Chameleon@Edge Community Workshop Report

Kate Keahey, Jason Anderson, Michael Sherman, Zhuo Zhen,
Mark Powers, Isabel Brunkan, Adam Cooper

▸ To cite this version:

Kate Keahey, Jason Anderson, Michael Sherman, Zhuo Zhen, Mark Powers, Isabel Brunkan, Adam Cooper. Chameleon@Edge Community Workshop Report. [Report] December 13, 2021. doi: 10.5281/zenodo.5777344.

Zenodo DOI: 10.5281/zenodo.5777344

https://zenodo.org/record/5777344

Published on December 13, 2021

Chameleon@Edge
Community Workshop Report

Kate Keahey, Jason Anderson, Michael Sherman, Zhuo Zhen,
Mark Powers, Isabel Brunkan, Adam Cooper

December 2021

Chameleon@Edge Community Workshop Report

Kate Keahey
keahey@mcs.anl.gov
Argonne National Laboratory

Jason Anderson
jasonanderson@uchicago.edu
University of Chicago

Michael Sherman
shermanm@uchicago.edu
University of Chicago

Zhuo Zhen
zhenz@uchicago.edu
University of Chicago

Mark Powers
markpowers@uchicago.edu
University of Chicago

Isabel Brunkan
isabel@uni.minerva.edu
Minerva University

Adam Cooper
adamcooper@uchicago.edu
University of Chicago

ABSTRACT

The widespread availability of inexpensive Internet of Things (IoT) and edge devices, that can be deployed at scale and used to observe, measure, and interact with our environment, are revolutionizing the opportunities available to science. At the same time building systems that use those devices dramatically changes the assumptions we make about infrastructure, system properties, and applications, and thus requires significant research to make good on the promise of this new technology. As a scientific instrument for systems research, the Chameleon testbed is evolving to make such research possible and has developed a preview set of new capabilities allowing users to allocate, reconfigure, and experiment with edge devices (CHI@Edge). To assess the alignment of the this set of capabilities with researchers' needs (i.e., the product definition for the testbed) the Chameleon team organized a community workshop where experimenters showcased projects developed on CHI@Edge and discussed testbed requirements needed for the advancement of current and future research.

The primary insights gained through the workshop is the evolution of infrastructure scientific instruments from focus on infrastructure ownership to focus on implementation of infrastructure management and sharing, user services, and federation. In particular, edge testbeds are likely to evolve to support mixed ownership of hardware (i.e., one in which one of the user roles is going to be that of hardware provider) due to importance of infrastructure placement, its configuration (specifically, association with IoT devices interacting with the environment), as well as affordability of the edge hardware. This shift of focus requires new features such as restricted hardware sharing, and new operational models taking into account not only innovative hardware types, but also its deployment context (i.e., new trade-offs in security, accessibility, and connectivity), and less specialized resource providers. The broad range of context configurations will have to take into account the variability of demands from various communities/contexts and potentially provide a spectrum of reconfiguration methods rather than rely on a one-size-fits-all approach. The ability to connect such lightweight, community-specific, and often ad hoc testbeds to larger pools of static shared network and datacenter resources via consistent interfaces was judged to be of paramount importance, as were facilities for sharing research in an actionably consumable way (i.e., via digital forms such as data, software, or packaged experiments) so that different groups can leverage results produced by others more easily.

Keywords: Edge computing, Internet of Things (IoT), infrastructure management, cloud computing, testbeds

1 INTRODUCTION

Chameleon [1] is an NSF-funded cyberinfrastructure scientific instrument supporting projects in computer science research, computer science education, and emergent applications. Originally established to provide a platform for exploration and learning in cloud computing, the instrument has evolved overtime to meet demand for research testbed supporting topics in programmable networking, artificial intelligence, and others. The recent emergence of research interest in the Internet of Things (IoT), edge computing, and related topics is driving further evolution to provide a platform that supports this type of investigation. Accordingly, in the Summer of 2021, the Chameleon team premiered a preview implementation of an edge testbed, called CHameleon Infrastructure (CHI) at Edge (CHI@Edge) and worked with interested segments of the user community to refine its needs.

This workshop provided a forum for the emergent community of Chameleon edge users – system scientists, educators, as well as users working on emergent applications – to share their experiences of using the CHI@Edge platform, highlight projects developed during the summer, and provide feedback and discuss future directions with the Chameleon team. The workshop was publicly announced and advertised to the Chameleon community as well as to related communities with interest in IoT, edge, and the cloud. The workshop was held virtually on September 9, 2021 with roughly 70 individuals participating live at different times.

The Chameleon@Edge workshop was a culmination of three months of intensive, hands-on engagement with the user community. Many insights presented in this report are derived from interactions with users over that period including support conversations, feedback shared on the user mailing list for the CHI@Edge project, as well as presentations and discussion during the workshop itself. We want to particularly acknowledge the insights and contributions of the following community members who presented their use cases at the workshop: Rick Anderson (Rutgers University), Sahithi Ankireddy (California Institute of Technology), Leonardo Bobadilla (Florida International University), Kevin Boswell (Florida International University), Isaac Darling (University of Chicago), Ewa Deelman (University of Southern California), Airam Flores (University of Texas at El Paso), Haryadi Gunawi (University of Chicago), Abdullah Imran (University of Texas at El Paso), Taimoor Ul Islam (Iowa State University), Junchen Jiang (University of Chicago), Eric Lyons (University of Massachusetts Amherst), Shirley Moore (University of Texas at El Paso), George Papadimitriou (University of Southern California), Ryan Tanaka (University of Southern California), Jonathan Tsen (FATEC Shunji Nishimura), Hongwei Zhang (Iowa State University), Michael Zink (University of Massachusetts Amherst).

The workshop itself started out with a keynote from Deepankar Medhi (NSF), outlining the ecosystem of NSF's testbeds and a presentation and a demonstration from the Chameleon team outlining the capabilities of CHI@Edge testbed. A further keynote from Weisong Shi (Wayne State University) outlined research directions in edge computing. The intent of these presentations was to create context for user sessions that focused on actual experiences when mapping research onto the testbed, the mechanics of running experiments, and discussion of requirements. The core of the workshop was made up of two sessions, each consisting of four lightning talks presenting user projects as well as panel discussion of presenters and the Chameleon team. The feedback from live panel discussions was organized around questions posed to the panel; the same questions were posed to wider attendees in the form of a virtual poll in order to absorb feedback from as many users as possible.

The key outcomes of the workshop are general requirements for an edge testbed serving the needs of computer science research and education as well as emergent applications, feedback on hardware as well as reconfiguration/isolation capabilities for this type of experimentation, and an assessment of priorities that the team can use in formulating a development roadmap for further capabilities.

2 CHI@EDGE: INTRODUCTION AND STATUS

Chameleon is a testbed for computer science systems research, with supported experiments ranging from designing new operating systems, virtualization solutions, or networking protocols to exploring challenges in

power management, performance variability, or artificial intelligence. To support this type of experimentation, Chameleon provides deep (bare metal) reconfiguration of resources fully allocated to its users, with a small part of the system configured as a KVM reconfigurable cloud for a different reconfigurability/efficiency trade-off. The project hardware balances investment into large-scale and diversity with partitions of tens of thousands of cores and a total of over 6PB of storage available for allocation in addition to an investment in a range of GPUs, FPGAs, a variety of storage solutions in various configurations (including NVMe and NVDIMM), diverse networking solutions, and heterogeneous processors including ARMs, Atoms, and low-power Xeons. This hardware is distributed over two major supercomputing sites connected by a 100g network. Since announcing its first availability in 2015, Chameleon has served 6,000+ users, working on 750+ projects in CS research, education, and emergent applications.

Over the last few years, it became evident that the Chameleon user community is increasingly interested in experiments in the edge to cloud continuum. Projects in topics such as biometrics [2], federated learning [3], and network fingerprinting for IoT [4] increasingly used the testbed to emulate IoT/edge capabilities, and a desire to experiment on real edge devices was increasingly evident in interactions with the user community. To adapt to this need, in phase 3 of the project the Chameleon team proposed extending the existing cloud testbed to support edge capabilities through interfaces consistent with the existing testbed.

A more specific testbed definition included features such as support for as non-prescriptive access/reconfiguration capabilities (i.e., as similar to bare metal reconfiguration as possible with support for high level of user privilege) over a range of devices via a set of interfaces and services (e.g., orchestration via Jupyter notebooks) consistent with the existing cloud testbed so that experiments using both edge and cloud can be supported easily. However, in a significant departure from the existing operations model, users requested that they can add to the testbed devices owned and operated by them, for restricted sharing (i.e., available only to selected individuals, usually collaborators of the device owner), but still consistent with the general testbed model. Further, there was also interest in support for "peripherals", sensors, or IoT devices, attached to the edge devices provided by the testbed.

These requirements gave rise to the design and development of a preview version of CHI@Edge made available for users in Summer 2021, a usable but still largely experimental extension of the Chameleon testbed. The preview implementation was built by adapting various OpenStack components to execute at the edge which allowed us to leverage user-facing services developed by the Chameleon team and ultimately provide a featureful and relatively reliable strawman for community evaluation – though one that will need significant evolution and refactoring to become production.

The initial CHI@Edge hardware deployment offered during the summer (referred to as the CHI@Edge *static pool* of devices in the rest of document) consisted of one Raspberry Pi 3, seven Raspberry Pi 4s, and four NVIDIA Jetson Nanos. This initial investment was supplemented by edge devices added by various users for limited sharing in the course of the summer (temporarily suspended at the end of the summer due to the difficulty of supporting a relatively immature implementation in this way). Despite that, by the end of August, users experienced a lack of resource availability due to significant demand which ultimately necessitated supplemental purchase of equipment. The current status of CHI@Edge deployment is shown on its resource page [5]; the rest of this section describes the capabilities of the system.

CHI@Edge provides support for full resource allocation capabilities as implemented in Chameleon infrastructure (CHI). This allows users to allocate edge devices and IP addresses either on-demand or via advance reservations. In particular, it allows users to allocate devices by type (e.g., a Raspberry Pi or NVIDIA Nano) as well as by name, i.e., a specific device. Though this capability was already useful in the datacenter, given the importance of deployment context for edge we expect it to be the primary allocation method on the edge testbed.

Given the lack of IPMI or equivalent capability on most edge devices we considered, as well as the need to provide relatively lightweight isolation implementation, the preview CHI@Edge provided reconfiguration via container deployment though in a single tenancy model (i.e., a device is fully dedicated to a user who allocated

it, the user can deploy as many containers as they wish, and will see some performance interference even if they deploy only one container). Similarly, to the main Chameleon offering we provide a library of images so that users don't have to configure their own images from scratch, as well as snapshotting functionality so that image configurations can be saved for future deployment. In addition however, we also provide a management framework for peripheral/IoT devices, together with some plugins covering the most common use cases such as cameras or software defined radios (SDRs); we worked with the user community during the summer to define requirements for more peripherals to support. Last but not least, users are able to program their devices via integration with Jupyter notebooks, consistent with that supported on the Chameleon testbed. This provides not only a more convenient interface but also support for orchestrating complex deployments, potentially over both edge devices and datacenter nodes.

Building on top of OpenStack ensured that all this functionality is available via the same set of interfaces as the main Chameleon offering, i.e., users can program their experiments edge to cloud using command line interfaces, the web GUI, or Jupyter notebooks via python-chi or bash interfaces. Furthermore, the system also supports federated login using user's institutional credentials for ease of use and integration with other facilities. Finally, building on top of OpenStack also had the advantage of leverage/adaptation of implementation of complex concepts rather than implementing them ourselves. All in all, these two important advantages of building the preview on top of OpenStack – the ability to leverage user-facing services (Jupyter, federated identity) developed on top of OpenStack in the main Chameleon testbed and adapting existing implementation for some of the complex concepts – allowed us to put together a preview implementation of the main edge testbed concepts extremely quickly. While the internals of this implementation will require very significant evolution to provide a stable implementation in the long run, the existing installation allowed us to give users a usable testbed, enable a significant range of projects over the summer, and generate valuable feedback as to the features that an edge testbed will require.

Already early into the experiment, our users voiced a need for a capability/SDK that would allow them to add their own devices to the testbed for limited/restricted sharing (i.e., sharing with collaborators only). The Chameleon team provided a rudimentary version of this capability in CHI@Edge, which enabled several projects to manage their devices in this way and implement their projects. The CHI@Edge SDK allows users to add their device to the device pool managed by Chameleon though under the assumption of significant control over the device by the device owner. This functionality needs to mature significantly to allow for sustained support; as a result, we had to temporarily suspend the availability of the SDK at the end of the summer as we improve the system. Long term, means outside of CHI@Edge will need to be found to address concern of long-term operations and security for devices that are not controlled by the device owner.

3 SCIENCE@EDGE USE CASES

This section presents summaries capturing the use cases presented during the workshop and highlights how each of them was using CHI@Edge.

3.1 Using CHI@Edge to Drive Autonomous Cars for Experiential Learning

Rick Anderson (Rutgers University), Sahithi Ankireddy (California Institute of Technology), Isaac Darling (University of Chicago), Michael Sherman (University of Chicago)

Autonomous vehicles – composed of a remote control toy car equipped with a camera and Raspberry Pi – can be used to teach machine learning concepts by using edge devices to process feedback from sensors (camera) and making decisions on how to drive the vehicle. Instructional materials for each class walk the students through collecting data with a working car, using the data to train new models, and measuring performance of the resulting model – in simulation on virtual tracks as well as in a physical environment – and can be tailored to

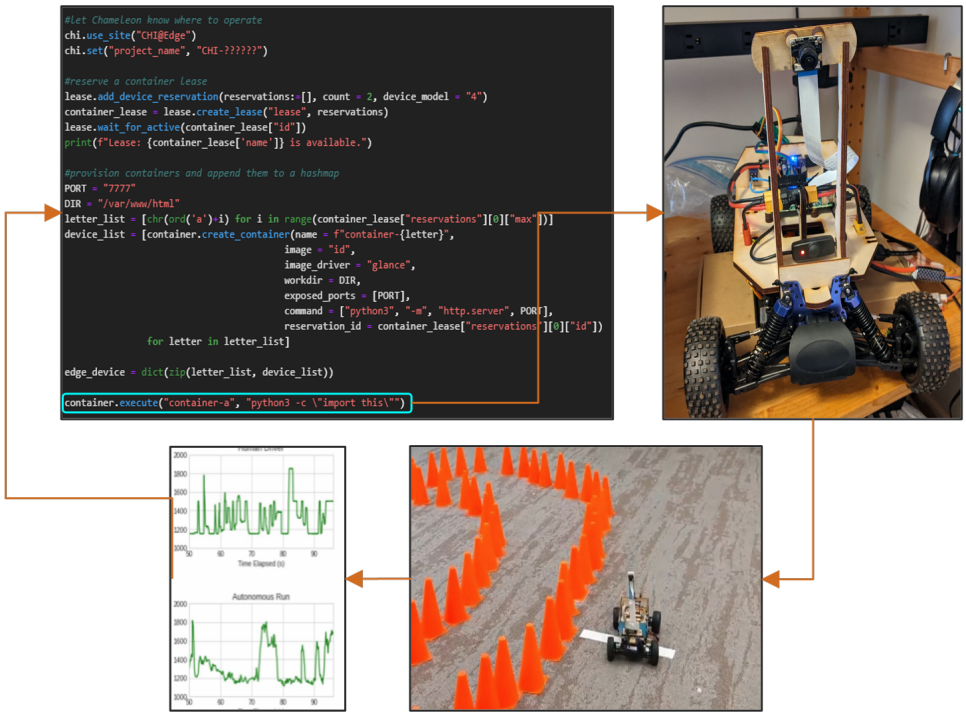

Figure 1: Edge devices connected to camera and actuators guiding the car's steering system are programmed by students, using edge and cloud from the same workflow via Jupyter notebook. (Included with the permission of Rick Anderson.)

the level of the class, from changing physical design (steering rack, wheel size) for an automotive vocational school, to implementing new machine learning models in a computer science course. A challenge of this type of experiential learning is guiding students through the initial set up process of building the car and loading code onto it, as there are many sources of friction, ensuring the sensors work, connecting the edge device to a network, loading the base software onto it, and initial configuration steps to allow machine learning models to run. An alternative is to start with a "working" car each time, allowing students to test incremental changes, and avoiding costly start-up activities each semester. CHI@Edge allows the edge devices to remain always connected, and the ability to provision software onto them remotely. In this way, the cars can be built and tested once with known working code, and then have course-specific code provisioned for each class.

This project used the CHI@Edge SDK to add several cars to the testbed for limited sharing (i.e., sharing only with the individuals taking the class). This configuration allowed the authors to package the software controlling the cars into containers, deployed onto the cars. The team developed containers with libraries to access the cars' interfaces, along with a python framework for data collection, training and testing. Then, they used Jupyter notebooks created in Chameleon's JupyterHub to orchestrate the deployment and trigger the cars to begin data collection runs. The data was copied off of the cars back into the Jupyter environment, where it was post-processed and analyzed. In some configurations, cloud resources were used to offload model training tasks to speed up development. Overall, in addition to the CHI@Edge features, specifically the SDK, the cyborg

plugin mechanism to add GPIO, serial, and camera devices, this project also made use of the python-chi API, the JupyterHub environment, cloud storage, and bare-metal GPU instances.

3.2 CHI@Edge as a baseline for advanced wireless research for agricultural and rural settings

Hongwei Zhang (Iowa State University), Taimoor UI Islam (Iowa State University)

ARA is an at-scale platform for advanced wireless research in agricultural and rural settings. By combining CHI@Edge and ARA cloud computing and storage resources, high capacity, low latency wireless applications can be applied for high-throughput crop phenotyping, precision livestock farming, agriculture automation and rural education.

This project installed CHI@Edge SDK on a variety of ARA devices and conducted a qualitative evaluation of various capabilities for ARA devices leading to its adoption by the project. Specifically, CHI@Edge can be used as a provisioning and fleet management infrastructure for the base stations and user equipment such as agricultural ground vehicles, robots, phenotypic cameras and sensors, which will act as edge devices enrolled on the CHI@Edge controller. CHI@Edge can be extended with ARA for wireless resource modeling and management and for wireless guards, extending its capabilities. The evaluation yielded, among others, the need for packaging of the CHI@Edge infrastructure for independent testbed deployment (CHI@Edge in a Box) as well as potential for extension for wireless spectrum reservations.

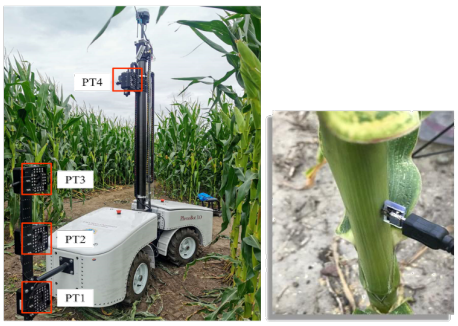

Figure 2: PhenoBot 3.0 with four tiers of stereo cameras for high-resolution phenotypic traits acquisition of maize plants – an NSF sponsored project conducted in Dr. Lie Tang's Automation Robotics Lab at Iowa State University's Agricultural Biosystems Engineering Department and Plant Sciences Institute. (Included with the permission of Dr Lie Tang.)

3.3 Using CHI@Edge to Choose Optimal Edge Topologies for Real-Time Autonomous Fish Surveys

Figure 3: UFL equipment used for autonomous fish surveys. (Included with the permission of Kevin Boswell and Leonardo Bobadilla, UFL)

Jonathan Tsen (FATEC Shunji Nishimura), Leonardo Bobadilla (Florida International University), Kevin Boswell (Florida International University), and Jason Anderson (University of Chicago)

Monitoring marine ecosystems and ensuring the sustainable use of marine resources presents unique challenges spanning socio-economic, technological, and dynamic ecological processes. This team is researching configurations for autonomous vehicle use that can recognize fish types and adjust floating vehicle paths in real-time to align their trajectory with the scientific mission.

The team first used the static pool of CHI@Edge resources to compare performance of different device types and device to cloud configurations. Additionally, the Chameleon GPUs were used to train the AI models on data collected from the CHI@Edge devices, as computational and storage needs of such training exceed edge device capability. Ongoing work is using CHI@Edge SDK, paired

with ArduCam camera, for deployment in the field trials. Overall, this project made use of both CHI@Edge and Chameleon cloud resources (cloud storage as well as bare-metal GPU instances), separately as well as in conjunction for integrated workflows, the JupyterHub environment, and the plugin mechanisms to enable camera integration with the container.

3.4 FlyNet: A platform for edge-to-core workflows

Michael Zink (University of Massachusetts Amherst) and Eric Lyons (University of Massachusetts Amherst) The Flynet project provides an architecture and tools that enable scientists to include edge computing devices in computational workflows. The edge computing resources can include video and weather processing, drone collision detection, and drone route planning. Currently, the project is investigating integrating compute and networking infrastructure, in network processing, end to end monitoring, and leveraging Pegasus workflow management for in-network and edge processing.

The team used Chameleon to run on edge devices available via the static device pool and combined edge devices with resources in the cloud for training. CHI@Edge has been used with iperf3 server images to test bandwidth measurements in smaller devices and FFMPEG servers, which receive the drone video. These tests help determine ideal routes for drones using an optimization function that takes into account the real time instantaneous load on CHI@Edge devices tracked via a Prometheus interface.

3.5 Pegasus at the Edge

Ryan Tanaka (University of Southern California), George Papadimitriou (University of Southern California), Ewa Deelman (University of Southern California)

The popularity of IoT devices, used in applications ranging across environmental monitoring, transportation, and surveillance, motivates the development of tools that support distributed computing on devices with limited resources at the edge. Pegasus [1], a fully featured workflow management system (WMS), was developed to enable scientists in a wide variety of domains including astronomy, seismology, and physics to easily develop, run, monitor, and debug their scientific workflows on a diverse range of computing platform such as grids, clouds, and HPC systems. The team explored the question of how workflow systems such as Pegasus can be adapted to orchestrate scientific workflows on this new resource limited paradigm.

The authors explored running Pegasus jobs on IoT devices to understand what adaptations are needed to use the devices effectively. The team used resources from the static CHI@Edge pool, in containers deployed via Chameleon's JupyterHub interface using python-chi. CHI@Edge provided a convenient tool for this exploration because the authors of the study didn't need to set up the devices themselves and the supported devices provided a sufficiently wide range of options for this exploration. FFmpeg, a multimedia framework which supports video streaming, was benchmarked on CHI@Edge Raspberry Pi and Jetson Nano edge devices to investigate various resolution streaming scenarios and understand which device is better for a given role and how the different architectures might affect potential bottlenecks.

3.6 Using CHI@Edge to Increase Pedestrian Safety

Airam Flores (University of Texas at El Paso), Abdullah Imran (University of Texas at El Paso), and Shirley Moore (University of Texas at El Paso)

Applying the same sensor technology used in self-driving cars to increase pedestrian safety, LiDAR (light detection and ranging) sensors and video cameras can be used to collect and analyze near-miss data to understand pedestrian and vehicle interactions. With a combination of sample data and data from installed devices at the University of Texas El Paso, the authors set out to evaluate the data and explore machine learning techniques to determine the best approaches for near-miss analysis.

The team's experimental plan was to proceed in the following stages: (1) use the sample data to orchestrate computation from edge to cloud using the static CHI@Edge device pool; (2) evaluate different options for edge servers, including computation power needed for video analytics on the edge and using specialized processors for lower power consumption, with the objective of this stage of exploration being to provide data for developing an eventual installation plan; (3) connect the LiDAR sensors and video cameras to CHI@Edge infrastructure via the SDK to experiment in the field. The team presented its experiences from the first stage of this plan and discussed its plans for later stages as well as strategies for performance and reproducibility optimization as well as privacy and confidentiality processing.

3.7 Understanding Reliability on Shared Edge

Haryadi Gunawi (University of Chicago), Junchen Jiang (University of Chicago)

Shared edge environments enable multiple edge applications to share network bandwidth and compute resources. Coupled with computing and bandwidth usage churns created by edge computing applications exploiting uneven distributions of input data streams, this could lead to significant challenges in performance reliability of the underlying edge devices. To investigate this, this project designed a plan to examine and minimize networking and computing workload churns, using the bursty resource demand of video analytics pipelines as a case study.

The project provisioned CHI@Edge devices from the static device pool, including Raspberry Pis and Jetson Nanos, and connected with Chameleon bare metal resources and Docker containers to run a server-driven video analytics pipeline which used server-side feedback to iteratively encode videos at low quality, while still allowing accurate computer-vision inference. The Raspberry Pis were used for simple vision tasks while the Jetson Nanos were a better fit for specific vision tasks though the authors found that the added benefit was easily connecting the edge devices to the bare metal Chameleon servers.

3.8 Applications and Network Measurements for Software Defined Radio on CHI@Edge

Taimoor Islam (Iowa State University), Isaac Darling (University of Chicago), Michael Sherman (University of Chicago)

Bandwidth and latency intensive applications are becoming increasingly important in rural areas – agricultural robots with high throughput cameras for phenotyping, real-time control for vehicles in precision livestock farming, and AR and VR for education. This need isn't well met by current cellular systems due to the lack of density and investment in infrastructure. Software defined radios (SDR) can be used to evaluate future wireless network technologies, sending data from cameras and sensors to be processed on the edge or the cloud.

This project enrolled two X86 hosts into CHI@Edge using the SDK, and attached SDRs to each. The hosts run a software stack – SRSLTE – that configures the radio to act as a 5G base-station or client, and creates an IP network interface on each host. This network interface was passed into containers provisioned by CHI@Edge. For the demonstration, Jupyterhub was used to launch Iperf in the deployed containers, measuring bandwidth across the radio link. The project compared the "default" network interface on the edge device to the new radio interface and logged metrics from the radios, showing how protocol parameters varied with the throughput.

4 DISCUSSIONS AND FEEDBACK

The workshop lightning talks were asked to comment specifically on desirable features of an edge-to cloud testbed and were followed by discussion sections with "drill down" questions exploring desires for general use case support, hardware and configuration features, and priorities. Discussion sections were accompanied by survey/poll questions where attendees were asked questions similar to the panel with multiple choices of top preferences/options. Much of the feedback was also obtained via discussion with CHI@Edge users on the mailing

list in the months leading up to the workshop. This section summarizes the collective insight gained from these discussions.

4.1 General Shape of an Edge to Cloud Testbed

From edge to cloud via consistent interfaces. Overwhelmingly, the ability to access both edge and cloud resources via a set of consistent interfaces and as part of the same infrastructure/experiment was noted as essential to experimentation. The use cases ranged from storing and serving edge data in/from the cloud, using cloud for training ML models to be deployed on edge, to defining experiments that span IoT, edge, and cloud as part of an integrated interaction. Of significant importance was also support for general features provided by a testbed, such as access via federated identity, the ability to structure experiments via non-prescriptive yet easy to use mechanisms such as a Jupyter notebook (in particular when this can be applied seamlessly across both edge and cloud resources), and mechanisms for sharing of such experiments or experimental templates/patterns such that they can be customized to address new problems.

SDK. The most innovative edge testbed capability was the ability for users to easily and dynamically add their devices to the testbed in such a way that they are connected to its resources and are shareable. This is also the hardest requirement to address fully, as this capability implies that resources that used to be operated in datacenters owned by the testbed provider, are now operating at large and outside of the control of testbed provider; this fundamentally changes the assumptions we make about resources and their operation in a way that can be only partially addressed via CHI@Edge. Significantly, edge devices are often associated with a deployment context, as they provide a processing backbone for a range of sensing and actuating "peripherals", i.e., IoT devices from cameras to software defined radios (SDRs), to environmental sensors. It often does not make sense to make those devices publicly shareable; for example, access to edge devices associated with self-driving vehicles in the autonomous vehicles use case described in Section 3.1 can only be meaningful to users actually engaged in the class. This observation led to the requirement for restricted sharing, i.e., a capability to restrict leases on vehicles only to a group of collaborators or students defined by the device owner. Overall, this requirement leads to the development of a *part-time ownership model* for the testbed in which users provide some devices/resources for limited sharing, while the testbed provides and supports a residual resource investment, as well as methods for sharing and connecting (i.e., in the form of an SDK that the user installs on the devices as well as shared services). In other aspects however, removing testbed resources from the datacenter raises a series of challenges that include physical vulnerability, complex ownership relationships (e.g., separate owners for devices and networks), and questions of accessibility for routine operations and the related operational models. Those will have to be addressed overtime as our understanding of the changed deployment context in the ways in which it can operate deepens.

Federation. With the higher distribution and more interesting deployment settings that the edge to cloud, distributed ownership, testbed implies, there is also a desire to experiment with, or simply account for, different networking capabilities. This implies the need for partnership and technological exchange with other testbeds, in particular testbeds providing networking capabilities such as FABRIC and the PAWR testbeds. Since connectivity in many scenarios is a "make or break" feature of experimentation there is much higher incentive for technology adoption as well as testbed use; this came out clearly across all types of interactions with users.

User to User Sharing. Finally, it was noted many times during the workshop that users are interested in leveraging artifacts created by other users ranging from educational modules, customizable experiment templates, specific experiments (e.g., evaluating efficiency of data transfer as in the Pegasus implementation), machine learning models and other specific tools that can provide a foundation block for new experiments, data feeds/streams that can be used to emulate specific experimental settings, and others. While such sharing is in principle always attractive, it is not currently a community practice to share and reference digital artifacts on the same level as e.g.,

research papers. Entering a new area of research means that the incentive to gain experience quickly – as can be done by leveraging existing content – is greater, fewer available artifacts and smaller community give sharing greater focus, and users bring less of a baggage of established habits – all making sharing of digital artifacts more feasible, especially if it comes in the context of immediate gratification provided by an artifact available in the context of a testbed on which it can be run.

4.2 Hardware, Peripherals, and Data Sources

Hardware makeup and availability is an important aspect of a testbed; however, unlike in a sole ownership case, a mixed ownership testbed may need to support more hardware types and configurations to integrate devices added by users. Also different from the situation in the datacenter is emphasis on diversity rather than scale: different types of hardware are important because use cases emphasize patterns of evaluation and reproducing an experiment in many different settings. It was also remarked that edge devices are tied to specific use cases so having more diversity in hardware allows them to cover more use cases. Thus, in a mixed ownership testbed, it is expected that the residual in-house part of the testbed will cover significant diversity at scale that will be driven primarily by the number of simultaneous experiments, while individual experiment scale and distribution will be achieved by adding user devices.

The particular hardware type requests confirm the interest in diversity in that they are relatively uniformly distributed across different hardware types with pointed preference for higher power edge devices, such as Jetson AGX or x86 edge servers, that are also more expensive and thus better amortized in a testbed than via individual ownership, especially for exploratory projects. Also interesting, though more from the perspective of dealing with the volume of experiments, were additional lower power edge devices and all types of devices with GPUs such as NVIDIA Nanos (the initial CHI@Edge deployment was fully utilized by the end of the summer so that it was difficult to find available resources) . Finally, more investment in diversity meant interest in devices with FPGAs, and to a lesser extent devices with TPUs such as Google Coral.

Users are interested in capturing the potential deployment context of an edge device in one way or another. This could be done via a range of peripheral devices that may be associated with edge; while this is less likely to be of interest for devices deployed in a datacenter (i.e., Chameleon owned), understanding those needs is important from the perspective of providing plugins for user owned devices. Another way of integrating deployment context is by emulation such as simulating camera data feeds; in this case user-shared data is going to be invaluable, underscoring the general need for sharing of digital artifacts. Of the most interest were data from cameras or environmental sensors though users also brought up data from acoustic sensors and software defined radios.

4.3 Reconfiguration, Images, and Deployment Styles

There is a trade-off between the level of reconfigurability, isolation, and privilege a system offers to users and its cost. Bare metal reconfiguration is typically considered "gold standard" for CS systems experimentation because it provides a lower bound for the typical set of requirements: users may need to access performance counters, leverage specific capabilities (e.g., NET_ADMIN to reconfigure the network of a device), or customize kernel to build modules for a peripheral, such as a camera, SDR, or other attached device. Further, it is not always easy or possible to say up front what capabilities will be needed, or how those needs might evolve as the experiment progresses, so stating bare metal reconfiguration as a requirement is often safest. At the same time, bare metal reconfigurability results in boot times orders of magnitude longer than from virtual machines and containers; single tenant usage means that expensive multi-core machine cannot be subdivided into smaller units; and in the case of edge devices specifically, power, feature set and accessibility considerations may favor lighter weight though less featureful reconfiguration methods. Not least, there is the difficulty of implementation: many edge

devices do not support features like IPMI and while carrier boards providing equivalent functionality are being developed it is not clear that they will fit all deployment contexts.

All in all, for relatively capable devices (roughly, ones capable of running Linux) bare metal reconfiguration using carrier boards for datacenter deployment should be both possible and useful; it will allow users to experiment/develop on a wide range of device types and port the results to more difficult deployment scenarios – though it will require significant adaptation to the existing system. Bare metal reconfiguration may also be possible in certain settings for devices when deployed in the field – though such deployment would require carrier boards (not always feasible) and further system adaptation to support the "out of datacenter" setting.

For most configurations in the field, lighter weight reconfiguration via some sort of container is more promising. However, while the much requested privileged containers would offer bare metal capabilities they cannot be supported as they incur the risk of an escape or breakout. A more feasible option is to use capabilities, finer-grained mechanisms to give Linux capabilities to the container process, which can e.g. allow performance monitoring while preventing access to more privileged areas of the system. In cases that require no elevated privilege and little or no reconfiguration (rare and hard to maintain in the long run) an approach that dynamically creates local user accounts (similar to grid computing) might be appropriate. Finally, for deployments consisting of less capable devices any interaction will have to depend on capabilities offered by the system they can support.

It seems likely that with the devices representing a spectrum of capabilities, and leaving the datacenter which creates a new set of security challenges, the "gold standard" of bare metal reconfiguration will be both less available and increasingly expensive. The quest to offer users the best possible package of reconfiguration, isolation, and access privilege – including on less capable devices – is therefore likely to be represented by a spectrum of reconfiguration capabilities pertaining to different capability/deployment options rather than the "one size fit all" bare metal reconfigurability.

4.4 Priorities

In terms of priorities for edge testbed development, both sessions in which this question was discussed expressed a preference for a stable (i.e., non preview) edge testbed with a limited set of features as a priority against new features, hardware, or otherwise expanding the set of supported experiments further. This indicates that while the presented initial implementation does not yet provide support for many types of experiments, it is well aligned with the generally desirable shape of a testbed and provides a baseline of useful features. Users also expressed a desire for more hardware available in this way which again testifies to the fact that the basic functionality captures the principal set of needs well.

The second most desired priority was general availability of an edge SDK that would allow all users to add their devices to the testbed, and indeed many of the presented applications would not have been possible without an early version of this feature. This is consistent with the outcome of discussion on the general shape of the testbed where the two features that emerged as most desirable were the ability to configure edge and cloud resources via a consistent set of interfaces and to make user-owned devices shareable via the testbed. An extension of this feature was the ability to have a package of CHI@Edge to configure a private testbed, potentially consisting of both edge and cloud resources, supporting sharing of edge devices.

Further discussed improvements consisted of adding more features (see above), improving the general quality of implementation including factors such as reliability, scalability, and (in the case of the SDK) manageability.

5 CONCLUSIONS

Experiments conducted on the Chameleon edge testbed during the summer of 2021 were largely successful in themselves but also offered important insights that led to refinement of our understanding of what the platform supporting edge-to-cloud experimentation should provide. Findings emphasized the ability to use edge devices in

conjunction with cloud resources through a consistent set of interfaces, access to a diverse set of resources, as well as general benefits of working within existing testbed such as federation with other testbeds and facilities for research sharing. At the same time, there is a clear interest in user operated devices via some form of SDK and a modality that would allow users to share devices that they own and operate with a limited set of collaborators (i.e., restricted sharing), effectively creating isolated sub-testbeds that use larger, residual resources of the main testbed (or testbeds) as well as its sharing services. In extreme circumstance, those devices could be insecure, operate in a variety of conditions (including limited accessibility, limited connectivity, device functionality, and power access), and thus the degree of reconfiguration supported or feasible may differ – while many experiments are possible without addressing these issues, ultimately solutions in this space will be required. By assuming a high level of control by a device owner, CHI@Edge currently avoids those issues at the cost of support for a more limited set of experiments.

One of the most important insights this points to is an evolution of the definition of a testbed. While the conventional idea of a testbed is that of expensive hardware resources, owned and operated by the testbed provider, and available in a datacenter, the edge testbed use case represents an inversion of that model in that hardware resources (typically inexpensive), are owned and operated by users, and often deployed outside of a datacenter. In this setting, the services provided by a testbed is an effective implementation of sharing as well as a connection to residual/cloud testbed resources, shareable in ways consistent with the edge devices. Specifically what effective sharing represents is increasingly important in this setting and refers both to defining non-prescriptive ways of sharing hardware (where "non-prescriptive" means supportive of as wide a range of experiments as possible) as well as shared services, tools, and methodology that facilitate experimentation and enable community sharing.

Another interesting insight is the trend towards heterogeneity of sharing/isolation instruments. In the past, bare metal reconfiguration has generally been the preferred approach because even if not all experiments needed it, many did, and it is hard to predict when starting an experiment how its requirements might evolve. This made bare metal reconfiguration a natural "lower bound" or "one size fit all" solution. However, bare metal reconfiguration also has a cost in terms of functionality requirements (e.g., for many edge devices bare metal reconfiguration is likely to require additional carrier boards), startup times, unit of hardware sharing (i.e. provisioning bare metal nodes implies allocation of a whole node to a user; in contrast, provisioning VMs can effectively split the hardware resources of a node between users), as well as operational factors such as power draw, particularly important for edge deployments. In the case of edge devices this cost is often a critical deployment factor and will thus determine what type of reconfiguration capability a particular installation might expose. It is therefore likely that in the future we may see a spectrum of reconfiguration methods, with different methods used for different devices and/or deployments.

6 LINKS AND REFERENCES

- CHI@Edge website: https://chameleoncloud.org/experiment/chiedge/
- CHI@Edge mailing list: https://groups.google.com/g/chameleon-edge-users
- Workshop Agenda: https://chameleoncloud.org/chiedge-community-workshop/
- Results form discussion polls: https://docs.google.com/document/d/1_3ygzDwKpitaLdDwo8VYMxQ8ykx3n9-j3tqBdCLMPeA/editheading=h.gs8a3qi30bws

REFERENCES

[1] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, M. Cevik, J. Colleran, H. S. Gunawi, C. Hammock, J. Mambretti, A. Barnes, F. Halbach, A. Rocha, and J. Stubbs, "Lessons learned from the chameleon testbed," in *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[2] K. Bahmani, “Biometric Research in the Cloud,” January 21, 2021. [Online]. Available: https://chameleoncloud.org/blog/2021/01/21/biometric-research-cloud/
[3] Z. Chai, “High-Performance Federated Learning Systems,” May 17, 2021. [Online]. Available: https://chameleoncloud.org/blog/2021/05/17/high-performance-federated-learning-systems/
[4] D. Cole, “A New Facial Authentication Pitfall and Remedy in Web Service,” November 16, 2021. [Online]. Available: https://chameleoncloud.org/blog/2021/11/16/a-new-facial-authentication-pitfall-and-remedy-in-web-service/
[5] “CHI@Edge.” [Online]. Available: https://chameleoncloud.org/experiment/chiedge/