COMPaaS DLV: Composable Infrastructure for Deep Learning in an Academic Research Environment

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

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

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

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

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

***Abstract*— In today's Big Data era, data scientists require new computational instruments in order to quickly analyze large-scale datasets using complex codes and quicken the rate of scientific progress. While Federally-funded computer resources, from supercomputers to clouds, are beneficial, they are often limiting – particularly for deep learning and visualization – as they have few Graphics Processing Units (GPUs). GPUs are at the center of modern high-performance computing and artificial intelligence, efficiently performing mathematical operations that can be massively parallelized, speeding up codes used for deep learning, visualization and image processing, more so than general-purpose microprocessors, or Central Processing Units (CPUs). The University of Illinois at Chicago is acquiring a much-in-demand GPU-based instrument, *COMPaaS DLV – COMposable Platform as a Service Instrument for Deep Learning & Visualization,* based on composable infrastructure, an advanced architecture that disaggregates the underlying compute, storage, and network resources for scaling needs, but operates as a single cohesive infrastructure for management and workload purposes. We are experimenting with a small system and learning a great deal about composability, and we believe COMPaaS DLV users will benefit from the varied workflow that composable infrastructure allows.**

Keywords—distributed systems, testbed implementation & deployment, composable infrastructure, deep learning, visualization

I. Introduction

The University of Illinois at Chicago (UIC) is acquiring a GPU-based system named *COMPaaS DLV* (currently on order). It will enable campus researchers to pursue science and engineering research – focused on *deep learning* (data mining, data analytics, computer vision, AI and natural language processing), *visualization* (simulation, rendering, 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) – in disciplines from biology and cybersecurity to healthcare and urban sustainability.

Its design utilizes state-of-the-art computer architecture known as *composable infrastructure* [3]. Traditional computer architecture is *static* (a set of components selected at design time); *composable infrastructure* is resource centric. It replaces a traditional environment with fluid (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, networking, and storage – that form a fluid pool of resources, such that different applications with different workflows can be uniquely configured and run simultaneously.

COMPaaS DLV is funded by NSF award #CNS-18282 to the University of Illinois at Chicago.

978-1-7281-2700-2/19/$31.00 ©2019 IEEE

Given composable infrastructure's scalability and agility, it is more beneficial than traditional clouds and clusters that are rigid, overprovisioned and expensive. While commercial public clouds, like Amazon Web Services, offer some functionality, they are primarily designed as a single-node model, not a collection of resources for research, and the networking is not practical, with slow transfer for data in and out at high cost.

II. Composable Infrastructure Experiences

One year ago, UIC purchased a small two-node composable infrastructure system from Liqid to learn about this new architecture. A GUI and REST API lets a user or administrator combine PCIe components from an available pool. The PCIe switch is then configured and elements are interconnected. A predefined OS image boots on the newly defined node. Given our experiences with a few applications (see below), we encountered issues with OS loading, GPU pass-through, OS hot-plug, and application deployment, which we addressed. Liqid's composable system has specific requirements to bring it online; i.e., all the host nodes and composable elements must be brought up before the PCIe switch (Liqid Grid) so the switch can traverse all the elements to do discovery and inventory. Our initial system was a mix of different hardware vendor chassis, which each handled power cycling and power on differently. The PCIe switch also needed a registry of all hardware installed in the system, so we initially found some network adapters incorrectly identified. We went through a variety of PCIe switch and composable software stack upgrades, including a number of manufacturer changes (e.g., Supermicro, Inspur, and most recently Dell servers) to address BIOS and management issues.

Central to our efforts was using composable SSDs as boot devices for host nodes. We planned to host OS images on the PCIe switch and write them to a composable SSD on demand to boot the system. We worked with Liqid to enable OS SSD hot-plug and deployment. Packaging OS images was challenging, with large file sizes, long delays to write images, and storage constraints. We moved our image hosting system to a Pre-Boot Execution Environment (PXE) server hosted externally from the Liqid system, which required a small RAMDisk be installed on the host nodes. We tried several techniques to deploy the OS, including a small compressed source file with a large target write, using software such as *Clonezilla* and *FOG,* and writing a small partition that could be expanded to full disk size. With these techniques, we reduced file size and write time, but it was insufficient to make the infrastructure composable in real time. Given the state of containers and their benefits, we transitioned to Docker to deploy GUI/GPU instances, and then transitioned to Kubernetes for scalability, deployment, and portability.

III. KUBERNETES

Science workflows are progressively moving to containers and cloud frameworks to execute reproducible experiments. Kubernetes provides a responsive software-driven deployment architecture that increases flexibility in running and moving jobs across different hardware configurations quickly. We introduced hardware composability to Kubernetes, enabling our users to create containerized applications with reproducible hardware and software deployment. Alternatives to Kubernetes, such as OpenStack [7] and the Chameleon Infrastructure (CHI) [2], are possible but difficult given the size of our team.

Our current recipe for success is using Ubuntu and CentOS distributions with custom Grub options to hot-plug composable elements within a running OS. The OS locks devices for actively running processes; these device locks must be removed before disabling a composable element while device drivers stay running. Specifically for GPUs running in non-graphical mode, it occasionally requires manually unbinding the device. Learned lessons enabled us to freely remove and add composable elements – i.e., GPU, NIC, SSDs.

Kubernetes is ephemeral in nature, a fluid environment to inherently support self-healing, auto-scalability and resource monitoring. The fluidity of resources of a composable infrastructure perfectly fits the Kubernetes model of execution. Within our Kubernetes deployment, we utilize several services to support application execution: reverse proxy (Traefik), load balancing (MetalLB), monitoring (Prometheus), and Kubernetes networking (services, ingress).

The required Kubernetes pod description file (YAML syntax) requests GPU, networking and storage resources. From this request, we extract composable requirements and make API calls to the PCIe switch requesting those devices be added to a node; composing the elements can happen in under a minute. Upon successful creation, Kubernetes detects the new hardware and makes it available for pod deployments. We then deploy our pod and hand off the location and networking details of the orchestration to the user. Grafana is used for system monitoring.

IV. USE CASES: DATA-INTENSIVE SCIENTIFIC APPLICATIONS

Composable infrastructure enables fast and fluid hardware configuration. We tested a few applications with different workflows (e.g., data gathering, filtering, compute, storing).

Simulations for Complex Turbulent Flows, to design efficient combustion systems, are computationally demanding and produce large quantities of data that cannot be output frequently for visualization due to the computational cost of writing and the sizes of the files [6]. UIC researchers want to investigate flexible approaches to feature extraction, including deep learning, to identify and store shock locations or regions with strong chemical reaction [5]. At SC18, we demonstrated this application running on our small Liqid development system. COMPaaS DVL, with its GPUs and fast I/O subsystem, should enable higher resolution simulations using deep learning.

The *Dynamics of Brain Function* application seeks to understand the dynamics of brain function using global yet very-high spatial and temporal resolution imaging techniques and tools to analyze dynamic social networks [4]. The project's data input is brain images, frames of different imaging techniques over time. Data preprocessing computes an all-pixel by all-pixel correlation for each frame. The correlation can be efficiently GPU accelerated. The application runs both CPU and GPU algorithms on a single computer optimized for 40G throughput. This system utilizes a Liqid SSD element to provide the necessary throughput. COMPaaS DLV may further enable real-time interaction and computational steering.

We are evaluating composing Data Transfer Nodes (DTNs) to receive and store high-resolution visualizations moving from physical storage to a composed GPU platform and then pushed to a large display wall. We are testing DTN services and running perfSONAR pods on our Liqid system. We plan to do performance testing (network and I/O throughput of 100G NICs and SSD cards) to evaluate PCIe interconnection overhead.

V. CONCLUSIONS

Liqid helped us discover and address problems as we learned more about the benefits and limitations of this new architecture. Some applications we tested required RAM and CPUs only, while others required CPUs and GPUs. There remains work to be done in regards to storage and networking. We continue to work with more UIC researchers to understand the nuances of their applications, optimize productivity, and create portable codes using containers and Kubernetes.

We recently placed an order for *COMPaaS DLV* – a moderate-sized Liqid system with 64 GPUs, 24 CPU nodes, NVMe SSDs, 100G networking and large memory pools (Intel Octane). It complements campus production resources, as well as Federally-funded community resources, such as NSF's XSEDE, Blue Waters and Chameleon, and DOE's Argonne Leadership Computing Facility. We tested integration to existing Kubernetes systems at StarLight and CHASE-CI/NRP [1]. The benefit of composability for our users is in the workflow that this infrastructure allows.

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] Chameleon Infrastructure: https://www.chameleoncloud.org/about/frequently-asked-questions/#toc-what-is-chi-in-a-box-

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

[4] C. Ma, R. Kenyon, A. Forbes, T.Y. Berger-Wolf, D. Llano, "SwordPlots: Exploring Neuron Behavior within Dynamic Communities of Brain Networks," Journal of Imaging Science and Technology, 60(1), January 2016, 10405–1-10405-13(13).

[5] M. Monfort, T. Luciani, J. Komperda, B. Ziebart, F. Mashayek, G.E. Marai, A Deep Learning Approach to Identifying Shock Locations in Turbulent Combustion Tensor Fields, In: Schultz T., Özarslan E., Hotz I. (eds) Modeling, Analysis, and Visualization of Anisotropy, Mathematics and Visualization, Springer, Cham, July 2017, pp 375-392, DOI: 10.1007/978-3-319-61358-1_16

[6] D.M. Peterson, M. Hagenmaier, C.D. Carter, and S.G. Tuttle, "Hybrid reynolds-averaged and large-eddy simulations of a supersonic cavity flameholder," AIAA Paper 2013-2483, 2013

[7] S. Telfer, The Crossroads of Cloud and HPC: OpenStack for Scientific Research, OpenStack consortium, 2016, www.amazon.com/Crossroads-Cloud-HPC-Scientific-scientific/dp/15396