2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

FogROS2: An Adaptive Platform for Cloud and Fog Robotics Using ROS 2

Jeffrey Ichnowski*[1A,3], Kaiyuan Chen*[1], Karthik Dharmarajan[1A], Simeon Adebola[1A], Michael Danielczuk[1A], Víctor Mayoral-Vilches[4,5], Nikhil Jha[1A], Hugo Zhan[1A], Edith LLontop[1A], Derek Xu[1A], Camilo Buscaron, John Kubiatowicz[1], Ion Stoica[1], Joseph Gonzalez[1], and Ken Goldberg[1,2,A]

***Abstract*— Mobility, power, and price points often dictate that robots do not have sufficient computing power on board to run contemporary robot algorithms at desired rates. Cloud computing providers such as AWS, GCP, and Azure offer immense computing power and increasingly low latency on demand, but tapping into that power from a robot is non-trivial. We present FogROS2, an open-source platform to facilitate cloud and fog robotics that is included in the Robot Operating System 2 (ROS 2) distribution. FogROS2 is distinct from its predecessor FogROS1 in 9 ways, including lower latency, overhead, and startup times; improved usability, and additional automation, such as region and computer type selection. Additionally, FogROS2 gains performance, timing, and additional improvements associated with ROS 2. In common robot applications, FogROS2 reduces SLAM latency by 50 %, reduces grasp planning time from 14 s to 1.2 s, and speeds up motion planning 45x. When compared to FogROS1, FogROS2 reduces network utilization by up to 3.8x, improves startup time by 63 %, and network round-trip latency by 97 % for images using video compression. The source code, examples, and documentation for FogROS2 are available at `https://github.com/BerkeleyAutomation/FogROS2`, and is available through the official ROS 2 repository at `https://index.ros.org/p/fogros2/`.**

I. Introduction

It is difficult for the onboard computing resources of robots to keep up with advances in robot algorithms and new computing hardware. Cloud computing offers on-demand access to immense computing resources and new and power-hungry computing platforms, such as GPUs, TPUs, and FPGAs. Prior work [1] showed that using the cloud for intensive computing in robotics can be practical and cost-effective. However, gaining access to evolving cloud computing resources requires expertise with many new and emerging software packages, and experience handling data security and privacy. In prior work [2], we introduced FogROS (henceforth *FogROS1*), a framework that extends the Robot Operating System (ROS) (henceforth *ROS 1*) to enable quick access to the cloud.

*Equal Contribution
[1]Department of Electrical Engineering and Computer Science
[A]The AUTOLab at UC Berkeley (automation.berkeley.edu).
[2]Department of Industrial Engineering and Operations Research
[1,2]University of California, Berkeley, CA, USA
[3]Robotics Institute, Carnegie Mellon University
[4]Acceleration Robotics, Ecuador 3, 1 I, Vitoria, Álava, Spain,
[5]System Security Group, Universität Klagenfurt, Universitätsstr. 65-67 9020 Klagenfurt, Austria
jeffi@cmu.edu, kych@berkeley.edu

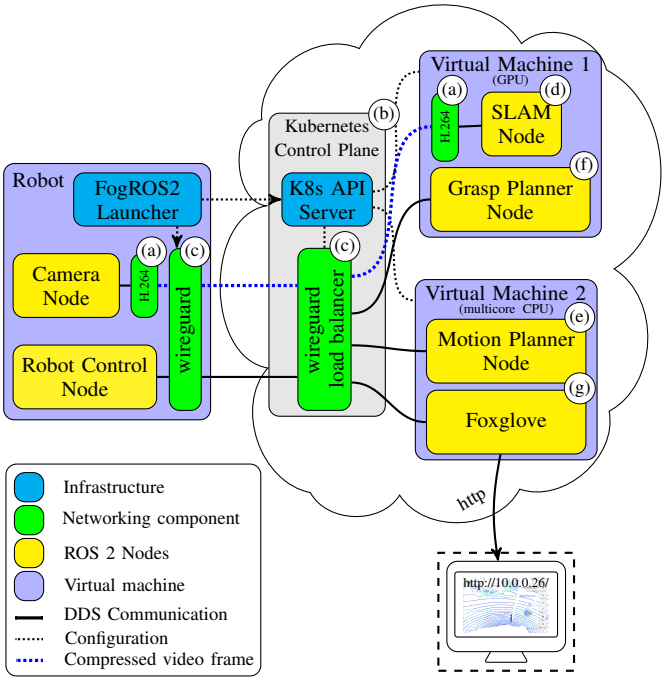

Fig. 1: FogROS2 addresses several deficiencies found in FogROS1, including latency, usability, and automation. For example, transmitting images from a camera to the cloud can require a lot of bandwidth, thus increasing latency. (a) FogROS2 transparently compresses video and image streams using the popular H.264 compression standard. FogROS2 further improves upon FogROS1 by speeding up launch times, improving scalability, and increasing compatibility by (b) migrating to Kubernetes (K8s), (c) switching to UDP over Wireguard, and updating to the ROS 2 ecosystem. In an example application, (d) FogROS2 moves a simultaneous localization and mapping (SLAM), (e) motion planning, (f) and grasp planning nodes to the cloud, taking advantage of the GPU and high CPU core counts available there to speed up processing. (g) By integrating Foxglove, users are also able to monitor robots from a browser any where in the world.

However, FogROS1 has limitations in latency, usability, and automation. In this paper, we introduce FogROS2 to reduce latency, improve usabillity, and automate additional components of launching robot code in the cloud, while extending to additional potential robot applications. Furthermore, we rewrote FogROS2 from scratch to fully integrate with ROS 2 to benefit from improvements in networking, launch configurability, and its command line interface; and we added integration points to Foxglove [3] to enable *remote* monitoring from anywhere in the world.

Latency, the time between when an event occurs and when the robot reacts to the event, is a critical factor in many

applications. FogROS1 and FogROS2 demonstrate that using the cloud can reduce latency of complex computations. (It is important to note that this is not a universal statement—some computation, such as feedback control loops, time-bounded, or safety-critical computations are not always suitable for the cloud due to unpredictable network round-trip times.) FogROS1 suffered from long cloud computer startup times (around 4 minutes) and high round-trip network times, particularly for images (e.g., 5 seconds per image). FogROS2 lowers these latencies by using application-specific cloud-computer images, using a Kubernetes backend to avoid overhead associated with creating new cloud computers, switching from TCP to UDP secured networking, and adding transparent H.264 video compression for image topics.

FogROS2 includes several usability and automation improvements over FogROS1 to facilitate adoption, including: (1) a command-line interface (CLI) to interact with FogROS2 cloud computers, (2) integration with Foxglove to enable remote monitoring, (3) a new launch process to allow for automating cloud-computer specification and region, and (4) creation of custom cloud-computer images.

Many of these improvements are facilitated by re-writing FogROS2 for ROS 2. ROS 2 [4], a rapidly growing replacement for ROS 1 [5], is a standard for developing robot applications. FogROS1 and FogROS2 enable moving computationally intensive parts (or *nodes*) of a computational graph to the cloud and securing communication channels for messages, all with a few small changes to the launch script and without changing a line of the robot code. By migrating to ROS 2, the FogROS2 launch system gains additional capabilities, such as: detecting the cloud server region nearest to the robot and automatically selecting computers and images based computational requirements. FogROS2 is now part of the official ROS 2 ecosystem, and installable with standard Ubuntu commands (`apt install ros-humble-fogros2`).

In 3 example applications, visual SLAM, grasp planning, and motion planning, we evaluate the ability of FogROS2 to reduce total computation times. We find that using cloud computers via FogROS2 speeds up the computation and can reduce compute times by 2x to 45x. Comparing to FogROS1, FogROS2 improves startup times by 63 %, and network latency by 97 % for images.

FogROS2 contributes 9 improvements over FogROS1 [2]: (1) FogROS2 extends the ROS 2 launch system introducing additional syntax in launch files that allow roboticists to specify at launch time which components of their architecture will be deployed to the cloud, and which ones on the robot. While FogROS1 existed outside the official ROS ecosystem, FogROS2 directly integrates with it. (2) FogROS2 provides launch script logic that allows robots to automate selection of cloud-computing resources, such as nearest region, computer image, and computer type. (3) FogROS2 adds support for streaming video compression between robot and cloud nodes—significantly improving the performance of image processing in the cloud, and potentially enabling new applications. (4) The architecture of FogROS2 is extensible, making it easy to plug in support for additional cloud computing providers, Data Distribution Service (DDS) providers (Sec. III), and message compression. (5) FogROS2 integrates with ROS 2 tooling and provides ROS 2 command-line interfaces to query and control cloud-robotics deployments. (6) FogROS2 interfaces with the new Foxglove web-based robot visualization software [3] to allow remote (anywhere-in-the-world) monitoring of FogROS2 applications. (7) FogROS2 supports a new backend based on Kubernetes that allows for faster warm starts and broader cloud-service provider support. (8) FogROS2 automates the building of cloud-based virtual machine images for faster startup time. (9) FogROS2 is part of the ROS 2 ecosystem and is accessible with the standard `apt install` command.

II. Related Work

Robots have limited onboard computing capabilities and as the computing demands of robotics grow, the cloud has become an increasingly necessary source of computing power. Kehoe et al. [6], [7] surveys the landscape of cloud robotics, including capabilities, potential applications, and challenges.

Cloud-robotics platforms facilitate offloading computation and data to the cloud. A notable example is RoboEarth [8], which shared information between robot and cloud. The main use case was to use the cloud to share databases between robots, but it did not leverage the cloud for offloading computing. Rapyuta [9] emerged from RobotEarth to become a platform for centralized management of robot fleets. In Rapyuta, robot nodes or Docker images are built on the cloud and pushed to the registered robots. A similar approach is taken by AWS Greengrass [10]. Using proprietary interfaces, Rapyuta and Greengrass allow building and deploying an entire pipeline for robotics applications [11], [12], [13], [14] from a centralized cloud interface. In contrast, FogROS2 approaches cloud deployment from the opposite perspective—instead of pushing applications from the cloud to a robot, FogROS2 pushes robot nodes from robot to the cloud. It uses an interface familiar to ROS 2 developers, allowing developers and researchers to access cloud resources without learning or conforming to an additional framework.

Researchers have explored using the cloud for grasp planning (e.g., Kehoe et al. [7], Tian et al. [15], and Li et al. [16]), parallelized Monte-Carlo grasp perturbation sampling [17], [18], [19], motion planning services (e.g., Lam et al. [11]), and splitting motion plan computation between robot and cloud (e.g., Bekris et al. [20] and Ichnowski et al. [21]). Researchers also have explored using new cloud computing paradigms as they emerge, such as serverless computing [22], [23], in which algorithms run (and are charged) for short bursts of intensive computing; while others have explored using the cloud to gain access to hardware accelerators such as FPGAs [24]. Others have explored some of these challenges, such as preserving privacy [25] and sharing models between robots [26]. In many of these examples, using the cloud requires a custom one-off implementation or interfacing with a proprietary library. FogROS2 and ROS 2 reduces this complexity.

For a robot to gain access to cloud resources, it must provision a cloud computer and establish a network connection to it.

As robots operate in the physical world, the connection to the cloud must be secured. However, setting this up is an involved process, in some cases requiring 12 steps for configuration and 37 steps for verification [27]. Hajjaj et al. [27] explored using SSH tunnelling for communication with ROS nodes running in the cloud. However, SSH tunnels do not support UDP which is needed when using ROS 2 Data Distribution Service (DDS) over UDP (while some DDS implementations support TCP, using TCP can introduce performance issues, and add unnecessary overhead for local communication). Crick et al. proposed rosbridge [28], Pereira et al. [29] proposed ROS Remote, and Xu et al. proposed MSA [30] as alternate ROS communication stacks with varying degrees of security and modifications required for their use in ROS applications. Wan et al. [31] and Saha et al. [32] propose unifying robot-cloud communication. Lim et al. proposed using VPNs [33], and FogROS2 builds on this approach. FogROS2 allows ROS 2 applications to easily use the cloud without code modification, and with secured communication.

III. Background on ROS

ROS 2 [4], the successor to Robot Operating System (ROS 1), includes many substantial improvements. A core improvement in ROS 2 is its change from a proprietary publication/subscription (pub/sub) system to the industry-standard middleware Data Distribution Service (DDS) [34]. DDS addresses robotics concerns such as providing real-time, high-performance, interoperable, and reliable communication [34]. As DDS is a specification, there are several implementations, and ROS 2 is agnostic to DDS implementation.

In ROS 2, computational units are abstracted into *nodes* that communicate with each other via a pub/sub system. Nodes subscribe to named *topics* and receive *messages* (data) as other nodes *publish* them. In an example application (Fig. 1), a camera node publishes images, a Simultaneous Location And Mapping (SLAM) node processes the images and publishes a location and map, a Motion Planner node receives the map and then computes and publishes a collision-free path, and a path following node drives the wheels to reach a target.

When orchestrating a robot application, often multiple nodes must be launched simultaneously. The ROS 2 launch system facilitates this by providing the ability to specify all required nodes, topic mappings, and relations between nodes in a single python script file. Launching the robot application is then a matter of running the command:

```
ros2 launch <package> <script>.
```

Listing 1, without the circled FogROS2 extensions, shows an example launch script that launches two nodes, a "grasp_motion" and a "grasp_planner" simultaneously.

IV. Approach

At the front end of FogROS2 is the launch system that specifies what nodes to launch and *where*. Unlike FogROS1, FogROS2's launch system is scriptable—allowing for launch-time logic to automate parts of the launch process. Listing. 1 shows an example in which a grasp planner needs a GPU to run efficiently. The script first defines a cloud machine with a GPU, then adds an attribute to the grasp planner node to tell the FogROS2 launch process to run it on the cloud machine. In more extensive use cases, multiple nodes can run on the same machine, and FogROS2 can launch multiple machines.

```
1  def generate_launch_description():
2    ld = FogROSLaunchDescription()
3    # configure cloud machine with a GPU (g4dn)
4    machine1 = AWSCloudInstance(
5      region="us-west-1",
6      ec2_instance_type="g4dn.xlarge")
7
8    # launch grasp_motion node on robot
9    grasp_motion_node = Node(
10     package="fogros2_examples",
11     executable="grasp_motion",
12     output="screen")
13   # launch grasp_planner node in on the cloud
14   grasp_planner_node = CloudNode(
15     package="fogros2_examples",
16     executable="grasp_planner",
17     output="screen",
18     machine=machine1 )
19   ld.add_action(grasp_motion_node)
20   ld.add_action(grasp_planner_node)
21   return ld
```

Listing 1: **FogROS2 Launch Script Example.** This example launches two nodes. Unlike ROS 1 and FogROS1, which used an XML launch file, FogROS2's launch files are python scripts. In this example, the FogROS2 launch extensions are circled. The first extension defines a machine on which to launch nodes. The second tells FogROS2 to launch the grasp planner node on that machine.

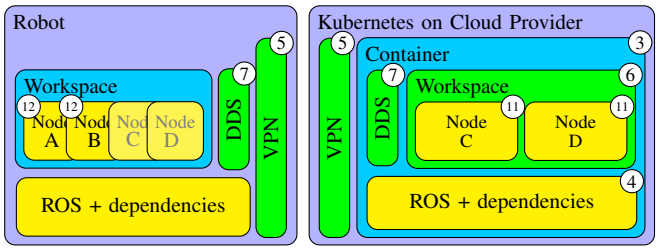

Fig. 2: **FogROS2 Launch Sequence** This high-level overview shows the steps FogROS2 takes in Section IV. Here we visualize a subset of the steps, keeping the same numbers as Section IV. The steps shown here are: (3) provision an instance, (4) install ROS and dependencies, (5) setup VPN, (6) copy ROS workspace to the instance, (7) setup DDS, (11) launch cloud-based nodes, and (12) launch robot nodes. The gray nodes on the Robot are copied to the cloud computer and only launched in the cloud.

The steps FogROS2 takes are (**bold** items are new to FogROS2): (1) **trigger launch from integration with the command-line interface**; (2) **process the launch script logic (e.g., to automate cloud selections, see Sec. IV-B)**; (3) connect to the cloud provider through its programmatic interface to create and start a new instance along with setting up access control to isolate from other cloud computers, and generating secure communication key pairs, **using the new Kubernetes backend when needed**; (4) install the ROS libraries and dependencies on the cloud machine needed for the robot application to run in the cloud (**skipped if using a pre-built custom image**, see Sec. IV-C); (5) set up Wireguard virtual private networking (VPN) on robot and cloud machine to secure the **ROS 2 DDS communication** between them; (6) copy the ROS nodes from the robot to the cloud machine; (7) **configure the DDS vendor's discovery mechanism to peer cloud and robot across the VPN**;

(8) **optionally configure streaming video compression (Sec. IV-E)**; (9) launch docker instances; (10) **optionally launch Foxglove for monitoring (Sec. IV-F)**; (11) launch cloud-based nodes; (12) launch nodes on the robot; (13) **once launched, users can use FogROS2 CLI integrations interact with FogROS-related cloud computers (Sec. IV-A)**.

Once the launch process is complete, the nodes running on the robot and on the cloud machine(s) securely communicate and interact with each other—and the only change needed was a few lines of the launch script.

A. ROS 2 Command Line Integration

FogROS2 integrates with the ROS 2 Command Line Interface (CLI), offering an intuitive way to interact with FogROS2 cloud instances not available in FogROS1 or ROS 1. To use the CLI, a user types into a terminal window:

```
ros2 fog <command> [args...]
```

where `command` specifies an interaction with FogROS2 along with additional arguments. For example, `command` can be `list`, which lists cloud instances, `delete`, to delete existing instances that are no longer in use, `image`, to create and manage cloud images, or `connect`, to connect via SSH [35] to running instances.

B. Launch Script Extensions

The new launch system in FogROS2 enables custom logic during launch, which was not possible with the launch system in FogROS1. We include several options:

- Since the distance between robot and cloud can dramatically affect network latency, the launch script can select nearest cloud computer based on the robot location.
- Since different computers (e.g., Intel vs Arm and with or without GPU) and regions require different images. The launch script can automate selecting the correct image.
- Selecting the best or most cost-effective cloud computer for a ROS 2 node can require significant effort [36]. By integrating ideas from Sky Computing [37], the launch script can select a machine type based on a specification of requirements (e.g., CPU type and core count, memory size, GPU type and memory, and more).

As an example, changing Listing 1 line 5 to call `region=find_nearest_aws_region()`, automates region selection. This function uses the robot's IP and a geolocation API to determine the nearest cloud data center. FogROS2 also offers a function to automate finding the cheapest instance type that matches a user's computing specifications.

C. Cloud Computer Virtual Machine Image Management

Cloud computers with FogROS1 suffered from long startup times, sometimes approaching 4 minutes. A significant portion of this time is due to installing ROS and dependencies on the cloud computer. Advanced users could address this problem by creating computer images with software pre-installed. FogROS2 adds a tool to automate this process using the command-line interface, allowing it to start up pre-installed instances in a fraction of the time of FogROS1.

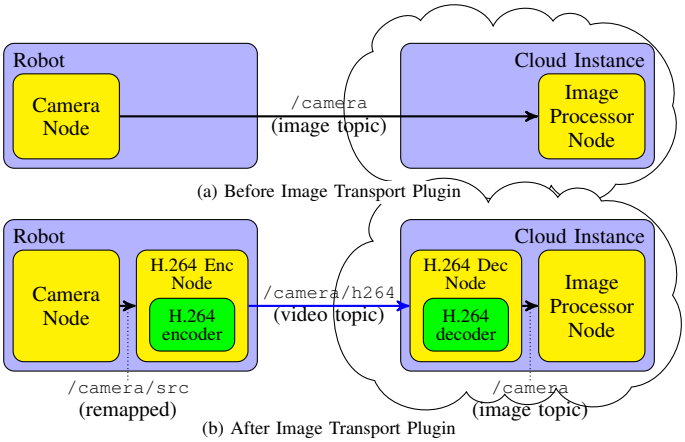

Fig. 3: **FogROS2 Streaming Video Compression** Unlike FogROS1, FogROS2 uses streaming video compression of image topics to reduce bandwidth and latency of cloud-based image processing. In (a), a cloud-based image processor node subscribes to the camera without streaming compression. In (b), FogROS2, using the Image Transport Plugin, introduces an H.264 encoder (compression) node on the robot, and pairs it with an H.264 decoder (decompression) node on the cloud. In both, the cloud-based Image Processor Node subscribes to the same topic.

D. New Kubernetes Backend

FogROS1 supported AWS only. To support additional cloud service providers, FogROS2 integrates a new Kubernetes [38] backend. Kubernetes (see Fig. 1) is a system that orchestrates running *containers*, or units of software packages and their dependencies—e.g., a robot ROS node, FogROS2, ROS 2, and underlying operating system components. With Kubernetes, computers can already be on and waiting to run a new container. This allows for significant speedup in startup time. There is a trade-off—machines managed by Kubernetes must already be on, and there can be significant initial delay when starting Kubernetes the first time.

E. Streaming Image Compression

Many robot algorithms depend on fast processing of image and video data, and these algorithms increasingly require hardware acceleration e.g., via GPUs. However, images and videos are data intensive in ROS, and the time to transmit data to the cloud can reduce the advantage of cloud-based acceleration. Processing images in the cloud was possible with FogROS1, but with high latency.

To address this, at launch time FogROS2 can setup transparent streaming compression between robot and cloud. The video compression we use is H.264 [39] from the open-source libx264 [40] library. Using H.264 allows FogROS2 to greatly reduce the latency of processing video in the cloud. We implement a ROS 2 *image_transport_plugin* [41] to make the compression transparent to the application—publishing nodes still publish a sequence of images and subscribing nodes still receive a sequence of images. Fig. 3 (b) shows how FogROS2 implements transparent streaming compression.

H.264 compression is often hardware accelerated, reducing the CPU utilization required to compress and decompress.

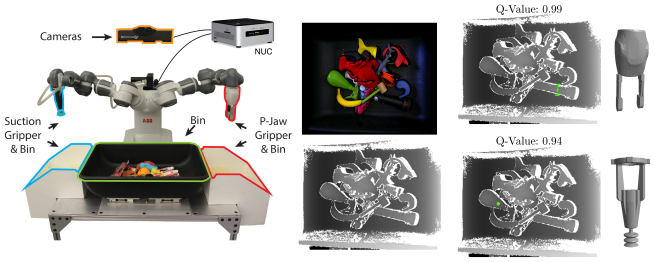

Fig. 4: **Grasp Planning with Dex-Net** A robot (left) with an attached computer (e.g., an Intel NUC) sends RGBD image observations from an overhead camera (middle) to compute grasps using a Dex-Net Grasp Quality Convolutional Neural Network (GQCNN) on a cloud computer with a GPU. The result (right) is grasps poses (green) and their relative quality score (q-value) for parallel-jaw (top-right) and suction gripper (bottom-right).

F. Remote Monitoring and Visualization with Foxglove

Users of robots running FogROS1 were only able to monitor robots locally, missing out on an advantage of the cloud. FogROS2 integrates Foxglove [3]—a browser-based tool that enables visualization of ROS 2 topics. Much like rviz, the 3D visualizer that is part of ROS, Foxglove operates by subscribing to ROS messages, then interpreting and displaying them. The chief difference is that Foxglove runs in a browser, and thus requires messages to be transmitted over web-based protocols. FogROS2 uses two components to integrate Foxglove: (1) a Foxglove server, that provides the web interface software, and (2) *ROS bridge*, a ROS 2 node that subscribes to topics as a ROS node and proxies them through websockets to a browser running the Foxglove software (Fig. 1 bottom). When visualizations are enabled, FogROS2 launches both the Foxglove server as a docker, and the ROS bridge node.

Once set up, FogROS2 provides the IP address of the server, allowing multiple users in different locations to monitor and visualize the robot application in a browser.

V. Evaluation

We evaluate the ability of FogROS2 to speed up robot computations using the cloud, to lower startup latency, to lower network latency and utilization.

We use an Intel NUC with an Intel® Pentium® Silver J5005 CPU @ 1.50 GHz with 2 cores enabled and with a 10 Mbps network connection to act as the robot, as it is representative of an efficient computing platform that could be onboard or attached to a robot (Fig. 4). We perform all evaluations with cloud nodes deployed to AWS unless specified otherwise. This setup differs from the examples in FogROS1, thus we re-run the experiments in FogROS1 [2] to compare FogROS1 and FogROS2 on equivalent hardware.

A. Streaming Video Compression

We evaluate the performance of using streaming H.264 video compression between robot and cloud. In this experiment, we have the robot node publish images to a topic to which a cloud node subscribes. The cloud node responds immediately with a small acknowledgement message. We record the round-trip time, and frames per second (FPS), and show it in Table I. We compare to **Uncompressed**, which is the raw pixel arrays native to ROS 1 and ROS 2, **Compressed**, which uses (non-streaming) image compression, **Theora** [42] streaming image compression, and FogROS2's H.264 compression. From the table, we observe the benefit of streaming video compression between the robot and the cloud, as the cloud can receive images 13× faster FPS, while shortening the latency by 97 %. We observe performance improvement of H.264 over Theora, the previous best available compression, with H.264 shortening the latency by 54 %. The reduced latency from 1401 ms to 38 ms may enable some real-time cloud-robotics applications not possible without compression.

Mode	FPS	Latency (ms)
Uncompressed	1.42	1401
Compressed	15.0	333
Theora	29.6	83
H.264	29.0	**38**

TABLE I: **Streaming FogROS2 video compression** A node on a robot publishes 3000 images to a ROS topic that FogROS2 transparently compresses and sends to a node in the cloud. We measure the frames per second (FPS) the cloud node receives. For every image, the cloud node publishes a small acknowledgement message that the robot uses to measure the round-trip time, or latency (ms).

	Robot	FogROS1	FogROS2	
Scenario	Only	Compressed	Compressed	H.264
fr1/xyz	0.52	1.62	0.82	**0.24**
fr2/xyz	0.43	1.61	0.75	**0.25**
fr2/loop	0.68	1.63	0.89	**0.22**

TABLE II: **ORB-SLAM2 results on FogROS2** We run ORB-SLAM2 [43] on 3 benchmarks from the TUM Dataset [46] and measure average per-frame round-trip latencies incurred by the ORB-SLAM2 node. Here, FogROS2 runs on a 36-core cloud computer. H.264 compression allows FogROS2 to outperform robot-only and image-compressed (not video-compressed).

However, there may be a tradeoff in some applications. Theora and H.264 are both lossy compression algorithms, meaning they are designed to compress videos by discarding some image information. This information loss is tailored to human perception [39], and thus may adversely affect computer vision algorithms.

B. Cloud Robotics Benchmark Applications

We evaluate FogROS2 in a benchmark on 3 example robot applications: SLAM with ORB-SLAM2 [43], Grasp Planning with Dex-Net [44], and Motion Planning with Motion Planning Templates (MPT) [45]. Refer to FogROS1 [2] for further details on these benchmarks. We compare to a baseline of robot-only computing and FogROS1 using equivalent cloud computers. For examples with cloud-based image processing, we compare to additional baselines of (a) raw/uncompressed, (b) PNG compressed, and (c) Theora [42] compressed, where Theora is an open-source video compression library with an existing image transport plugin [41].

In SLAM experiment (Table II), H.264 compression lowers per-frame latency from 0.82 s to 0.24 s, a 3.4× improvement, allowing FogROS2 to achieve lower latency than robot-only

Scenario	Robot Only	Cloud Compute	FogROS1 Network	FogROS1 Total	FogROS2 Network	FogROS2 Total
Uncompressed	14.0	0.6	5.0	5.6	5.0	5.6
Compressed	14.0	0.6	1.3	1.9	0.7	1.3
H.264	14.0	0.6	-	-	0.6	**1.2**

TABLE III: **Dex-Net results on FogROS2** We measure compute time in seconds for 10 trials on a robot with a CPU, and compute and network time using cloud computer with an Nvidia T4 GPU via FogROS2.

Scenario	Robot Only	Cloud Compute	FogROS1 Network	FogROS1 Total	FogROS2 Network	FogROS2 Total	Speedup
Apartment	157.6	3.46	0.07	3.53	0.04	3.50	45×
Cubicles	35.8	1.51	0.07	1.58	0.05	1.56	23×
Home	161.8	4.73	0.08	4.81	0.05	4.78	34×
TwistyCool	167.9	4.76	0.08	4.84	0.05	4.81	35×

TABLE IV: **MPT Motion Planning results on FogROS2** We run multi-core motion planners from Motion Planning Templates (MPT) [45] on 4 motion planning problems from the Open Motion Planning Library OMPL [47].We record the planning time running on a 96-core cloud computer, and the network round-trip time between robot and cloud.

Computer Image	FogROS1	FogROS2 AWS	FogROS2 GCP (K8s)	FogROS2 Local (K8s)
Default	228±25 s	275±61 s	**29±2.1 s**[b]	**27±0.83 s**[b]
Custom	155±32 s[a]	**85±11 s**		

[a] FogROS1 custom image manually created
[b] Kubernetes cluster already started.

TABLE V: **Startup time for cloud computer** FogROS2 automates the creation of custom computer images with pre-installed ROS and dependencies. This speeds up cloud-computer startup allowing the robot to operating sooner. Kubernetes uses containers that have many of the ROS component already installed—thus they save time installing ROS, but are not fully customized like the AWS images.

	west coast	east coast
us-west-1	**6.1 ms**	72 ms
us-east-2	74 ms	**13 ms**

TABLE VI: Example round-trip data times based on robot location (west or east coast) and the cloud computer's data center (us-west-1 vs us-east-2). FogROS2 launch extension can automatically select the nearest data center resulting in lower network latency.

computation. In Dex-Net experiments (Table III), FogROS2 and H.264 results in a 12× speedup. In MPT experiments (Table IV), planning speeds up by up to 45× (FogROS1 also speeds up similarly due to hardware changes in experiments).

C. Cloud-Computer Startup Times

We test if FogROS2 can shorten startup times compared to FogROS1. Short startup times benefit software development cycles and robots that intermittently operate (e.g., vacuuming robots). In this experiment, we use the new `image` command in FogROS2 to generate a custom computer image, and measure the time between launch and first robot-cloud ROS 2 node interaction with and without the custom image. For comparison, we manually create a custom image for FogROS1 by using the AWS web console.

In Table V, we observe that the custom image in FogROS2 reduces AWS startup times by 63 %.

Using Kubernetes on local cluster or Google Cloud Platform (GCP) reduces startup times further, due to having the computers in the Kubernetes cluster already running. The AWS backend must create a new cloud computer each time, accounting for approximately 40 s of delay.

There is a tradeoff to be made in startup times. Kubernetes requires starting up a cluster of computers, which can take on the order of 10 minutes. If one is willing to spend this time upfront, Kubernetes allows users to redeploy ROS nodes over and over, which may be beneficial when rapidly prototyping changes to robot code.

D. Automating Region Selection

We test the launch script extensions for automatic region selection to allow robots to select the cloud data center that is nearest to them. We test deploy a robot on the US west coast, and the launch script selects the `us-west-1` data center. When we test to deploy the robot on the US east cost, the robot selects the `us-east-2` data center. Table VI shows example round-trip network times in this experiment, with the best latencies in bold and automatically selected by the geolocation script.

VI. Conclusion

We present FogROS2, an adaptive cloud-robotics platform for running compute-intensive portions of ROS 2 applications in the cloud. FogROS2 addresses 9 shortcomings of FogROS1, integrates with the ROS 2 launch and communication systems, to provision and start cloud computers, configure and secure network communication, install robot code and dependencies, and launch robot and cloud-robotics code. As a redesigned and distinct successor to FogROS, FogROS2 supports ROS 2, transparent video compression, improved performance and security, access to more cloud computing providers, and remote visualization and monitoring. In experiments, we observe a significant performance benefit to using cloud computing, with the additional improvement from transparent video compression.

In future work, we will continue to improve performance and capablities of FogROS2. We will explore additional models of computing, such as serverless, spot instances, and more. We will also explore extending the networking capabilities of FogROS2 to allow multiple robots to communicate, collaborate, and share data more easily [48].

Acknowledgements

This research was performed at the AUTOLab at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the CITRIS "People and Robots" (CPAR) Initiative, the Real-Time Intelligent Secure Execution (RISE) Lab, and by the NSF/VMware Partnership on Edge Computing Data Infrastructure (ECDI) Secure Fog Robotics Project Award 1838833. We thank our colleagues for helpful discussions and testing. We thank Brandon Fan and Emerson Dove for their contributions and discussion related to the new Kubernetes backend.

REFERENCES

[1] J. Ichnowski, J. Prins, and R. Alterovitz, "The economic case for cloud-based computation for robot motion planning," in *Int. S. Robotics Research (ISRR)*, 2017, pp. 1–7.

[2] K. E. Chen, Y. Liang, N. Jha, J. Ichnowski, M. Danielczuk, J. Gonzalez, J. Kubiatowicz, and K. Goldberg, "FogROS: An adaptive framework for automating fog robotics deployment," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 2035–2042.

[3] Foxglove Technologies Inc, "Foxglove," https://foxglove.dev/.

[4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[5] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[6] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[7] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, "Cloud-based robot grasping with the google object recognition engine," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 4263–4270.

[8] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo, *et al.*, "RoboEarth," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[9] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2014.

[10] "AWS IoT Greengrass," https://aws.amazon.com/greengrass/, accessed: 2021-02-15.

[11] M.-L. Lam and K.-Y. Lam, "Path planning as a service PPaaS: Cloud-based robotic path planning," in *Proc. IEEE Int. Conf. on Robotics and Biomimetics (ROBIO)*, 2014, pp. 1839–1844.

[12] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel, "Cloud-based collaborative 3d mapping in real-time with low-cost robots," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, 2015.

[13] C. Mouradian, S. Yangui, and R. H. Glitho, "Robots as-a-service in cloud computing: Search and rescue in large-scale disasters case study," in *IEEE Consumer Communications & Networking Conference (CCNC)*, 2018, pp. 1–7.

[14] J. Rosa and R. P. Rocha, "Exportation to the cloud of distributed robotic tasks implemented in ros," in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 235–240.

[15] N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, and K. Goldberg, "A cloud robot system using the dexterity network and Berkeley robotics and automation as a service (BRASS)," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 1615–1622.

[16] P. Li, B. DeRose, J. Mahler, J. A. Ojea, A. K. Tanwani, and K. Goldberg, "Dex-Net as a service (DNaaS): A cloud-based robust robot grasp planning system," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, 2018, pp. 1420–1427.

[17] B. Kehoe, D. Berenson, and K. Goldberg, "Estimating part tolerance bounds based on adaptive cloud-based grasp planning with slip," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, 2012, pp. 1106–1113.

[18] B. Kehoe, D. Berenson, and G. Ken, "Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 576–583.

[19] B. Kehoe, D. Warrier, S. Patil, and K. Goldberg, "Cloud-based grasp analysis and planning for toleranced parts using parallelized Monte Carlo sampling," *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 455–470, 2014.

[20] K. Bekris, R. Shome, A. Krontiris, and A. Dobson, "Cloud automation: Precomputing roadmaps for flexible manipulation," *IEEE Robotics & Automation Magazine*, vol. 22, no. 2, pp. 41–50, 2015.

[21] J. Ichnowski, J. Prins, and R. Alterovitz, "Cloud-based motion plan computation for power-constrained robots," in *Workshop on the Algorithmic Foundation of Robotics (WAFR)*. Springer, 2016.

[22] J. Ichnowski, W. Lee, V. Murta, S. Paradis, R. Alterovitz, J. E. Gonzalez, I. Stoica, and K. Goldberg, "Fog robotics algorithms for distributed motion planning using lambda serverless computing," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020, pp. 4232–4238.

[23] R. Anand, J. Ichnowski, C. Wu, J. M. Hellerstein, J. E. Gonzalez, and K. Goldberg, "Serverless multi-query motion planning for fog robotics," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. IEEE, 2021.

[24] S. Murray, W. Floyd-Jones, Y. Qi, D. J. Sorin, and G. D. Konidaris, "Robot motion planning on a chip." in *Proc. Robotics: Science and Systems (RSS)*, 2016.

[25] J. Mahler, B. Hou, S. Niyaz, F. T. Pokorny, R. Chandra, and K. Goldberg, "Privacy-preserving grasp planning in the cloud," in *Proc. IEEE Conf. on Automation Science and Engineering (CASE)*, 2016, pp. 468–475.

[26] A. K. Tanwani, R. Anand, J. E. Gonzalez, and K. Goldberg, "RILaaS: Robot inference and learning as a service," *IEEE Robotics & Automation Letters*, vol. 5, no. 3, pp. 4423–4430, 2020.

[27] S. S. H. Hajjaj and K. S. M. Sahari, "Establishing remote networks for ROS applications via port forwarding: A detailed tutorial," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417703355, 2017.

[28] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, "Ros and rosbridge: Roboticists out of the loop," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2012, pp. 493–494.

[29] A. B. M. Pereira, R. E. Julio, and G. S. Bastos, "ROSRemote: Using ROS on cloud to access robots remotely," in *Robot Operating System (ROS)*. Springer, 2019, pp. 569–605.

[30] B. Xu and J. Bian, "A cloud robotic application platform design based on the microservices architecture," in *Int. Conf. on Control, Robotics and Intelligent System*, 2020, pp. 13–18.

[31] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[32] O. Saha and P. Dasgupta, "A comprehensive survey of recent trends in cloud robotics architectures and applications," *Robotics*, vol. 7, no. 3, p. 47, 2018.

[33] J. Z. Lim and D. W.-K. Ng, "Cloud based implementation of ROS through VPN," in *Int. Conf. on Smart Computing & Communications (ICSCC)*. IEEE, 2019, pp. 1–5.

[34] Object Management Group (OMG), "Data distribution service 1.4," Object Management Group (OMG)," Standard, Mar. 2015. [Online]. Available: https://www.omg.org/spec/DDS/1.4/

[35] T. Ylonen and C. Lonvick, "The secure shell (SSH) protocol architecture," Internet Requests for Comments, RFC Editor, RFC 4251, January 2006. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4251.txt

[36] N. J. Yadwadkar, B. Hariharan, J. E. Gonzalez, B. Smith, and R. H. Katz, "Selecting the best vm across multiple public clouds: A data-driven performance modeling approach," in *Proceedings of the 2017 Symposium on Cloud Computing*, 2017, pp. 452–465.

[37] S. Chasins, A. Cheung, N. Crooks, A. Ghodsi, K. Goldberg, J. E. Gonzalez, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, M. W. Mahoney, A. Parameswaran, D. Patterson, R. A. Popa, K. Sen, S. Shenker, D. Song, and I. Stoica, "The sky above the clouds," 2022. [Online]. Available: https://arxiv.org/abs/2205.07147

[38] "Kubernetes," https://kubernetes.io/, 2022, [Online; accessed 15-Sep-2022].

[39] I. E. Richardson, *The H.264 advanced video compression standard*. John Wiley & Sons, 2011.

[40] The VideoLAN Organization, "x264," https://www.videolan.org/developers/x264.html.

[41] "image_transport_plugins," https://github.com/ros-perception/image_transport_plugins.

[42] Xiph.Org Foundation, "Theora specification," Xiph.Org Foundation, Tech. Rep., June 2017. [Online]. Available: https://theora.org/doc/

[43] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[44] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Proc. Robotics: Science and Systems (RSS)*, 2017.

[45] J. Ichnowski and R. Alterovitz, "Motion planning templates: A motion

planning framework for robots with low-power CPUs," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019.

[46] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2012.

[47] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012. [Online]. Available: http://ompl.kavrakilab.org

[48] K. Chen, J. Yuan, N. Jha, J. Ichnowski, J. Kubiatowicz, and K. Goldberg, "Fogros g: Enabling secure, connected and mobile fog robotics with global addressability," 2022. [Online]. Available: https://arxiv.org/abs/2210.11691