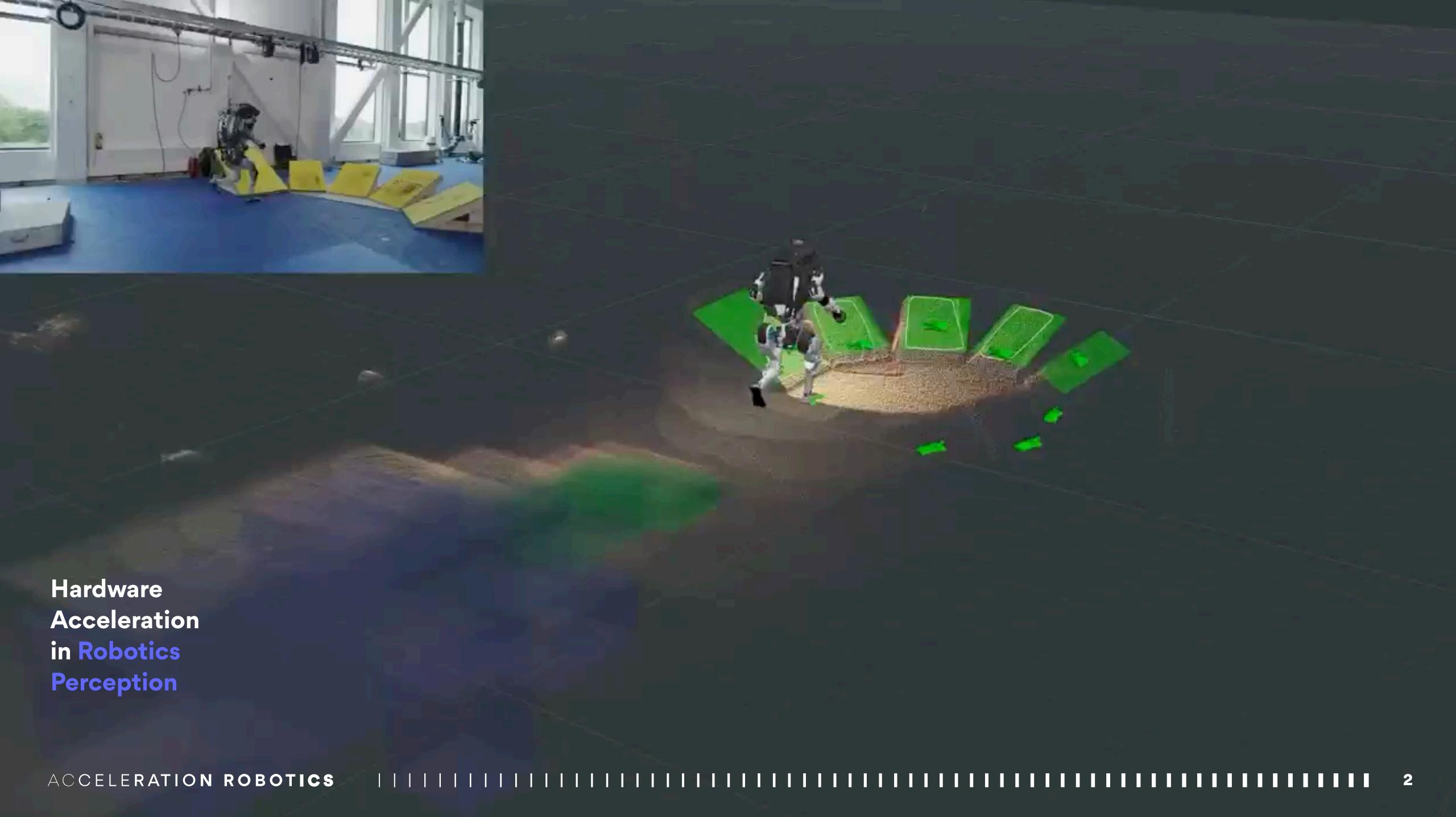
An open architecture for Hardware Acceleration in ROS 2

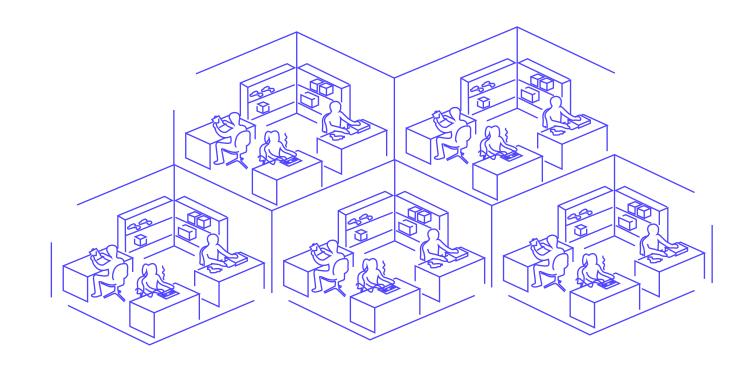


Easily leverage hardware acceleration in a ROS-centric manner and <u>build custom</u> <u>compute architectures for robots</u>, or <u>robot cores</u>.



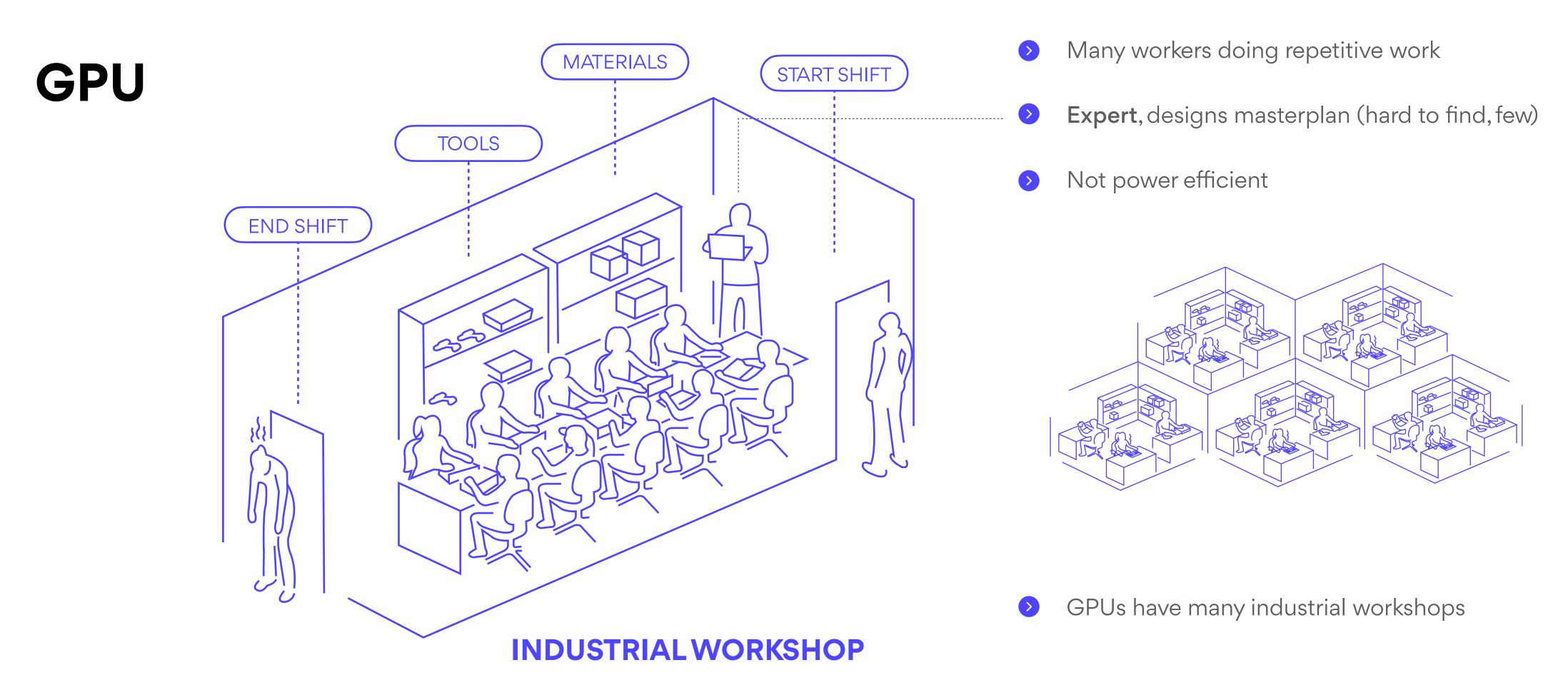
Understanding robotics computations in CPU TOOLS MATERIALS INSPECTION **CPU ASSEMBLY WORKSHOP** SOLDERING

- Refined workshops that can build pretty much anything, each workshop is a core
- Work sequentially, task after task until finalizing
- Multiple workers or, "threads", on each CPU (though it eventually fills the space)

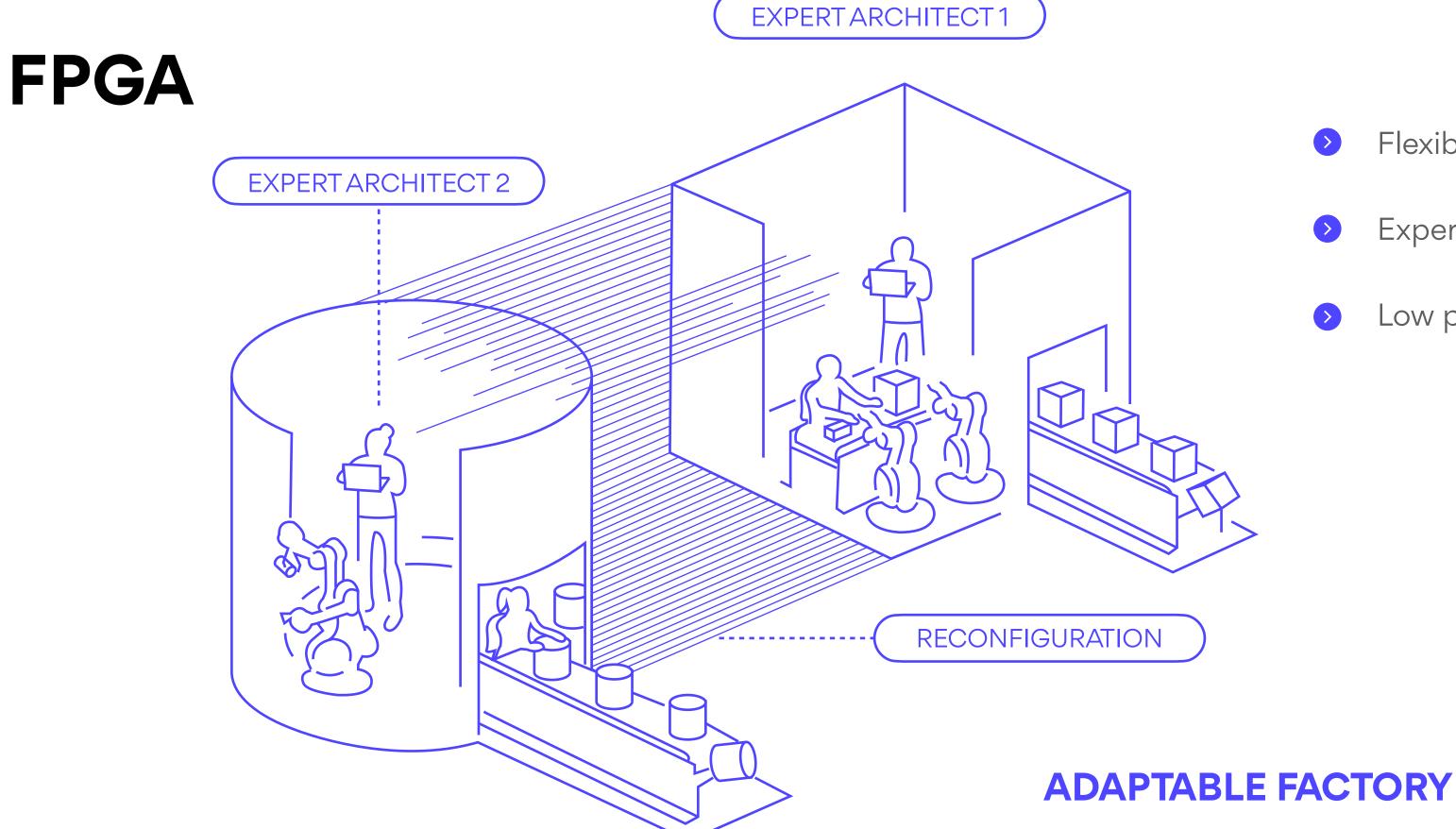


Modern CPUs have various cores

Understanding robotics computations in GPUs



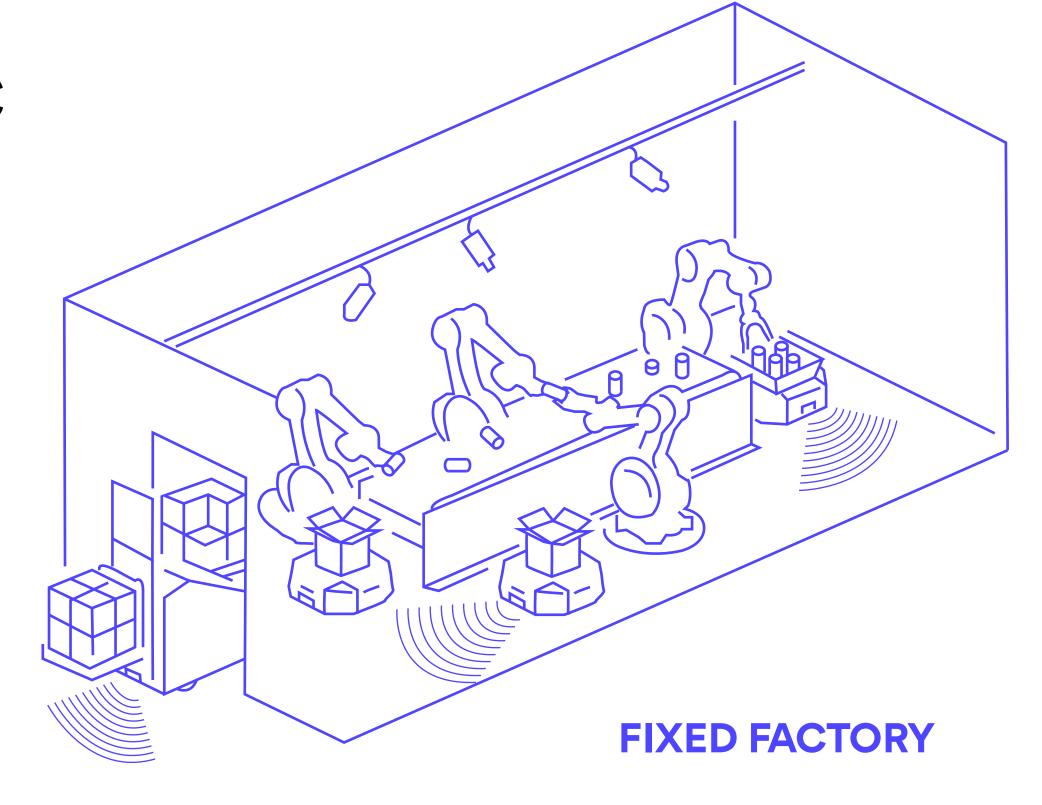
Understanding robotics computations in FPGAs



- Flexible and adaptable factory (software defined)
- Expert-driven architectures
- Low power, high performance

Understanding robotics computations in ASICs

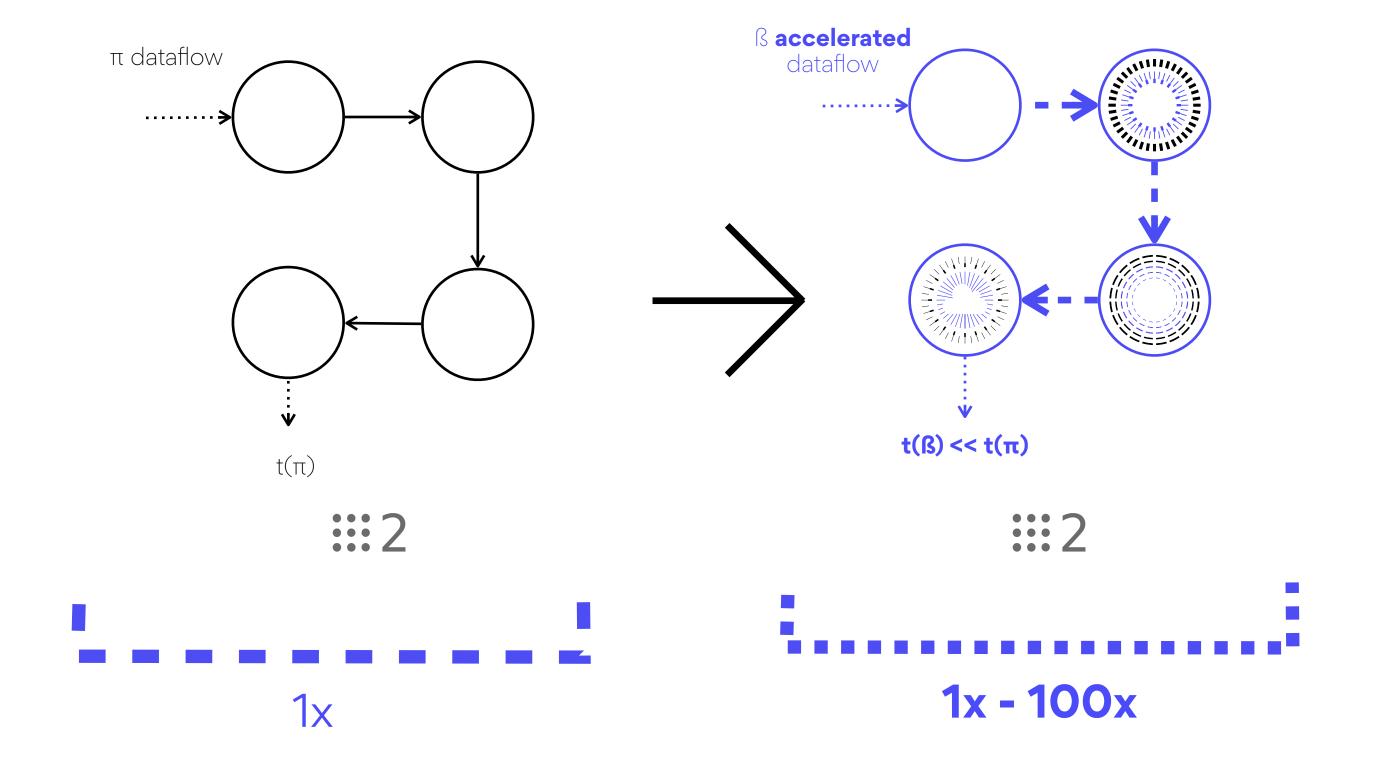
ASIC

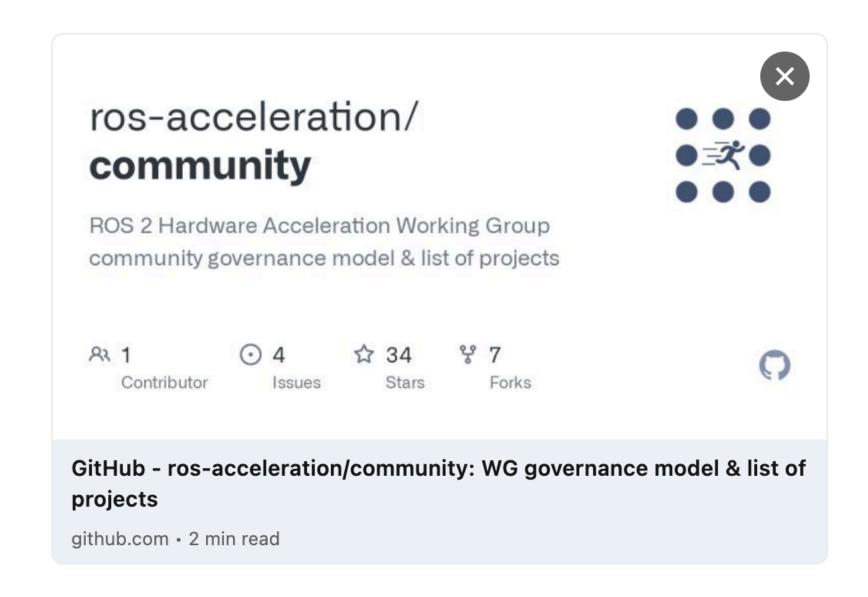


- Fixed, fully automated factory
- Specialized in one process
- Very expensive to design and build
- Best throughput at lowest power consumption

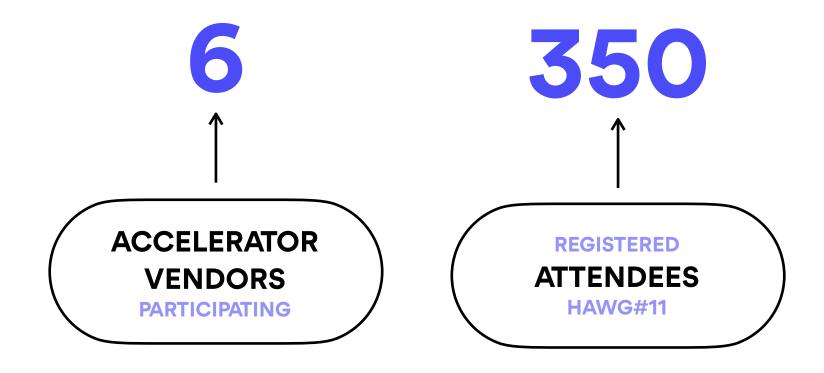
ROS 2 Hardware Acceleration Working Group (HAWG)

The **ROS 2 Hardware Acceleration Working Group** drives the creation, maintenance and testing of acceleration kernels on top of open standards for optimized ROS 2 and Gazebo interactions over different compute substrates (including FPGAs, GPUs and other accelerators).





https://github.com/ros-acceleration/community

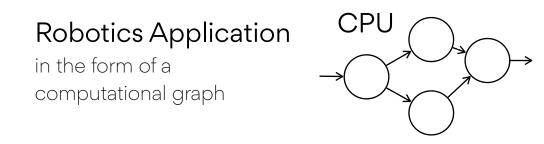


ROBOTCORE Framework:

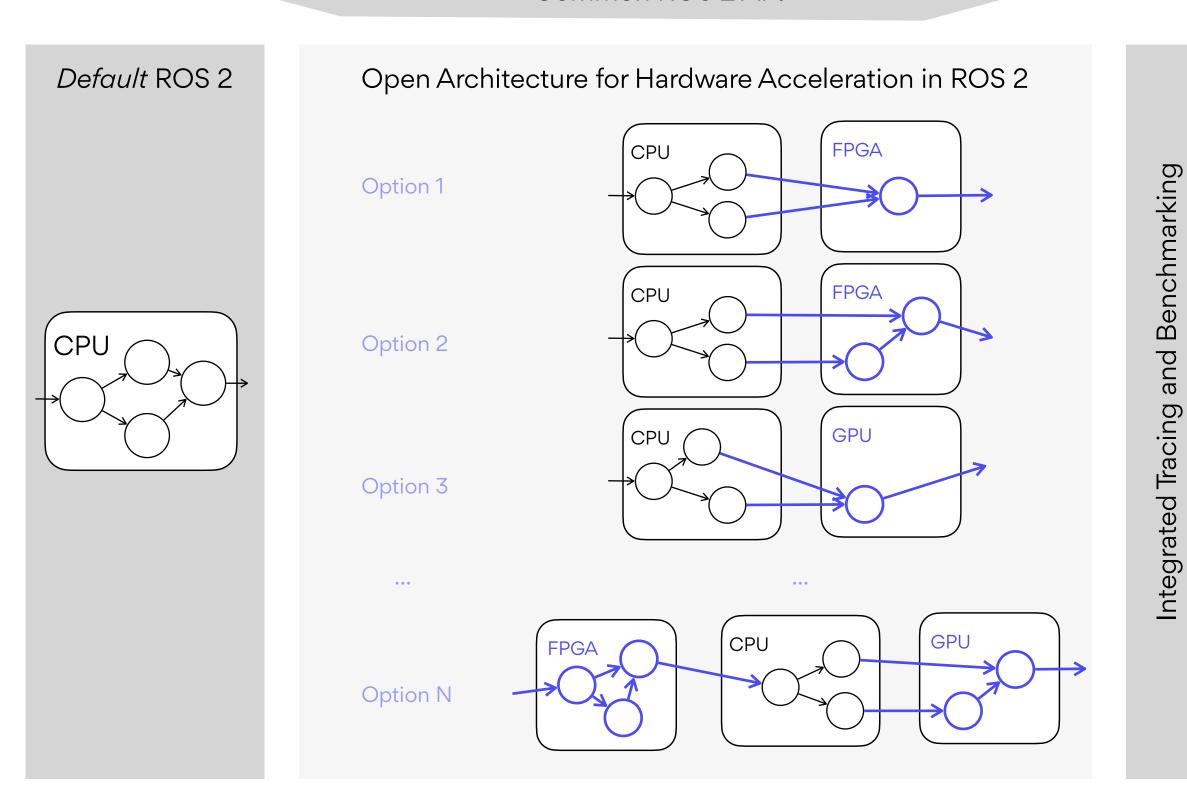
An open architecture for Hardware Acceleration in ROS 2

Hardware Acceleration Framework for ROS

It helps <u>build custom compute architectures</u> <u>for robots</u> through acceleration kernels, or <u>robot cores</u>, that make robots faster, more deterministic and power-efficient. <u>Simply put, it provides a development, build and deployment experience for creating robot hardware and hardware acceleration kernels similar to the standard, non-accelerated ROS development flow.</u>



Common ROS 2 API

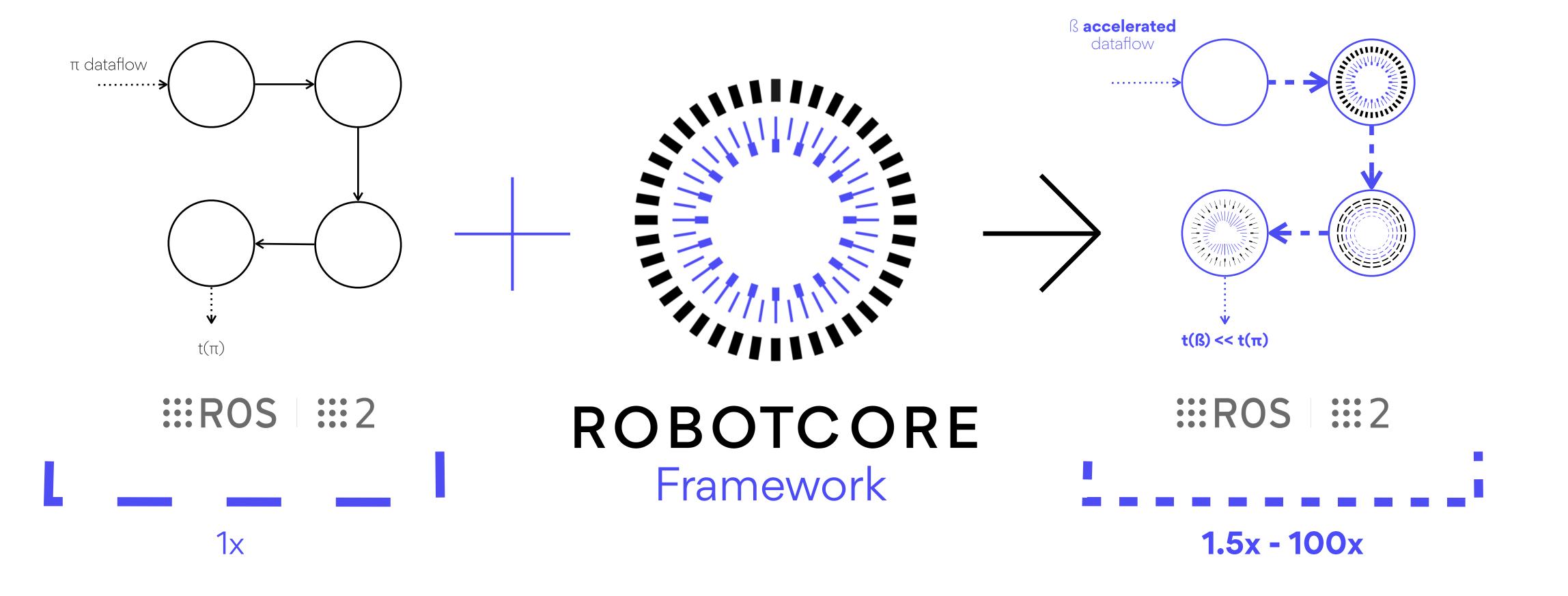


https://arxiv.org/pdf/2205.03929.pdf

ROBOTCORE Framework

Hardware Acceleration Framework for ROS. It helps <u>build custom compute</u> <u>architectures for robots</u>, or <u>robot cores</u>, that make robots faster, more deterministic and power-efficient. Simply put, it provides a development, build and deployment experience for creating robot hardware and hardware accelerators similar to the standard, non-accelerated ROS development flow.

Public community-driven and open source implementation available at https://github.com/ros-acceleration



ROBOTCORE Framework:

Production-grade multi-platform ROS support with Yocto (REP 2000 →)

Instead of relying on common development-oriented Linux distros (such as Ubuntu), our contributions to Yocto allow to build a customized Linux system for your use case with ROS, providing unmatched granularity, performance and security.

















ZCU102





Jetson Nano



Jetson Xavier NX



Jetson AGX Xavier



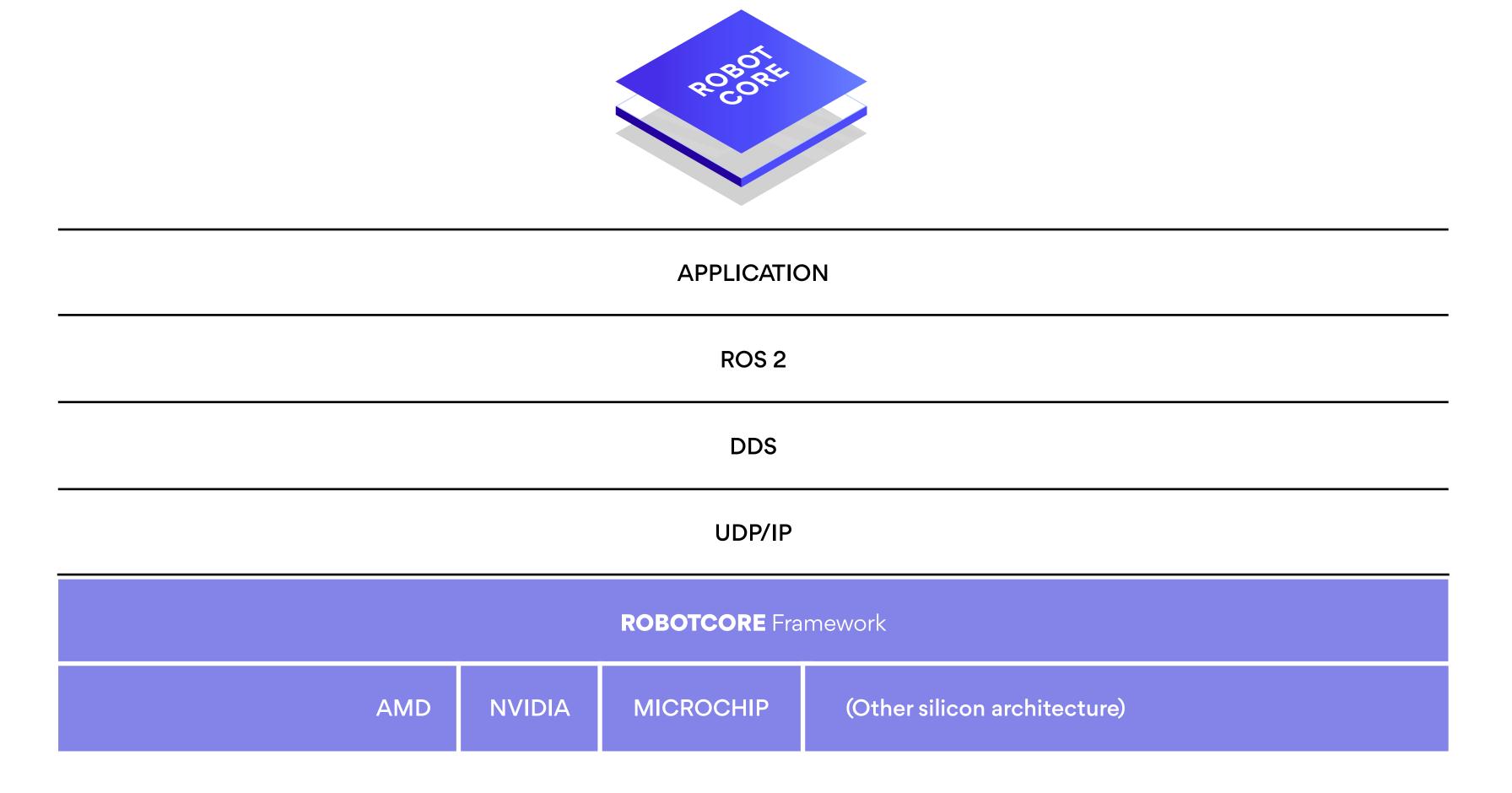
PolarFire Icicle

From ROBOTCORE Framework to an open standard

ROS 2 Hardware Acceleration Architecture and Conventions (REP 2008 →)

We are bringing the lessons learned while developing ROBOTCORE Framework into a REP that describes the architectural pillars and conventions required to introduce hardware acceleration in ROS 2 in a vendor-neutral, scalable and technology-agnostic manner.

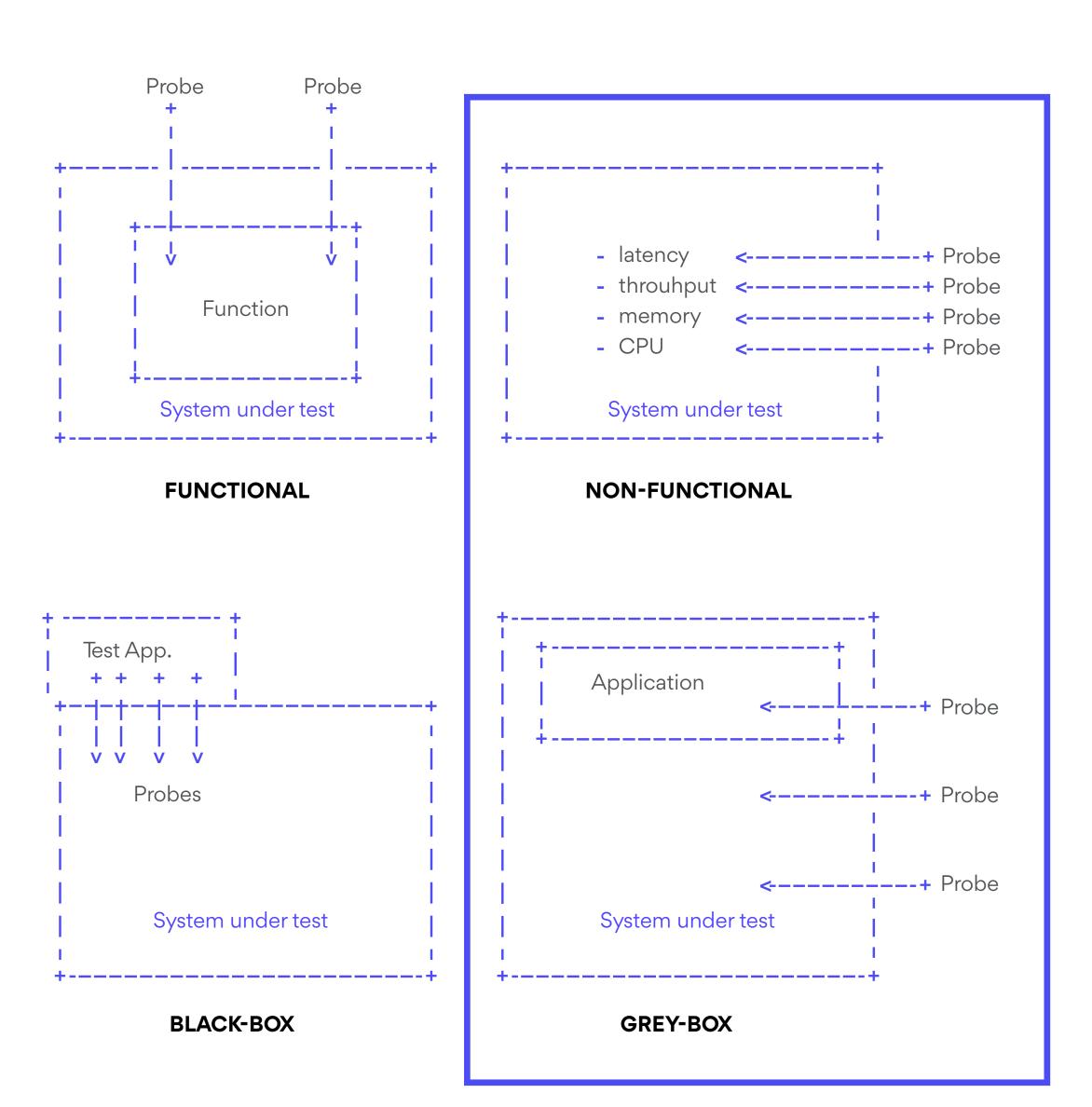
All while maintaining the common ROS development flow.



From ROBOTCORE Framework to an open standard Benchmarking performance in ROS 2 (REP 2014 →)

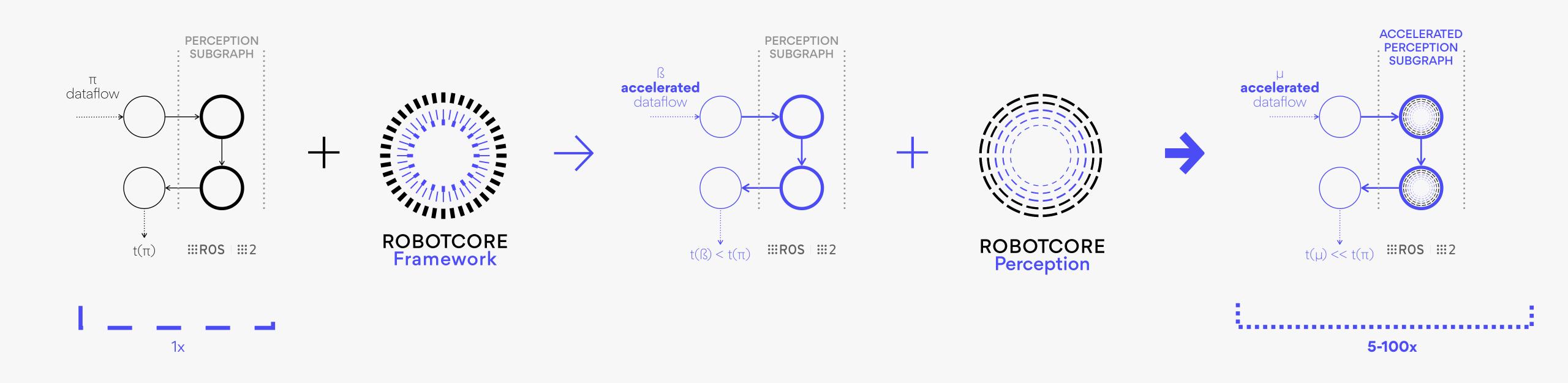
Benchmarking is the act of running a computer program with a known workload to assess the program's relative performance.

We adopt a grey-box and nonfunctional benchmarking approach
for hardware acceleration with a lowoverhead tracing and benchmarking
framework, and select the Linux
Tracing Toolkit next generation (LTTng)
to implement it.



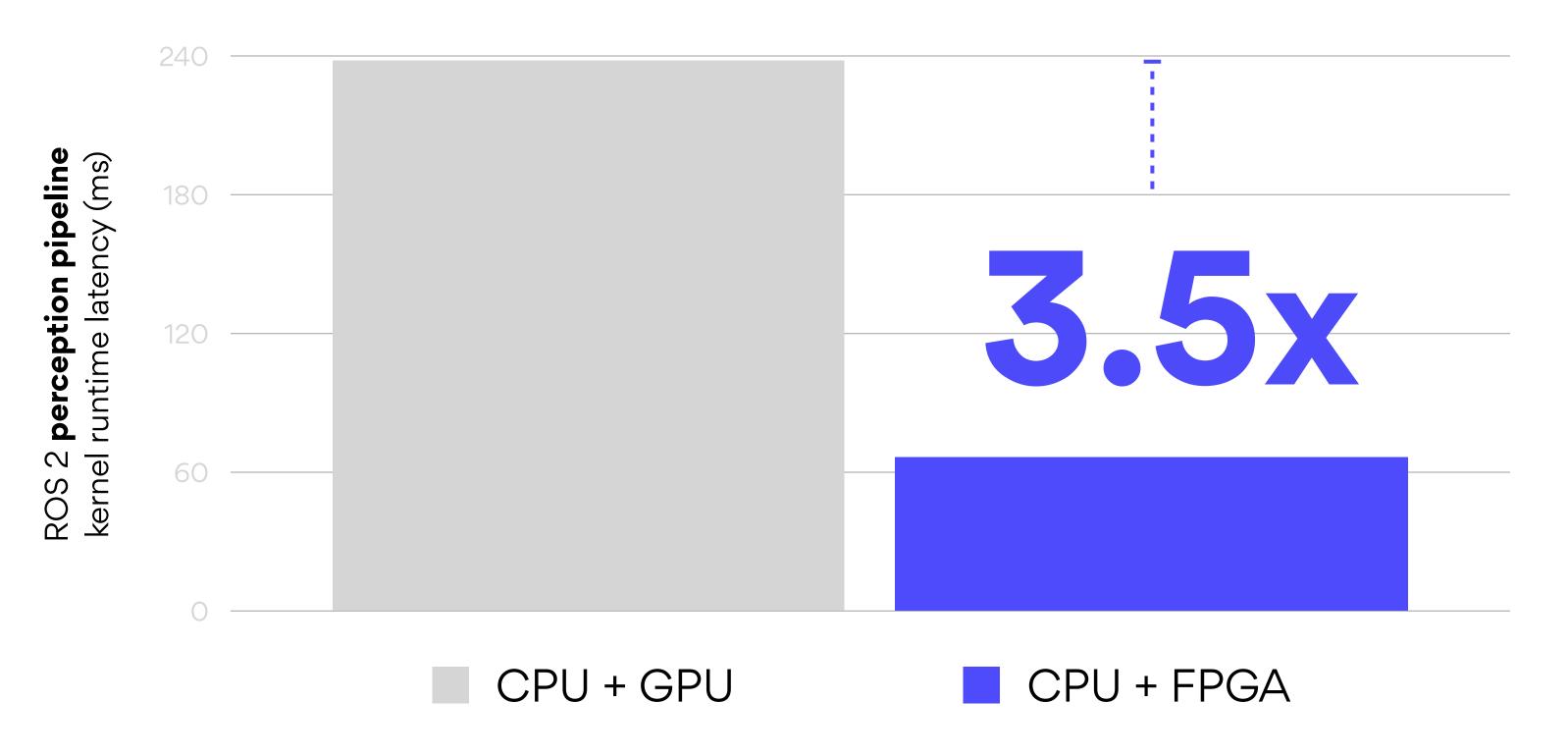
Case Study: ROS 2 Perception graph

ROBOTCORE Perception is an **optimized robotic perception** stack built with ROBOTCORE Framework that leverages hardware acceleration to provide a speedup in your perception computations. <u>API-compatible with the ROS 2</u> perception stack, ROBOTCORE Perception delivers high performance, real-time and reliability to your robots' perception.

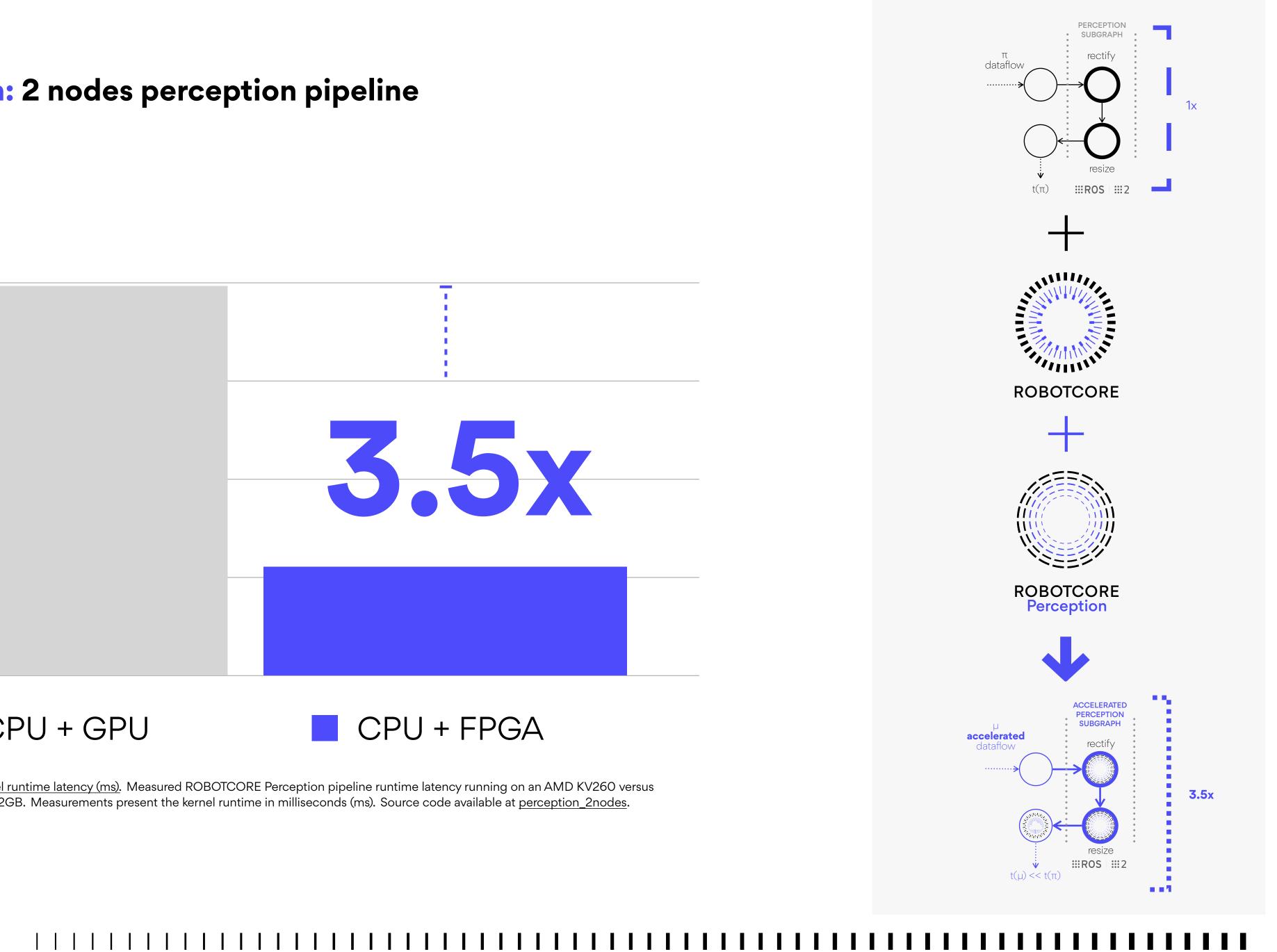


ACCELERATION ROBOTICS

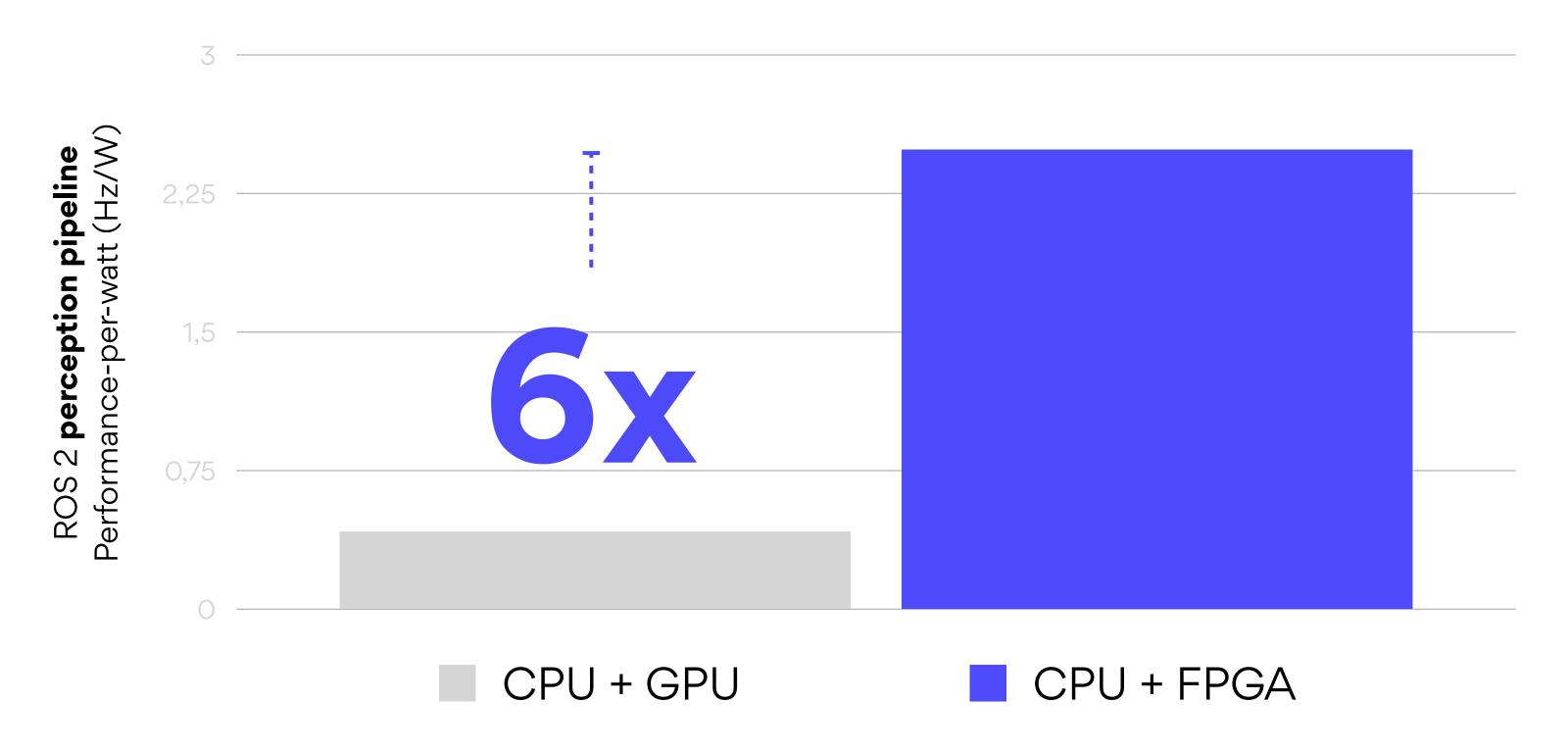
ROS 2 Perception graph: 2 nodes perception pipeline



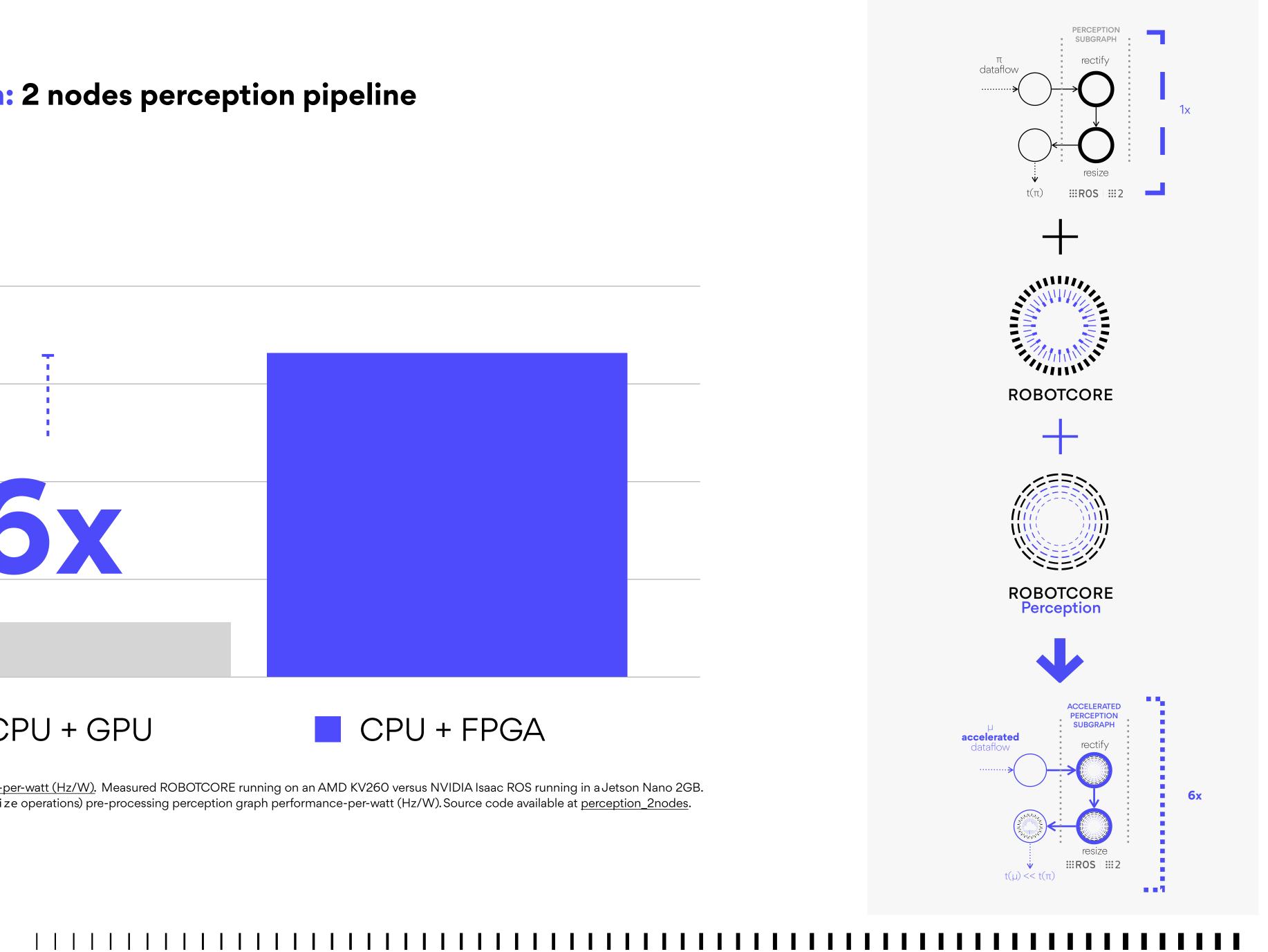
ROS 2 perception pipeline (2 nodes) - kernel runtime latency (ms). Measured ROBOTCORE Perception pipeline runtime latency running on an AMD KV260 versus NVIDIA Isaac ROS running in a Jetson Nano 2GB. Measurements present the kernel runtime in milliseconds (ms). Source code available at perception_2nodes.



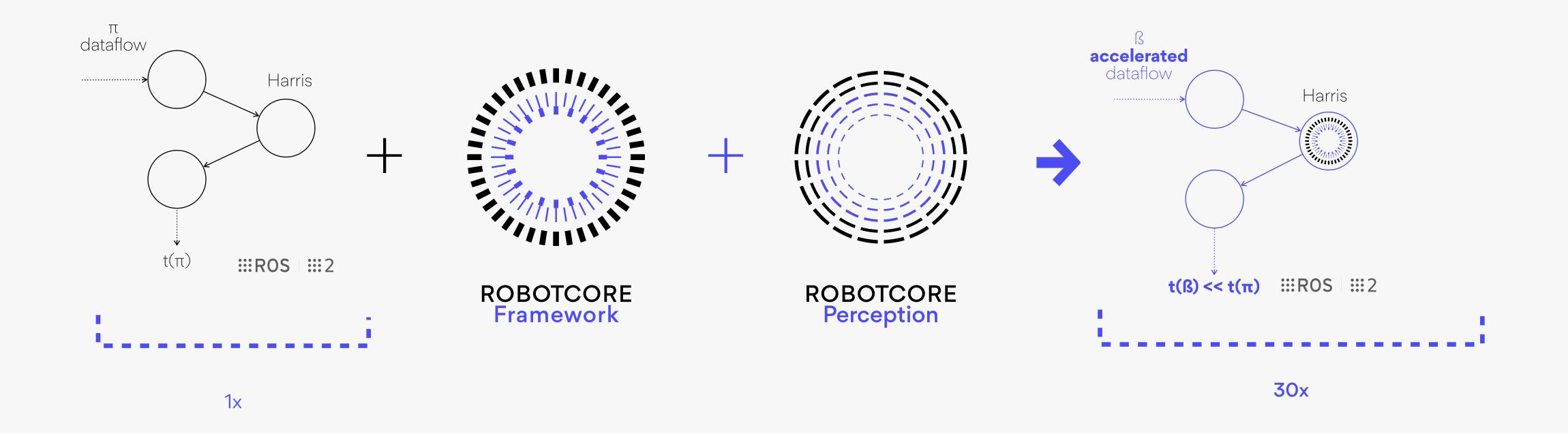
ROS 2 Perception graph: 2 nodes perception pipeline



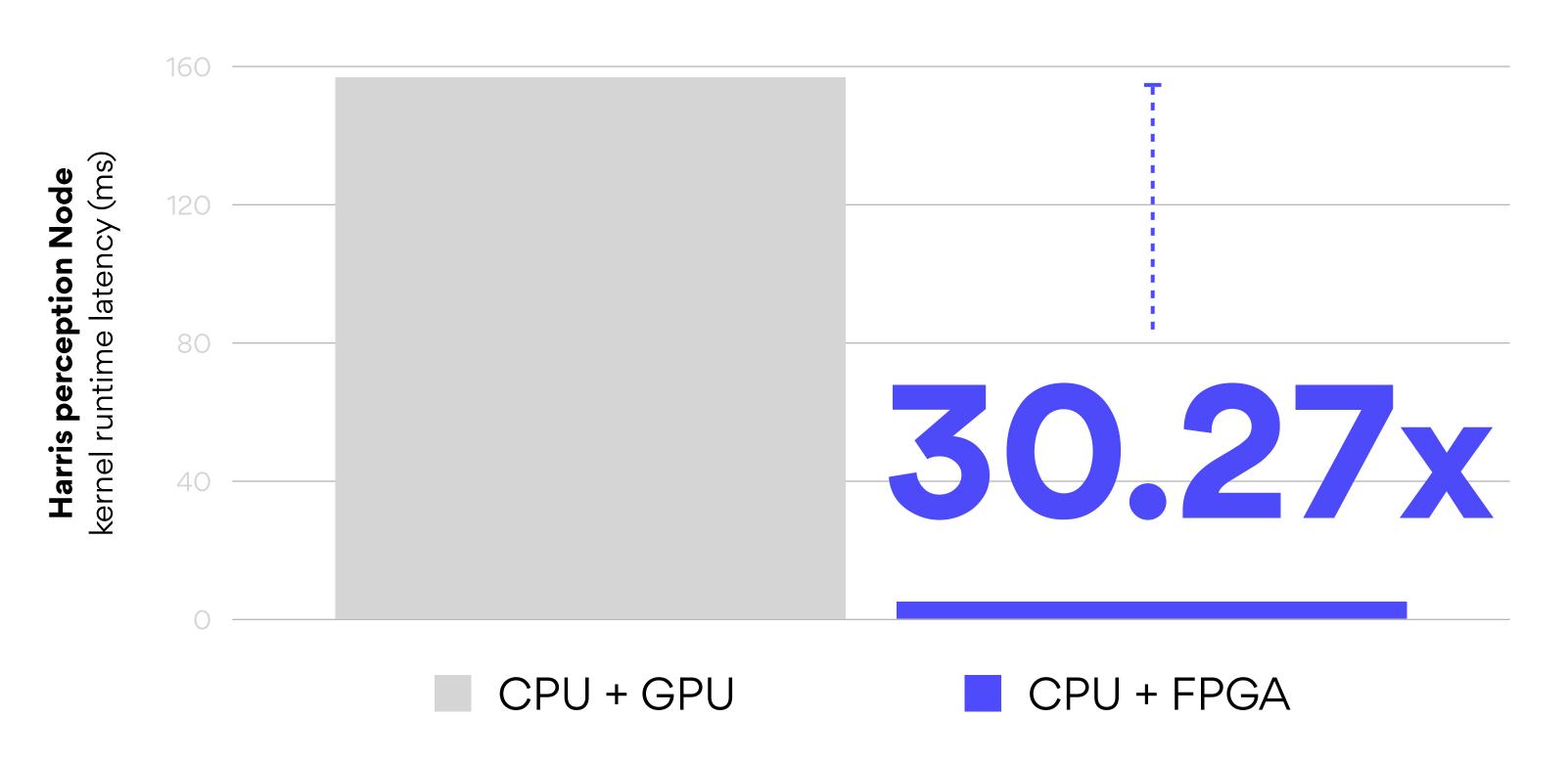
ROS 2 perception pipeline (2 nodes) - Performance-per-watt (Hz/W). Measured ROBOTCORE running on an AMD KV260 versus NVIDIA Isaac ROS running in a Jetson Nano 2GB. Measurements present a 2-Node (rectify and resize operations) pre-processing perception graph performance-per-watt (Hz/W). Source code available at perception_2nodes.



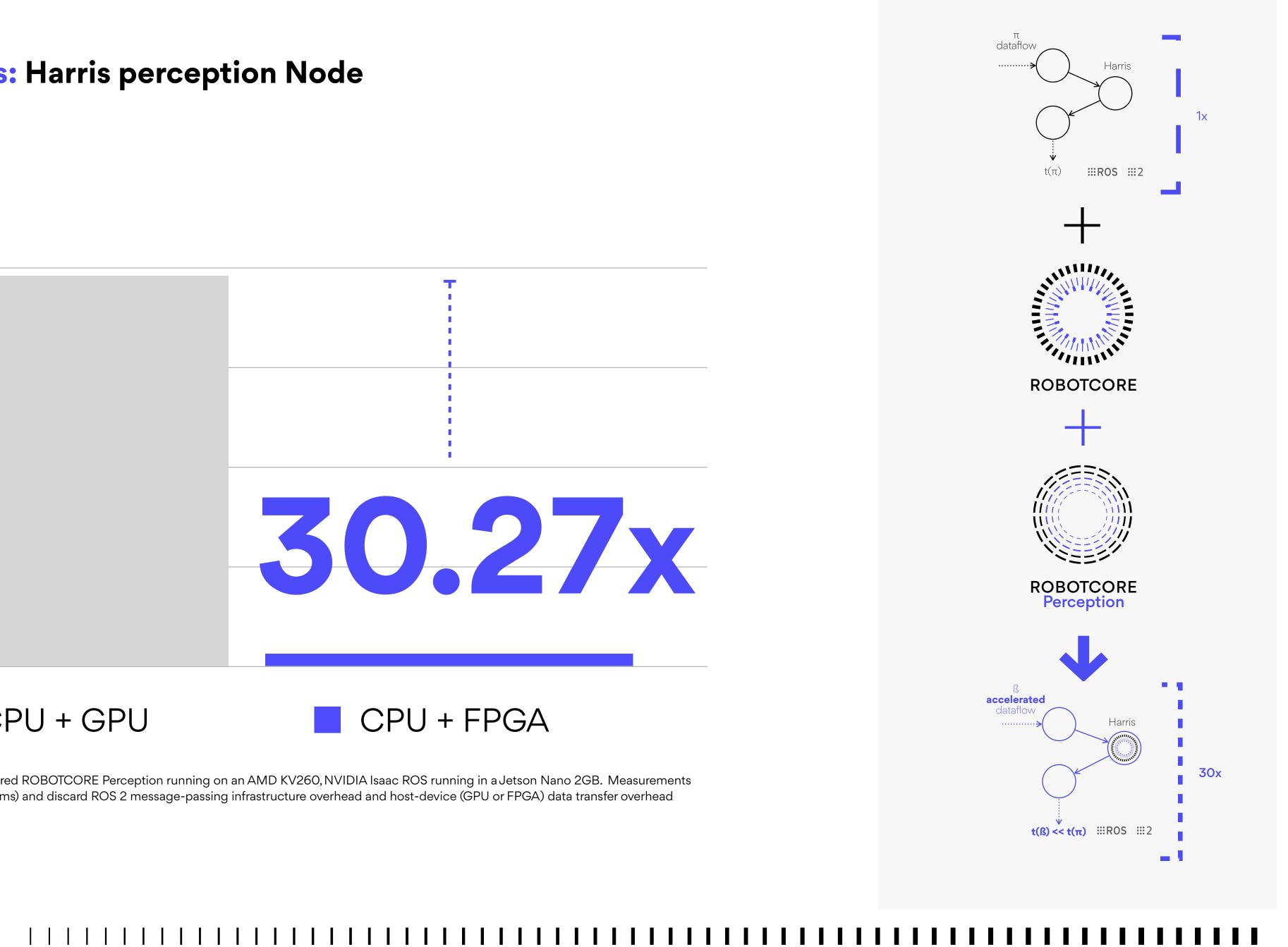
Case Study: ROS 2 Perception Nodes



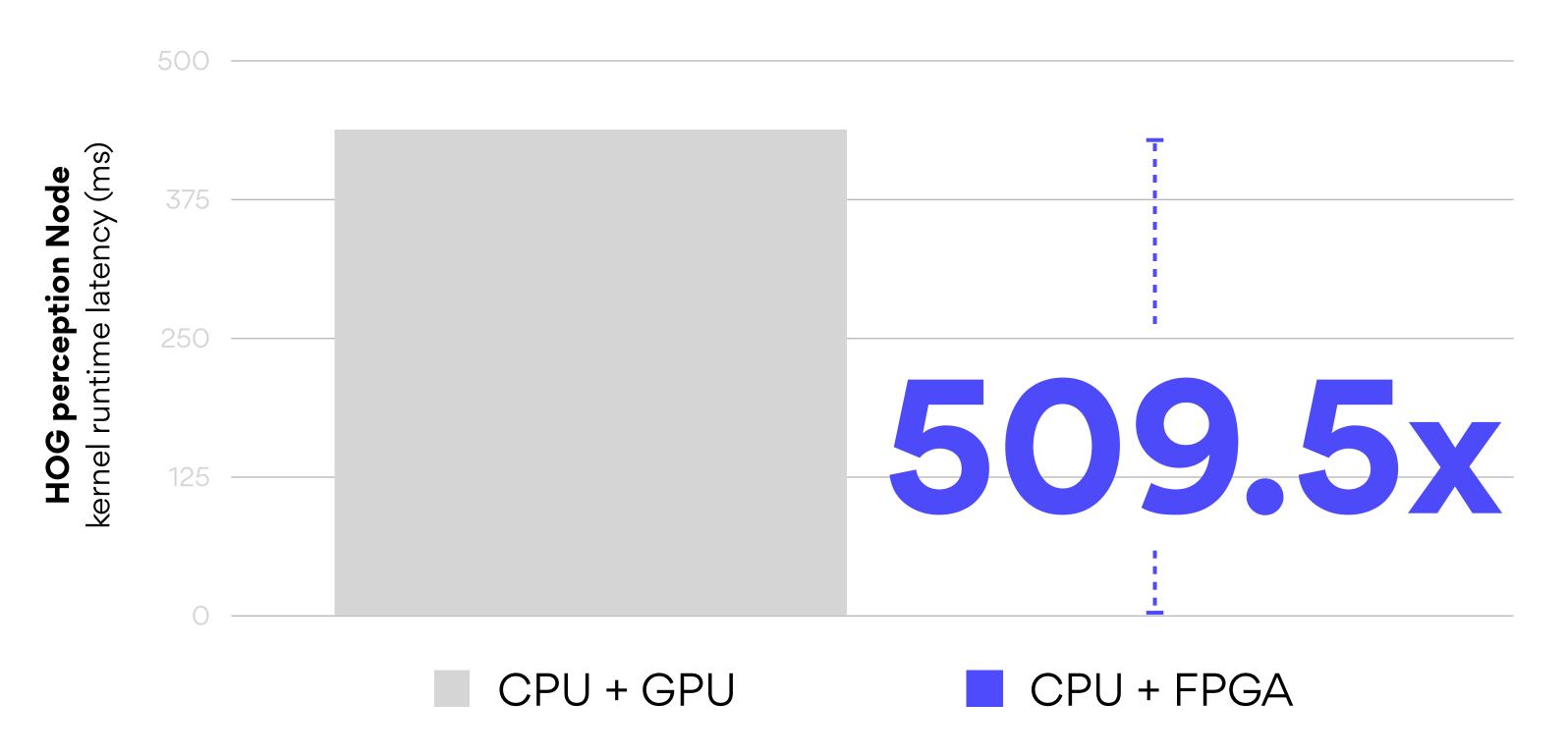
ROS 2 Perception Nodes: Harris perception Node



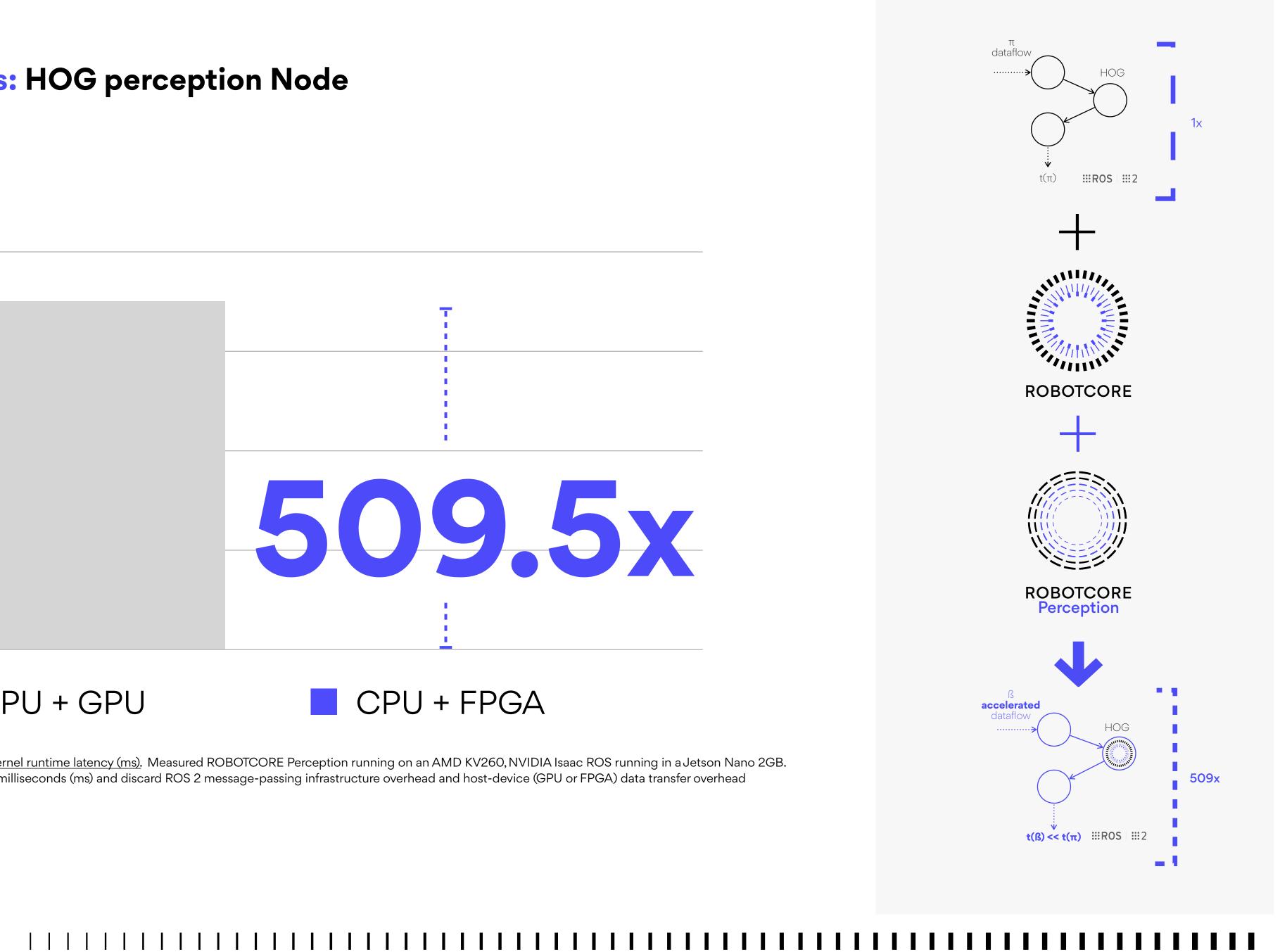
Harris - kernel runtime latency (ms). Measured ROBOTCORE Perception running on an AMD KV260, NVIDIA Isaac ROS running in a Jetson Nano 2GB. Measurements present the kernel runtime in milliseconds (ms) and discard ROS 2 message-passing infrastructure overhead and host-device (GPU or FPGA) data transfer overhead



ROS 2 Perception Nodes: HOG perception Node



Histogram of Oriented Gradients (HOG) - kernel runtime latency (ms). Measured ROBOTCORE Perception running on an AMD KV260, NVIDIA Isaac ROS running in a Jetson Nano 2GB. Measurements present the kernel runtime in milliseconds (ms) and discard ROS 2 message-passing infrastructure overhead and host-device (GPU or FPGA) data transfer overhead



COELERATION ROBOTICS Jarokation Agreetion Agreement to the post of the post in the post

2022

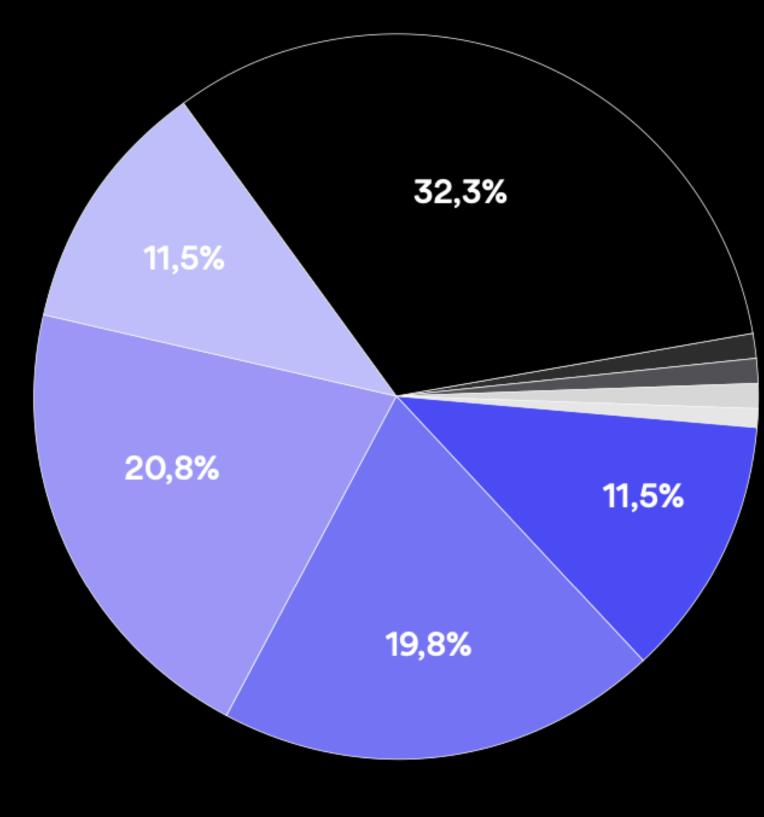
Hardware Acceleration Report in Robotics

Captures the state-of-the art of hardware acceleration in robotics by following a quantitative approach and presents robotic architects with a resource to consider while designing their robot computational architectures.

- First phase, a community survey conducted in both the ROS and the overall robotics communities helped grasp the interest behind the use of hardware acceleration in robotics. Input from this community survey was then used to drive the
- Second phase, a hardware acceleration benchmarking effort

2022 Hardware Acceleration Report in Robotics

First phase: We're pushing forward REP-2008 initiative to better integrate hardware acceleration with ROS and Gazebo, what's most important for you? (report



96 answers

Integration with ROS 2 build system (ament)

Integration with ROS 2 build tools (colcon)

Acceleration firmware integrated into ROS 2 workspaces (cross-compilers, hypervisors, etc.)?

Benchmarking capabilities for hardware acceleration

Capabilities to
easily switch
between hardware
accelerated and
CPU-centric Nodes

They are all equally important to make offloading transparent for the user

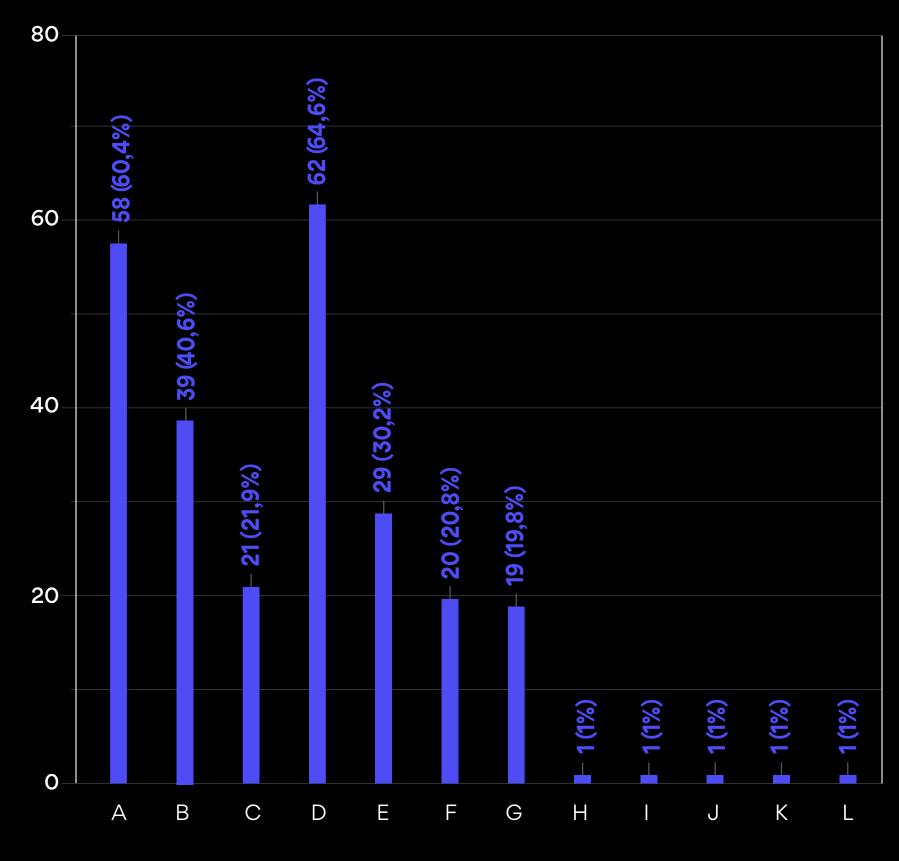
None of the above

Adding acceleration to important packages that just work without thinking about it

Complete and accurate documentation is the priority to me.
Likely, large part of the community is unfamiliar with hardware acceleration

2022 Hardware Acceleration Report in Robotics

First phase: What packages/components do you think weshould prioritize when it comes to hardware accelerating ROS 2 and/or Gazebo? (report=)



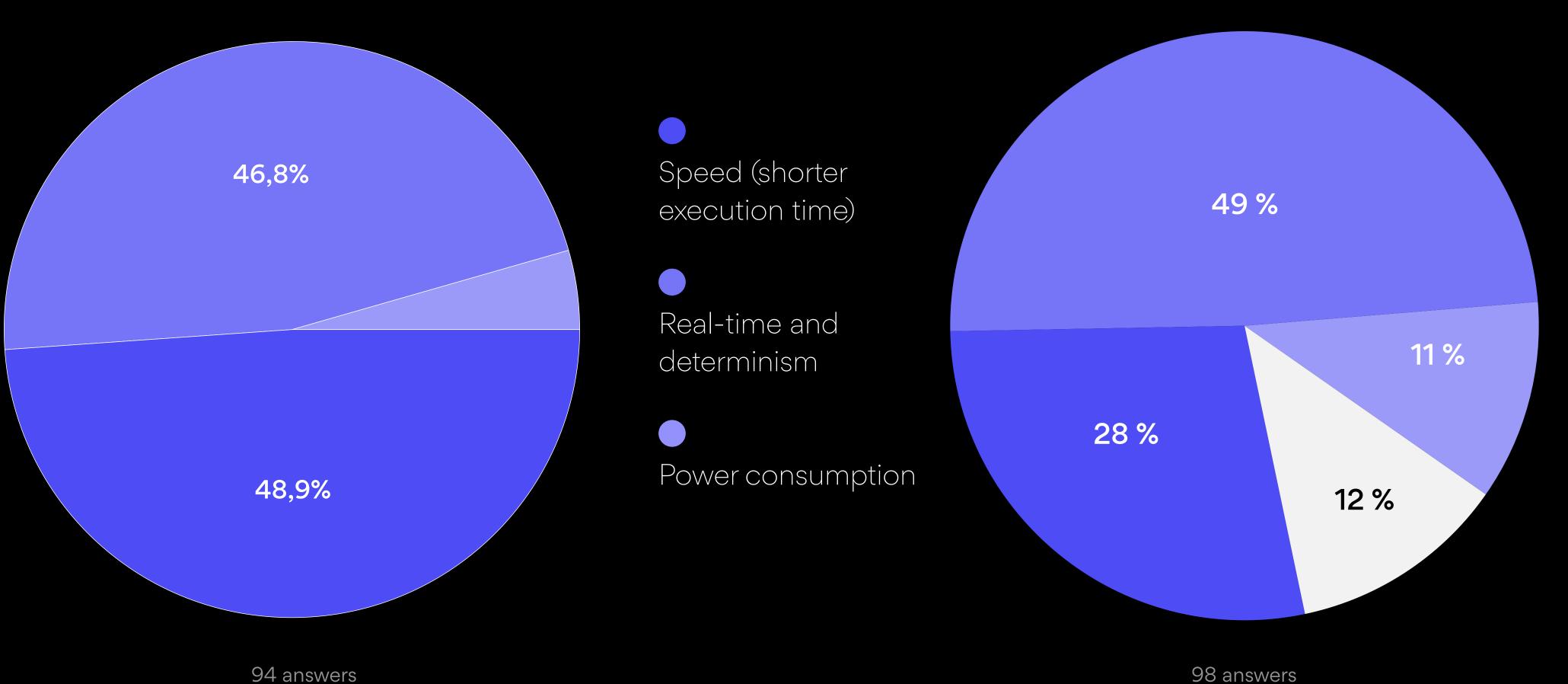
96 answers (multiple answers allowed)

- A. Gazebo/Ignition physic engines
- B. ROS 2 navigation stack (navigation2)
- **C.** ROS 2 manipulation stack (Movelt2)
- D. ROS 2 perception stack
- E. ROS 2 communication middleware (DDS, i.e. offloading it to hardware)
- F. ROS 2 networking stack (UDP/IP/Ethernet, more deterministic network interactions)
- G. ROS 2 control stack
- **H.** All of them are important. Accelerating Gazebo could be useful when working with synthetic environment for RL or DRL. The other are both for timings and deterministic properties of the nodes
- I. Webots physics engine
- J. Image and depth data processing pipelines. Improvements on image and depth data compression and their integration with rosbag recording.
- K. Lidar drivers and perception

L. I'd like to see more general tools that can be implemented as nodes or library calls that allow me to quickly build accelerated alternatives for my system.

2022 Hardware Acceleration **Report in Robotics**

First phase: What do you care more about when it comes to hardware acceleration? (<u>report</u>→)



Bandwidth or throughput

(the total amount of work done in a given time for a task)



Speed

(shorter execution time between the start and the completion of a task)



Real-time

(ability of completing a task's computations while meeting time deadlines)



Power

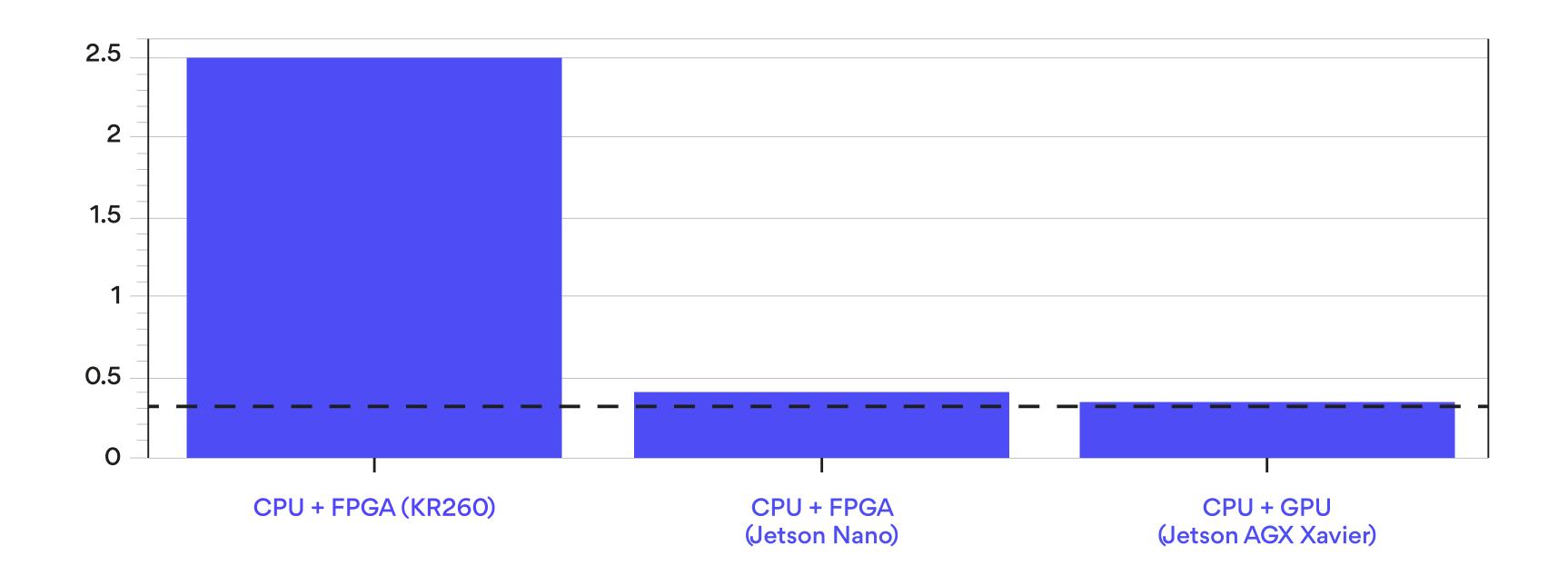
(the electrical energy per unit of time consumed while executing a given task)

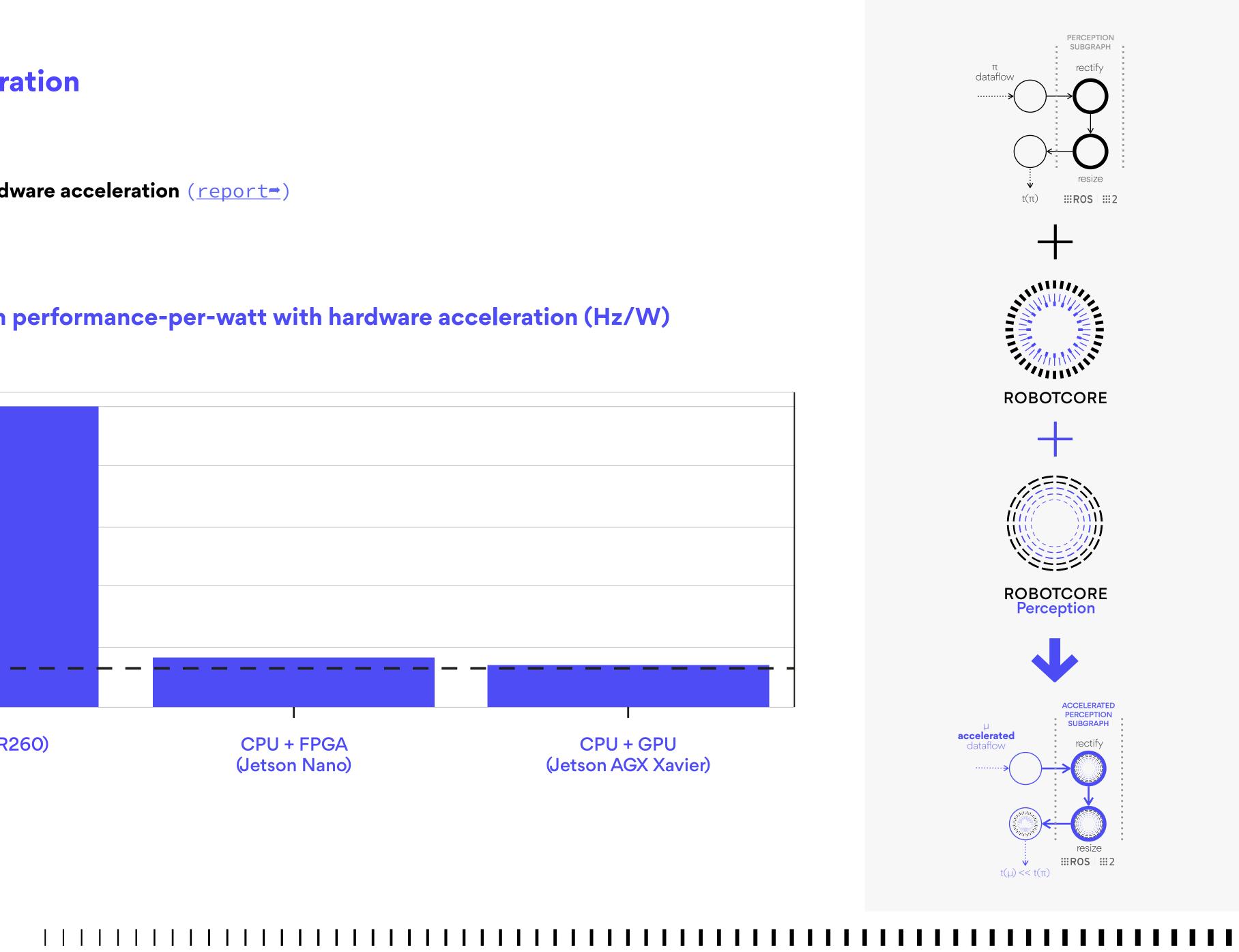
98 answers

2022 Hardware Acceleration **Report in Robotics**

Second phase: **Benchmarking hardware acceleration** (<a href="report")

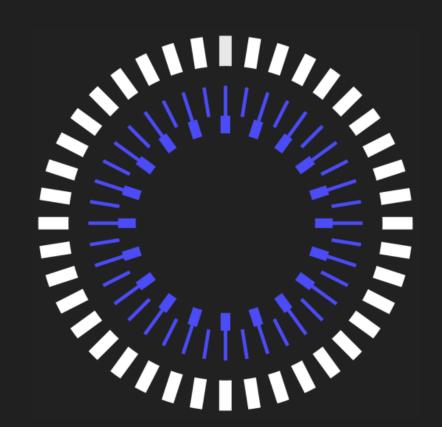
ROS 2 perception graph performance-per-watt with hardware acceleration (Hz/W)





ROBOTCORE tools and robot cores

ROS 2 API compatible hardware acceleration tools and robot Intellectual Property (IP) cores. Increasing your robot's performance, including latency, power efficiency and platform scalability.



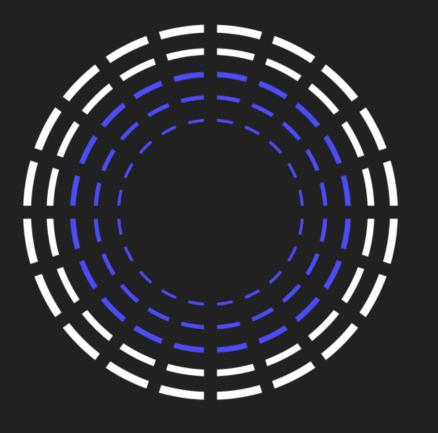
ROBOTCORE® Framework

Hardware acceleration framework for ROS and ROS 2.



ROBOTCORE® Cloud

Speed-up ROS 2 graphs with/in the cloud.



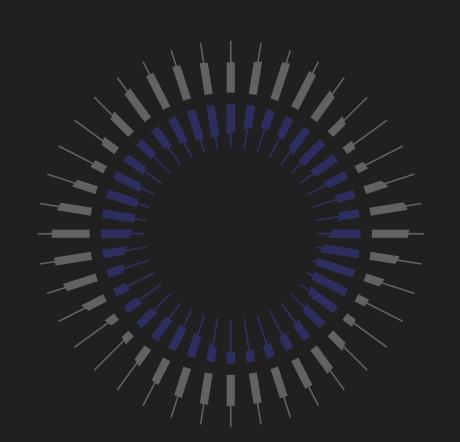
ROBOTCORE® Perception

Accelerated ROS 2 perception stack.



ROBOTCORE® Transform

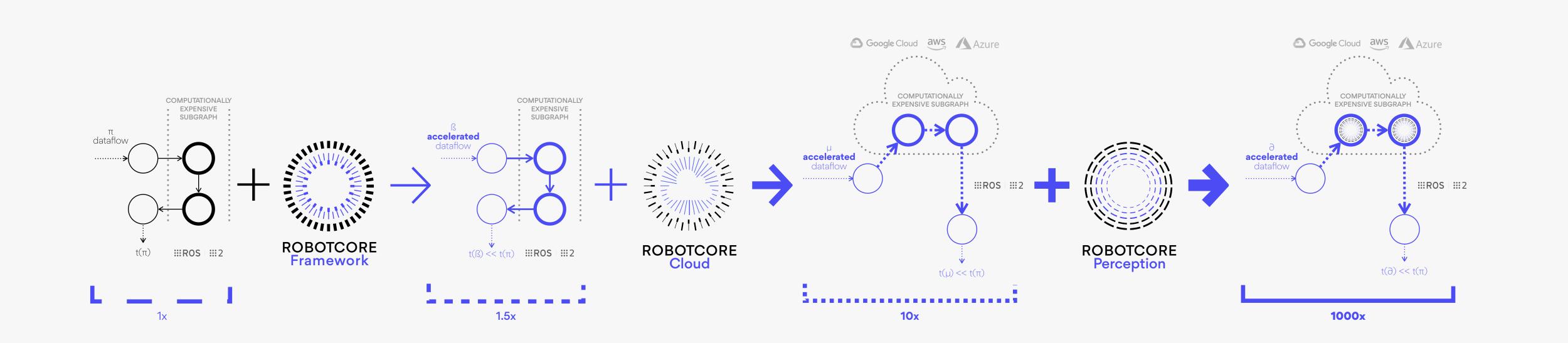
Accelerated ROS 2 coordinate transformations (**tf2**).



ROBOTCORE®

Accelerated ROS 2 ****:

FogROS2 and ROBOTCORE Cloud: Tools to speed-up ROS 2 graphs with the cloud, and in the cloud.



https://github.com/BerkeleyAutomation/FogROS2

Robotics MCU: A robotics microcontroller unit (MCU) powered by RISC-V and ROS 2

OMG DDS ROS 2 mW 100 Mbps budget applications Target: Iron Irwini (May'23) 1x Ethernet and RTPS interoperability DDS v1.4 and RTPS 2.3 More peripherals 2xUART, 2xI2C master, 1xI2C slave, 2xQSPI master, 1xSDIO Robotics 32-bit 1-core in-order, 4 MCU **RISC-VISA** pipeline-stage 4x PWM Silicon-proven Nodes from 130 nm to 22nm Apache 2.0 and MIT Best-effort and reliable **512 KB** energy QoS policies available Commercially-friendly license efficient SRAM memory

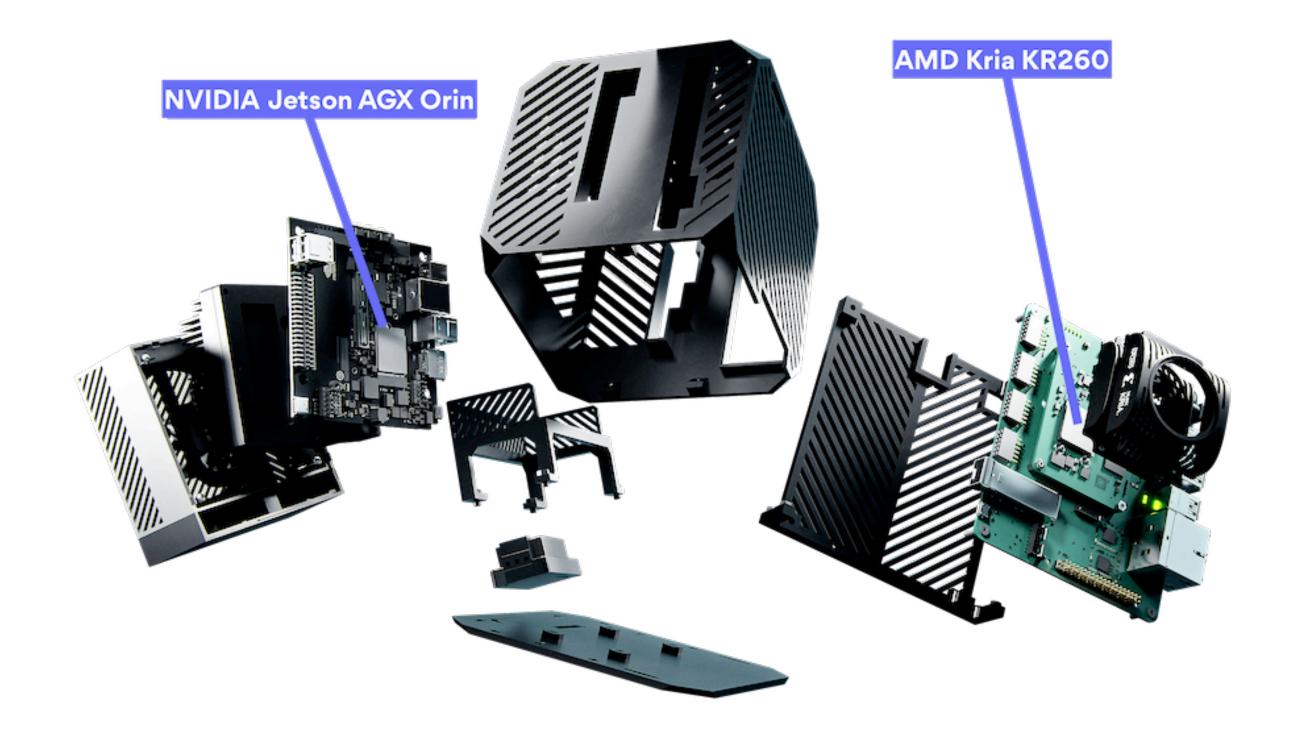


ROBOTCORE:

The Robotic Processing Unit specialized in ROS computations

ROBOTCORE® is a <u>robot-specific processing unit</u> that helps map Robot Operating System (ROS) computational graphs to its CPUs, GPU and FPGA efficiently to obtain best performance. It empowers robots with the ability to **react faster**, consume **less power**, and deliver **additional real-time** capabilities.





https://github.com/ros-acceleration/robotic_processing_unit

ACCELERATION ROBOTICS

Thanks

Q&A