Composable Infrastructures for an Academic Research Environment: Lessons Learned

Lance Long
Electronic Visualization Lab, Computer Science
Univ Illinois Chicago
Chicago, Illinois
llong4@uic.edu

Timothy Bargo
Electronic Visualization Lab, Computer Science
Univ Illinois Chicago
Chicago, Illinois
tbargo2@uic.edu

Luc Renambot
Electronic Visualization Lab, Computer Science
Univ Illinois Chicago
Chicago, Illinois
renambot@uic.edu

Maxine Brown
Electronic Visualization Lab, Computer Science
Univ Illinois Chicago
Chicago, Illinois
maxine@uic.edu

Andrew E. Johnson
Electronic Visualization Lab, Computer Science
Univ Illinois Chicago
Chicago, Illinois
ajohnson@uic.edu

***Abstract*—Composable infrastructure holds the promise of accelerating the pace of academic research and discovery by enabling researchers to tailor the resources of a machine (e.g., GPUs, storage, NICs), on-demand, to address application needs. We were first introduced to composable infrastructure in 2018, and at the same time, there was growing demand among our College of Engineering faculty for GPU systems for data science, artificial intelligence / machine learning / deep learning, and visualization. Many purchased their own individual desktop or deskside systems, a few pursued more costly cloud and HPC solutions, and others looked to the College or campus computer center for GPU resources which, at the time, were scarce. After surveying the diverse needs of our faculty and studying product offerings by a few nascent startups in the composable infrastructure sector, we applied for and received a grant from the National Science Foundation in November 2019 to purchase a mid-scale system, configured to our specifications, for use by faculty and students for research and research training.**

This paper describes our composable infrastructure solution and implementation for our academic community. Given how modern workflows are progressively moving to containers and cloud frameworks (using Kubernetes) and to programming notebooks (primarily Jupyter), both for ease of use and for ensuring reproducible experiments, we initially adapted these tools for our system. We have since made it simpler to use our system, and now provide our users with a public facing JupyterHub server. We also added an expansion chassis to our system to enable composable co-location, which is a shared central architecture in which our researchers can insert and integrate specialized resources (GPUs, accelerators, networking cards, etc.) needed for their research.

In February 2020, installation of our system was finalized and made operational and we began providing access to faculty in the College of Engineering. Now, two years later, it is used by over 40 faculty and students plus some external collaborators for research and research training. Their use cases and experiences are briefly described in this paper. Composable infrastructure has proven to be a useful computational system for workload variability, uneven applications, and modern workflows in academic environments.

***Keywords*—composable infrastructure, deep learning, visualization, resource management, workload management, user workflow, composable co-location, infrastructure as code**

I. Introduction

Upon being introduced to *composable infrastructure* [13] in 2018, we saw tremendous potential to College of Engineering faculty who are pursuing fundamental science and engineering research and research training in *deep learning* (data mining and data analytics, computer vision, natural language processing, artificial intelligence, machine learning), *visualization* (simulation, rendering, visual analytics, video streaming, image processing), and a *combination of deep learning and visualization* (e.g., when data is so large that it cannot be easily visualized, then deep learning is used to extract features of interest to be visualized).

There was growing demand among faculty for GPU systems; many purchased their own individual desktop or deskside systems that required power, maintenance and support, a few pursued more costly cloud and HPC solutions, and others looked to the College or campus computer center for GPU resources which, at the time, were scarce. We recognized that composable infrastructure's scalability and agility would provide benefits for on-premise computation over traditional cloud platforms and clusters that are rigid, overprovisioned and expensive. It would address the needs of our academic researchers with a sandboxed environment to discover and assess new techniques and approaches to solving problems while simultaneously providing secure environments for sensitive research.

Major cyberinfrastructure (CI) projects realize the benefit and expressive power of *Infrastructure as Code,* where the user (administrator, CI researcher or data scientist) describes the required hardware and configuration, not through a portal and series of panels (web-based gateways) but through code and APIs running inside a programming notebook (primarily in Python within Jupyter). Projects such as Nautilus (PRP/UCSD) [1], Chameleon [2] and FABRIC [3] are doing this at different levels. Nautilus is a hypercluster that runs Big Data applications supporting Jupyter Notebooks. Chameleon provides bare-metal nodes that can be provisioned and configured through Python to build reproducible experiments. Similarly, FABRIC lets a user build virtual machines with specific requirements (in terms of SSD, NIC and GPU) that are passed through from the host machine to a virtual machine.

In 2019, after surveying the diverse needs of our faculty and studying product offerings by a few nascent startups in the composable infrastructure sector, we applied for and received a grant from the National Science Foundation to purchase a mid-scale system, configured to our specifications, for use by faculty and students for research and research training. We purchased a system that we named *COMPaaS DLV – COMposable Platform as a Service Instrument for Deep Learning & Visualization [15].* It was delivered in November 2019 and access was provided to

COMPaaS DLV is funded by NSF award #1828265 to the University of Illinois Chicago.

faculty in February 2020. Two years later, it is now used by 40 research faculty and students in four College of Engineering departments and some external collaborators.

II. COMPaaS Architecture

Composable infrastructure is not standardized; different companies (e.g., Liqid, GigaIO, HPE, Fungible, etc.) use different fabric technologies and have different models of composable. In 2018-2019, there were very few companies to choose from, but our solution was guided by the diverse needs of data-intensive scientific research and research training in academia, not the needs of commercial companies or data centers, and flexibility was key. We went with the Liqid solution, described here.

"As traditional computing environments falter under the demands of AI-centric, dynamic applications driving economic expansion, Liqid's innovation in composable infrastructure provides a comprehensive platform to optimize and efficiently architect data centers to address the evolving requirements of a data-rich world," said Sumit Puri, Liqid CEO [4]. Liqid's composable infrastructure solution supports multi-tenant orchestrations (VM, containers), bare-metal, analytics, and Artificial Intelligence/Machine Learning (AI/ML), providing the flexibility to dynamically adjust requirements and scale (storage, GPUs and other accelerators, and 100Gbps and specialty NICs).

COMPaaS replaces a traditional unified environment with agile (modular and extensible) pools of CPUs, GPUs, storage and networking, interconnected with a high-bandwidth configurable fabric (PCI-express, or PCIe). It reduces the essence of a server to bare-metal elements – compute, GPU, storage and networking – that form a fluid pool of resources that can be uniquely configured and appropriately sized to run multiple applications simultaneously.

The COMPaaS system is:

- *Highly flexible.* Computer components are treated as pools of resources. Each application defines what resources it needs and the infrastructure composes, or combines, them on the fly. Bare-metal servers are provisioned 'right sized' and resized as needed.
- *Scalable.* As more infrastructure is added, it is auto-integrated with existing infrastructure and becomes part of the pool of capacity, supporting composable co-location of academic resources.
- *High throughput.* Its components are interconnected with a high-speed internal fabric. Big Data moves quickly among CPU, GPU, networking and storage at optimum speed with little to no bottlenecks.

COMPaaS (Figure 1) was designed and built in collaboration with Liqid and Dell (who provided the compute nodes and top-of-rack network management switches). It consists of two 42U racks with a total of 24 compute nodes (Dell servers) with PCIe HBA interfaces connected to a PCIe infrastructure (switches and enclosures). The enclosures host PCIe composable devices: 64 high-end Nvidia GPUs (32x V100 and 32x T4), 153TB of NVMe SSDs, 6TB of nonvolatile Intel Optane memory, and 16 100Gbps network interfaces. The two classes of GPUs were selected as a balance between cost and application requirements: large GPUs (V100) are used for training workloads and smaller GPUs (T4) are mostly used for inference tasks and for development and testing. Each rack is a separate composable infrastructure consisting of multiple edge PCIe fabric switches interconnected to a PCIe fabric management switch. The enclosures hosting the devices along with the compute nodes are connected to the edge switches.

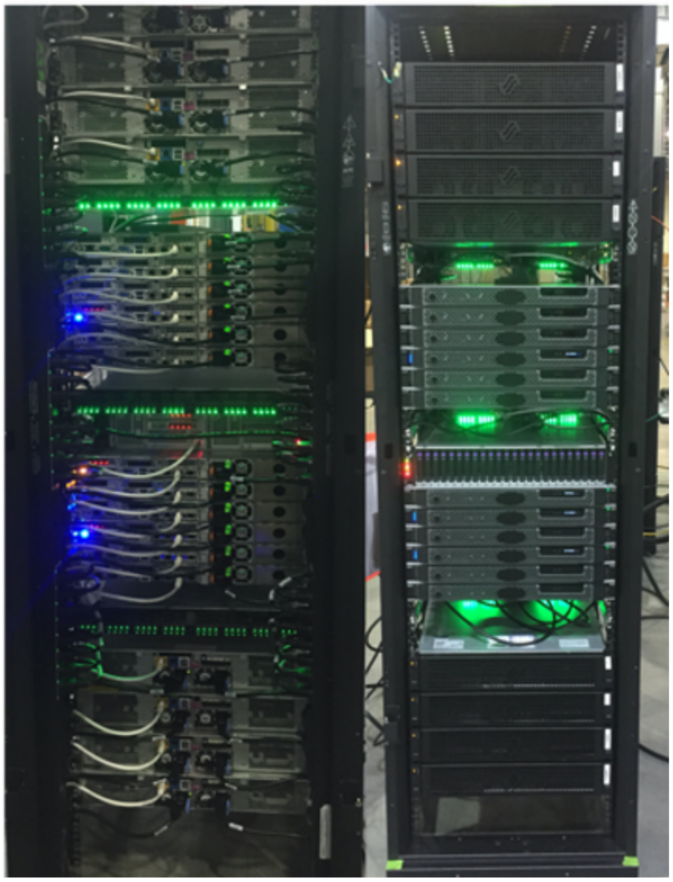

Fig. 1. COMPaaS Hardware Racks

III. Resource Management

COMPaaS came with Liqid's resource management layer that consisted of a proprietary point-and-click graphical user interface (GUI) and REST API (Figure 2). A REST API (also known as RESTful API) is an application programming interface that conforms to the constraints of REST architectural style and allows for interaction with web services. These services enable an infrastructure engineer using the GUI to combine PCIe-connected resources from an available pool, configure the PCIe fabric switch, and connect the composable elements. While this approach maintains system security, the user interface is time consuming to use and difficult to deploy to our end users – data scientists and computer science researchers – who lack knowledge of the underlying hardware.

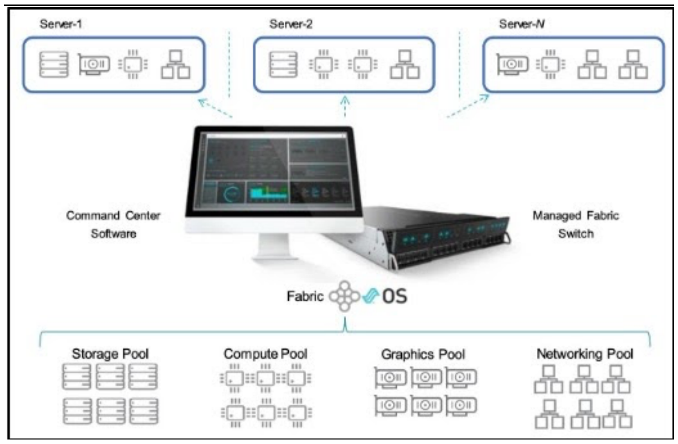

Fig. 2. Composable Infrastructure (Liqid)

Given that science workflows can be quite diverse, it was our goal to implement a solution to enable our users to dynamically reconfigure, on demand, their own application-specific machine resources. To ensure adoption, we wanted to seamlessly integrate our solution with their scientific workflows. Modern workflows are progressively moving to containers and cloud frameworks (using Kubernetes) and to programming notebooks (primarily Jupyter), both for ease of use and for ensuring reproducible experiments. We succeeded in reducing the complexity and making operations more manageable by using Jupyter Notebooks to manage infrastructure, applications and visualizations. Recently, we expanded our system with a public-facing JupyterHub server providing all users with the ability to enter through a Jupyter instance. It will soon become the only way to access the system; i.e., users will not use external terminal access with *ssh.*

Our solution entailed our development of a management layer that abstracts Liqid's REST API [16]. This higher-level layer provides Python support for the composition and management of pool resources. Once a machine is composed with the specified resources, we use MaaS.io (Metal as a Service) [6] to provision, commission, and deploy servers, either as bare-metal instances (custom OS images) or as container environments (using Kubernetes). MaaS.io is used for node management and operating system deployment. Ubuntu and CentOS distributions are provided with custom GRUB options to hot-plug composable elements within a running OS. Hot-plug capabilities in modern systems enable system engineers to reconfigure the capabilities of a machine (GPUs, storage, NICs), on-demand, from application specifications. Dell, working with Liqid, provides BIOS integration to facilitate these features. Dynamically, we can programmatically *unplug* components from one compute node (as long as the device is not in use) and plug it into another compute node using low-level fabric APIs (reconfiguration using PCIe switches) to achieve a *Software Defined Infrastructure.*

IV. Workload Management

COMPaaS did not come with any Liqid-supported workload/application management tools, so we implemented a software layer to handle it. Using Containerd and the Nvidia GPU operator, we implemented Kubernetes as the container orchestrator for GPU workloads. Kubernetes provides a responsive software-driven deployment architecture that increases flexibility in running and moving jobs across different hardware configurations quickly. By layering Kubernetes over composable hardware, we enabled our users to create containerized applications with reproducible hardware and software.

Specific GPUs can be inventoried, reused, or tested against, to reduce and understand variability between application runs. Kubernetes is ephemeral in nature, an environment that inherently supports self-healing, auto-scalability and resource monitoring. The fluidity of composable infrastructure resources balances well with the Kubernetes model of execution. Within our Kubernetes deployment, we utilize several services that support application execution: reverse proxy (Traefik), load balancing (MetalLB), monitoring (Prometheus), and Kubernetes networking (services, ingress). Storage is provided through local NVMe persistent drives for applications requiring high-speed storage or through a NFS (Network File System) to access large, remote data storage at slower rates or when users prefer to directly connect to their IDE (integrated development environment) for development. The required Kubernetes pod description file (YAML syntax) requests GPU, networking and storage resources. From this request, we extract a user's composable requirements and make API calls to the resource management layer requesting that these devices be added to a node. These hardware changes can be quickly updated for applications by restarting the Kubernetes deployments.

V. User Workflow

Once Kubernetes pods (or deployments) are running, our researchers then launch a JupyterLab Notebook from inside their container (Figure 3). JupyterLab enables researchers to put code, documentation and visualizations into a single computational notebook and then run their code on a remote server through a web interface. They use this instance of JupyterLab to access COMPaaS's resources, exposed externally using Traefik, to execute their code and applications. Such notebooks can be shared and reproduced. JupyterLab is the current version of Jupyter offering a modern experience, with file management, multi-windows, and interactive layout.

We now provide JupyterHub as the frontend where each user receives a dedicated Jupyter instance. The Hub runs on a powerful server, part of our 2021 COMPaaS hardware expansion, providing secure, web-based, public facing access to the infrastructure All backend APIs (MaaS.io, composable hardware, and monitoring) can be packaged into Python modules and pre-loaded into JupyterLab Notebooks. This work demonstrates the potential for a user to 'program' a machine and 'program' an experiment as code using notebooks that are persistent and can be shared.

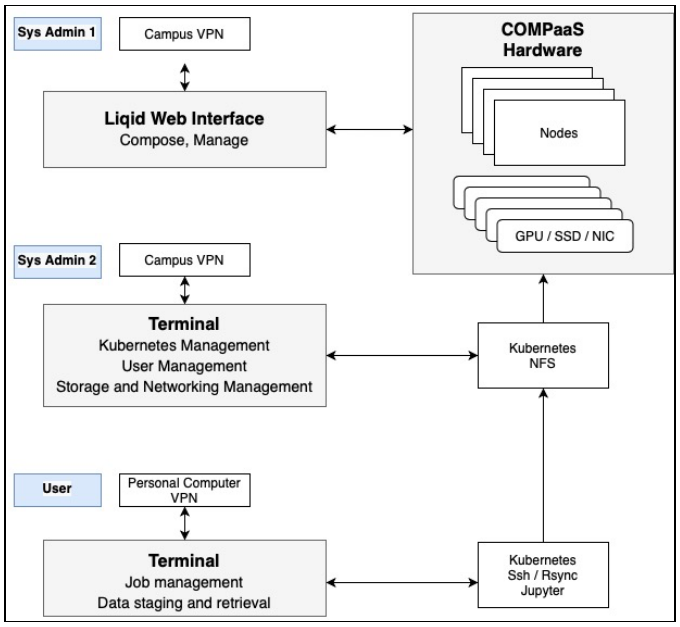

Fig. 3. COMPaaS User Workflow

VI. Composable Co-Location

On-premise hardware is still required for many applications, such as robotics and visualization, that require specialized resources. As familiarity with composable infrastructure grows, the idea of using *composable co-location* instead of traditional servers co-located in racks is promising. By *composable co-location,* we mean the integration of emerging hardware needs without changing the existing core infrastructure.

The ability for departments and researchers to provide and quickly add (co-locate) only those resources required by their applications – GPUs (or other accelerators), specialized networking, etc. – to a shared central composable infrastructure system is far more cost effective than buying desktop or deskside systems that need maintenance and support (often wasting the time of graduate students and/or department IT staff, without building persistent institutional knowledge).

In 2020, we began implementing a GPUoE (GPU over Ethernet) prototype. Using a GPU expansion chassis connected to compute nodes over Ethernet, we were able to compose remote GPUs into our existing composable infrastructure. Our APIs developed for composable infrastructure have been extended to support these remote GPUs.

In 2021, we enhanced COMPaaS with a public-facing JupyterHub server, a modern PCIe fabric and a supporting expansion chassis, thereby making composable co-location available to our users (Figure 4). Researchers can now co-locate their own accelerators or compute nodes with COMPaaS. The JupyterHub server provides secure web-based access to resources while supporting their experiments.

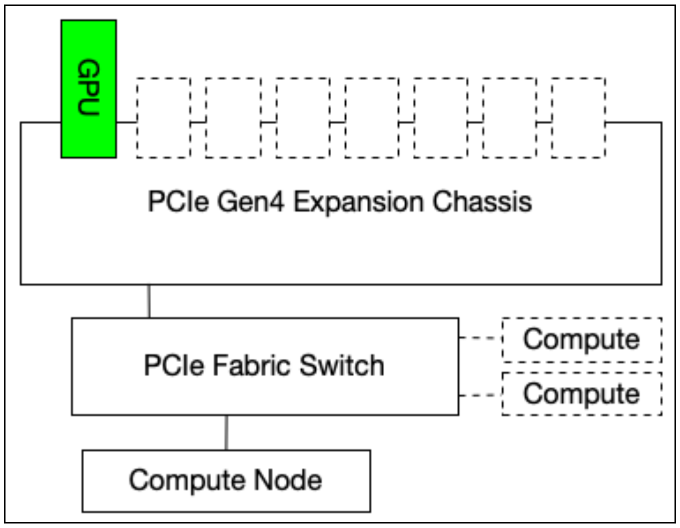

Fig. 4. 2021 Expansion Supporting Composable Co-location

VII. Use Cases

The infrastructure needs of science and engineering research and education are varied. This is exemplified in a recent anecdote by Larry Smarr, UCSD Distinguished Professor: “It all started while UC San Diego computer science and engineering professor Larry Smarr was waiting for coffee in the ‘Bear’ courtyard at the Jacobs School of Engineering a little more than three years ago. While standing in line, Smarr overheard a student say, ‘I can't get a job interview if I haven’t run TensorFlow on a GPU on a real problem’” [5]. Smarr’s PRP research platform [1], developed at UCSD, would later be leveraged to support data-science classes on real-world problems. We envision a similar path for COMPaaS at our institution.

COMPaaS users – faculty from four College of Engineering departments (Computer Science, Civil, Materials & Environmental Engineering, Mechanical and Industrial Engineering, and Electrical and Computer Engineering) – primarily run applications that are GPU-centric for compute, with significant variability in storage and networking around their data requirements.

Computer Science. Applications primarily focus on security, data science, computer vision and Machine Learning (ML). Security projects explore the complexity of modern web applications and the intricacies of security mechanisms that often result in flaws that expose users to significant security and privacy threats. These projects try to develop methods and tools that enable users to understand and more effectively manage retrospective privacy in the context of modern, long-lived, online archives. Composable resources were used to develop Natural Language Processing (NLP)-based domain-specific classifiers that identified data practices stated in privacy policies. Adherence of corresponding applications were then adjusted based on this ground truth [14].

Data-science applications have an intuitive framework that integrates state-of-the-art AI technologies with applications, workflows, smart visualizations and collaboration services to help users access, share, explore and analyze their data, whether local or remote, come to conclusions, and make decisions with greater speed, accuracy, comprehensiveness and confidence. One such project is developing and advancing tools that identify image data in biomedical literature to locate beneficial, targeted publications [7]. This work involves training image classifiers, integrating classifiers into labeling pipelines, designing retrieval user interfaces, and identifying related visual representations.

Computer vision projects include semantic segregation and 3D human pose estimation. Researchers are developing a novel network architecture, termed DependencyNet (dependency network), for semantic segmentation [8]. They also achieved experimental results that demonstrate an effective approach for 3D human pose estimation [9]. Over the past two years, they found COMPaaS to be consistently stable and efficient, with result output as expected, and the group's models achieved state-of-the-art performance on their respective benchmarks.

ML applications include frameworks for many complex real-world reinforcement learning problems, such as the coordination of autonomous vehicles, network packet delivery, and distributed logistics.

Civil, Materials & Environmental Engineering. Applications focus on simulation and modeling. Researchers run data-driven models on high-performance computers to develop an accurate and general neural network ML model that uses crystallographic data to study patterns of synthesizability [10]. They also perform simulations of mass transport in alloys and ceramics. COMPaaS has been performing 1.5-10 times faster than comparable infrastructures they are familiar with. Additionally, they found our use of Jupyter Notebooks to be a significant asset.

Mechanical and Industrial Engineering. Researchers use COMPaaS for three research projects: feature extraction in fluid flow using a Convolutional Neural Network (CNN); column height detection in metallic nanoparticles using a CNN [11]; and, electric vehicle battery state-of-charge estimation using different ML methods [12].

Electrical and Computer Engineering. Researchers recently started running mathematical models of ML algorithms and training language models using long-short term memory (LSTMs).

VIII. CONCLUSIONS

Composable infrastructure enables academic researchers to accelerate the pace of research and discovery by providing them with an on-premise centralized resource and the ability to quickly deploy bare metal or containers with appropriately sized resources, as required.

COMPaaS is a cost-effective, mid-scale, agile resource for College of Engineering faculty. It can serve as an "on ramp" where codes are first developed and optimized before being scaled and ported to more costly cloud and large HPC environments. It has proven to be a useful tool for workload variability, uneven applications, and modern workflows in academic environments. Also, with our introduction of composable co-location, faculty can now add additional specialized components without having to purchase separate on-premise systems. These factors are very important in an academic environment.

COMPaaS's two-rack composable infrastructure system has demonstrated that it is appropriately sized for a multi-department university college. Additional racks can be added based on a college's size and anticipated number of users. For a single department, one rack would likely provide sufficient composable resources to support a broad range of applications. At the time of writing, our infrastructure is fully utilized in terms of GPU, and we are onboarding several new users each week. Our 2021 expansion system is representative of what would be appropriate for a small research lab.

Our Kubernetes orchestration and Jupyter Notebook implementations enable our users to get started quickly and to fully utilize COMPaaS. Without COMPaaS, researchers would continue to utilize desktop or deskside systems that are typically supported and maintained by student researchers who are also trying to do research. COMPaaS enables faculty and student researchers to quickly come to task with dedicated resources that are scaled to their applications. Using containers, multiple researchers can work on the same application in parallel. The skills learned developing codes on composable infrastructure enable users to build scalable applications faster, with knowledge that transfers directly to industry problems.

ACKNOWLEDGMENT

This publication is based on work supported by NSF award #1828265 to the University of Illinois Chicago. We wish to acknowledge our industry collaborator Liqid, Inc., who provided invaluable technical support for COMPaaS. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of our funding agency or collaborator.

REFERENCES

[1] I. Altintas, K. Marcus, I. Nealey, S.L. Sellars, J. Graham, D. Mishin, J. Polizzi, D. Crawl, T. DeFanti, L. Smarr, "Workflow-Driven Distributed Machine Learning in CHASE-CI," https://arxiv.org/pdf/1903.06802.pdf

[2] 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, J. Stubbs, "Lessons Learned from the Chameleon Testbed," Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20). USENIX Association. July 2020

[3] I. Baldin et al., "FABRIC: A National-Scale Programmable Experimental Network Infrastructure," IEEE Internet Computing, Vol. 23, No. 6, pp. 38-47, 1 Nov-Dec 2019, https://doi.org/10.1109/MIC.2019.2958545.

[4] "Liqid Announces $28 Million Series B Funding to Deliver Industry-leading Composable Infrastructure Solutions," BusinessWire, November 13, 2019, https://tinyurl.com/6mrmnj3f

[5] Daniel Kane, UC San Diego News: "From Coffee Cart to Educational Computing Platform," January 20, 2022, https://ucsdnews.ucsd.edu/feature/from-coffee-cart-to-educational-computing-platform

[6] https://maas.io

[7] Juan Trelles Trabucco, Pengyuan Li, Cecilia Arighi, Hagit Shatkay, G. Elisabeta Marai, "Modality-Classification of Microscopy Images Using Shallow Variants of Deep Networks," 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 2379-2385, 2020, IEEE, 2020, https://doi.org/10.1109/BIBM49941.2020.9313467

[8] Mingyuan Liu, Dan Schonfeld, Wei Tang, "Exploit Visual Dependency Relations for Semantic Segmentation," Proc. IEEE/CVF Conference on

Computer Vision and Pattern Recognition (CVPR), pp. 9726-9735, 2021, https://openaccess.thecvf.com/content/CVPR2021/papers/Liu_Exploit_Visual_Dependency_Relations_for_Semantic_Segmentation_CVPR_2021_paper.pdf

[9] Kenkun Liu, Zhiming Zou and Wei Tang, "Learning Global Pose Features in Graph Convolutional Networks for 3D Human Pose Estimation," Proceedings of the Asian Conference on Computer Vision (ACCV) 2020, Kyoto, Japan, November 30-December 4, 2020, https://openaccess.thecvf.com/content/ACCV2020/papers/Liu_Learning_Global_Pose_Features_in_Graph_Convolutional_Networks_for_3D_ACCV_2020_paper.pdf

[10] Ali Davariashtiyani, Zahra Kadkhodaie, and Sara Kadkhodaei, "Predicting Synthesizability of Crystalline Materials via Deep Learning," Communications Materials (a Nature research journal), Vol. 2, Issue 115, 2021, https://doi.org/10.1038/s43246-021-00219-x2021

[11] Marco Ragone, Lance Long, Matthew Tamadoni, Reza Shahbazian-Yassar, Farzad Mashayek and Vitaliy Yurkiv, "Deep Learning for Mapping Element Distribution of High-Entropy Alloys in Scanning Transmission Electron Microscopy Images," Computational Materials Science, Vol. 201, 2022, p. 110905, https://doi.org/10.1016/j.commatsci.2021.110905

[12] Marco Ragone, Vitaliy Yurkiv, Ajaykrishna Ramasubramanian, Babak Kashir, Farzad Mashayek, "Data driven estimation of electric vehicle battery state-of-charge informed by automotive simulations and multi-physics modeling," Journal of Power Sources, Vol. 483, 2021, pp. 229108, https://doi.org/10.1016/j.jpowsour.2020.229108

[13] S.D. Lowe, Composable Infrastructure for Dummies, HPE, John Wiley & Sons, 2016, www.hpe.com/us/en/resources/composable-infrastructure-for-dummies.html

[14] https://gwusec.seas.gwu.edu/privacylabels/root/

[15] M. Brown, L. Renambot, L. Long, T. Bargo, A. Johnson, "COMPaaS DLV: Composable Infrastructure for Deep Learning in an Academic Research Environment," MERIT (Midscale Education and Research Infrastructure and Tools) Community Event Workshop, 27th IEEE International Conference on Network Protocols (ICNP 2019), Chicago, Illinois, USA, October 7, 2019, http://doi.org/10.1109/ICNP.2019.8888070

[16] Zhongyi Chen, Luc Renambot, Lance Long, Maxine Brown, Andrew Johnson, "Moving from Composable to Programmable," First Workshop on Composable Systems (COMPSYS '22), co-located with the 36th IEEE International Parallel and Distributed Processing Symposium, Lyon, France, June 3, 2022, accepted