ACCELERATION ROBOTICS

2022

Hardware Acceleration Report in Robotics

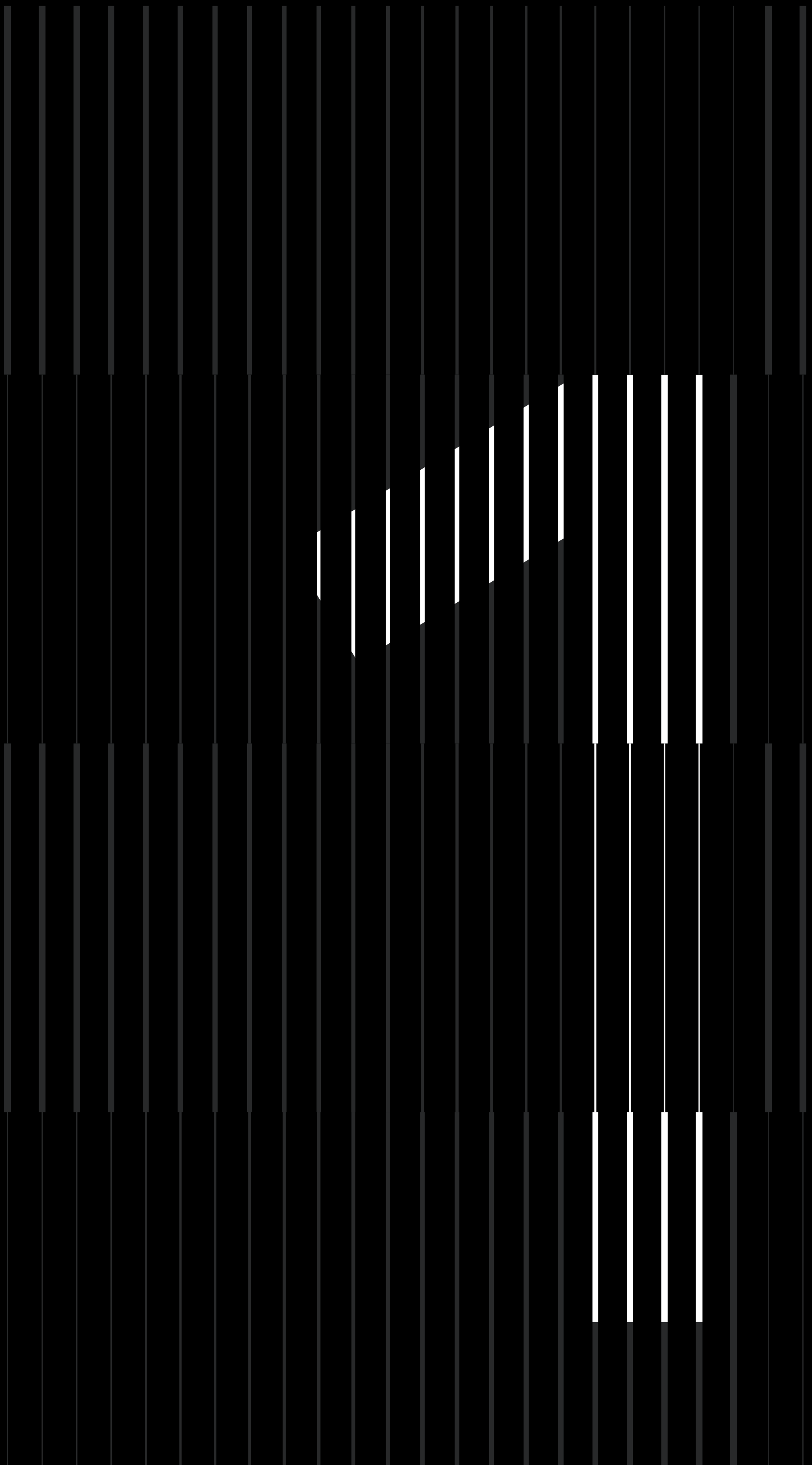
About the "2022 Hardware Acceleration Report in Robotics"

Hardware acceleration will revolutionize robotics, enabling new applications by speeding up robot response times while remaining power-efficient. However, the diversity of acceleration options makes it difficult for roboticists to select the right computational resource for each task, defaulting to CPUs. This report 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. The report compares the most popular computation solutions in robotics used today through reproducible and measurable examples available at the ROS 2 Hardware Acceleration Working Group GitHub organization.

About Acceleration Robotics

Acceleration Robotics is a firm focused on designing customized brains for robots to hasten their response time. Founded by top robotic experts to deliver semiconductor building blocks for robots, the company leverages GPUs and FPGAs to create custom hardware that speeds up a robot's operation.

Index



Executive Summary

Robots are deterministic machines. Meeting time deadlines in their computations (*real-time*) is the most important feature however other characteristics are also of relevance while designing robotic computations including the time between the start and the completion of a task (*latency*), the total amount of work done in a given time (*bandwidth or throughput*) or that a task happens in exactly the same timeframe, each time (*determinism*). **CPUs are widely used in robotics due to their availability however they hardly provide real-time and safety guarantees while delivering high throughput**. **Hardware acceleration** (with either FPGAs, GPUs or other accelerators) **presents an answer to this problem**. One that allows the robotics architect to create custom computing architectures for robots that comply with real-time and bandwidth requirements, while lowering power consumption.

Hardware acceleration has the potential to revolutionize robotics, enabling new applications by speeding up robot response times while remaining power-efficient. However, the diversity of acceleration options makes it difficult for roboticists to select the right computational resource for each task. **This report 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.** The report compares the most popular computation solutions in robotics used today through reproducible and measurable examples available at the ROS 2 Hardware Acceleration Working Group GitHub organization.

Since **most companies building real robots today use ROS** or similar event-driven software frameworks, **this report uses ROS as the common baseline in robotics to conduct the study** (section 2.2). In particular, we use ROS 2 which presents a modern industry-accepted framework for robot application development and consider both bandwidth and latency to benchmark performance in robotics (section 2.3) using a *grey-box* and *non-functional* benchmarking approach (section 2.4).

The work presented in this report happened in two phases. First, 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. The most relevant results from these two phases are summarized below:

Community Survey

↘

Only about half of the respondents (**51%**, section 3.1) is confident about the value and differences between hardware acceleration solutions for robotics:

→ Only **62.5%** (section 3.2) have ever programmed an acceleration kernel.
→ This suggests that there's still a lot of work to be done from silicon vendors' side to further simplify the use and integration of their solutions.

↘

The majority of the roboticists currently use GPUs (**69.8%**, section 3.11) versus FPGAs (**21.9%**):

→ Roboticists seem to care about speed or latency (**48.9%**, shorter execution time) as much as *real-time* and determinism (**46.8%**). Only a reduced **4.3%** would prioritize power consumption (section 3.10).
→ This indicates that there's margin for FPGA usage growth in the ROS robotics community.

↘

When asked about the most relevant aspects of hardware acceleration (section 3.3), **52.1%** of the roboticists that answered indicate that a simpler integration with ROS 2 and its ecosystem of tools is of most relevance to them:

→ **52.1%** Integration with ROS 2 (`ament` build system **11.5%**, `colcon` build tools **19.8%** and acceleration firmware **20.8%**).
→ **32.3%** Capabilities to easily switch between hardware accelerated and CPU-centric Nodes.
→ **11.5%** Benchmarking capabilities for hardware acceleration.
→ **4.1%** Others.

↘

ROS 2 Perception stack with a **64.6%** (section 3.5, *multiple selections allowed*) is the most demanded group of packages to be accelerated:

→ **64.6%**	ROS 2 Perception stack
→ **60.4%**	"Gazebo physics engines"
→ **40.6%**	Navigation2
→ **30.2%**	"DDS communication middleware"
→ **21.9%**	MoveIt 2
→ **20.8%**	ROS 2 networking stack (UDP/IP/Ethernet)
→ **19.8%**	ROS 2 control stack

↘

The majority of the respondents (**92.7%**) indicated that they'd prefer the commercially friendly Apache 2.0 license for hardware acceleration resources (section 3.8).

→ **74.8%** would prefer source code access to acceleration kernels with code examples (section 3.9)

↘

Ubuntu seems to be the dominant (**79.5%**, section 3.15) operating system requested by ROS roboticists for hardware acceleration.

→ Ubuntu 20.04 is the preferred option (**59%**) followed by Ubuntu 22.04 (**20.5%**).
→ Yocto-based rootfs is preferred after Ubuntu (**7.7%**).

↘

For packaging accelerators, deb files are the preferred option (**59%**, section 3.16) followed by Docker containers (**23.1%**).

Benchmarking hardware acceleration

↘

Results obtained across benchmarks performed on a ROS 2 perception graph show that *from a latency point of view* **optimized FPGA accelerators outperform their GPU counterparts**, even when using powerful GPUs.

→ Considering mean runtime measurements (in ms, Figure 15, section 4.2), the use of a CPU + FPGA combination delivers a **3.56x speedup** over a comparable CPU + GPU, and a **1.36x speedup** over a comparable CPU.

→ When considering a more powerful CPU + GPU combination (Figure 17, section 4.2), the FPGA still outperforms it with a **1.59x speedup**.

↘

ROS 2 Perception Nodes running in an FPGA also outperform those running in a GPU by relevant speedups.

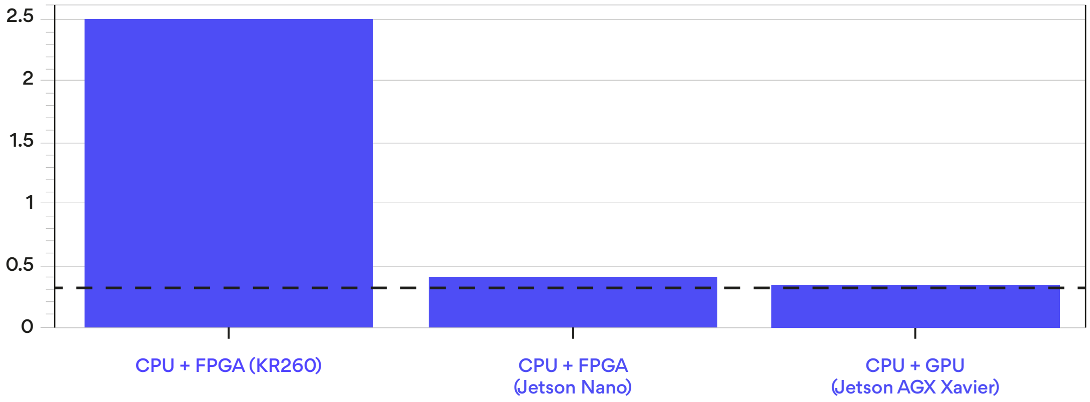
→ To discriminate between any possible differences between the CPU cores, measurements were collected isolating perception computations by discarding both the ROS 2 message-passing infrastructure overhead, as well as the host-device (GPU or FPGA) data transfer overhead.

→ Popular perception algorithms such as the Histogram of Oriented Gradients (HOG) show a **500x speedup** in an FPGA and relative to a comparable GPU (Figure 22).

↘ **Figure 0**

Performance-per-watt benchmark of a simple ROS 2 perception graph across various accelerators. The computational graph studied is described in section 4.2. Bigger is better.

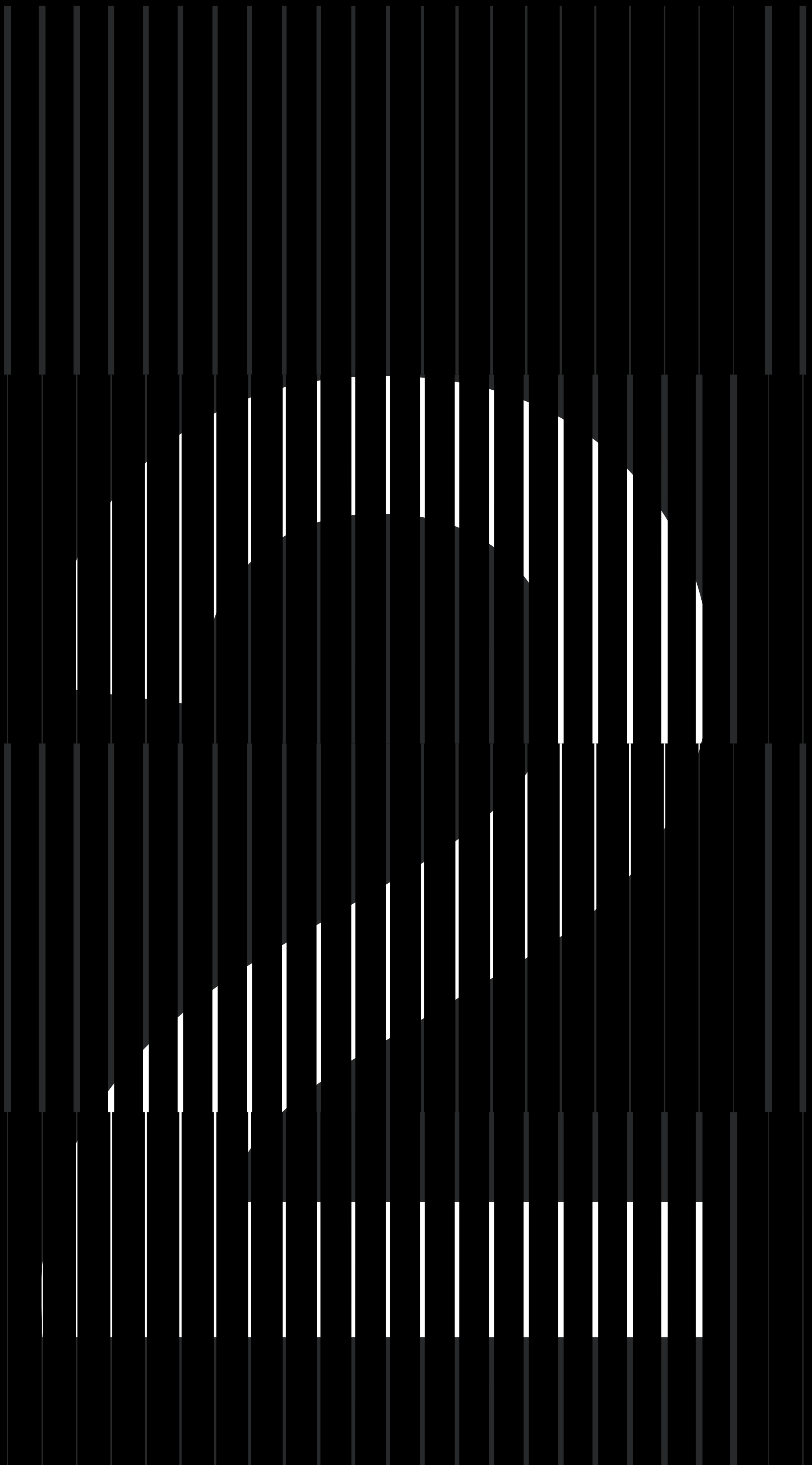
ROS 2 perception graph performance-per-watt with hardware acceleration (Hz/W)



Overall, results hint that the **rate at which the energy consumption grows with GPU solutions seems to be smaller than the rate at which the latency performance improves, which leads to a decaying performance-per-watt in our ROS 2 perception measurements** with these GPU + CPU solutions. Instead, FPGA-enabled solutions present a performance-per-watt figure that's 6x (5.93x) better than the one observed in comparable GPU + CPU sets and 8x (7.95x) better than the one in more power GPU + CPU sets.

These results indicate that **using bandwidth as the only measure of performance can be misleading in ROS and robotics**. Moreover, data suggests that when considering latency as the measure of performance, GPU sets may struggle to find themselves on equal footing with their FPGA counterparts.

There are nevertheless various advantages that GPUs inherently have and that should be considered while building complex robotic computations. Moreover, though FPGA kernel runtime execution outperforms their GPU counterparts, it's relevant to note that FPGAs are resource-limited and thereby it's important to consider that only a fixed set of accelerators would be able to fit within an FPGA at any given point in time whereas the GPUs don't have this limitation due to their architectures. **Scalable robot compute architectures that consider hardware acceleration should look at combining CPUs, GPUs and FPGAs to obtain the best trade-off.**



Introduction

2.1 The CPU *whack-a-mole* in robotics

Robots are deterministic machines. Meeting time deadlines in their computations (*real-time*) is the most important feature however other characteristics are also of relevance while designing robotic computations including the time between the start and the completion of a task (*latency*), the total amount of work done in a given time (*bandwidth or throughput*) or that a task happens in exactly the same timeframe, each time (*determinism*).

There's a critical relationship between the hardware and the software capabilities in a robot. Robotic systems usually have limited on-board resources, including memory, I/O, disk or compute capabilities, making it hard to balance between *real-time* and *bandwidth* requirements (due to limited shared resources), and restricting robots' reaction capabilities and speed. A key challenge in robotics using general purpose **CPUs**[1] is that they **hardly provide *real-time* and safety guarantees while delivering high *throughput*.** The de *facto* strategy in industry [1] to meet timing deadlines is a laborious, empirical, and case-by-case tuning of the system. **This *"CPU whack-a-mole"* approach in robotics is unsustainable and hard to scale** due to the lack of a hardware-supported timing-safe event driven programming interface in CPUs.

CPUs hardly provide real-time and safety guarantees while delivering high throughput. This *"CPU whack-a-mole"* approach in robotics is unsustainable and hard to scale

[1] CPUs are widely used in commercial compute platforms in robotics due to their availability and generalized use. The general purpose nature of CPUs makes them specially interesting for roboticists to kickstart projects, however this comes at a cost when translating into real applications: their fixed architectures and limited amount of resources difficult adaptability to new (computing) robotic scenarios and always impose a trade-off between performance and *determinism*.

Hardware acceleration with dedicated compute architectures (in either FPGAs, GPUs or other accelerators) is presented as an alternative to CPUs. One that allows the architect to adaptively generate custom computing architectures to meet the robotic computing demands, delivering **a solution that can comply with *real-time* and *bandwidth* requirements** while increasing reliability and lowering power consumption.

Hardware acceleration is presented as an alternative to CPUs delivering a solution that can comply with *real-time* and *bandwidth* *requirements*

This report presents robotic architects with a resource to consider while designing their robot computational architectures that describes how hardware acceleration can improve their performance. To study the capabilities of hardware acceleration in robotics, this article follows a quantitative approach [2] to measure performance and compares the most popular hardware computation solutions in robotics used today through ROS 2.

Results presented in this report are meant to be reproducible and disclosed as open source examples made publicly available at the ROS 2 Hardware Acceleration Working Group GitHub organization repositories.

2.2 ROS 2 as the common baseline in robotics

Robot behaviors take the form of computational graphs, with data flowing between computation Nodes, across physical networks (communication buses) and while mapping to underlying sensors and actuators. The popular choice to build these computational graphs for robots these days is the Robot Operating System (ROS) [3], a framework for robot application development. ROS enables you to build computational graphs and create robot behaviors by providing libraries, a communication infrastructure, drivers and tools to put it all together. Most companies building real robots today use ROS or similar event-driven software frameworks. ROS is thereby the *common language in robotics*, with several hundreds of companies and thousands of developers using it everyday. ROS 2 [4] was redesigned from the ground up to address some of the challenges in ROS and solves many of the problems in building reliable robotics systems.

ROS 2 presents a modern and popular framework for robot application development most silicon vendor solutions support, and with a variety of modular packages, each implementing a different robotics feature that simplifies performance benchmarking in robotics.

2.3 Bandwidth, latency and performance considerations

The field of robotics is changing rapidly and must be studied with real examples and measurements on real robotic computations, rather than simply as a collection of definitions, designs and marketing actions. The quantitative approach [2] to robotics systems architecture fits well in this context and helps robotic architects come up with better performing architectures through an empirical strategy, and case-by-case tuning of the system.

In robotics ***bandwidth or throughput*** is the ***total amount of work done in a given time,*** such as the publication frequency (in frames per second) of a ROS 2 perception feed resulting from processing the data of a camera, or the data transfer rate in a give ROS 2 Topic (in megabytes per second) of a processed point cloud coming from a depth sensor. In contrast, **latency or response time** is the ***time between the start and the completion of a task,*** such as milliseconds for the reception of an image from a ROS 2 Topic subscription in a computational graph.

When speaking about performance in robotics, both bandwidth and latency should be taken into consideration. In particular, given the importance of real-time in robotics we'd generally be interested in the latency for performance benchmarking.

When speaking about performance in robotics, both bandwidth and latency should be taken in consideration

A final consideration is the bandwidth/latency performance improvement ratio in robotics. A simple rule of thumb in (general) computation is that **bandwidth grows by at least the square of the improvement in latency.** Robotic architects should take this into consideration while designing their robotic systems.

2.4 Methodology for benchmarking performance

Benchmarking is the act of running a computer program to assess its relative performance. In the context of hardware acceleration, it's fundamental to assess the relative performance of an acceleration kernel versus its CPU scalar computing baseline. Similarly, benchmarking helps comparing acceleration kernels across hardware acceleration technology solutions (e.g. `FPGA_A` vs `FPGA_B` or `FPGA_A` vs `GPU_A`, etc.) and across kernel implementations (within the same hardware acceleration technology solution).

There're different types of benchmarking approaches. The following diagram depicts the most popular inspired by [5]:

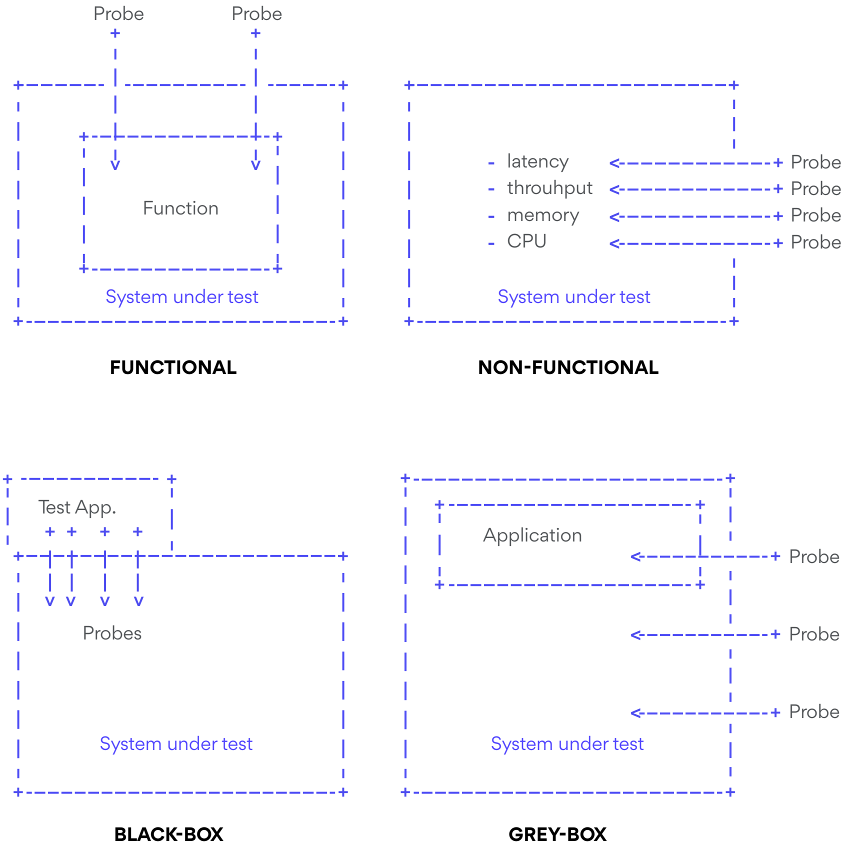


Figure 1

Performance benchmarking approaches. Functional (top-left), Non-functional (top-right), Black-Box (bottom-left) and Grey-box (bottom-right).

In addition, the following aspects should be considered when benchmarking ROS 2 robotics computations:

Embedded: Benchmarks should run in embedded *easily.*

ROS 2-native: Benchmarks should consider the particularities of ROS 2 and its computational graph. If necessary, they should instrument the communications middleware and its underlying layers.

Intra-process, inter-process and intra-network: Measures conducted should consider communication within a process in the same SoC, between processes in an SoC and between different SoCs connected in the same network (intra-network).

Compute substrate-agnostic: benchmarks should be able to run on different hardware acceleration technology solutions. For that purpose, a CPU-centric framework (as opposed to an acceleration technology-specific framework) that can be integrated in various accelerators for benchmarking and/or tracing is the ideal choice.

Automated: benchmarks and related source code should be designed with automation in mind. Once ready, creating a benchmark and producing results should be (ideally) a fully automated process.

Hardware farm mindset: benchmarks will be conducted on hardware embedded platforms sitting in a farm-like environment (redundancy of tests, multiple SoCs/boards) with the intent of validating and comparing different technologies.

Accounting for all of this, and similar to the ROS Enhancement Proposal (REP) REP-2008 proposal [6], in this report we adopt a **grey-box and non-functional benchmarking approach for hardware acceleration** that allows to evaluate the relative performance of accelerated ROS 2 individual Nodes as well as complete computational graphs. To realize it in a technology agnostic-manner, we select the Linux Tracing Toolkit next generation (LTTng) which will be used for tracing and benchmarking.

2.4.1

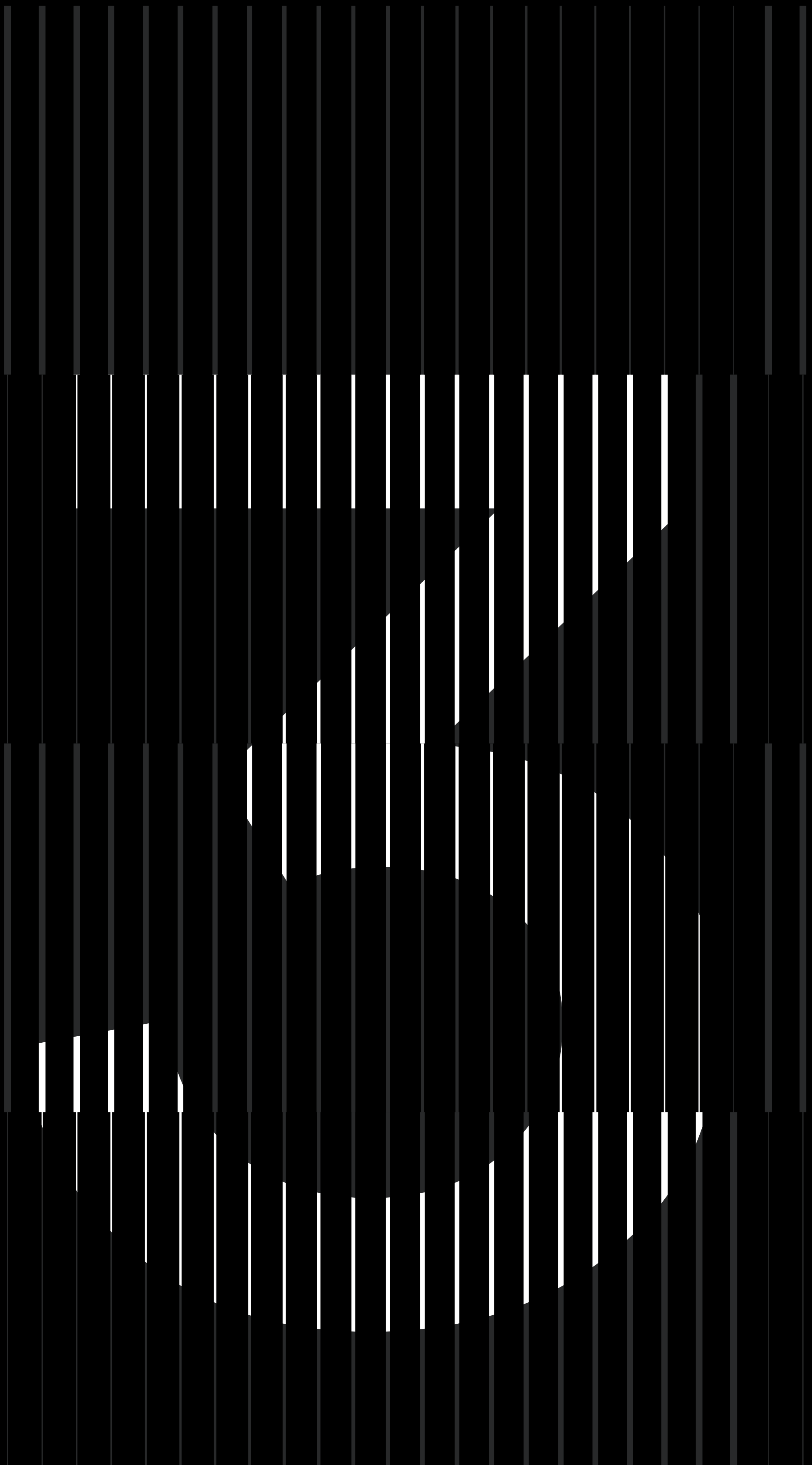
Differences between tracing and benchmarking

Tracing and benchmarking can be better understood as follows:

Tracing: a technique used to understand what goes on in a running software system.

Benchmarking: a method of comparing the performance of various systems by running a common test.

From these definitions, inherently one can determine that both benchmarking and tracing are connected in the sense that the test/benchmark will use a series of measurements for comparison. These measurements will come from tracing probes. In other words, tracing will collect data that will then be fed into a benchmark program for comparison.



Community Survey

3.1 Are you familiar with the different hardware acceleration solutions and their advantages for ROS 2 and Gazebo?

(e.g. FPGAs vs GPUs)

96 answers

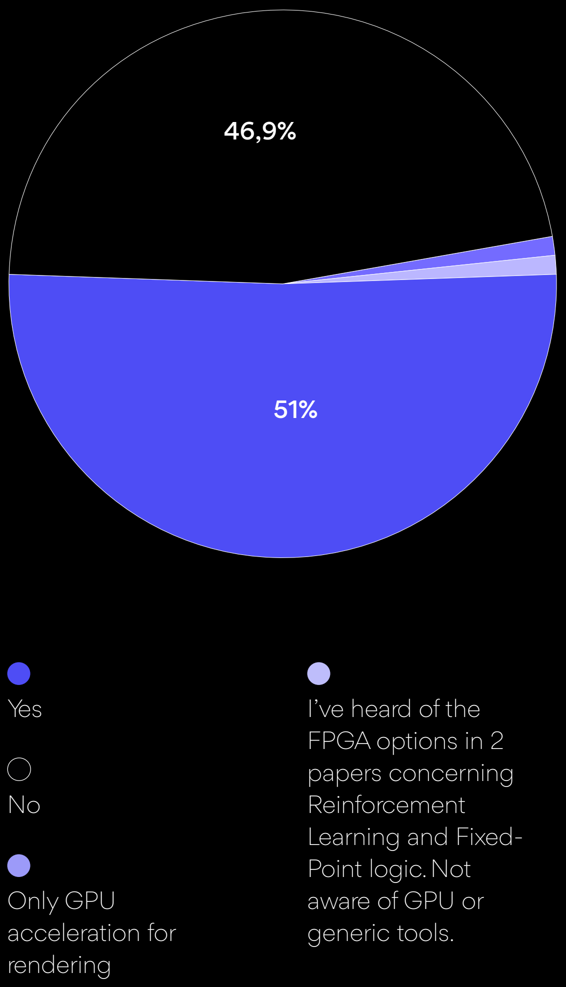


Figure 2

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Are you familiar with the different hardware acceleration solutions and their advantages for ROS 2 and Gazebo? (e.g. FPGAs vs GPUs)?".*

3.2 Have you ever programmed an acceleration kernel?

96 answers

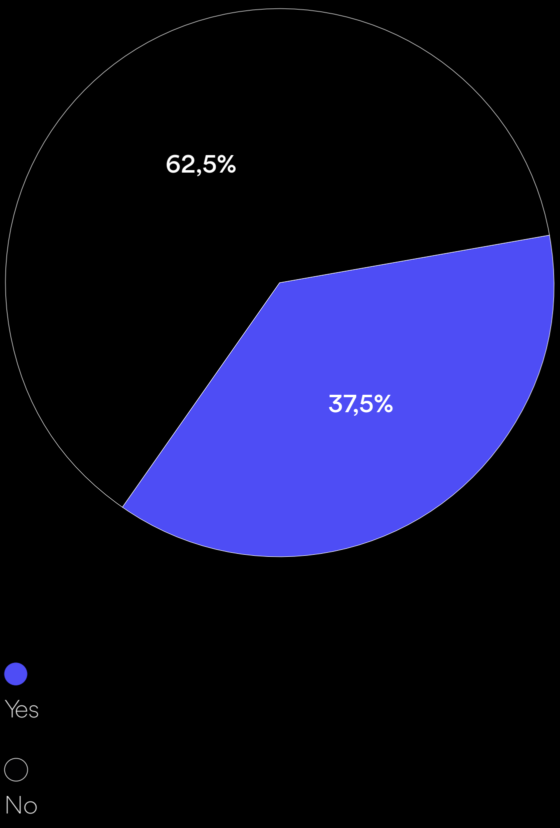


Figure 3

Results from the “Hardware acceleration in ROS 2 and Gazebo survey” (link) question: *“Have you ever programmed an acceleration kernel?”.*

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

96 answers

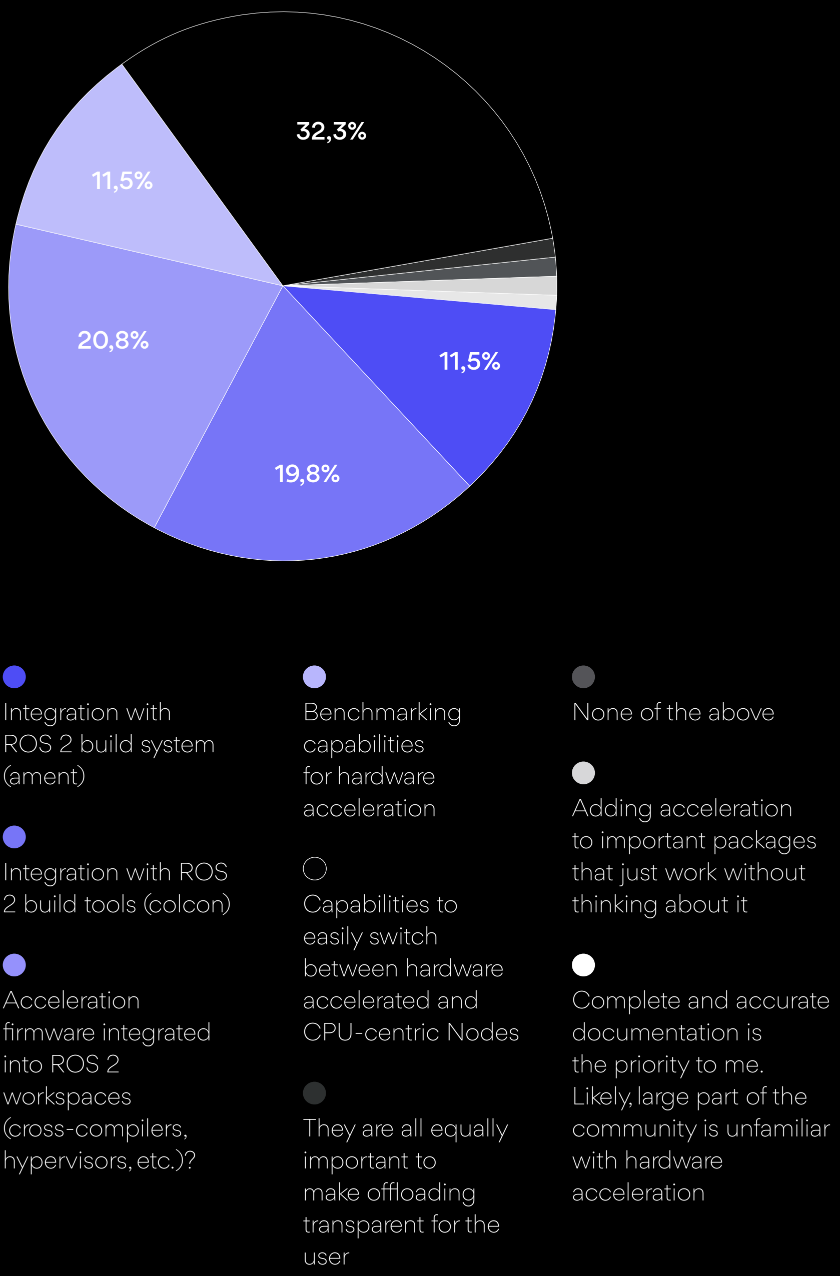


Figure 4

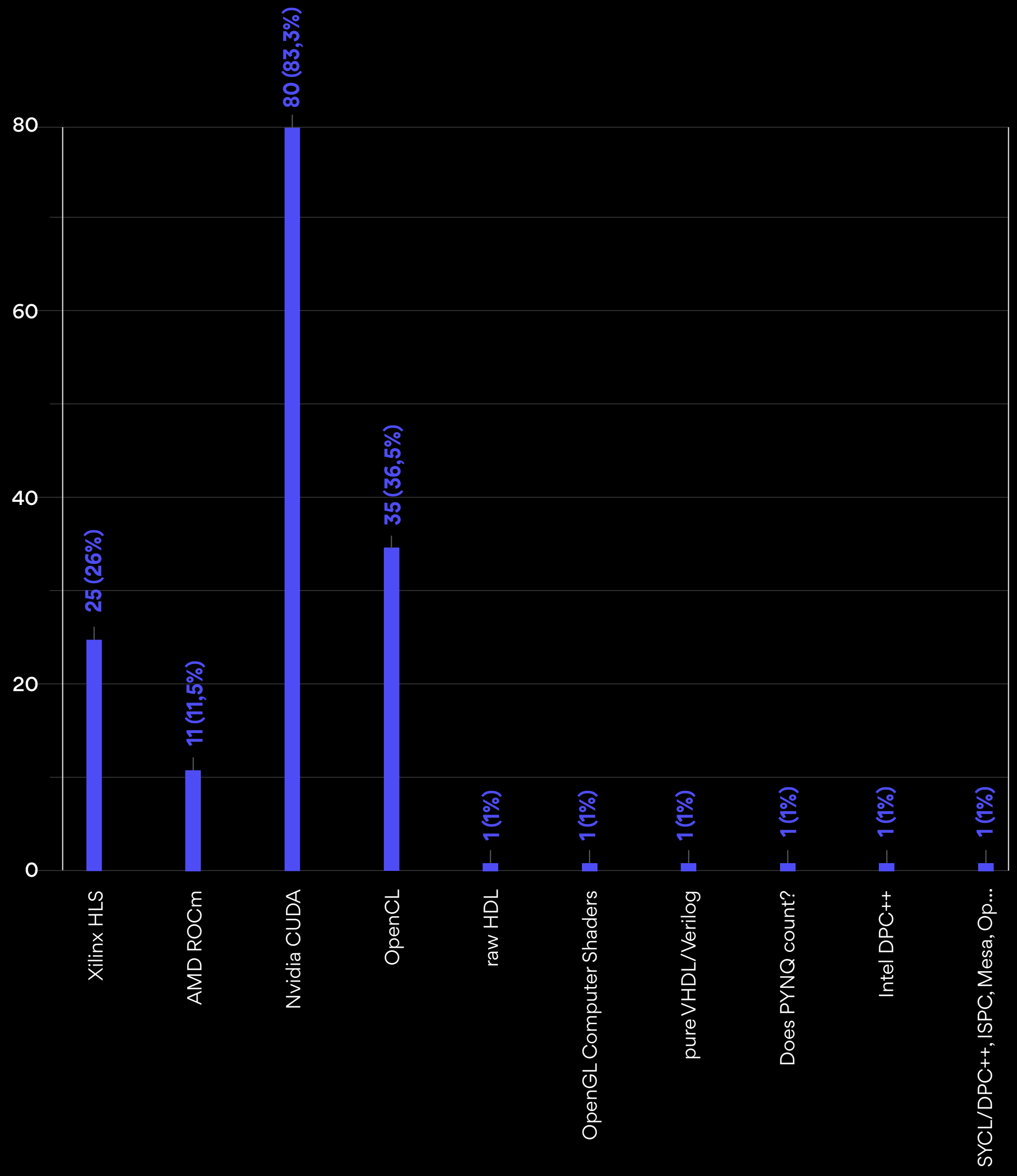
Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"We're pushing forward REP-2008 initiative (Hardware Acceleration Architecture and Conventions, https:// github.com/ros-infrastructure/rep/ pull/324) to better integrate hardware acceleration with ROS and Gazebo, what's most important for you?".*

3.4 Which hardware acceleration platform/framework are you familiar with?

96 answers (multiple answers allowed)

Figure 5

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Which hardware acceleration platform/ framework are you familiar with?"*



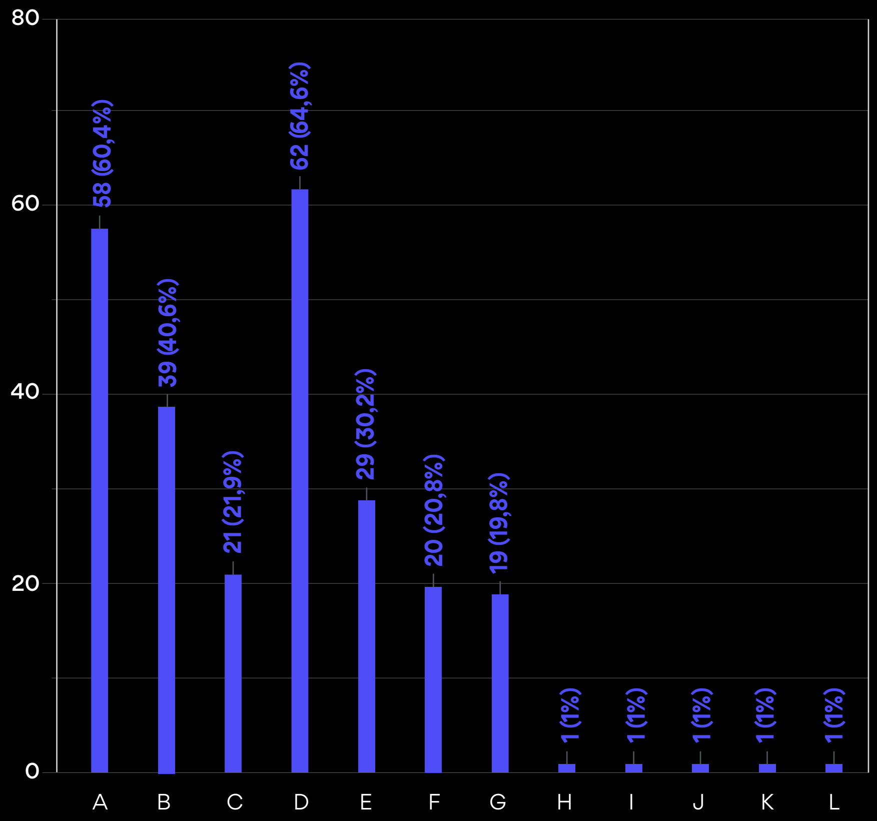
3.5

What packages/components do you think we should prioritize when it comes to hardware accelerating ROS 2 and/or Gazebo?

Figure 6

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"What packages/components do you think we should prioritize when it comes to hardware accelerating ROS 2 and/or Gazebo?".*

96 answers (multiple answers allowed)



A. Gazebo/Ignition physic engines

B. ROS 2 navigation stack (navigation2)

C. ROS 2 manipulation stack (MoveIt2)

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.

3.6 What specifically would you like to see accelerated in ROS 2 or Gazebo in the short term?

96 answers

Selected Answers

ROS 2

→ Perception (3D SLAM, VIO package, image_proc)

→ Nav2

→ Moveit2 ompl

→ ROS 2 control stack

→ The ability to create hardware based timers in ros2 for deterministic call back times

→ The ROS 2 executor and counterparts in DDS. For example, a scheduler implemented in hardware

Gazebo

→ Allow fast rendering like in Unity

→ 3D camera simulation (libgazebo_ros_openni_kinect)

→ Physics engine in Gazebo and enable ML training

→ Accelerated simulated sensors/sensor processing

→ Gazebo physics

3.7 What type of examples would you like to see on how hardware acceleration can improve Gazebo/Ignition and ROS 2?

45 answers

Selected Answers

→ Accelerated simulations (2+ times faster than RT) with Nav2, multiple AMRs in simulations, SLAM

→ An out of the box hardware accelerated velodyne simulator

→ Multi-Agent with computational expensive sensors like 3D cameras

→ Latency and timing cycles. Additional capabilities unlocked due to lower latency

→ Perception and planning examples, with source

3.8

Silicon vendors often use concerning licenses to lock users into their hardware. Which type of licenses would you like to see in the packages that your vendor maintains/provides?

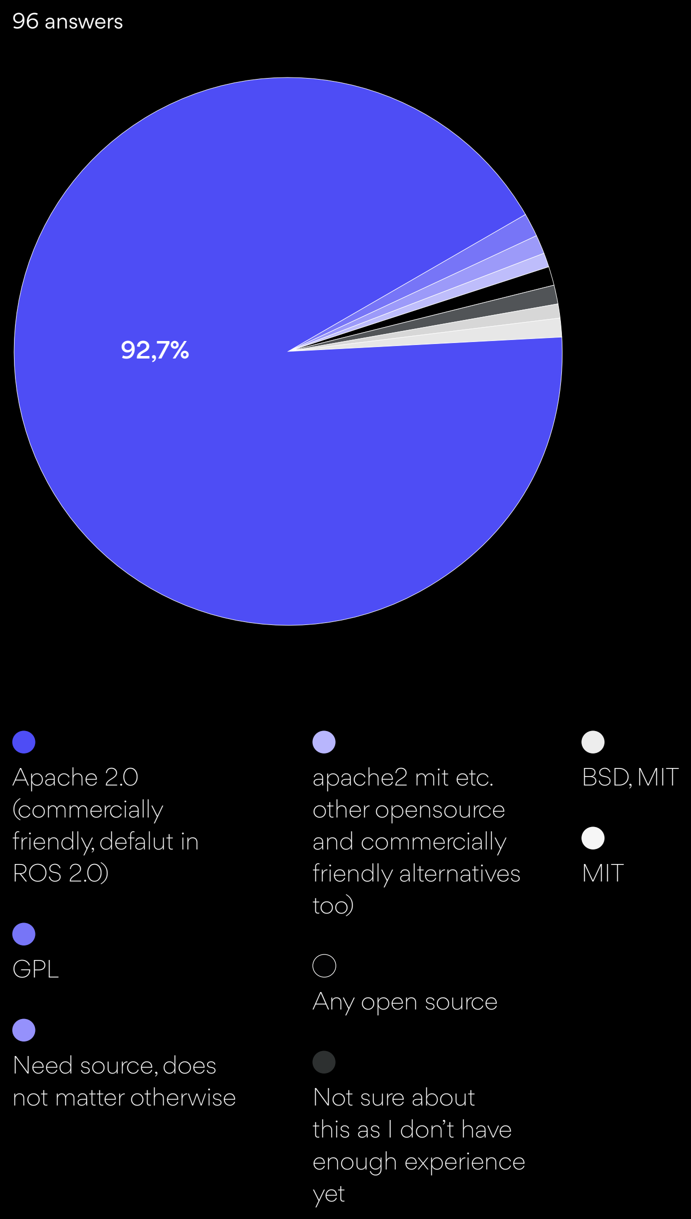


Figure 7

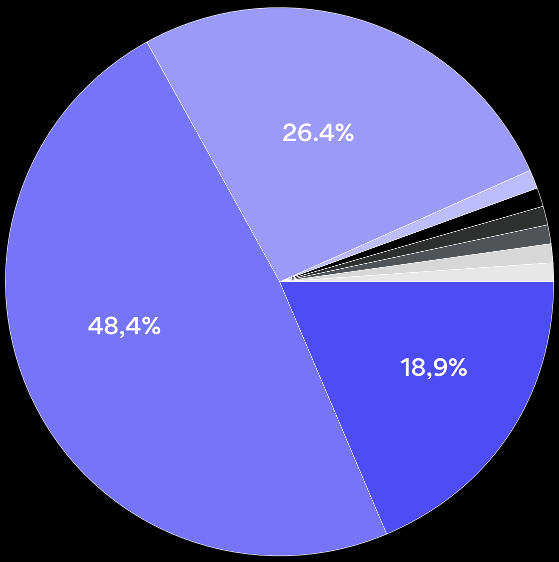
Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Silicon vendors often use concerning licenses to lock users into their hardware. Which type of licenses would you like to see in the packages that your vendor maintains/provides?".*

3.9 Do you prefer kernels integrated in ROS 2 packages as binaries or built as part of the ROS 2 workspace from source?

Figure 8

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Do you prefer kernels integrated in ROS 2 packages as binaries or built as part of the ROS 2 workspace from source? (Note that hardware skills to develop or extend acceleration kernels are scarce and learning what's required may take years)?".*

96 answers



● Binaries are just fine, I just want a plug and play solution

● Binaries are fine and what I'll use but it'd be great to have source code examples

● I need the source code of kernels, and plan to build them from source

● Why not the standard approach? sources on github and binaries in APT? You can always overlay the system-installed package with a custom-built one should you need custom-built kernels

○ Source code if it means faster availability. Binaries could follow once community of users if large enough

● We are interested in safety and safety certifications, so source would be ideal for us unless we can have safety rated modules that are binary

● I'd like both. Binaries for plug and play for users with common hardware, but source for custom projects and memory hardware constrains

● Binaries would be fine but there are always corner cases when reading the source code may give you a hint on whats is going wrong. Not sure if I will spend time building it but having the code available is useful for keeping track of things

● Not sure if the other 3 bullets on this answer cover what I'd like to see. I want binaries for the "common" use cases - say...a semantic segmentation module of YOLOvX and a module of ResNet-50 + Mask R-CNN + FPN. That way, I can grab something from the single-shot detector and 2-stage detection schemes for quick inclusion in a system. But I want the source code when I look to implement my own accelerated system. In which case, I'd like to be able to test the source and see it run just like the ResNet binary, such that I can start making the changes I need

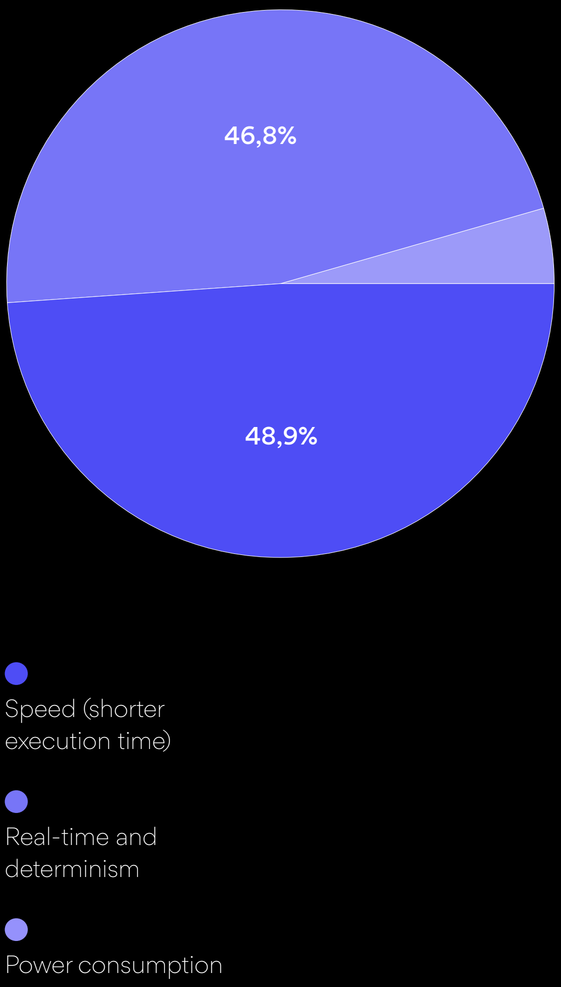
3.10

What do you care more about when it comes to hardware acceleration?

94 answers

Figure 9

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"What do you care more about when it comes to hardware acceleration?"*



Speed (or latency): time between the start and the completion of a task

Real-time: Meeting time deadlines in their computations

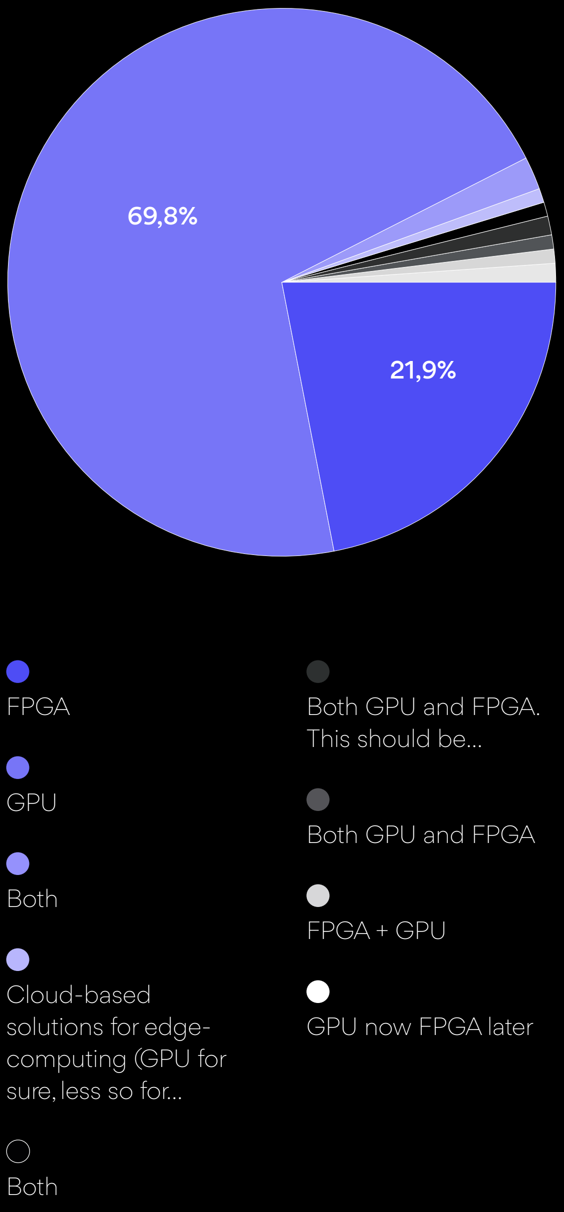
Determinism: that a task happens in exactly the same timeframe each time

3.11 Which hardware acceleration solution are you using or planning to use?

96 answers

Figure 10

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Which hardware acceleration solution are you using or planning to use?".*



3.12 Why have you picked this hardware acceleration solution?

51 answers

Selected Answers

- Easy access
- More in line with my area of expertise and build methods
- Relatively low-cost off-the-shelf hardware is available
- Every proper laptop or PC has a GPU, unfortunately OpenCL is not spreaded that widely to be independent of GPU manufacturer
- There are already some resources for GPU integration that developers can "easily" hack together with ROS 2 or Ignition for parallelised stuff. However, using FPGA with ROS 2 sounds like a steeper learning curve - but having a group focusing on simplifying its integration/availability would open the use of FPGAs to the community
- We are using NVIDIA because it is widely available and easy to use. However we are looking at FPGAs now because some of the vendors have SIL and ASIL ratings. We cannot get SIL rated AGX SOMs from NVIDIA
- FPGA is more versatile to make hardware acceleration
- GPU are more widely spread
- FPGA's can enable better power consumption for similar accuracy and speed of GPUs. But GPU libraries make it easier to put them in systems. I want to compare FPGA and GPU performance

3.13 Which computing hardware solution are you using in your robots?

96 answers

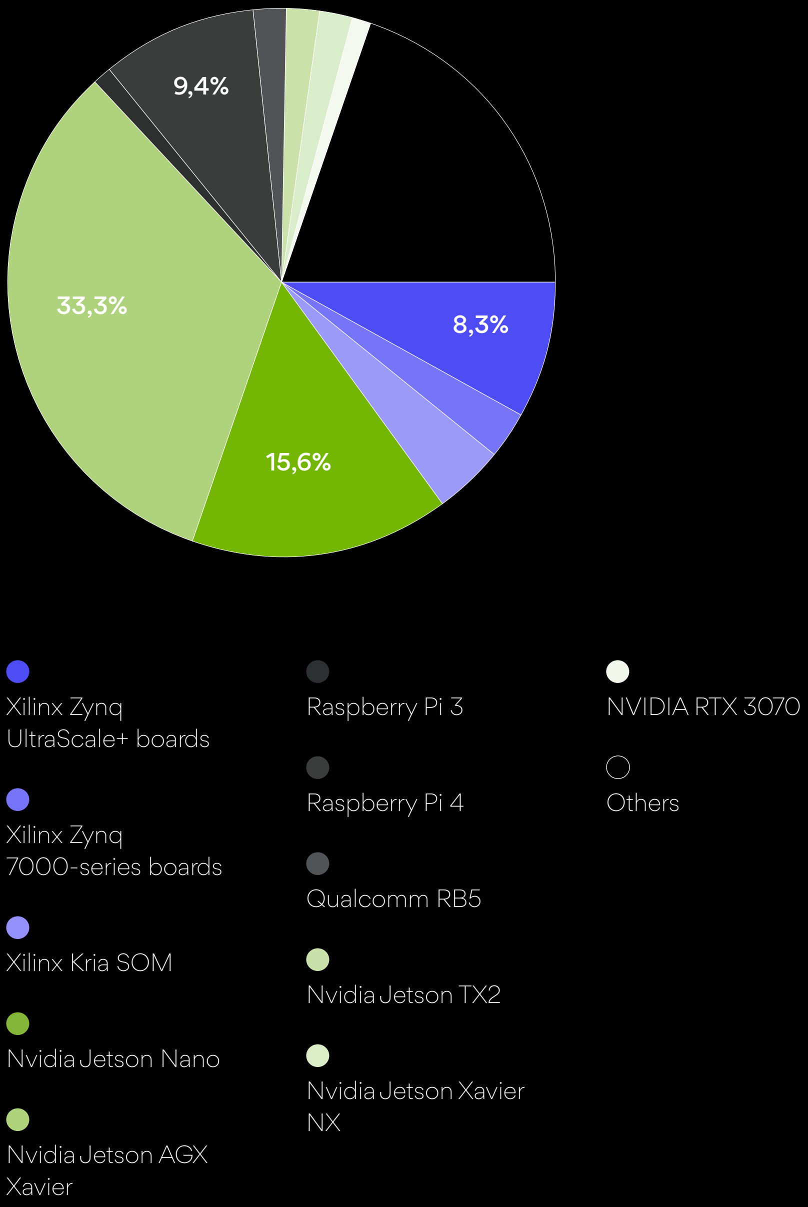


Figure 11

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Which computing hardware solution are you using in your robots? (please specify if other)".*

3.14 What type of robot are you creating?

96 answers

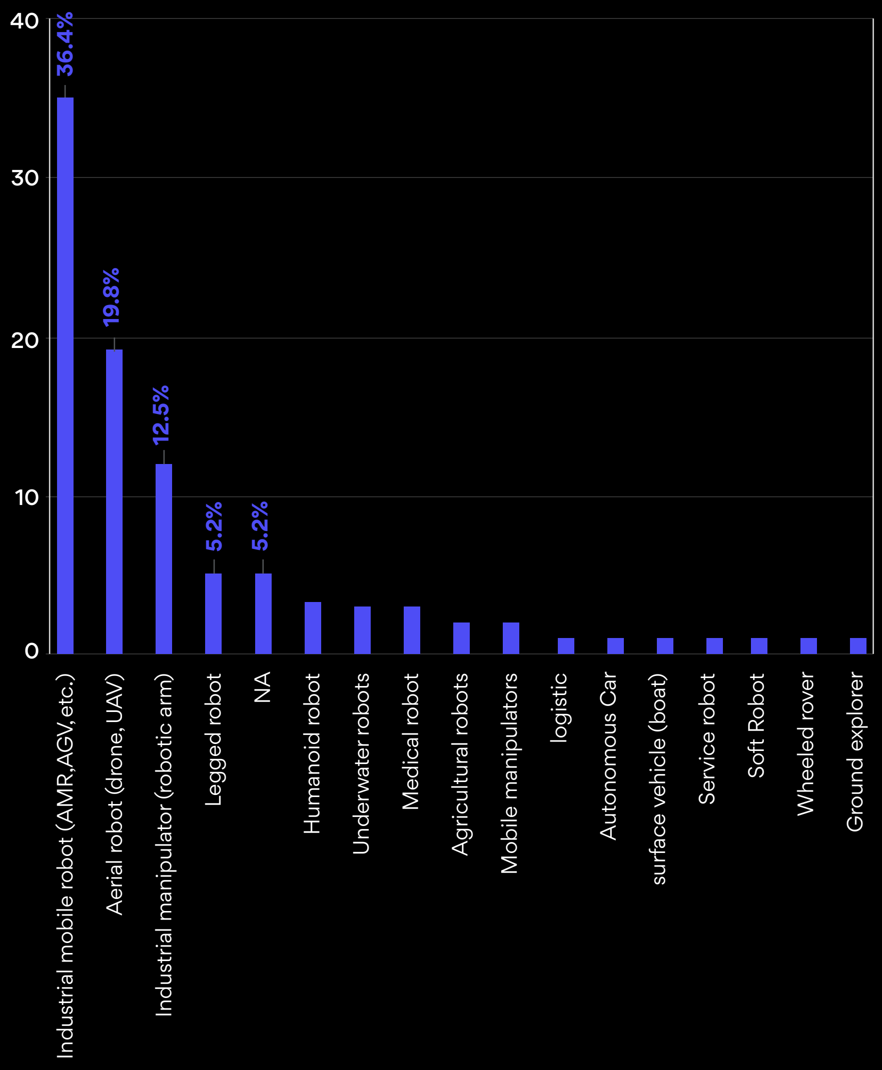


Figure 12. a

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"What type of robot are you creating? (please specify if other)"*. Processed answers.

3.15 Which Operating System (rootfs) should we be focusing on?

39 answers

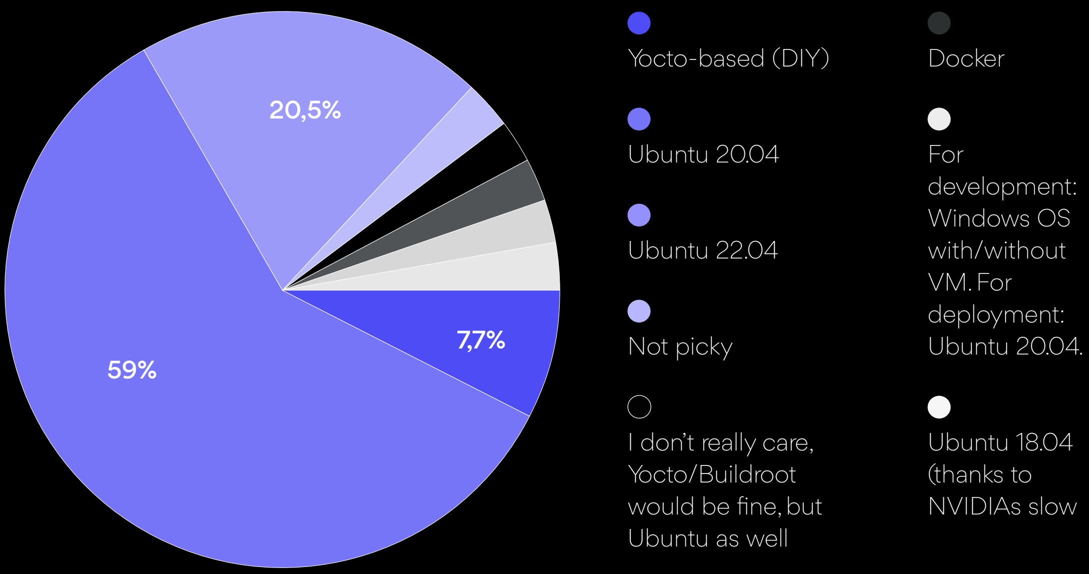


Figure 12.b

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"Which Operating System (rootfs) should we be focusing on?."*

3.16 How do you want your accelerators packaged for production use?

39 answers

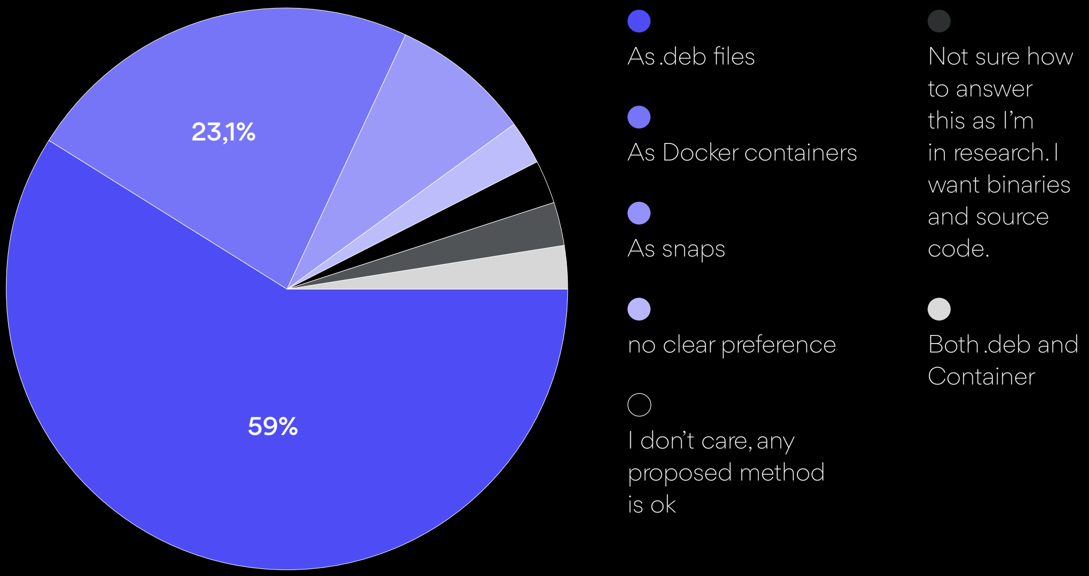
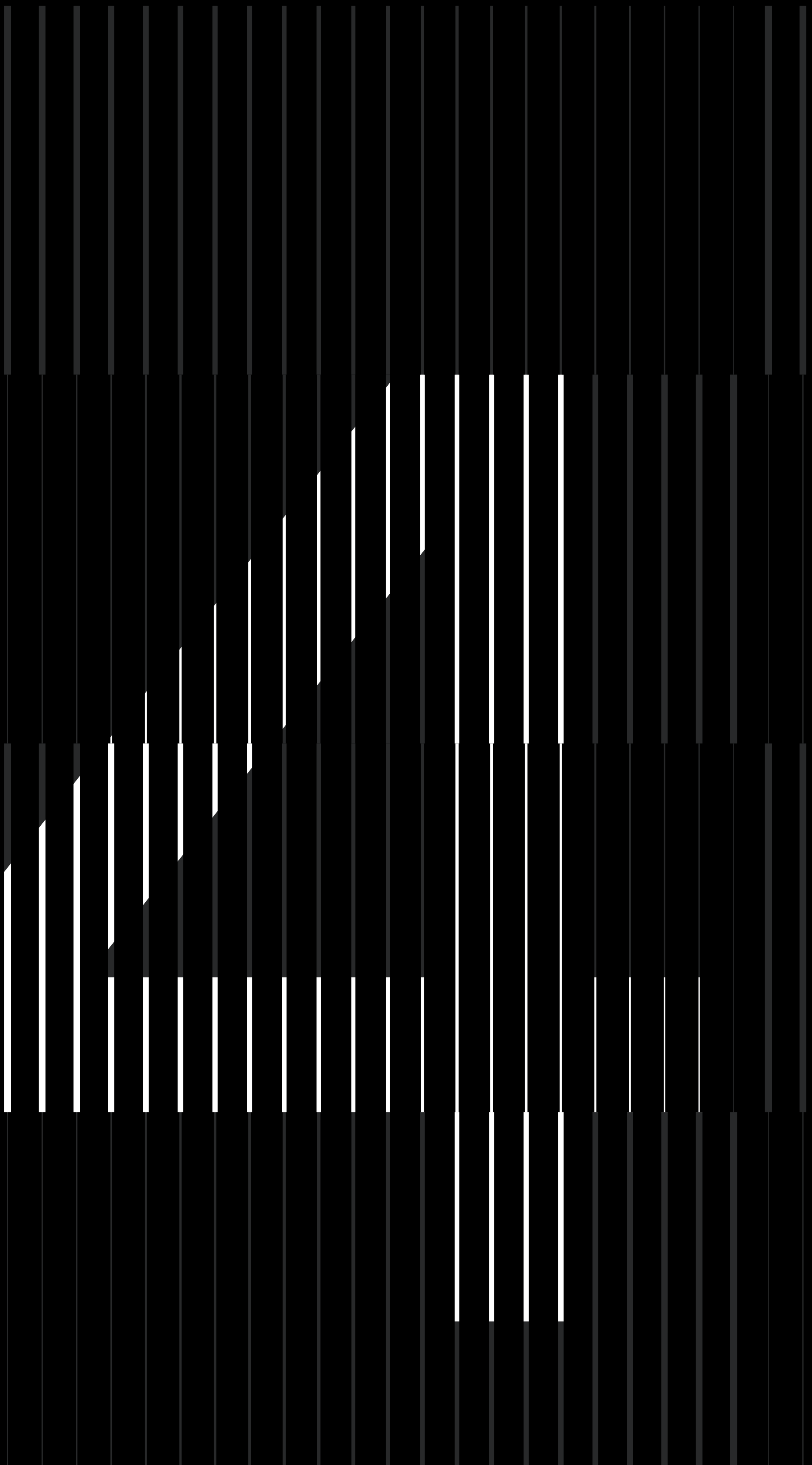


Figure 13

Results from the "Hardware acceleration in ROS 2 and Gazebo survey" (link) question: *"How do you want your accelerators (and accelerated applications) packaged for production use?."*



Benchmarking hardware acceleration

The community survey conducted in both the overall robotics and ROS communities hinted that **62 out of 96 respondents (64,6%) believed that the ROS 2 Perception stack should be prioritized** (section 3.5, Figure 6) from a hardware acceleration perspective. In addition, **46 out of 94 respondents** (48,9%, the most popular option amongst the available) **indicated that speed or latency (shorter execution time) is what they care most about** (section 3.10 , Figure 9). Accordingly, this report's performance benchmarking will focus on reporting around the latency perceived in a series of ROS 2 Perception scenarios.

Source code used to perform these benchmarks is open and available in GitHub. In particular, the ros-acceleration organization contains various related resources including the perception_2nodes meta-package which is used to benchmark ROS 2 graphs. Additional examples used to benchmark ROS 2 Nodes and produce some of these results can be found at acceleration_examples.

Each benchmark studying ROS 2 computational graphs was instrumented with LTTng and was traced during 60 seconds, which was then used to produce comparisons.

4.1 Case study: Hardware Accelerating ROS 2 Perception

Robotics perception helps sense the static and dynamic objects, and build a reliable and detailed representation of the robot's environment using computer vision and machine learning techniques. Data obtained in a robot from its sensors like cameras and LIDAR is typically fed into the perception layer turning into something useful for decision making and planning physical actions. The **perception layer in a robot is thereby responsible for object detection, segmentation and tracking**. Traditionally, **a perception pipeline starts with image pre-processing, followed by a region of interest detector and then a classifier** that outputs detected objects. ROS 2 provides various pre-built Nodes (Components more specifically) that can be used to build perception pipelines easily.

Robotics perception helps sense the static and dynamic objects, and build a reliable and detailed representation of the robot's environment

To benchmark ROS 2 Perception, the following subsections will follow the methodology described in section 2.4. First by analyzing the performance of a simple ROS 2 Graph involving 2 perception pre-processing Nodes and later by measuring the acceleration kernel execution time of various perception operations, including more complex filters. In both cases, measurements will be made to meet the preferences collected during the previous survey capturing runtime execution (or more specifically, the *latency*).

4.2 Benchmarking hardware acceleration in a ROS 2 Perception Graph

The ROS 2 perception graph studied is depicted below and taken from [7]:

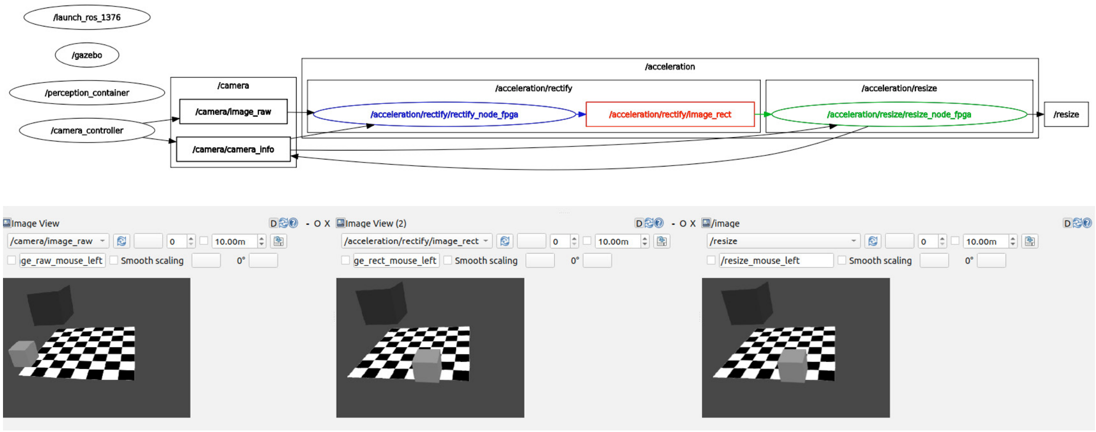


Figure 14

Computational graph of our case study is a ROS 2 Perception graph that leverages the image_pipeline ROS package.The graph contains two ROS 2 perception nodes: (1) RectifyNode which subscribes to the /camera/image_raw and /camera/camera_info topics from Gazebo (which is simulating the camera and the scene) and publishes a rectified image to (2) ResizeNode, which publishes the final resized image.

The following results are obtained while benchmarking this ROS 2 graph following 2.4 with a CPU and with combinations of popular hardware acceleration solutions used in robotics and their frameworks[2]:

[2] For AMD solutions the Kria Robotics Stack (KRS 1.0) and the Vitis Vision Library (2021.2) have been used with a 250 MHz clock while producing accelerators. For NVIDIA solutions, NVIDIA Isaac ROS DP 1.1 packages and the Vision Programming Interface (VPI) 2.1 have been used.

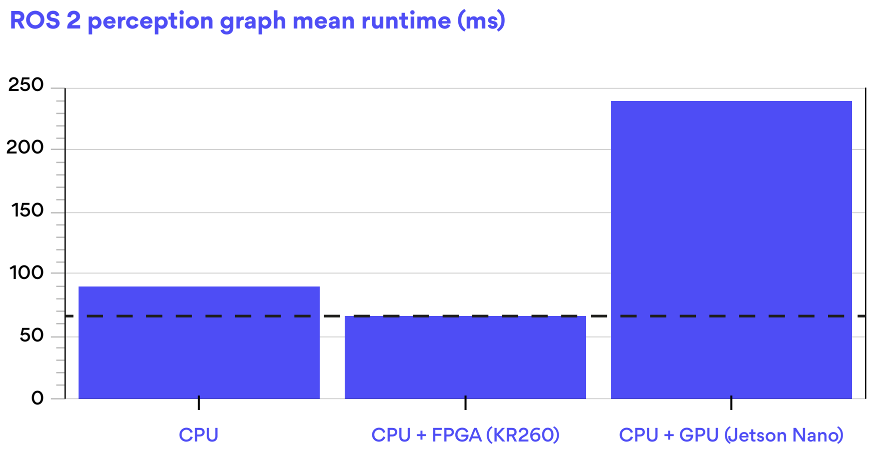


Figure 15

Benchmark of the ROS 2 perception graph of Figure 14 following methodology of section 2.4. Depicted is the mean runtime in milliseconds (ms) of the graph on various compute substrate combinations.

ROS 2 perception graph runtime (ms)	Mean (speedup)	RMS (speedup)
CPU[3]	**91.48** ms	**92.05** ms
CPU[3] + FPGA[4] (AMD's Kria® KR260)	66.82 ms (1.36x)	67.82 (1.35x)
CPU[5] + GPU[6] (NVIDIA's Jetson Nano 2GB)	238.13 ms (▼0.38x)	243.73 (▼0.37x)

An interesting observation can be made while discarding the ROS 2 message-passing infrastructure overhead in the graph and focusing solely on the perception computations:

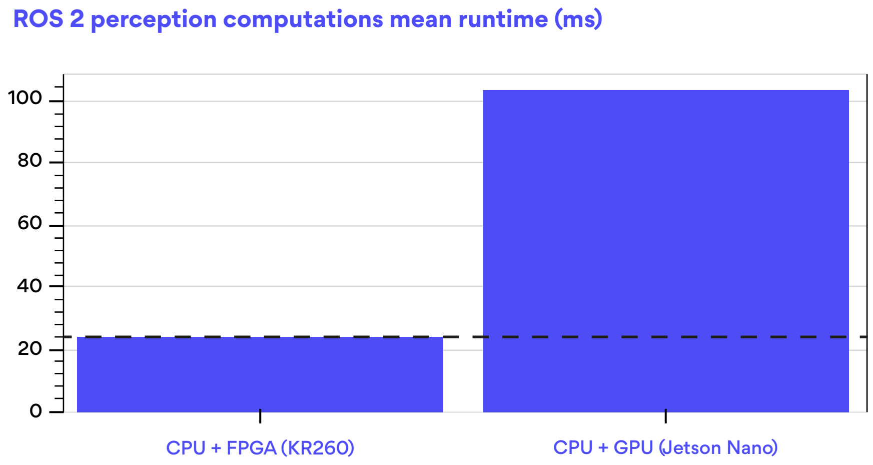


Figure 16

Benchmark of the ROS 2 perception computations of the graph in Figure 14 following methodology of section 2.4. Note that the ROS 2 message-passing infrastructure overheads in the graph have been discarded. Depicted is the mean runtime in milliseconds (ms) of the perception computations on various compute substrate combinations.

[3] Quad-core arm Cortex-A53.

[4] 256K System Logic Cells, 1248 DSPs, 26.6Mb on-chip memory (LUT: 117K, FF: 256K, DSP: 1248, BRAM: 144, URAM: 64)

[5] Quad-core arm Cortex-A57.

[6] 128-core NVIDIA Maxwell™

ROS 2 perception computations (`rectify + resize`) runtime (ms)	Mean (speedup)	RMS (speedup)
CPU + FPGA (AMD's Kria® KR260) - `rectify` **and** `resize` **kernels**	23.90 ms	24.05 ms
CPU + GPU (NVIDIA's Jetson Nano 2GB) - `rectify` **and** `resize` **kernels**	102.29 ms	102.58 ms

when considering more powerful GPUs and CPUs we obtain the following results:

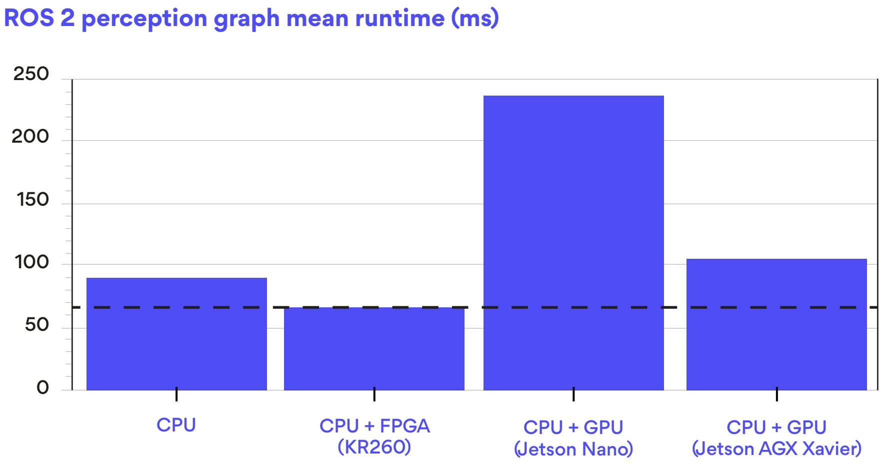


Figure 17

Benchmark of the ROS 2 perception graph of Figure 14 following methodology of section 2.4. Depicted is the mean runtime in milliseconds (ms) of the graph on various compute substrate combinations.

ROS 2 perception graph runtime (ms)	Mean (speedup)	RMS (speedup)
CPU	**91.48** ms	**92.05** ms
CPU + FPGA (AMD's Kria® KR260)	66.82 ms (1.36x)	67.82 (1.35x)
CPU + GPU (NVIDIA's Jetson Nano 2GB)	238.13 ms (▼0.38x)	243.73 (▼0.37x)
CPU[7] + GPU[8] (NVIDIA's Jetson AGX Xavier)	106.34 ms (▼0.86x)	107.30 (▼0.85x)

[7] 8-core ARM v8.2 64-bit CPU

[8] 512-core Volta GPU with Tensor Cores

4.3 Benchmarking hardware acceleration in ROS 2 Perception Nodes

To benchmark hardware acceleration in individual ROS 2 Nodes of the Perception stack we will conduct measurements of the acceleration kernels runtime in milliseconds (ms) using two comparable accelerators (hardware): AMD's Kria KR260 and NVIDIA's Jetson Nano 2GB.

To discriminate between any possible differences between the A53 cores in KR260 and the A57 cores in Jetson Nano, **measurements will discard both the ROS 2 message-passing infrastructure overhead**[9]. In addition, so that performance is more comparable across accelerators, we will **collect data while discarding the host-device (GPU or FPGA) data transfer overhead**[10].

Benchmark results for various robotics perception operations are presented below:

[9] The ROS 2 intra-process and inter-process message-passing overheads are often significant in individual ROS 2 Nodes.

[10] Host (CPU) to device (GPU or FPGA) data transfer often happens over shared memory using various libraries and/or runtimes (VPI, Vitis Vision Library, CUDA, XRT, OpenCL, etc.). We discard these overheads by using device-specific tools that allow introspecting the runtime execution of each kernel for both accelerators. For more details on this refer to [6].

Rectify - 7.34x

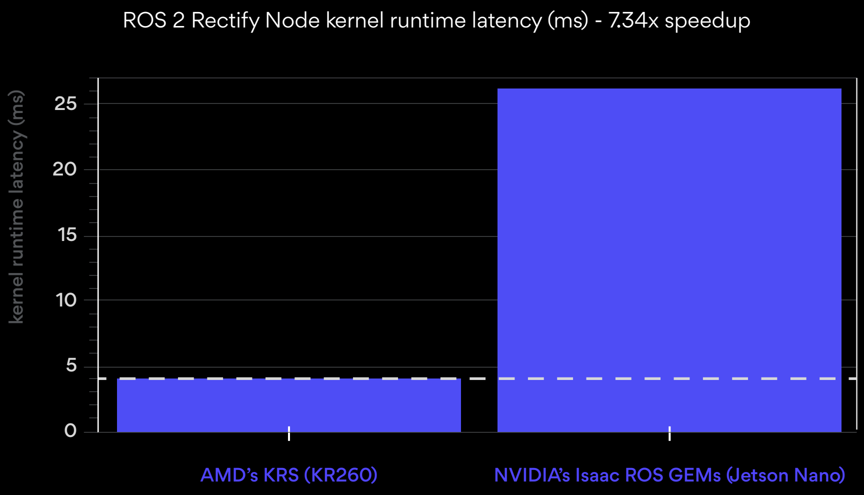


Figure 18

Benchmark of a ROS 2 `rectify` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

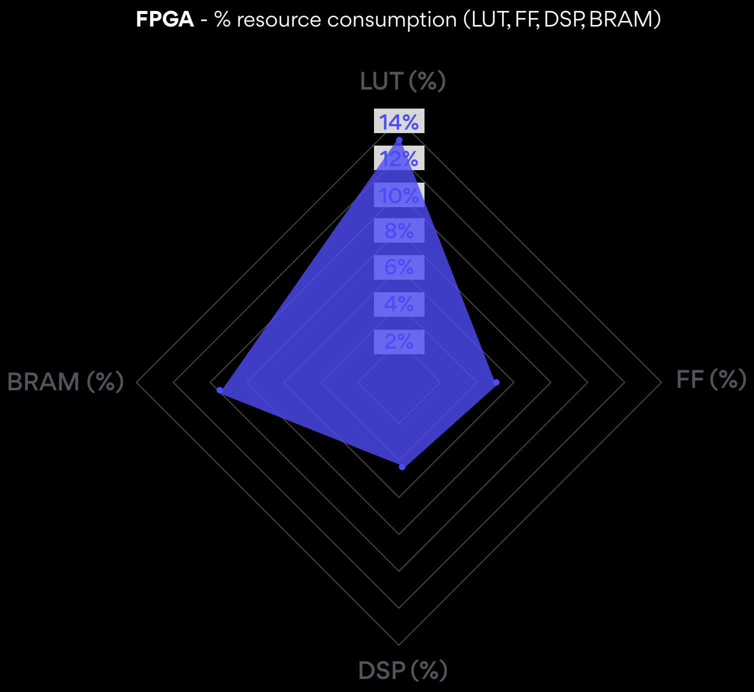


Figure 19

ROS 2 `rectify` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.

Resize - 2.62x

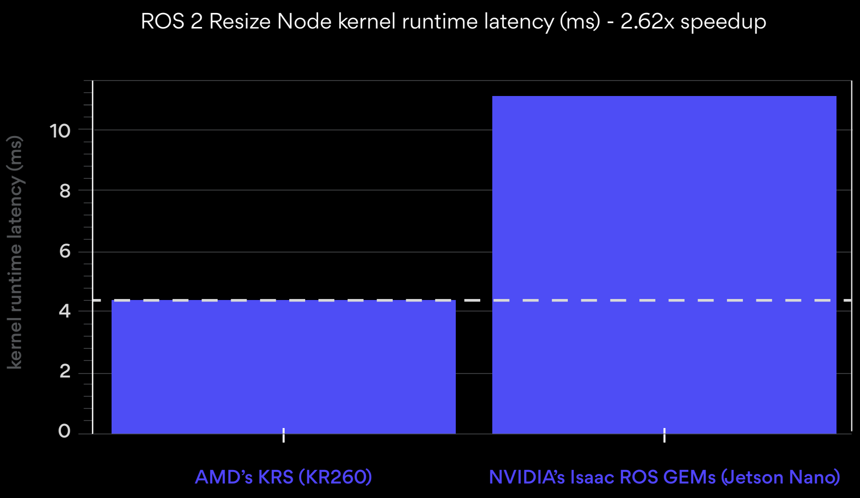


Figure 20

Benchmark of a ROS 2 `resize` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

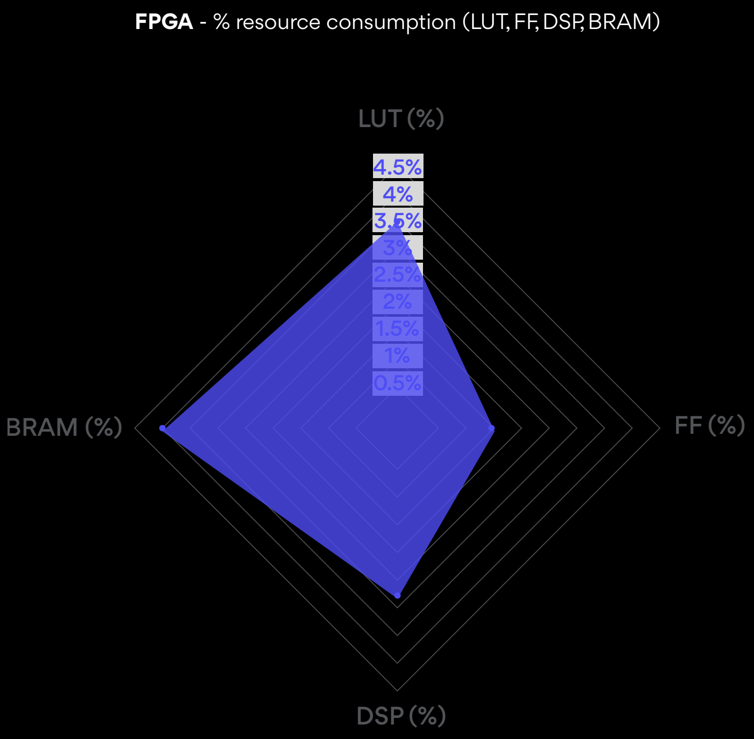


Figure 21

ROS 2 `rectify` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.

Histogram of Oriented Gradients - 509.52x

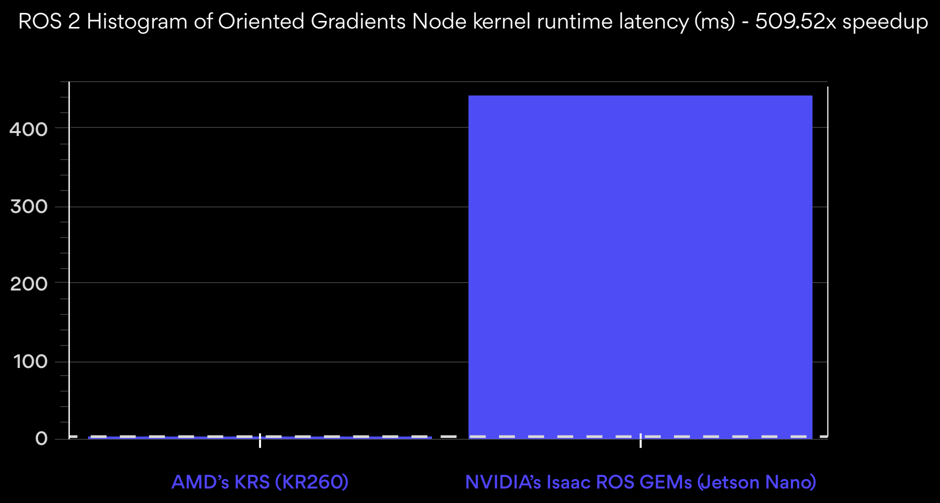


Figure 22

Benchmark of a ROS 2 HOG Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

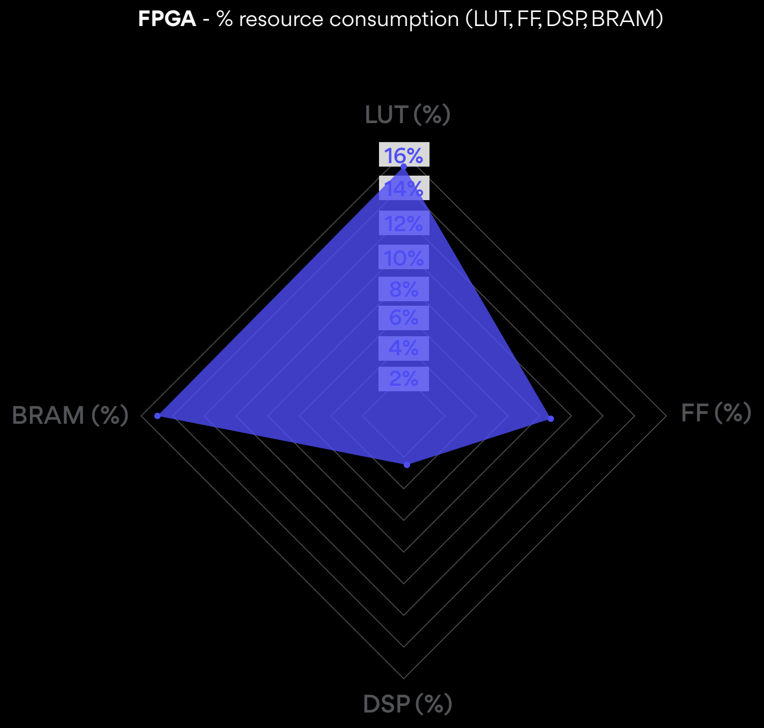


Figure 23

ROS 2 HOG Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.

Figure 24

Benchmark of a ROS 2 `Harris` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

Harris - 30.27x

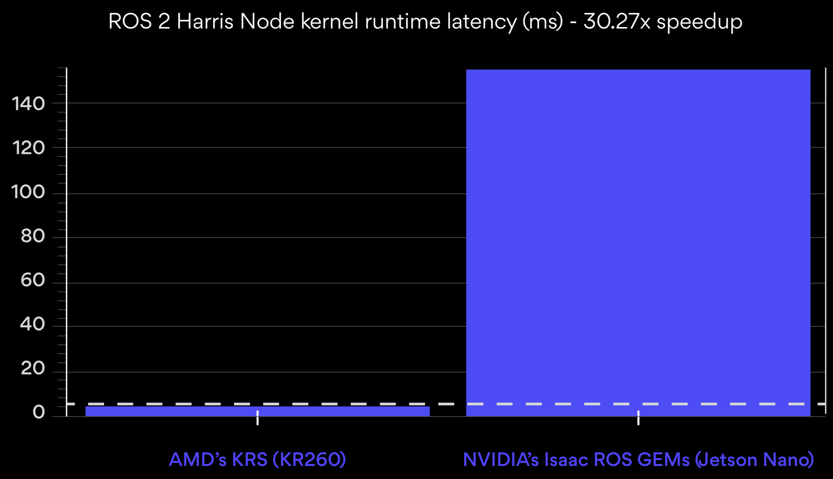


Figure 25

ROS 2 `Harris` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.

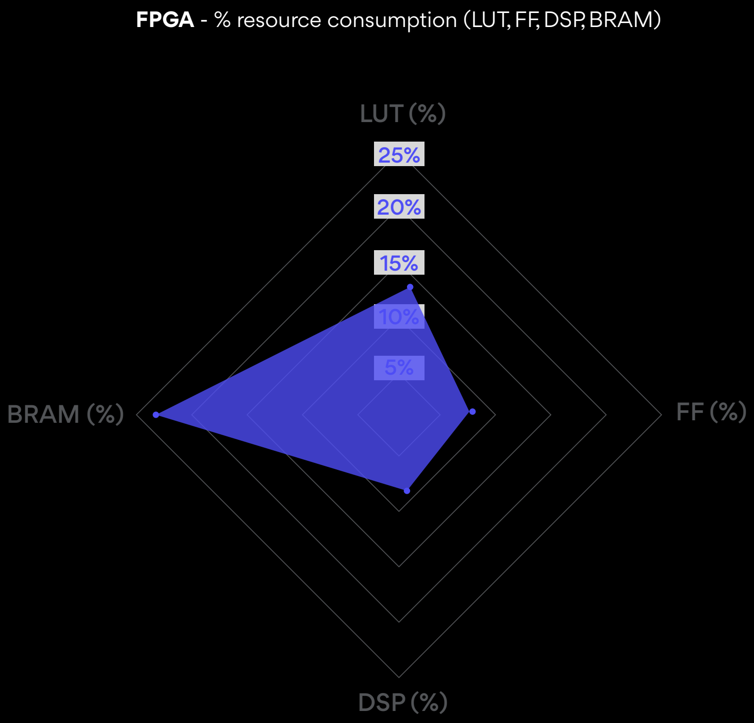


Figure 26

Benchmark of a ROS 2 `Canny Edge` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

Canny Edge Tracing - 3.26x

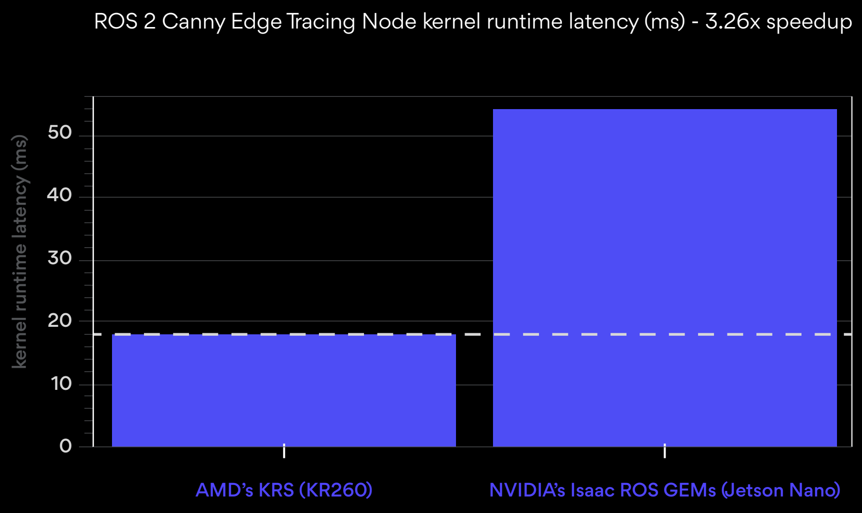


Figure 27

ROS 2 `Canny Edge` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.

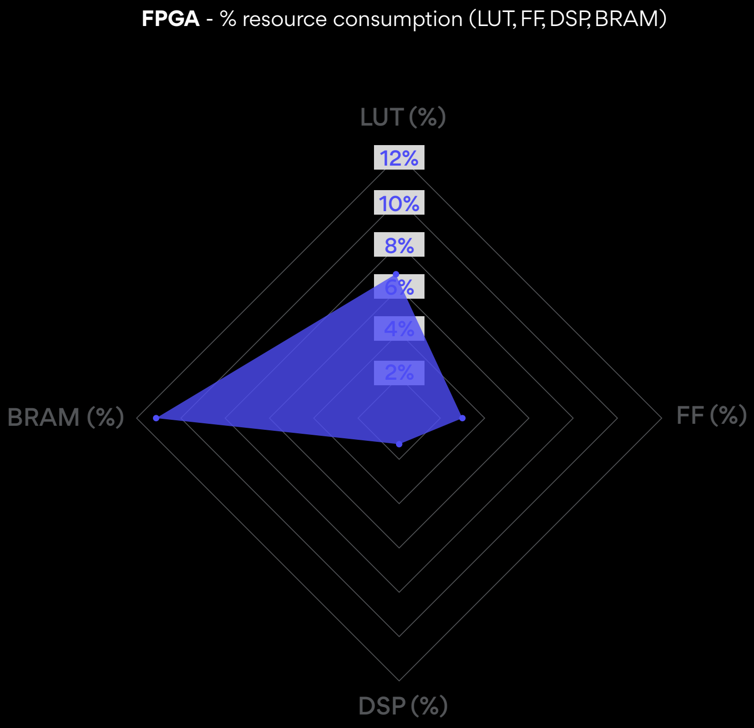


Figure 28

Benchmark of a ROS 2 `Fast Corner Detection` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIAJetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

Fast Corner Detection - 8.43x

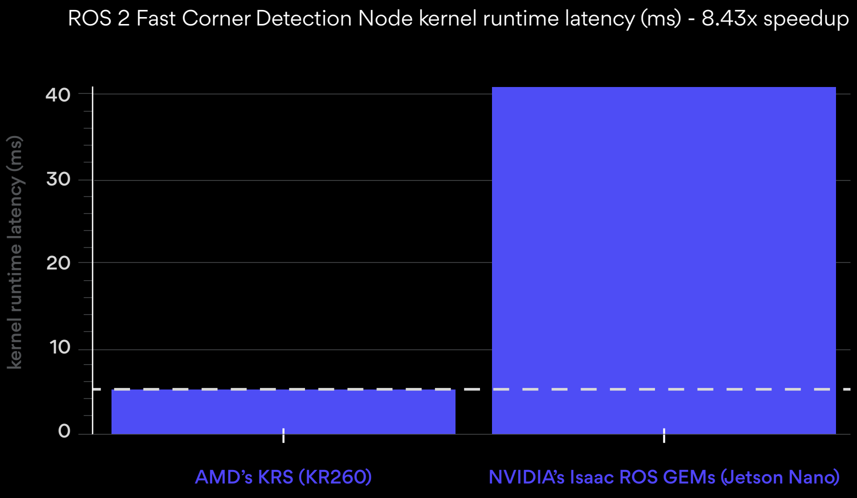


Figure 29

ROS 2 `Fast Corner Detection` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.

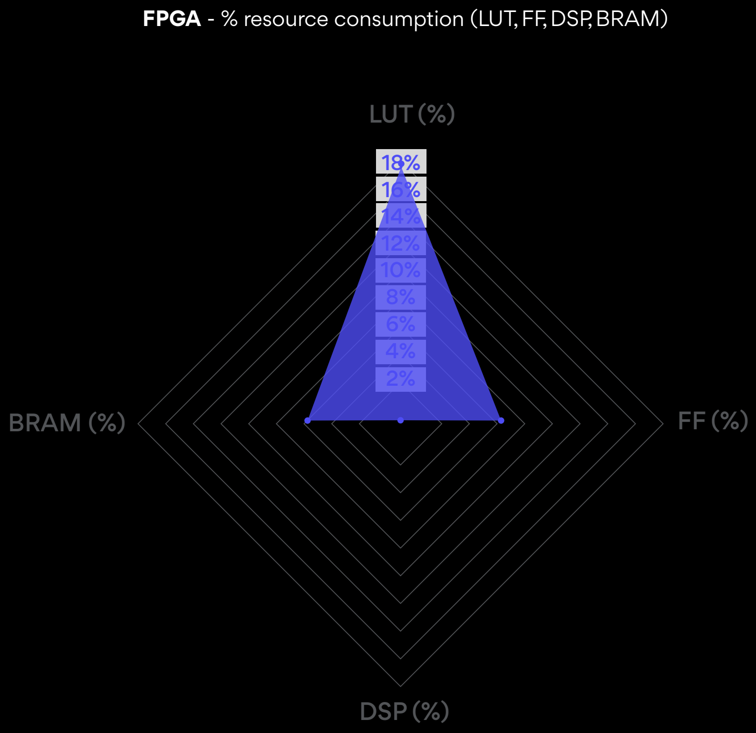


Figure 30

Benchmark of a ROS 2 `Gaussian Difference` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead.

Gaussian Difference - 11.94x

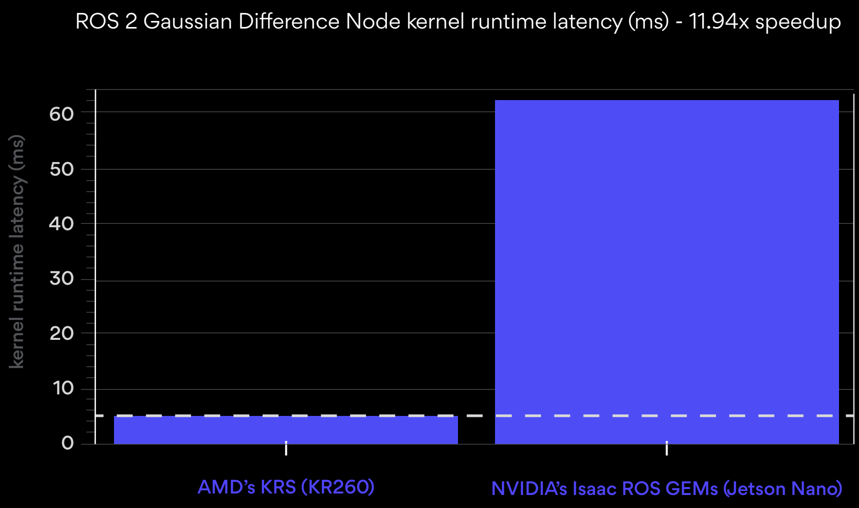
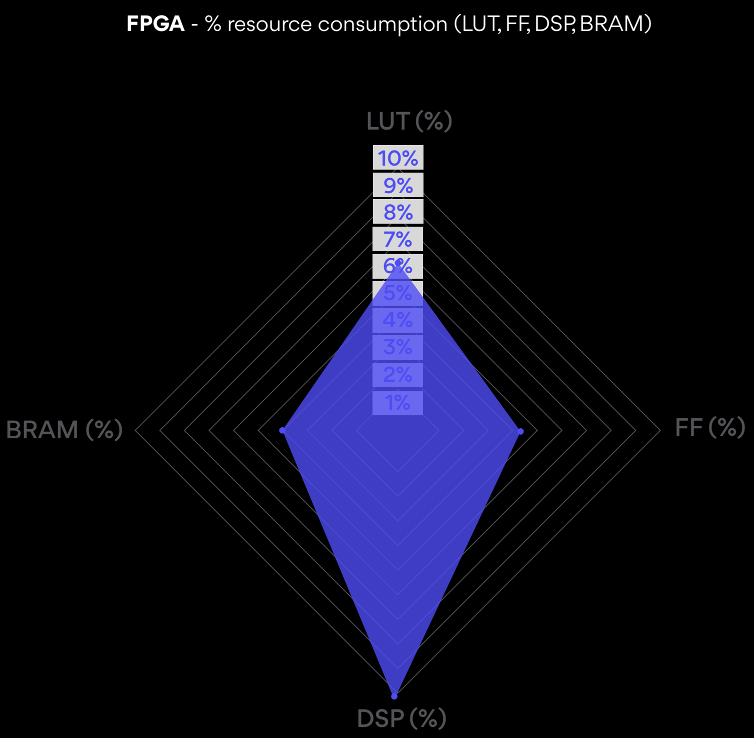


Figure 31

ROS 2 `Gaussian Difference` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.



Bilateral Filter - 9.33x

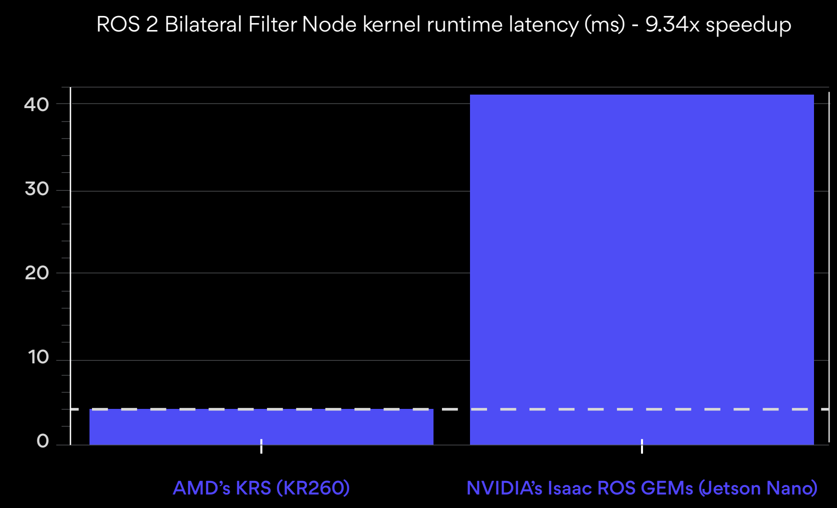


Figure 32

Benchmark of a ROS 2 `Bilateral Filter` Node acceleration kernel runtime latency (ms) running on an AMD KR260 and in an NVIDIA Jetson Nano 2GB. So that performance is comparable across accelerators and for the particular perception function, measurements discard the ROS 2 message-passing infrastructure overhead and the host-device (GPU or FPGA) data transfer overhead..

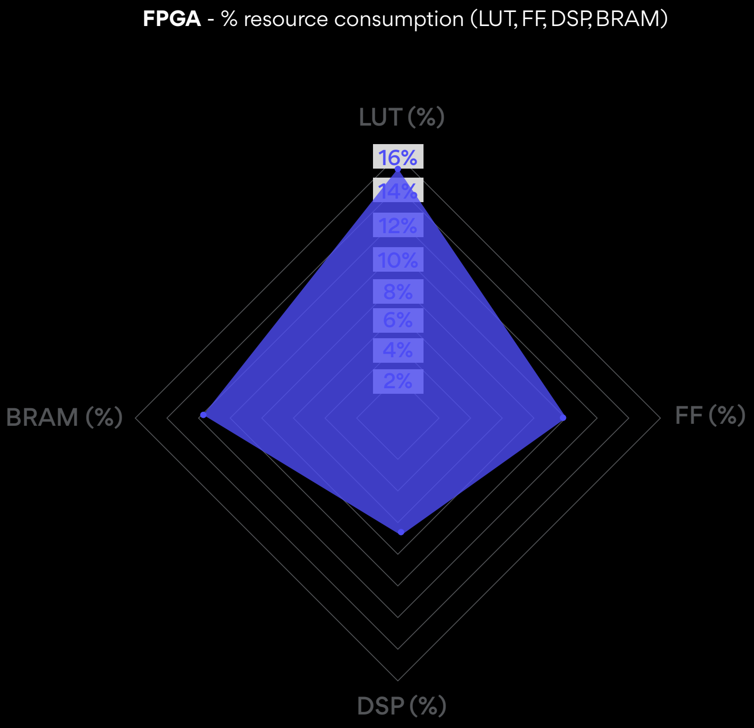
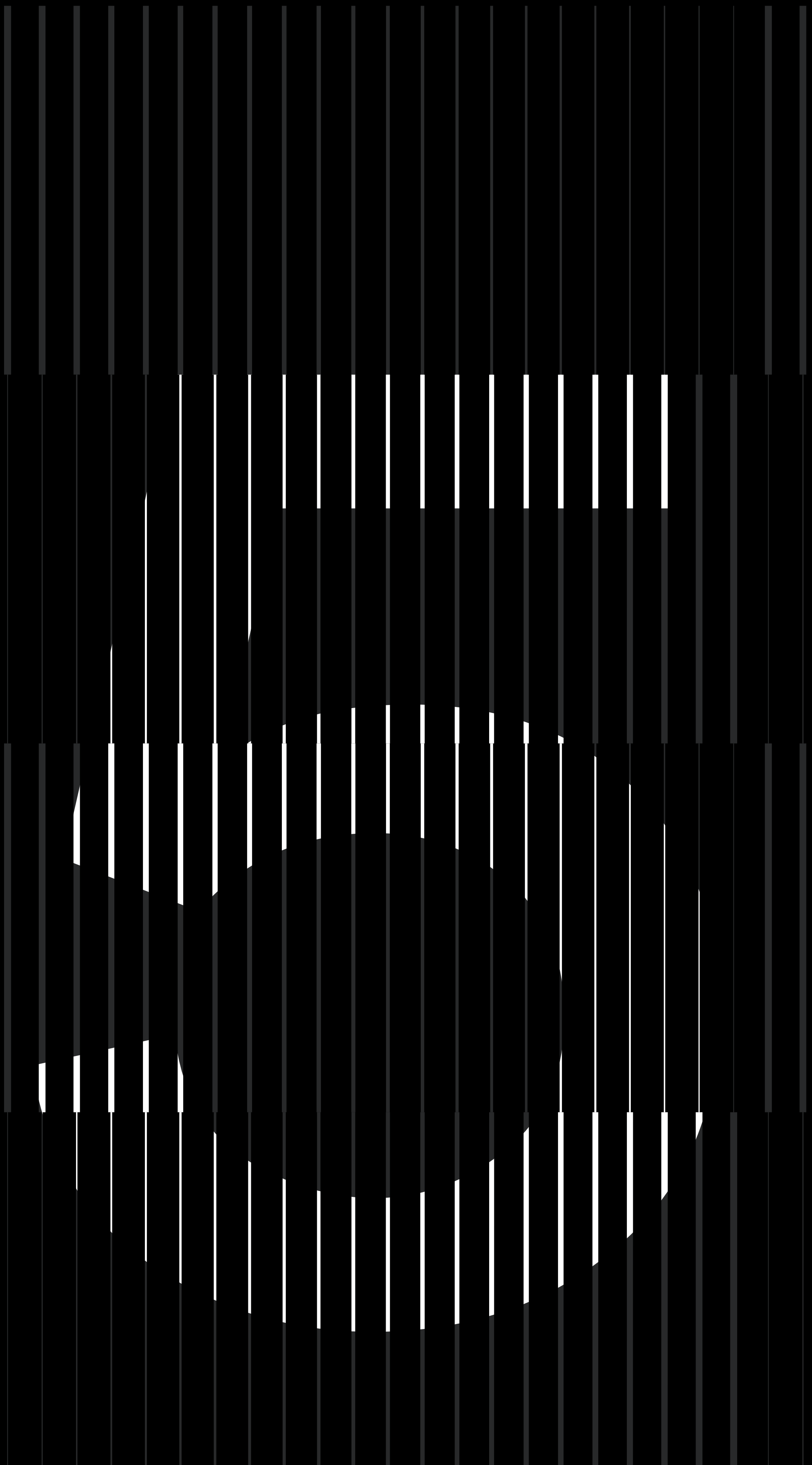


Figure 33

ROS 2 `Bilateral Filter` Node acceleration kernel resource consumption in the FPGA (%) and relative to LUTs, FFs, DSPs and BRAM.



Discussion

The ROS 2 Hardware Acceleration Working Group was born in 2021 and its monthly gatherings have managed to engage with a wide community of roboticists, attracting many users (from both industry and academia) to ROS 2 and its use. With plenty of attendance in each one of its meetings[12] and with the participation of multiple silicon vendors, the group is arguably one of the most active ones in the ROS robotics domain.

> The success of the ROS 2 Hardware Acceleration Working Group during 2021 and 2022 suggests that there's an increasingly evident interest in the use of hardware acceleration solutions in robotics

The success of the ROS 2 Hardware Acceleration Working Group during 2021 and 2022 suggests that there's an increasingly evident interest in the use of hardware acceleration solutions in robotics, however, the community survey conducted hints that **only about half of the respondents** (**51%**, section 3.1) **is confident about the value and differences between hardware acceleration solutions for ROS 2 and Gazebo**. Moreover, only **62.5%** (section 3.2) **have ever programmed an acceleration kernel**, the majority of which used NVIDIA CUDA (80.3%, section 3.4). This suggests that there's still a lot of work to be done from silicon vendors' side to further simplify the use and integration of their solutions in the ROS robotics ecosystem and provide comprehensive documentation. The previous statement is confirmed by section 3.3 which highlights what **aspects of hardware acceleration are of most relevance to roboticists using ROS**. Unsurprisingly, we find that more than half of the respondents (52.1%) indicate that a simpler integration with ROS 2 and its ecosystem of tools is of most relevance to them:

- → **52.1%** - Integration with ROS 2 (`ament` build system *11.5%*, `colcon` build tools *19.8%* and acceleration firmware *20.8%*)
- → **32.3%** - Capabilities to easily switch between hardware accelerated and CPU-centric Nodes
- → **11.5%** - Benchmarking capabilities for hardware acceleration
- → **4.1%** - Others

[12] See https://github.com/ros-acceleration/community#meetings for a community-maintained list of meeting minutes and recordings.

When looking at which packages or components of ROS 2 and Gazebo roboticists would like to accelerate first (section 3.5, *multiple selections allowed*), we find that the **ROS 2 Perception stack with a 64.6% is the most demanded group of packages to be accelerated.** This is closely followed by "Gazebo physics engines" (60.4%), navigation2 (40.6%), "DDS communication middleware" (30.2%), MoveIt 2 (21.9%), ROS 2 networking stack (UDP/IP/Ethernet, 20.8%) and the ROS 2 control stack (19.8%). These numbers are also confirmed by individual answers provided in sections 3.6 and 3.7.

> There's still a lot of work to be done from silicon vendors' side to further simplify the use and integration of their solutions in the ROS robotics ecosystem

The **majority of the respondents (92.7%) indicated that they'd prefer the commercially friendly Apache 2.0 license** for hardware acceleration resources and libraries (section 3.8). When asked about the format of acceleration kernels, opposed to 18.9% which would be fine with just binaries, **74.8% would prefer source code access to acceleration kernels with code examples** (section 3.9).

One surprising aspect encountered while conducting the survey is that **roboticists seem to care about speed or *latency*** (48.9%, shorter execution time) **as much as *real-time* and *determinism*** (46.8%). Only a reduced 4.3% would prioritize power consumption as indicated in section 3.10. This is somewhat counter-intuitive when looking at how the **majority of the roboticists currently use GPUs** (**69.8%**, section 3.11) **versus FPGAs (21.9%)**, since after all, it's widely accepted that FPGAs outperform GPUs and CPUs while delivering real-time and determinism in computations, and with lower power consumption. To add to this conflict, section 3.14 hints that **hardware acceleration is mostly used to create battery-powered robots (and thereby power-sensitive)** with roboticists creating autonomous mobile robots (AMRs, 36.4%) followed by drones (19.8%), industrial robotic arms (12.5%) and legged robots (5.2%). **This indicates that there's margin for FPGA usage growth in the ROS robotics community.**

The numbers of type of accelerator usage (section 3.11) are coherent with the most popular commercial solutions (section 3.13) with NVIDIA Jetson AGX Xavier being the leading solution (33.3%) followed by both the NVIDIA Jetson Nano (15.6%) and the AMD's Zynq UltraScale embedded portfolio (15.6%, including Kria® boards).

> Roboticists seem to care about speed or latency (48.9%, shorter execution time) as much as real-time and *determinism* (46.8%)

Finally, Ubuntu seems to be the dominant (79.5%, section 3.15) operating system requested by ROS roboticists for hardware acceleration. Ubuntu 20.04 is the preferred option (59%) versus Ubuntu 22.04 (20.5%). Yocto-based rootfs are the second most popular choice with 7.7% of the respondents preferring it. As for packaging mechanisms, .deb files are the preferred option (59%, section 3.16) followed by Docker containers (23.1%).

The results obtained across benchmarks performed on a simple pre-processing ROS 2 perception graph that focus on capturing speed or latency (time between the start and the completion of a task) show that optimized FPGA accelerators outperform their GPU counterparts, even when using powerful GPUs. Considering mean runtime measurements (in ms, Figure 15, section 4.2), the use of CPU + FPGA[13] deliver a **3.56x speedup** over a comparable CPU + GPU[14], and a **1.36x speedup** over a comparable CPU[15]. When considering a more powerful CPU + GPU[16] combination (Figure 17, section 4.2), the FPGA still outperforms it with a 1.59x speedup.

To further study these results and to discriminate between any possible differences between the two CPUs used (A53 cores in KR260 and the A57 cores in Jetson Nano), Figure 16 (section 4.2) discards the ROS 2 message-passing infrastructure overhead and reports on the perception computations in the graph. Results show how perception computations in the FPGA have a 4.27x speedup relative to their GPU counterparts when running in the Jetson Nano.

Latency results obtained across benchmarks performed on a ROS 2 perception graph show that optimized FPGA accelerators outperform their GPU counterparts, even when using powerful GPUs

Performance improvements in the form of latency with dedicated acceleration kernels in FPGAs are further evident when considering the power domain. Figure 34 depicts the performance-per-watt of the ROS 2 Perception graph studied in section 4.2 and across all the accelerators considered before:

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

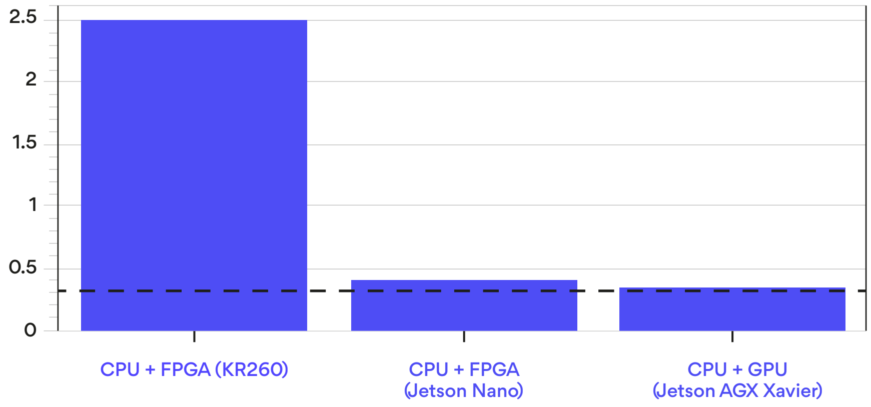


Figure 34

Performance-per-watt benchmark of a simple ROS 2 perception graph across various accelerators. The computational graph studied is described in section 4.2.

[13] AMD Kria® KR260

[14] NVIDIA Jetson Nano 2GB

[15] Quad-core arm Cortex-A53.

[16] NVIDIAJetson AGX Xavier

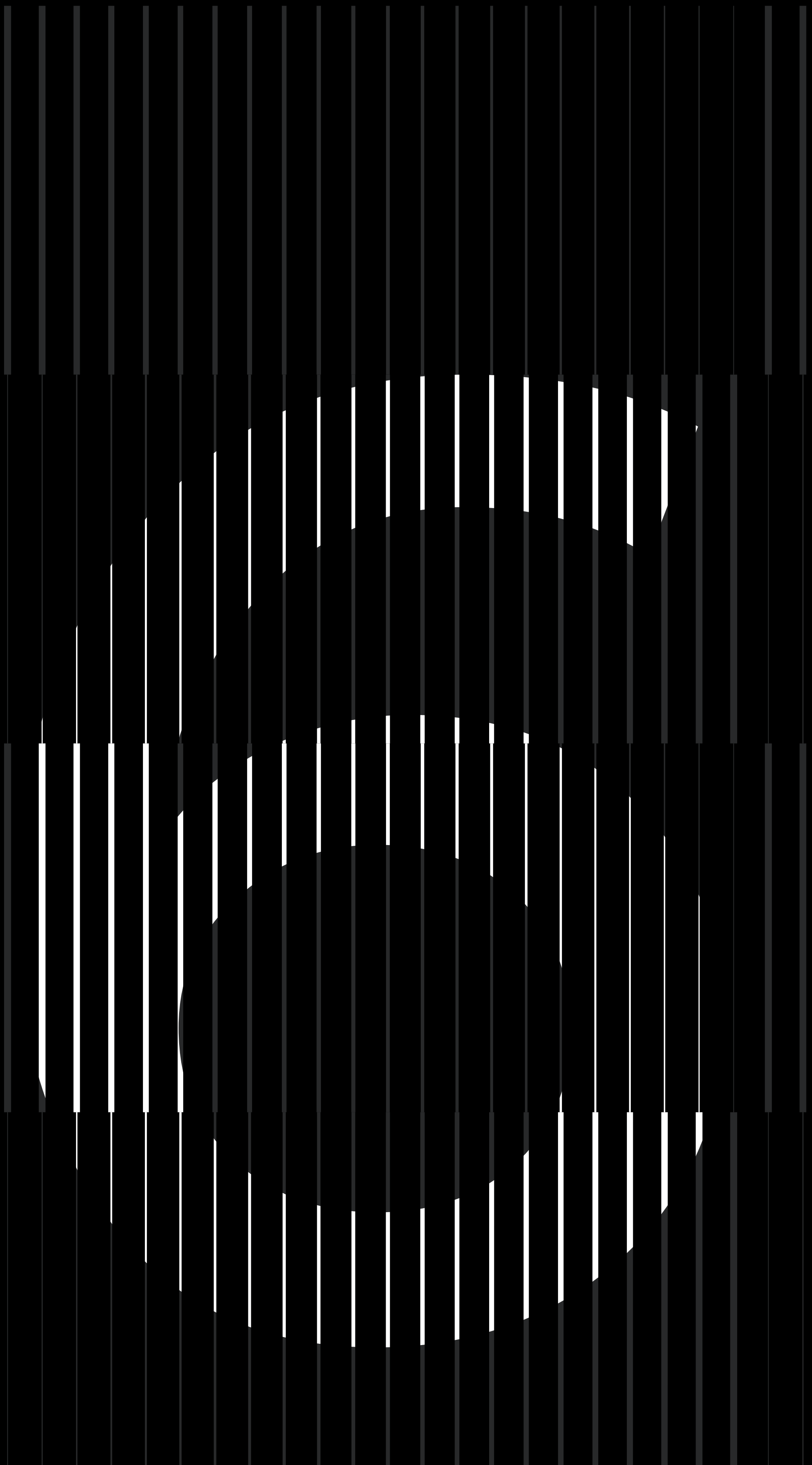
While measuring power consumption in a ROS 2 perception graph, we observe that the FPGA designs are much more power efficient than their GPU counterparts. The KR260 presents a performance-per-watt figure that's 6x (5.93x) better than the one in the Jetson Nano and 8x (7.95x) better than the one in the Jetson AGX Xavier. An interesting observation can be made here comparing the performance-per-watt results obtained from the Jetson Nano and the Jetson AGX Xavier: ***the Xavier features a more powerful CPU and GPU, which consumes more energy while performing computations, however the latency performance of these computations do not scale in the same manner as the energy consumption.*** What these results hint is that the rate at which the energy consumption grows with NVIDIA Jetson (CPU + GPU) solutions seems to be smaller than the rate at which the latency performance improves, which leads to a decaying performance-per-watt in our ROS 2 perception measurements. This statement links back to the rule of thumb shared in section 2.3 that emphasized how *"bandwidth grows by at least the square of the improvement in latency".* With GPUs often focusing on bandwidth to measure performance, when considering latency as the measure of performance GPUs struggle to find themselves on equal footing with their FPGA counterparts.

The rate at which the energy consumption grows with NVIDIA Jetson (CPU + GPU) solutions seems to be smaller than the rate at which the latency performance improves, which leads to a decaying performance-per-watt in our ROS 2 perception measurements

There are nevertheless various advantages that GPUs inherently have and that should be considered while building complex robotic computations. Moreover, though FPGA kernel runtime execution outperforms their GPU counterparts, it's relevant to note that FPGAs are resource-limited and thereby it's important to consider that only a fixed set of accelerators would be able to fit within an FPGA at any given point in time whereas the GPUs don't have this limitation due to their architectures.

ROS 2 Nodes running in an FPGA outperform those running in a GPU by relevant speedups. All the way up to 500x in popular perception algorithms such as the Histogram of Oriented Gradients (HOG)

Section 4.3 further dives into this and focuses on studying perception performance in individual ROS 2 Nodes while estimating resources required in an FPGA. To do so, it isolates perception computations by discarding both the ROS 2 message-passing infrastructure overhead, as well as the host-device (GPU or FPGA) data transfer overhead. Results depicted over figures 18-33 indicate that perception ROS 2 Nodes running in an FPGA outperform those running in a GPU by relevant speedups. All the way up to 500x (Figure 22) in popular perception algorithms such as the Histogram of Oriented Gradients (HOG).



Hardware Acceleration solutions for the robotics architect

Acceleration Robotics is amongst the top experts globally on the Robot Operating System (ROS), including ROS and ROS 2 computational graphs. Our hardware acceleration efforts are accelerator-agnostic (FPGAs or GPUs) and robot-agnostic. We focus on what works best to improve robotics computations. Our work is well known, widely distributed and used.

The following solutions are meant to help robotics architects design specialized robot compute architectures and streamline various robotic processes using open source including ROS and Gazebo, so that you don't spend time reinventing the wheel and re-developing what already works.

Robotic Processing Units (RPUs)

Robotic Processing Units (RPUs) are robot brains, processing units for robots that map efficiently robot behaviors (programmed as ROS computational graphs) to underlying compute resources. They empower robots with the ability to react faster, consume less power, and deliver additional real-time capabilities.

	Name	Description
	ROBOTCORE™	A robot-specific processing unit specialized in ROS computations. Features 16x CPUs, a GPU and an FPGA. This is the processing unit for the robotics architect targeting autonomous mobility, industrial manipulation and healthcare robotics applications.

Services

The following consulting services help rapidly augment your engineering capabilities with a robotics deep domain expertise.

	Name	Description
	Robotics consulting	Hardware acceleration framework for ROS and ROS 2 extending support for leading FPGAs and GPUs.
	Robot FPGA and GPU IP design services	Tools to speed-up ROS 2 graphs with the cloud, and in the cloud. Helps roboticists launch parts of their ROS 2 computational graphs into the cloud leveraging CPU, FPGA and/or GPU cloud instances.

Tools and Robot IP Cores

ROS 2 API-compatible hardware acceleration tools and robot Intellectual Property (IP) cores (**robot cores**). Increase your robot's performance, including latency, throughput and power efficiency.

	Name	Description
	ROBOTCORE™ Framework	Hardware acceleration framework for ROS and ROS 2 extending support for leading FPGAs and GPUs.
	ROBOTCORE™ Cloud	Tools to speed-up ROS 2 graphs with the cloud, and in the cloud. Helps roboticists launch parts of their ROS 2 computational graphs into the cloud leveraging CPU, FPGA and/or GPU cloud instances.
	ROBOTCORE™ Perception	Accelerated ROS 2 robotics perception stack. API-compatible with the ROS 2 perception stack.
	ROBOTCORE™ Transform	Accelerated ROS 2 coordinate transformations (tf2). API-compatible with the ROS 2 transform (tf2) library

References

[1]: Liu, S., Zhu, Y., Yu, B., Gaudiot, J. L., & Gao, G. R. (2021). The Promise of Dataflow Architectures in the Design of Processing Systems for Autonomous Machines. arXiv preprint arXiv:2109.07047.

[2]: Hennessy, J. L., & Patterson, D. A. (2011). *Computer architecture: a quantitative approach*. Elsevier.

[3]: Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J.,... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

[4]: Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66), eabm6074.

[5]: A. Pemmaiah, D. Pangercic, D. Aggarwal, K. Neumann, K. Marcey, "Performance Testing in ROS 2". https://drive.google.com/file/d/15nX80RK6aS8abZvQAOnMNUEgh7px9V5S/view

[6]: REP-2008 - ROS 2 Hardware Acceleration Architecture and Conventions https://github.com/ros-infrastructure/rep/pull/324

[7]: Victor Mayoral-Vilches, Sabrina M. Neuman, Brian Plancher, and Vijay Janapa Reddi. Forthcoming. "RobotCore: An Open Architecture for Hardware Acceleration in ROS 2." In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. Preprint