BDEC2 Platform White Paper

Motivation

Ubiquitous data and compute are no longer the sole province of high-end HPC, and plans for future exascale platforms must account for the emergence of big data and edge computing in addition to traditional scientific simulation. Huge datasets are now routinely processed in the cloud, and edge devices—sensors, phones, watches, scientific instruments, etc.—generate data more rapidly than we can process it. Widespread use of machine learning has ramifications across the board, both in its traditional role in knowledge extraction, and also in, but also now as an accelerator. Surrogate models enable edge devices to make decisions rapidly based on current data, instead of waiting for expensive calculations, and the same types of models can be used to drastically speed up HPC simulations.

A Converged Platform

Scientists are growing to depend on simulation, measurement, *and* analytics in large-scale data pipelines. There is a need for a converged programming paradigm so that applications can easily take advantage of the entire gamut from HPC facility to cloud to edge, and back again. The challenge is that the computing models in these environments could not be more different. One the HPC side, tightly coupled, high-powered nodes talk to one another in large-scale batch jobs over low-latency networks. In the cloud, persistent services respond on-demand to requests from users and devices, punctuated by bursty machine learning workloads. On the edge, mostly embedded applications collect data and execute in memory- and power-restricted environments over high-latency, low-bandwidth connections. A programming model for all three of these must span orders-of-magnitude differences in scales and computational requirements. Code must be quickly deployable anywhere, regardless of machine architecture, and data needs to be delivered quickly to and from all parts of the pipeline. Machine learning must be applicable at every stage, both as an accelerator and as a way to let the system adapt quickly.

What has to Change

Building a data pipeline that spans form HPC to cloud to edge will require major architecture changes both in hardware and in software. However, we are careful to propose a “converged” platform rather than a unified one. We cannot succeed at our goals by reinventing work that has already been done in the cloud and HPC worlds. We must leverage existing solutions and open standards (cloud APIs like lambdas, Kubernetes, OpenStack), and we must adapt existing cloud and HPC solutions to end-to-end use cases.

Reproducibility and deployment

Users should be able to reliably deploy code anywhere in the pipeline, and there should be no hard software stack requirements at any point in the system. Solutions like containers and serverless computing ("lambdas") provide partial solutions to this problem, but changes are needed to ensure performance. On the HPC and cloud side, containers are insulated from hardware to provide isolation, and they provide a convenient way to bundle dependencies with an application. However, hardware (or lightweight VM) support for isolation at the network, accelerator, and storage level is needed to enable containers to run fast and securely. Moreover, much of today's container infrastructure is targeted at a homogeneous, generic x86_64-Linux platform. We will need better ways to rapidly deploy architecture-optimized containers if applications are to run fast on the converged platform.

Data Architecture

One of the largest challenges facing an end-to-end data analysis platform is data movement. On the HPC and cloud side, the compute power and bandwidth exist to handle massive data sets, but users are responsible for finding them and moving them into place within the HPC center. On the edge, devices frequently need to fetch data, but there is no standard, fast, power-efficient delivery mechanism for these devices. Even in the cloud, databases are typically not deployed in a containerized way, as it is difficult to manage stateful services with today's container frameworks. Rather, cloud services and lambdas rely on object stores like S3 and fetch their data at runtime. If we begin fetching the data only when it is to be used, it is already too late.

The current pull model, where client code must know where and how to fetch each data set, is not sustainable or viable for a converged platform. The system needs full control over scheduling both code *and* data. Workflows' data requirements should be expressed in advance, similar to how we express code requirements with container names in registries today. To achieve this, we need a data naming framework – a registry of data sets that allows them to be referenced in whole or in part by unique names. Users should be able to specify their inputs and run the same job at an HPC center or in the cloud without having to orchestrate data transfers manually. Similarly, code that runs on edge devices should be scheduled with its associated data, and the system should ensure that named data are on the edge with the code at runtime.

Workflows and Security

If there is to be a convergence, the platform must allow scheduling across HPC centers, clouds, and edge devices. Currently, on the HPC side, there are security hurdles that prevent many types of workflows from being scheduled without a human in the loop. At the edge of HPC centers, there is typically 2-factor authentication, so automating anything inside the center from outside is very difficult. Moreover, HPC schedulers are typically controlled with commands that need to be run on login nodes – they do not expose APIs that would allow job scheduling to be automated from cloud applications or orchestrators. Security in a converged platform must be end-to-end, with token-based authentication and well-defined, automatable APIs at every level. HPC centers must engineer solutions that enable automation without compromising security, and a standard

for identity federation and trust needs to be developed to allow centers to interoperate with a world increasingly driven by web services and JSON APIs. Without this, HPC sites will continue to be siloed as they are today.

Ubiquitous Machine Learning

In a converged data-intensive computing environment, machine learning will be present at every level of the platform, and will likely be used to drive decisions for the platform itself. Today, to apply ML to a domain, users must construct their own training sets, build their own models, and retrain them when input data deviates significantly from the training set. For ML to truly be ubiquitous, the barriers to apply it must be minimal. We will need to develop generic simulation harnesses that allow us to generate surrogate models for the kernels in a simulation, and to transfer them quickly to edge devices for decision making. Likewise, for the system to make runtime optimization decisions based on models (e.g., choosing architecture-specific variants of a kernel, or making resource-aware scheduling decisions), the system will need to be profiled and monitored, and system data will be available through the same naming and content delivery mechanisms described above. Ideally, scientists can easily choose when to use raw simulations and when to use surrogates, depending on the situation. Likewise, system engineers can easily learn from past monitoring data and choose when to use heuristics or expert decisions, and when to rely on ML-derived models to make a choice.