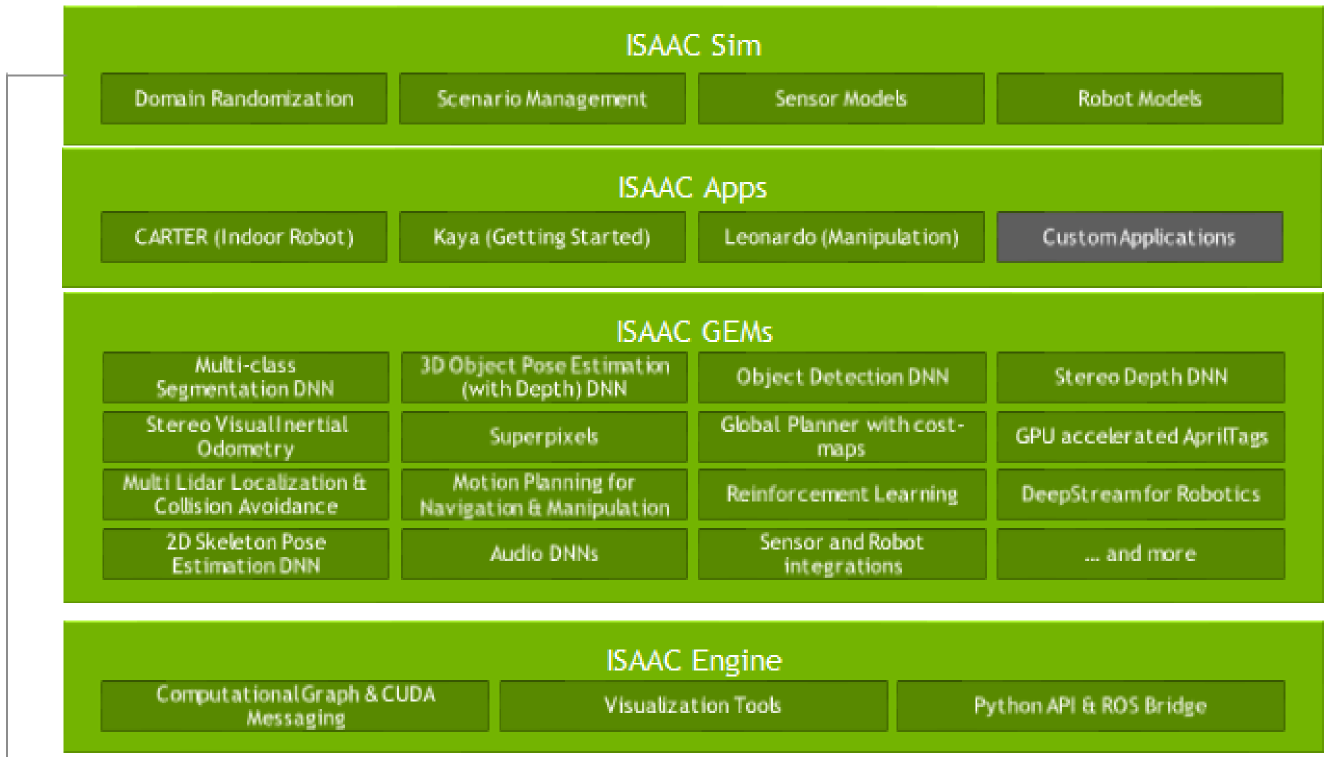
CARTER (Indoor Robot) | Kaya (Getting Started) | Leonardo (Manipulation) | Custom Applications

ISAAC GEMs

Multi-class Segmentation DNN	3D Object Pose Estimation (with Depth) DNN	Object Detection DNN	Stereo Depth DNN
Stereo Visual Inertial Odometry	Superpixels	Global Planner with cost-maps	GPU accelerated AprilTags
Multi Lidar Localization & Collision Avoidance	Motion Planning for Navigation & Manipulation	Reinforcement Learning	DeepStream for Robotics
2D Skeleton Pose Estimation DNN	Audio DNNs	Sensor and Robot integrations	... and more

NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework



ISAAC Sim

Domain Randomization | Scenario Management | Sensor Models | Robot Models

ISAAC Apps

CARTER (Indoor Robot) | Kaya (Getting Started) | Leonardo (Manipulation) | Custom Applications

ISAAC GEMs

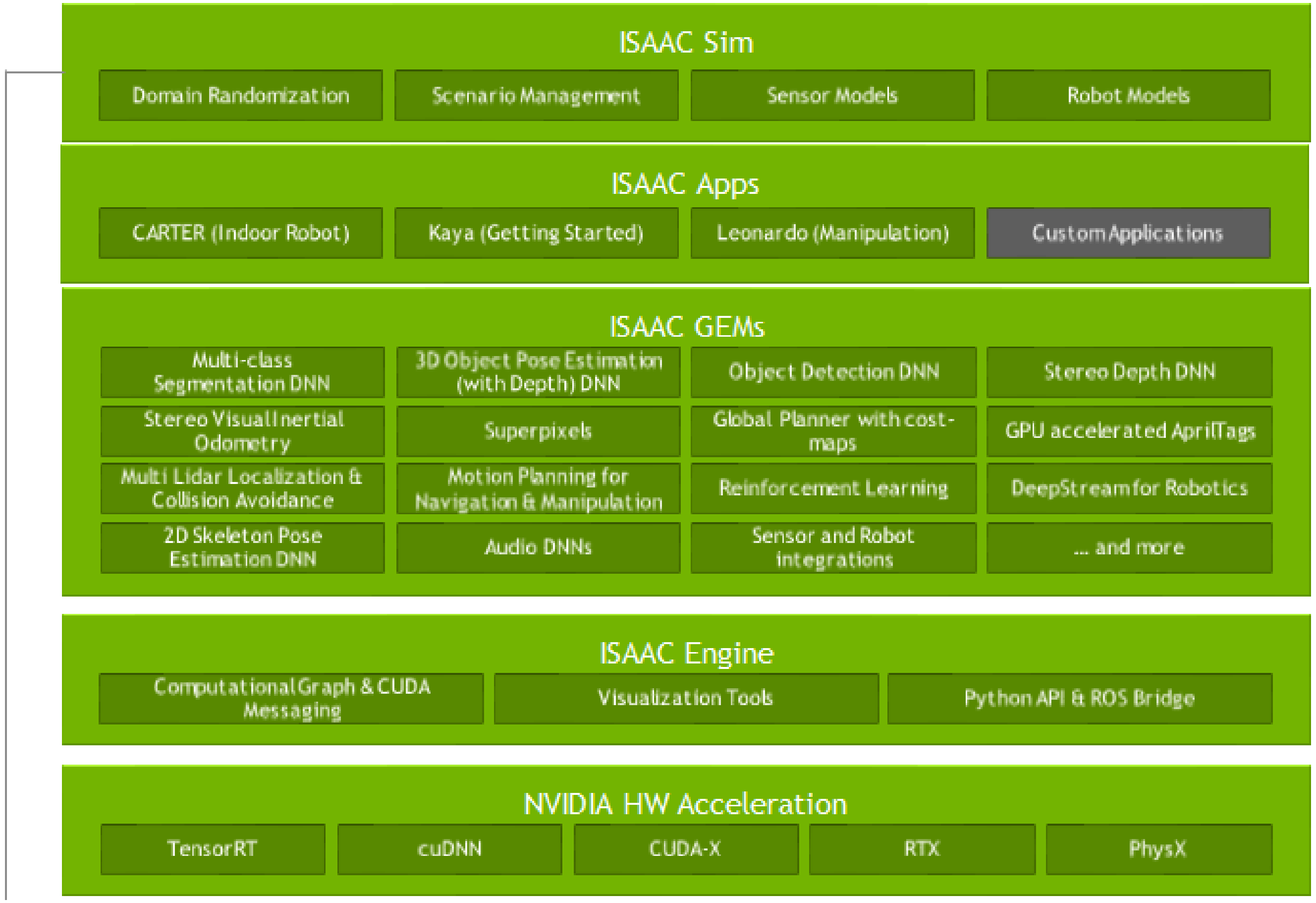
Multi-class Segmentation DNN	3D Object Pose Estimation (with Depth) DNN	Object Detection DNN	Stereo Depth DNN
Stereo Visual Inertial Odometry	Superpixels	Global Planner with cost-maps	GPU accelerated AprilTags
Multi Lidar Localization & Collision Avoidance	Motion Planning for Navigation & Manipulation	Reinforcement Learning	DeepStream for Robotics
2D Skeleton Pose Estimation DNN	Audio DNNs	Sensor and Robot integrations	... and more

ISAAC Engine

Computational Graph & CUDA Messaging | Visualization Tools | Python API & ROS Bridge

NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework



ISAAC Sim

Domain Randomization | Scenario Management | Sensor Models | Robot Models

ISAAC Apps

CARTER (Indoor Robot) | Kaya (Getting Started) | Leonardo (Manipulation) | Custom Applications

ISAAC GEMs

Multi-class Segmentation DNN | 3D Object Pose Estimation (with Depth) DNN | Object Detection DNN | Stereo Depth DNN

Stereo Visual Inertial Odometry | Superpixels | Global Planner with cost-maps | GPU accelerated AprilTags

Multi Lidar Localization & Collision Avoidance | Motion Planning for Navigation & Manipulation | Reinforcement Learning | DeepStream for Robotics

2D Skeleton Pose Estimation DNN | Audio DNNs | Sensor and Robot integrations | ... and more

ISAAC Engine

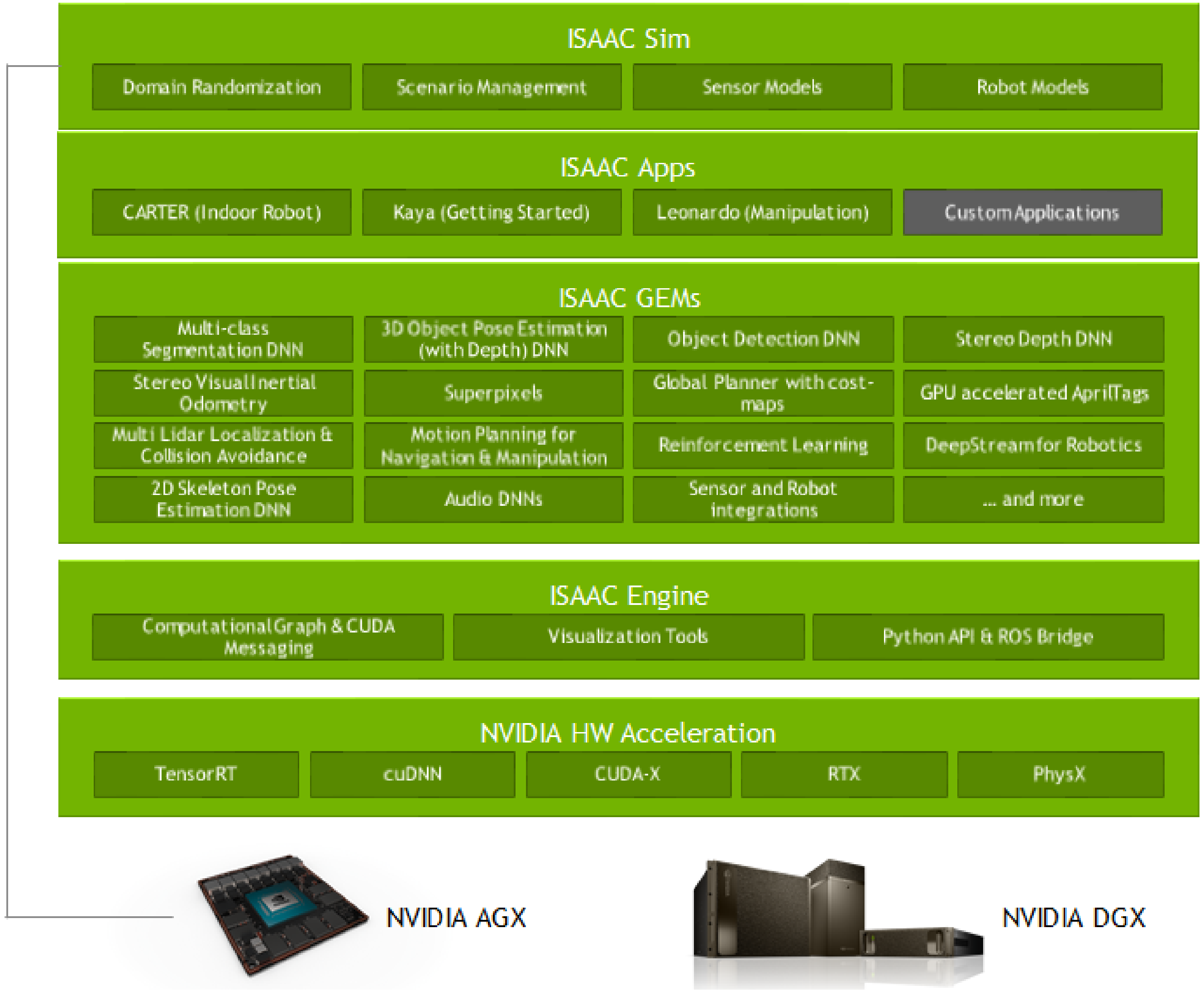
Computational Graph & CUDA Messaging | Visualization Tools | Python API & ROS Bridge

NVIDIA HW Acceleration

TensorRT | cuDNN | CUDA-X | RTX | PhysX

NVIDIA SDK

- The SDK includes the Isaac Engine: an application framework



ISAAC Sim

Domain Randomization | Scenario Management | Sensor Models | Robot Models

ISAAC Apps

CARTER (Indoor Robot) | Kaya (Getting Started) | Leonardo (Manipulation) | Custom Applications

ISAAC GEMs

Multi-class Segmentation DNN | 3D Object Pose Estimation (with Depth) DNN | Object Detection DNN | Stereo Depth DNN

Stereo Visual Inertial Odometry | Superpixels | Global Planner with cost-maps | GPU accelerated AprilTags

Multi Lidar Localization & Collision Avoidance | Motion Planning for Navigation & Manipulation | Reinforcement Learning | DeepStream for Robotics

2D Skeleton Pose Estimation DNN | Audio DNNs | Sensor and Robot integrations | ... and more

ISAAC Engine

Computational Graph & CUDA Messaging | Visualization Tools | Python API & ROS Bridge

NVIDIA HW Acceleration

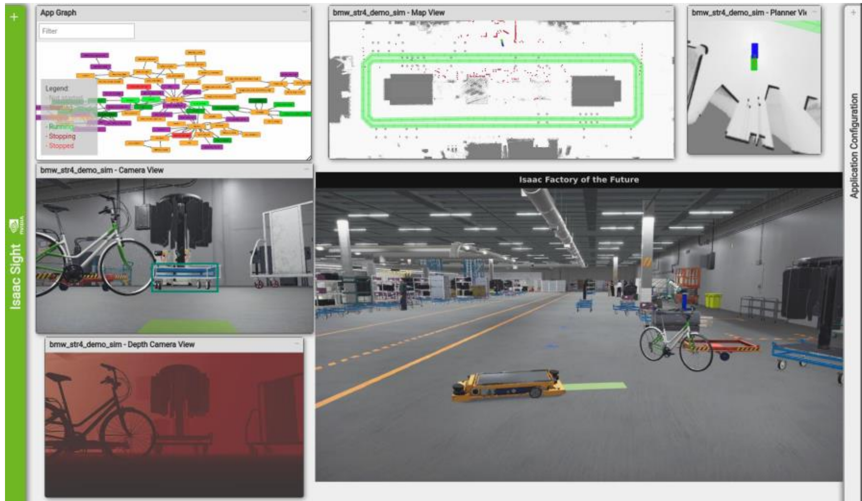
TensorRT | cuDNN | CUDA-X | RTX | PhysX

NVIDIA AGX

NVIDIA DGX

NVIDIA SDK

- Isaac is NVIDIA's open platform for intelligent robots
- The Isaac SDK provides a large collection of powerful GPU-accelerated algorithm GEMs for navigation and manipulation
- Isaac SDK Engine is a framework to easily write modular applications and deploy them on a real robot
- Isaac SDK comes with various example applications from basic samples that show specific features to applications that facilitate complicated robotics use cases
- Isaac SDK also works hand-in-hand with Isaac SIM, which allows for development, testing, and training of robots in a virtual environment.

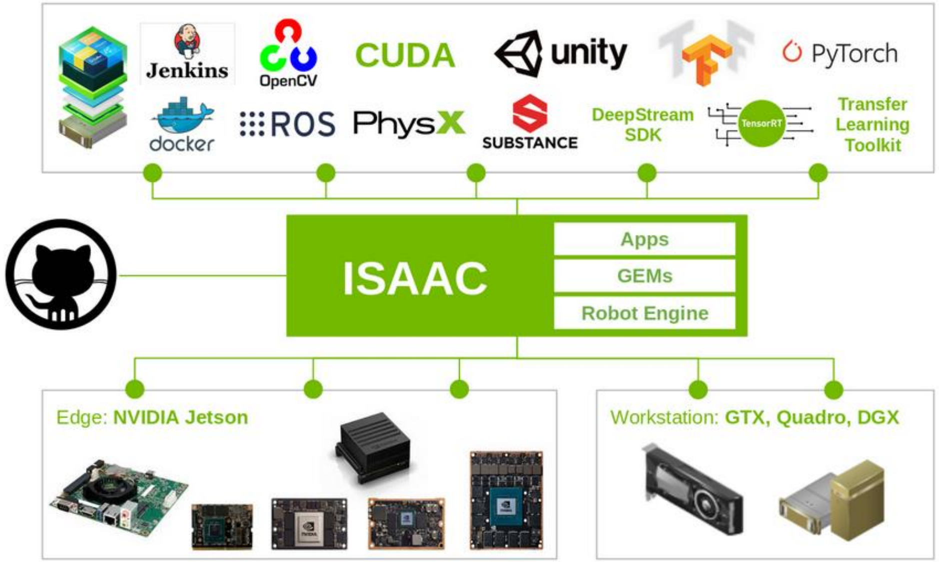

Isaac Sight
App Graph
Filter
Legend
- Running
- Stopping
- Stopped
bmw_str4_demo_sim - Map View
bmw_str4_demo_sim - Planner Vi
bmw_str4_demo_sim - Camera View
Isaac Factory of the Future
bmw_str4_demo_sim - Depth Camera View
Application Configuration

NVIDIA SDK

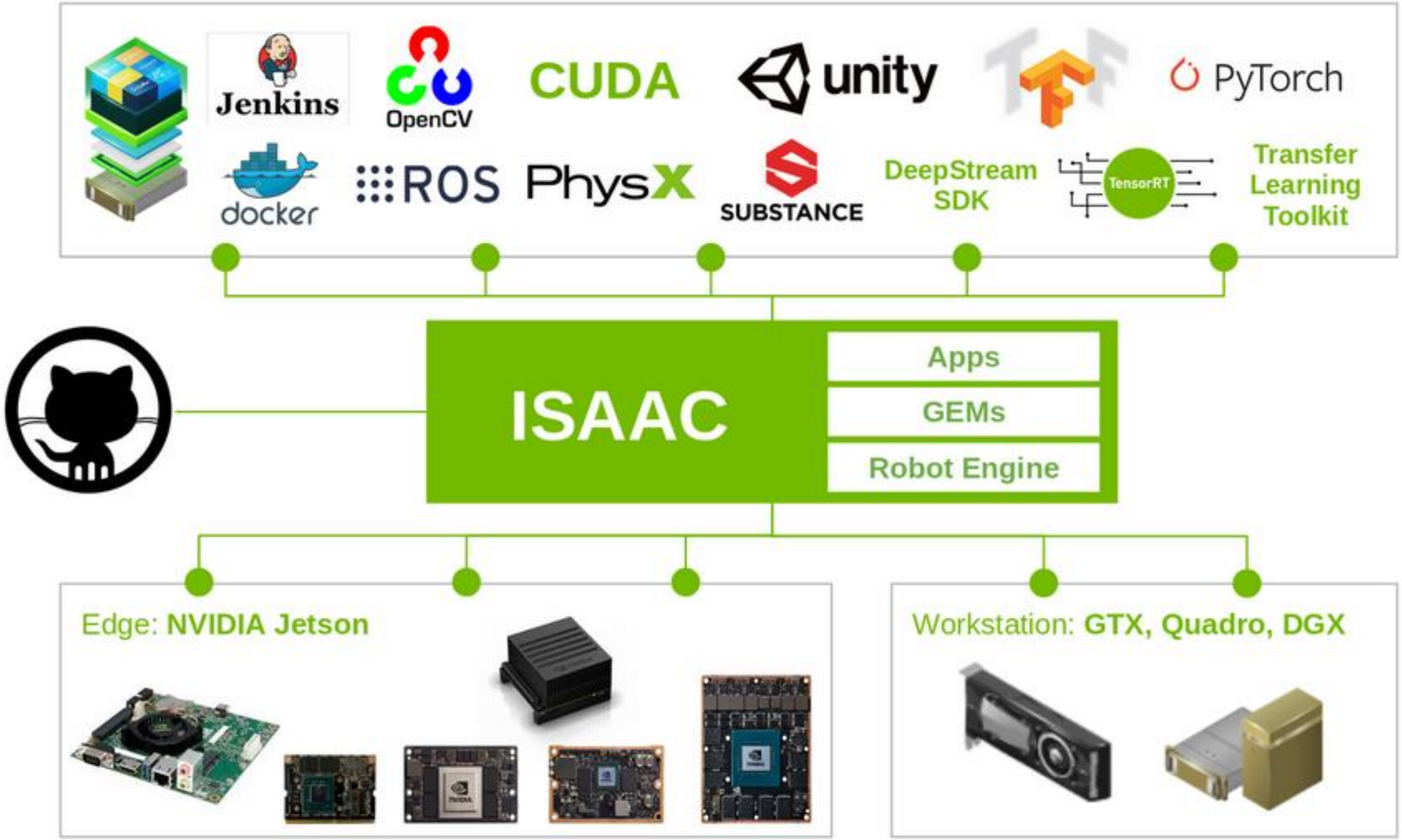
- Isaac GEMs
  - Robotics combines many different disciplines
    - low-level hardware drivers
    - Safe planning algorithms
    - Fast and accurate computer vision
    - Deep neural networks
    - High-level artificial intelligence
  - GEMs: accelerate the development of challenging robotics applications
    - Isaac provides planning and perception GEMs for navigation and manipulation use cases
    - GEMs also provide support for key hardware components and robotic peripherals.

NVIDIA SDK

- Isaac Applications
  - Isaac SDK provides various sample applications
  - Highlight features of Isaac SDK Engine or focus on the functionality of a particular Isaac SDK GEM. These sample applications are good starting points for learning Isaac
  - The Isaac SDK is meant for development of applications for complicated use cases like a delivery robot
  - The Carter application gives you a starting point for building your own delivery robot
  - Carter can drive to a goal location, patrol a building, or similar
  - The Carter navigation stack is based on a Lidar.
  - Isaac SDK is also supported by a rich ecosystem, and Isaac SDK Engine connects Isaac GEMs to existing packages like OpenCV, ROS, PCL, and others.
    - Very similar to ROS

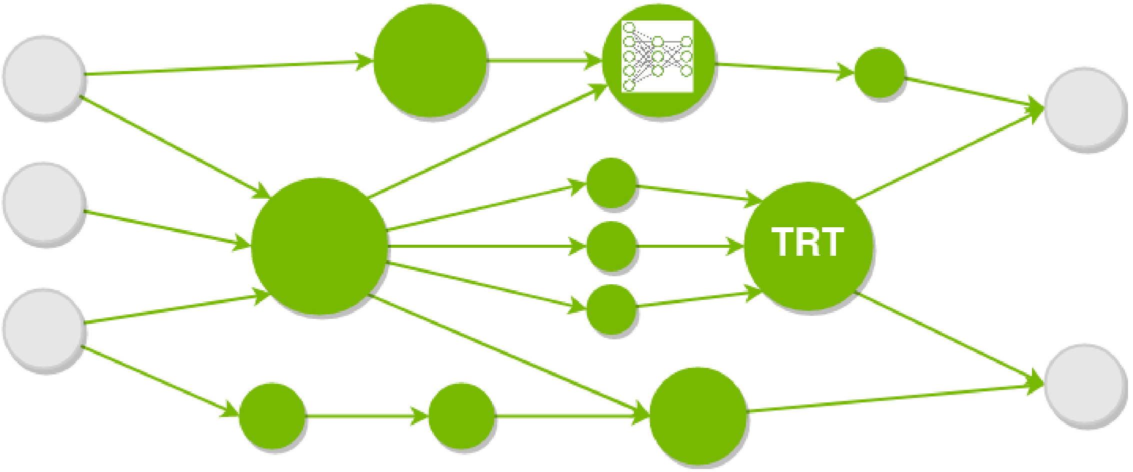


NVIDIA SDK


Jenkins
OpenCV
CUDA
unity
PyTorch
docker
ROS
PhysX
SUBSTANCE
DeepStream SDK
TensorRT
Transfer Learning Toolkit
ISAAC
Apps
GEMs
Robot Engine
Edge: NVIDIA Jetson
Workstation: GTX, Quadro, DGX

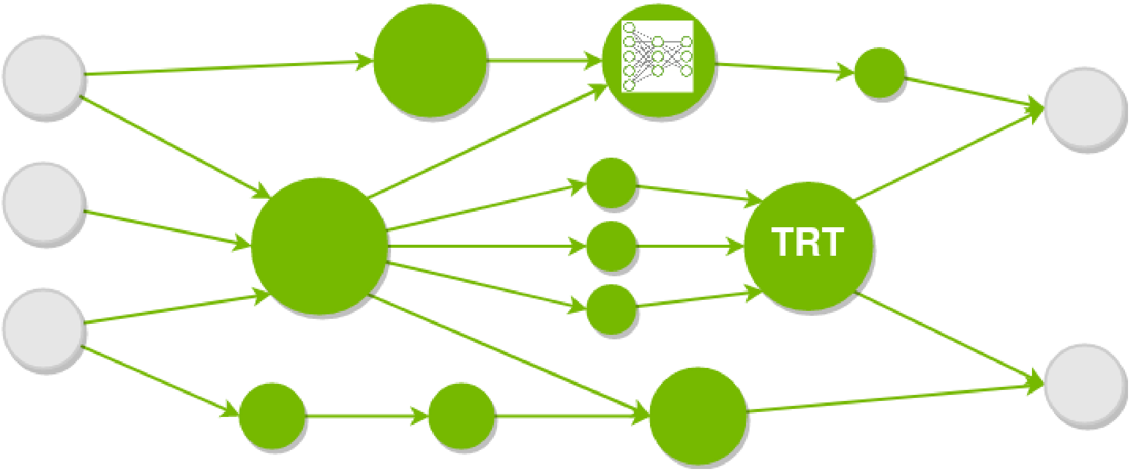
NVIDIA SDK

- Isaac Engine
  - ISAAC SDK Engine: a feature-rich framework for building modular robotics applications
    - With Isaac, you can build an application out of small components, which pass messages between each other and can be configured to your custom use case.
  - Toolchains based on the Bazel build system for building and deploying applications
  - You can build and run applications with a command as simple as *bazel run*
  - All external dependencies are pulled automatically to your system without any additional setup
  - The Setup section of this document explains the few steps necessary for getting started
  - Isaac SDK Engine fully supports NVIDIA GPUs and CUDA



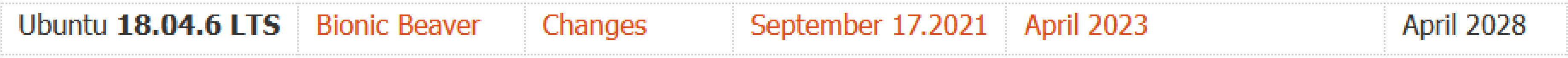
NVIDIA SDK

- Bazel
  - Large software projects need a reliable and efficient build system and Isaac SDK uses Bazel
  - Bazel enables clean module dependencies, hermetic builds, and cross-compilation for various hardware platforms like the Jetson TX2 or Jetson Xavier developer kits
  - Bazel is installed by the dependency script.



Install

- The last version, ISAAC SDK 2021.1 is only supported from Ubuntu 18.04

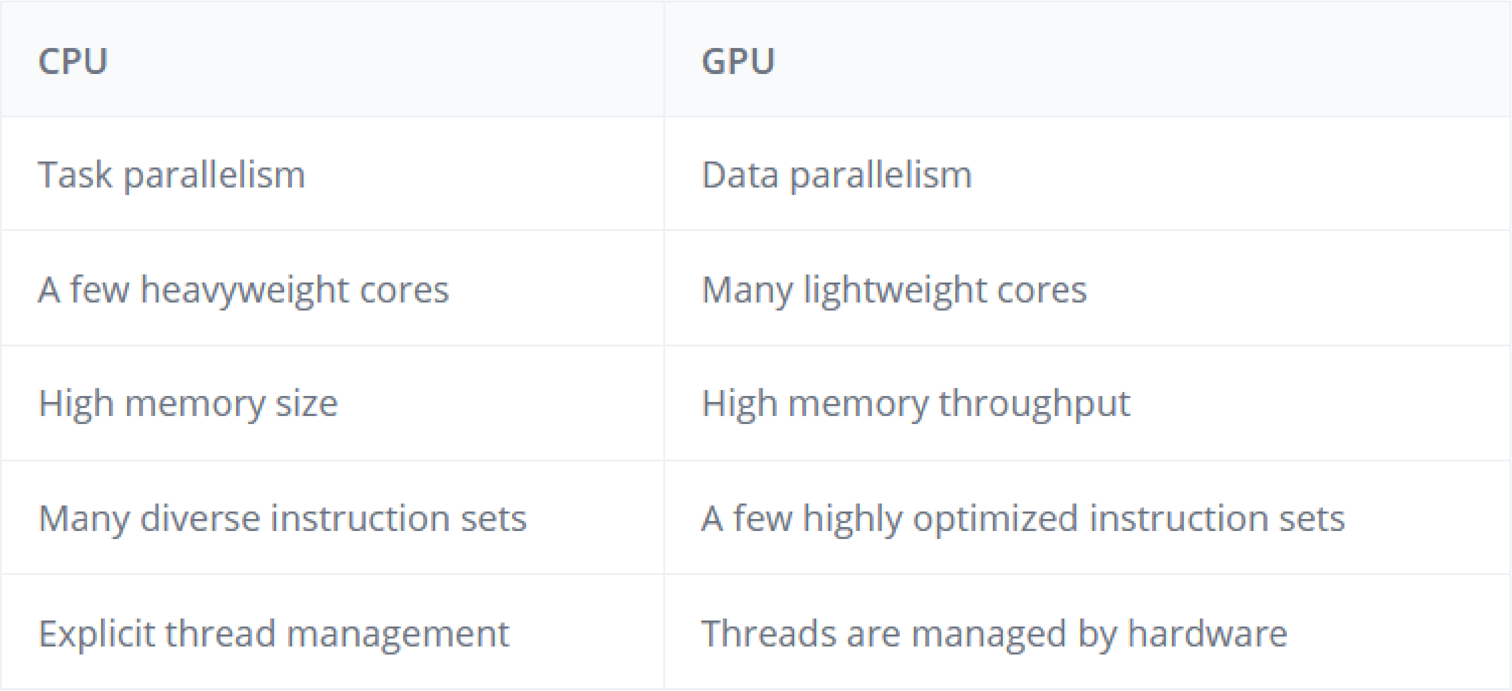


Ubuntu **18.04.6 LTS**	Bionic Beaver	Changes	September 17.2021	April 2023	April 2028

- install recent NVIDIA graphics card drivers on your workstation; we recommend using version >= 440
  - How?
- Isaac SDK requires that your desktop system include a GPU with a compute capability of 6.1 or higher.
- For deployment of your robotics applications, Isaac works best with these developer kit
  - Jetson Nano
  - Jetson Nano 2GB
  - Jetson Xavier
  - Jetson Xavier NX
  - Jetson TX2
- We will use a standard computer simulating the environment

CPU vs GPU

- A Central Processing Unit (CPU) is a latency-optimized general-purpose processor that is designed to handle a wide range of distinct tasks sequentially
  - Latency-optimized: fast response on small requests. Execute as many instructions as possible belonging to a single serial thread, in a given window of time
- Graphics Processing Unit (GPU) is a throughput-optimized specialized processor designed for high-end parallel computing.
  - Throughput-optimized: fast response on big dimension problems



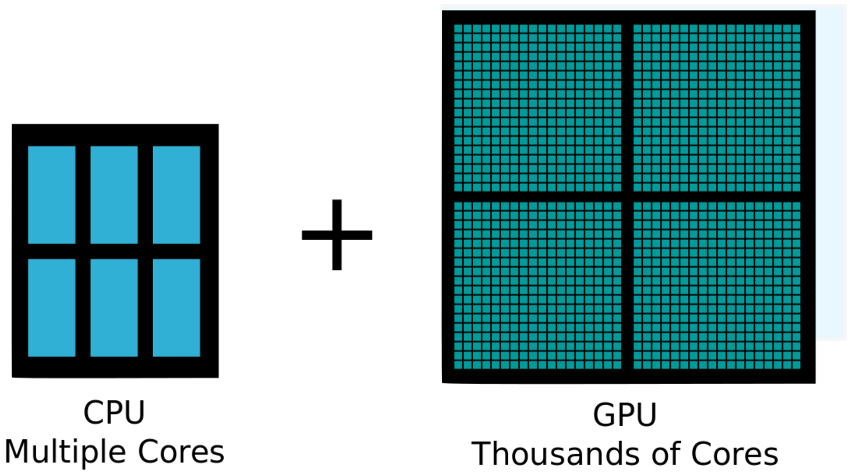
CPU	GPU
Task parallelism	Data parallelism
A few heavyweight cores	Many lightweight cores
High memory size	High memory throughput
Many diverse instruction sets	A few highly optimized instruction sets
Explicit thread management	Threads are managed by hardware

CPU vs GPU

- A Central Processing Unit (CPU) is the brain of your computer
- The main job of the CPU is to carry out a diverse set of instructions through the fetch-decode-execute cycle to manage all parts of your computer and run all kinds of computer programs
- A CPU is very fast at processing your data in sequence, as it has few heavyweight cores with high clock speed
- It's like a Swiss army knife that can handle diverse tasks pretty well
- The CPU is latency-optimized and can switch between a number of tasks really quick, which may create an impression of parallelism
    - Nevertheless, fundamentally it is designed to run one task at a time.

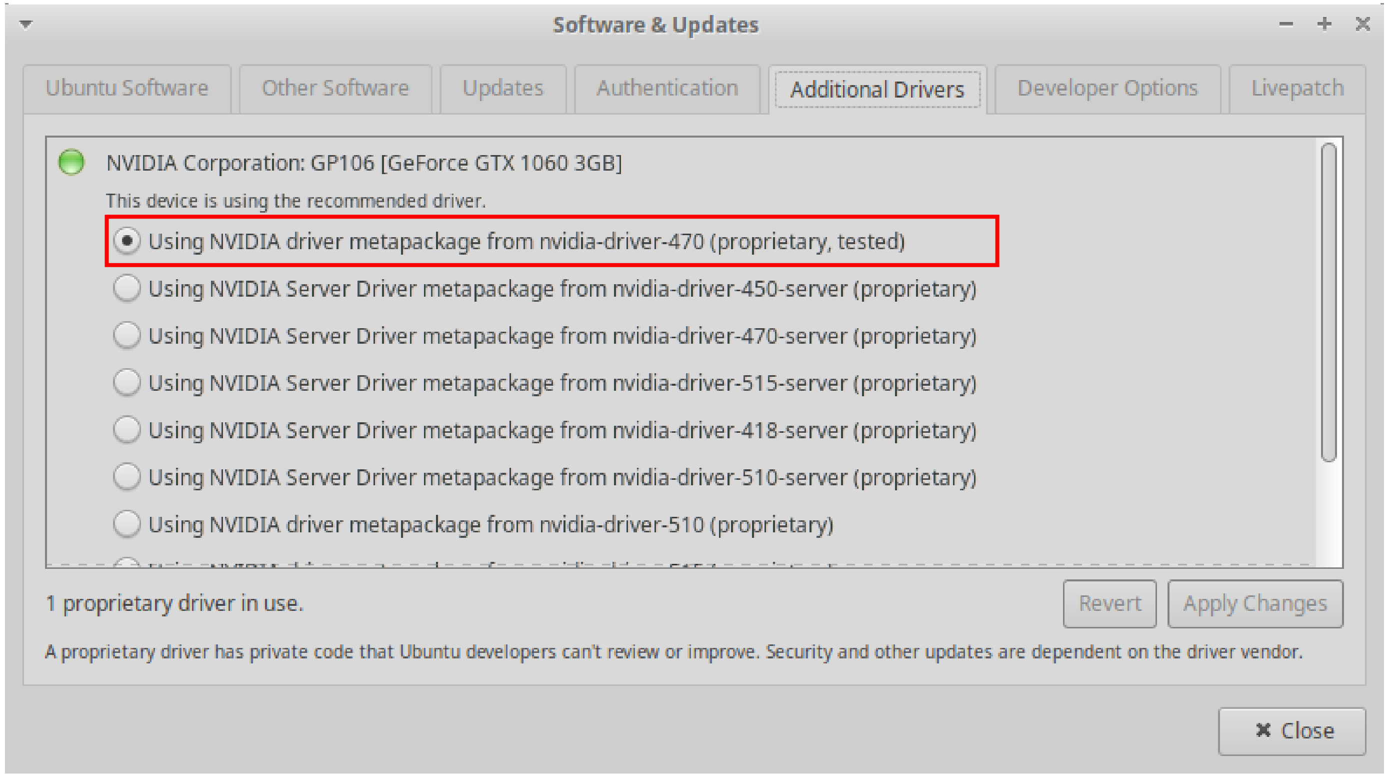
CPU vs GPU

- A Graphics Processing Unit (GPU) is a specialized processor whose job is to rapidly manipulate memory and accelerate the computer for a number of specific tasks that require a high degree of parallelism.
- As the GPU uses thousands of lightweight cores whose instruction sets are optimized for dimensional matrix arithmetic calculations, it is extremely fast with linear algebra and similar tasks that require a high degree of parallelism.
- As a rule of thumb, if your algorithm accepts vectorized data, the job is probably well-suited for GPU computing.
- Architecturally, GPU’s internal memory has a wide interface with a point-to-point connection which accelerates memory throughput and increases the amount of data the GPU can work within a given moment.
- It is designed to rapidly manipulate huge chunks of data all at once.



ISAAC SDK

- Main documentation: https://docs.nvidia.com/isaac/doc/index.html
- install recent NVIDIA graphics card drivers on your workstation; we recommend using version >= 440
  - $ sudo apt-get install software-properties-gtk
    - If you address dependencies problems, use the following command
      - sudo apt-get -f install
  - $ software-properties-gtk --open-tab=4



NVIDIA Corporation: GP106 [GeForce GTX 1060 3GB]

This device is using the recommended driver.

● Using NVIDIA driver metapackage from nvidia-driver-470 (proprietary, tested)

○ Using NVIDIA Server Driver metapackage from nvidia-driver-450-server (proprietary)

○ Using NVIDIA Server Driver metapackage from nvidia-driver-470-server (proprietary)

○ Using NVIDIA Server Driver metapackage from nvidia-driver-515-server (proprietary)

○ Using NVIDIA Server Driver metapackage from nvidia-driver-418-server (proprietary)

○ Using NVIDIA Server Driver metapackage from nvidia-driver-510-server (proprietary)

○ Using NVIDIA driver metapackage from nvidia-driver-510 (proprietary)

1 proprietary driver in use.

A proprietary driver has private code that Ubuntu developers can't review or improve. Security and other updates are dependent on the driver vendor.

ISAAC SDK

- Main documentation: https://docs.nvidia.com/isaac/doc/index.html
- install recent NVIDIA graphics card drivers on your workstation; we recommend using version >= 440
- To install NVIDIA ISAAC SDK, go to the following link: https://developer.nvidia.com/isaac/downloads
  - A free-member account is needed
  - A EULA license must be accepted
- Craete destination directory
  - $ mkdir isaac-sdk
  - $ cd isaac-sdk
- Extract the archive
  - $ tar -xf ../isaac-sdk-20210609-e336b5195.tar.xz
- Subscribe and download the Isaac SDK archive file
- Time 5-10 minutes
- Install ISAAC:
  - $ cd issac-sdk/engine
  - $ sudo apt-get install libvpx-dev
  - $ ./engine/build/scripts/install_dependencies.sh

ISAAC SDK

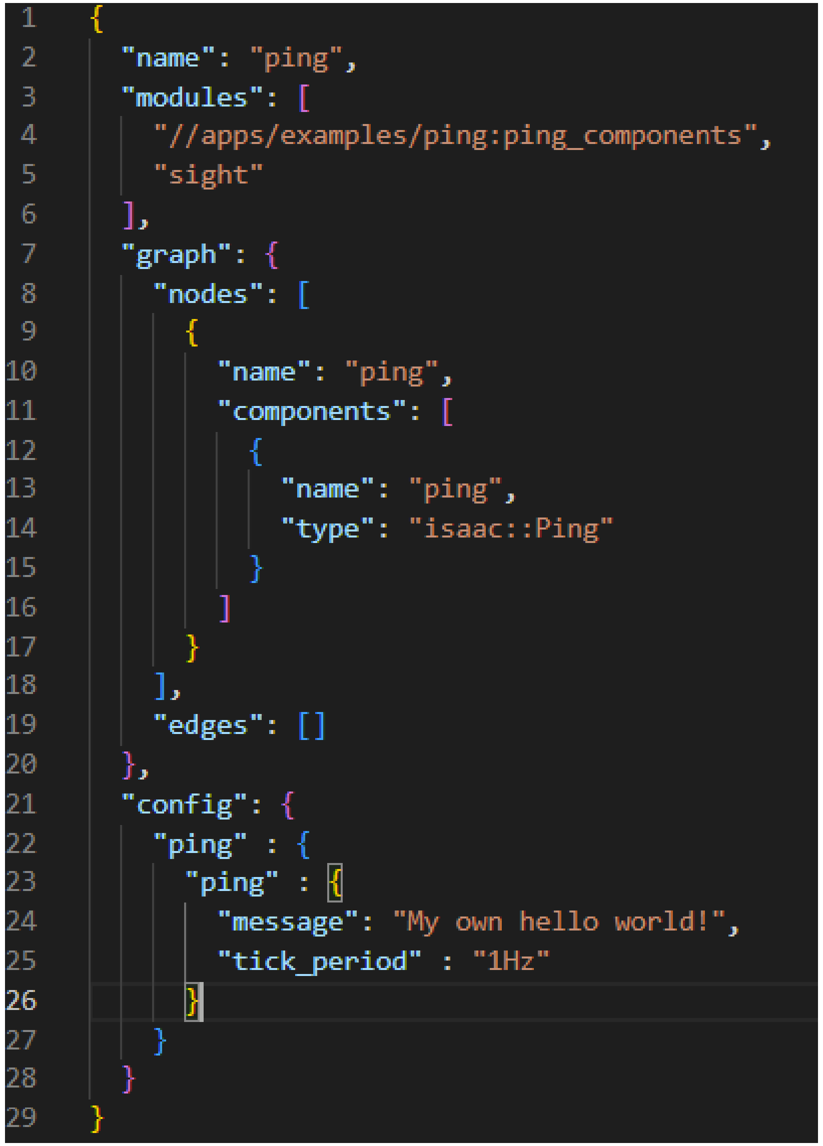
- Developing timeline
    - First release 2019
        - Main components: SDK, ISAAC ENGINE
        - ISAAC SIM: Simulation
    - Current release 2021
        - New releases of the SDK are not planned
    - New release of ISAAC SIM: 2022.1
        - Not supports anymore the ISAAC SDK

Example 1.5

- Start with an example: stereo_dummy
- How to run?
  - A bazel target name for example has the following form: app/samples/stereo_dummy
  - This refers to the application *stereo_dummy* in the folder app/samples/stereo_dummy
  - If you want to run a different application, you have to change the target name correspondingly
- Note that all bazel build and bazel run commands should be executed at the root folder of your repository
- For example, if your root folder is /home/bob/isaac you first go to the directory /home/bob/isaac and then run the commands mentioned below.
- Open the Isaac sdk folder
- build
  - $ bazel build apps/samples/stereo_dummy
- run
  - $ bazel run apps/samples/stereo_dummy
- Go to the webpage: localhost:3000
- What is behind?
- Examples: https://github.com/robotic-software/examples.git
  - Must be put in sdk/apps/

Example 2.5

- ISAAC ping example
  - File needed:
    - ~~BUILD~~
    - JSON configuration file
    - ~~Source code~~
- An Isaac application is defined by a JavaScript Object Notation (JSON) file
- To define a new ISAAC application, we need four sections (we will come back on this point) in the JSON file



```json
{
  "name": "ping",
  "modules": [
    "//apps/examples/ping:ping_components",
    "sight"
  ],
  "graph": {
    "nodes": [
      {
        "name": "ping",
        "components": [
          {
            "name": "ping",
            "type": "isaac::Ping"
          }
        ]
      }
    ],
    "edges": []
  },
  "config": {
    "ping" : {
      "ping" : {
        "message": "My own hello world!",
        "tick_period" : "1Hz"
      }
    }
  }
}
```

Example 2.5

- Isacc ping example
- An Isaac application is defined by a JavaScript Object Notation (JSON) file
- To define a new ISAAC application, we need four sections:
  - Name is a string with the name of the application
  - Modules are a list of libraries in use
    - We include ping:ping_components so that “apps/examples/ping/Ping.cpp” is available
    - The Isaac SDK comes with high-quality and well-tested packages that we can import as modules
    - We can write our modules
  - The graph has two subsections to define the functionality of the application:
    - nodes are the fundamental blocks of our application
      - In this simple example, we have just one node named “ping” that has a single component
      - Note that the type of this component, isaac::Ping, matches the last line of Ping.hpp
      - A typical Isaac application has multiple nodes, and each node typically has multiple components
    - edges connect components together and enable the communication between them
      - This example does not require any components to be connected
  - config lets you tune the parameters of each node depending on our use case.
    - In this example, it is specified that the ping component should tick at one hertz.

Example 2.5

- Other application files
  - Source file: hpp, cpp
  - BUILD

Example 2.5

- Create a new directory in the apps folder
  - $ cd sdk/apps
  - $ mkdir examples
  - $ cd examples
  - $ mkdir ping
- Create a BUILD file
- Create a ping.app.json file
- Create a Ping.cpp source file
- Create a Ping.hpp header file

Example 2.5

- Header and source:
  - Represent the core of the application
  - Some applications don’t have source code, because they exploit other modules already available on the ISAAC stack

Example 2.5

- JSON
  - Defines the application graph:
    - The connection among all the modules and their configuration

Example 2.5

- BUILD:
  - Contains the definition of the modules of the application
    - Source files
  - Contains the definition of the ISAAC application
    - The one that we want to run

Example 2.5

- After created all the necessary application file you can compile it with bazel build
  - $ bazel build ping
    - The name of the application is defined from the json file name
- To run the application
  - $ bazel run ping

Fine lezione 5