A Collection of White Papers from the BDEC2 Workshop in San Diego, California

October 15–17, 2019

Acknowledgment

This workshop was supported in part by the National Science Foundation under Grant No. 1849625.

Big Data and Extreme Scale Computing, 2nd Series, (BDEC2)
Workshop 4: San Diego, California,
October 15-17, 2019

Call for White Papers

As noted in the invitation to this BDEC2 meeting, many governments around the world are ramping up major initiatives to apply artificial intelligence (AI) in nearly every domain of science and engineering, and in many other areas of modern life as well. The meeting in San Diego will focus on issues emerging at the intersection of the global drive to use "AI everywhere," and the need to develop a shared cyberinfrastructure (CI) for science and engineering that spans the "computing continuum," from cloud and HPC data centers to the plethora of new instruments, IoT, and other prolific data generators in the wide "data periphery."

Recognizing the powerful impact that AI will continue to have on numerous fields of science and engineering, previous BDEC2 meetings have highlighted application domains that need both big data and extreme-scale computing, including the need for novel architectures and cyberinfrastructure (CI) to provide edge to cloud computing. The first leading idea that emerged —***AI for Science***— captured the community's recognition that machine learning and deep learning are transforming the world of scientific computing generally, including traditional simulation and modeling. This is a momentous phase change in our approach to scientific inquiry, and it is receiving an enormous amount of attention from many governments around the world. Previous work of the BDEC2 Application working group, under the leadership of Geoffrey Fox, has given us an excellent snapshot of what is happening today in AI for Science (http://bit.ly/bdec2_HPC_ML_taxonomy).

For the San Diego meeting, we want to focus on the other two related concepts that are particularly relevant to the goals of BDEC, namely, *the application of Artificial Intelligence in Cyberinfrastructure* (**AI-in-the-CI**) and *Cyberinfrastructure that supports Artificial Intelligence* (**CI-for-the-AI**). Accordingly, we invite you to submit 1-2 page whitepapers that address these topics:

***AI-in-the-CI*:** Artificial Intelligence can help make our current CI more capable, dynamic, adaptable, and efficient. The current scientific computing software stack must embrace and evolve with an "AI Everywhere" approach to performance tuning, job scheduling, edge to cloud data movement and resource allocation. Reimagining the current cyberinfrastructure software stack, with machine learning everywhere, is a daunting task. Innovation will be required across the stack, from the operating system to the workflow tools. Questions that whitepapers might address include the following:

- How could Machine/Deep Learning be useful in, e.g., improving data locality, replication, and movement?
- How can we automate the generation of code that can achieve portable performance on heterogeneous architectures?
- Can we use AI to predict and avert bottlenecks in data intensive workflows that have low latency requirements?
- Are there technology gaps that need to be addressed in order to support AI-in-the-CI? For example, right now most system information is stovepiped and therefore not adapted to be used in a collective way for ML training. Is a common infrastructure for system data in order to collect, aggregate, learn from, anticipate, and dynamically react to different system behaviors?

***CI for the AI*:** Fresh innovations in software infrastructure will be needed to enable edge-to-HPC ("transcontinuum") workflows and data logistics that support the scalable deployment of AI everywhere. Furthermore, to support AI computational workloads,

computer architectures and software environments must be updated. Large-scale systems will need to support both AI workloads and classic simulation and modeling. Data logistics from the edge to the cloud will require orchestration with HPC platforms. Topics that whitepapers could address include the following:

- How are AI workflows different from classic HPC workflows? What's unique to AI workflows?
- What are good examples of automated edge-to-HPC computation?
- Are there technology gaps that need to be addressed in order to support AI-in-the-CI? For example, what CI will be required to support AI datasets that are continually and dynamically? What kind of CI will be required "in the missing middle," i.e. between the HPC center and the edge, in order to support efficient data logistics, in transit processing, and continuously streaming data?

Of course the issues and questions for the topics given above are meant to be illustrative; you should feel free to formulate your own questions for either of these topics. Good ideas about potential lines of important research, emerging new types of relevant technology, or discussions of notable technology gaps are always welcome. Putting such ideas in the context of your own experience, or the experience of your research community, is especially valuable. As at previous meetings, as many white papers will be selected for short (4-5 slide) presentations by their authors at the meeting, but all white papers contributed will be published online and form an important part of the workshop's results.

Please send your contributions Terry Moore (tmoore@icl.utk.edu) by October 10. We are looking forward to seeing you in San Diego.

SmartFlows: "AI for CI" on top of the Computing Continuum for AI-Integrated Applications

Ilkay Altintas[1], Kyle Marcus, Volkan Vural, Shweta Purawat, Daniel Crawl
UC San Diego

1. Motivation

The growth of new processors over the last decade including GPUs, FPGAs, and edge accelerators open the way to a diverse set of applications using machine learning on top of nontraditional hardware. The common theme to these applications, mostly composed of artificial intelligence (AI) workloads, is their need to run in specialized environments for reasons such as on demand or 24x7 nature of task they are performing, and difficulties regarding their portability, latency, privacy and performance optimization. Moreover, there is a need for integration of these applications with traditional high-throughput computing (HTC) or high-performance computing (HPC) tasks for integrated dynamic data-driven, e.g., AI-integrated science.

A typical example to these applications is the role of real-time edge processing and use of machine learning and big data in wildfire behavior modeling applications within the WIFIRE cyberinfrastructure [1]. WIFIRE's dynamic-data driven fire modeling workflows depend on continuous adjustment of fire modeling ensembles using the observations on fire perimeter generated from imagery captured by a variety of data sources including ground based cameras, satellites and aircraft. Typically, the computing for the perimeter generation takes place in an environment built for big data and/or edge computing while the fire modeling needs to take place in an HTC or HPC environment depending on the fire modeling codes that need to be executed. Figure 1 depicts this integrated workflow and various steps in relation to the underlying execution infrastructure required to run these applications.

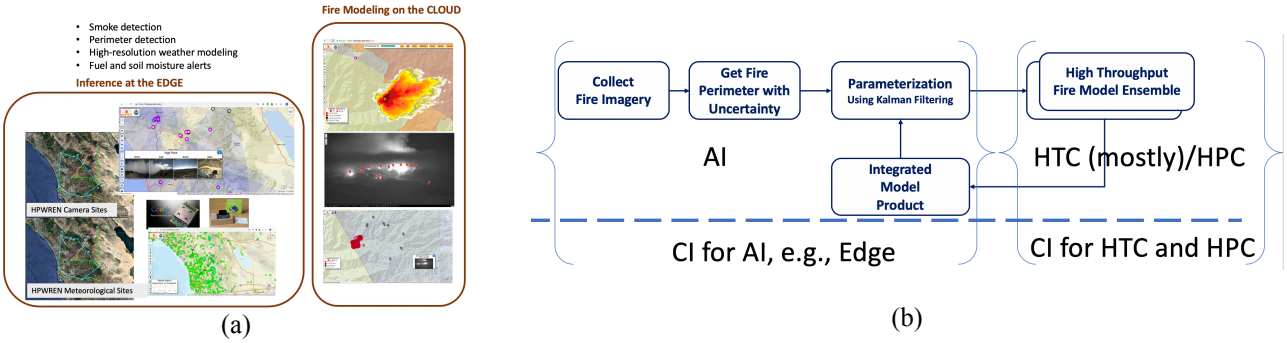


Figure 1. AI-integrated fire modeling at the continuum of computing

A generalized case of this example applies to many domains including molecular dynamics simulations (See Figure 2). Learning from imaging to decide on parameterization of simulations and combining this learning with the insights gained from analyzing the time-series output of simulations promises many benefits ranging from shorter run times and time to discovery to less energy usage for computing.

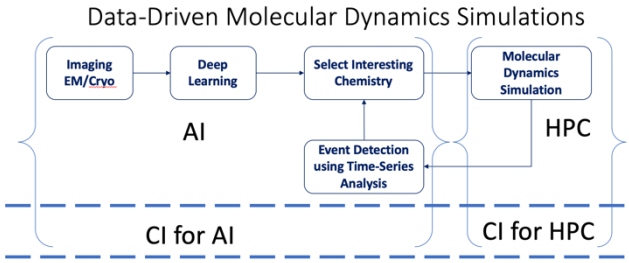


Figure 2. Data-driven molecular dynamics simulations

Currently, there are many new tools and technologies, e.g., for scheduling and resource monitoring, to scale and manage the components of these application workflows homogeneously. However, these application workflows are composed out of steps that require a heterogenous CI ecosystem that involves dynamic resource management and coordination. Current CI lacks an integrated environment that can pull data from a number of resource monitoring tools and turn these federated data into predictive intelligence needed to steer application workflows in a dynamically scalable and data-driven fashion at the time workflow execution.

[1] Corresponding Author Email: ialtintas@ucsd.edu

2. SmartFlows: Artificial Intelligence for Cyberinfrastructure

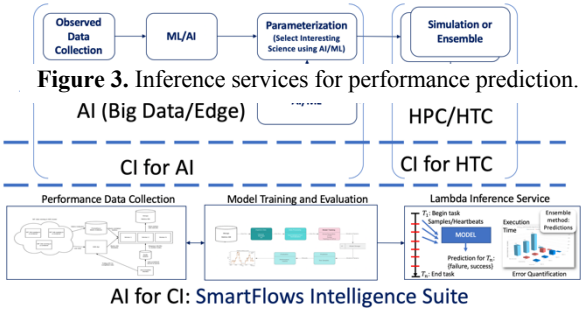


Figure 3. Inference services for performance prediction.

As an approach to answer the need for a heterogeneous CI performance prediction, we created an extensible data collection, monitoring and prediction framework called *SmartFlows* [2]. SmartFlows analytical services are designed to provide inference for the performance of a workflow step based on the predictive models derived from data generated by prior heterogenous AI workloads. Having predictive intelligence on all steps of the workflow prior to the execution allows for running every part of an application workflow in a way that achieves the optimal performance for the whole workflow rather than optimizing for one step of the workflow. This approach also enables optimization of the collective workflow based on different objectives including cost, speed, and energy consumption.

We have tested our approach within the NSF CHASE-CI project for two workflows for Belle2 and Belle2-network [3-5]. Figure 3 illustrates the three main components of SmartFlows running for a generalized heterogenous workflow.

- Performance Data Collection: The workflows were executed on Nautilus in Docker images running as Kubernetes jobs. A scalable and extensible performance data collection service was implemented to use Prometheus for execution metrics and collect data for all workflow tasks on Redis. The data collection service also includes "worker" services that query Prometheus for the collected job metrics and the Kubernetes API for additional job information, e.g., the specification of the computing node the job was executed on. All the collected information is then turned into JSON-based objects and stored in MongoDB for scalable access.
- Model Training and Evaluation: In order to predict resource utilization for a given task, we train machine learning algorithms, namely LSTMs, using the metrics that were collected from simulations as training data stored in MongoDB. After the initial data processing and split into training, validation and test datasets, and normalization of training data, and LSTM model was trained with Keras and Tensorflow on the CHASE-CI GPUs and tuned using iterative hyper-parameter tuning. This resource-centric predictive performance modeling approach captures the essential characteristics of hardware resource and specializes in forecasting performances of never-before-seen workflows and input loads on a hardware resource [6]. After the model training step, we publish the final model and related parameters as a service to be executed for prediction.
- Prediction using a Lambda Inference Service: The developed model is then used for prediction resource utilization and visualized. Although this test service is not fully implemented, we plan to implement further automation both for training and deployment to increase the efficiency, and live prediction and visualization of the resource requirements while the tasks are running.

We believe the combination of SmartFlows with new workflow tools that can ingest these predictions will allow us to experiment with dynamic resource allocation based on these predictions. Even though dynamic resource allocation is not straightforward, any possible solutions we can produce would be a groundbreaking solution to runtime failures due to hitting the resource limits assigned to this task. Also, allocating resources only when needed optimizes the efficiency of the resource utilization, which will lead to completing the tasks waiting in the queue in optimum time.

References

[1] Altintas I., Block J., de Callafon R., Crawl D., Cowart C., Gupta A., Nguyen M., Braun H.W., Schulze J., Gollner M., Trouve A., Smarr L., Towards an Integrated Cyberinfrastructure for Scalable Data-Driven Monitoring, Dynamic Prediction and Resilience of Wildfires. Procedia Computer Science, Vol. 51, pp 1633-1642, 2015. https://doi.org/10.1016/j.procs.2015.05.296.
[2] I. Altintas, S. Purawat, D. Crawl, A. Singh, K. Marcus. Towards A Methodology and Framework for Workflow-Driven Team Science, IEEE Computing in Science & Engineering, 2019.
[3] Belle2 Experiment:, https://www.belle2.org/, 2019.
[4] Schram, M., Tallent, N., Friese, R., Singh, A., Altintas, I., Application of Deep Learning on Integrating Prediction, Provenance, and Optimization, in 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP 2018), 214(2019). https://doi.org/10.1051/epjconf/201921406007
[5] Singh, A., Schram, M., Tallent, N., and Altintas I., "Deep Learning for Enhancing Fault Tolerant Capabilities of Scientific Workflows" in IEEE International Workshop on Benchmarking, Performance Tuning and Optimization for Big Data Applications, at the IEEE Big Data 2018 Conference, Seattle, WA
[6] A. Singh, A. Rao, S. Purawat, and I. Altintas, "A Machine Learning Approach for Modular Workflow Performance Prediction," in Proceedings of the 12th Workshop on Workflows in Support of Large-Scale Science, New York, NY, USA, 2017, p. 7:1–7:11. https://doi.org/10.1145/3150994.3150998

ZettaFlow: Towards High-Performance ML-based Analytics across the Digital Continuum

Gabriel Antoniu, Alexandru Costan, Ovidiu Marcu
Univ Rennes, Inria, CNRS, IRISA
October 12, 2019

At the previous BDEC2 workshops we introduced the Sigma data processing architecture [Antoniu2019], which aims to enable unified data processing across hybrid infrastructures combining Edge, Cloud and HPC systems. In this white paper, we discuss how this architecture could be leveraged to enable the seamless execution of machine learning algorithms at the intersection of these domains.

The Sigma architecture combines batch-based and stream-based Big Data processing techniques (i.e., by extending the traditional Lambda architecture) with in situ/in transit data processing techniques inspired from the HPC area (Figure 1). It allows to collect, process and analyze extreme volumes of past data, real-time data, jointly with simulated data both in situ (where the data is generated) and in transit (e.g., on specific resources dedicated to analytics), before the data is shipped to long-term persistent storage.

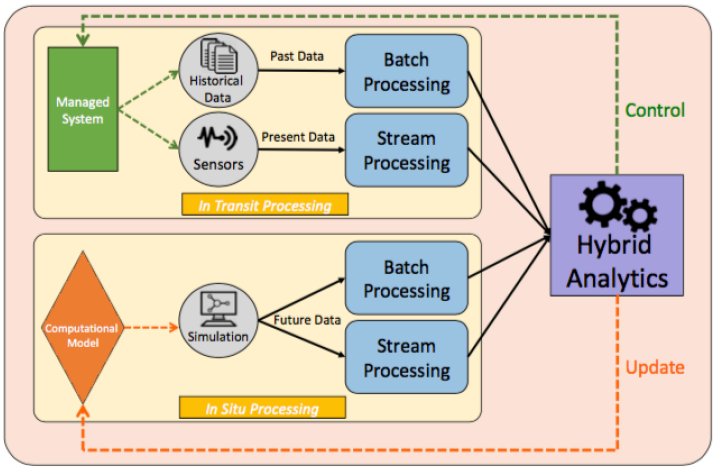


Figure 1. The Sigma data processing architecture.

To provide a reference implementation of the Sigma architecture, our approach consisted of jointly leveraging two existing software components: the Damaris [Damaris,Dorier2013] middleware for scalable in situ/in transit processing and the KerA [Marcu2018] unified system for ingestion and storage of data for scalable stream processing.

Data shifts to the Edge. By 2022 Gartner predicts that 75% of enterprise-generated data will be created and processed outside of the data center and cloud infrastructures, compared with 10% today [Gartner2018]. Recent developments brought by **5G networks** will be a key enabler of the future digital world. They will bring new service capabilities for industrial/research stakeholders thanks to the unprecedented on-demand performance and real-time reactivity. For example, energy, transport, manufacturing and water utilities will be capable of connecting to millions of networked devices, taking real-time, intelligent and autonomous decisions (roundtrip latency in 1ms range) [Ericsson2019]. All these data will be continuously **streamed** at high rates to the processing sites (at the Edge, in the Fog or in the Cloud), requiring frameworks able to cope with their **real-time processing requirements**.

Unified framework for stream storage, analytics and machine learning. Our vision to support the next generation of analytics and machine learning applications relies on a **unified approach for stream storage and processing**, spanning from the Edge (where data is generated) to the Fog/Cloud and HPC centers (where it is processed).

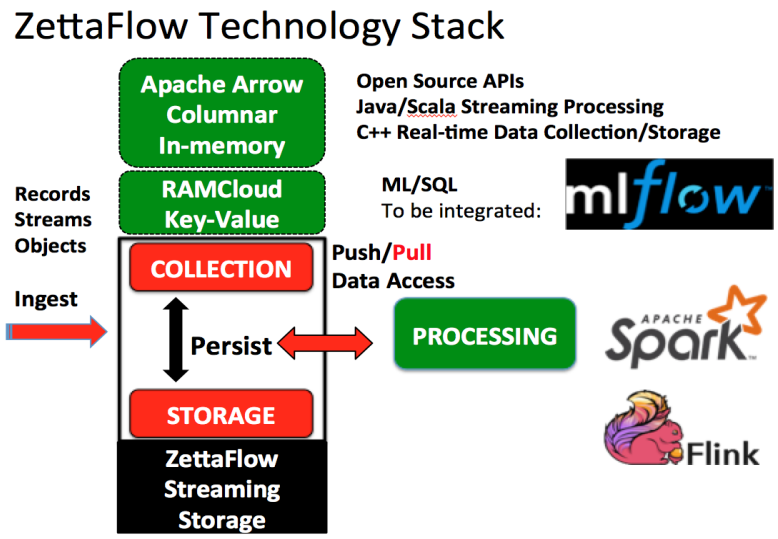


Figure 2. The ZettaFlow architectural overview.

To illustrate this vision, we introduce ZettaFlow, a high-performance multi-model analytics-oriented storage system, that extends the KerA approach by proposing a unified multi-model storage engine (i.e., supporting *streams*, *key-value*, *columnar* APIs). To do so, it relies on RAMCloud [Ousterhout2015] (a low-latency key-value store) and Apache Arrow [Arrow2019] (a high-performance in-memory columnar storage system).

Dedicated support for machine learning. ZettaFlow is currently integrated with:

- The Apache Flink data analytics framework, through a shared memory approach effectively co-locating storage and processing;
- The Damaris middleware through streaming RPC (a)synchronous interfaces. The implementation relies on native C++ integration, to avoid Big Data Java serialization overheads, and to efficiently support high-performance networks like Infiniband and Intel DPDK.

We are now considering integration of ZettaFlow with **MLflow** [Zaharia2019] (an open-source platform for the end-to-end machine learning lifecycle) and Apache Spark. The overall goal is to build **a reference high-performance architecture for unified real-time analytics and machine learning**. This unified approach will provide machine learning workloads a high performance experience similar to the one currently enabled for stream-oriented analytics.

Questions and Answers

1) *What innovative capabilities/functionalities will the proposed candidate platform demonstrate (e.g. transcontinuum workflow, edge computing, data logistics, distributed reduction engine, etc.)?*

The above integration provides the possibility to explore the possibility to **dynamically push to storage** the critical aspects of machine learning workloads (i.e., *in-storage processing*). As a step further, we will investigate specific optimizations by studying machine learning data access patterns to effectively optimize the real-time flow of data that traverses both ZettaFlow and machine learning frameworks.

2) *What applications/communities would/could be addressed?*

5G applications like smart vehicles and transport, critical services and infrastructure control, human-machine interaction and critical control of remote devices.

3) *What is the "platform vision," i.e. what kind of shared cyberinfrastructure (CI) for science would further research/design/development of this platform lead to?*

A reference high-performance multi-model analytics storage platform supporting workflows that **combine machine learning and real-time analytics**.

4) *How available/ready/complete is the set of software components to be used to build the demonstrator?*

We are working towards an efficient integration of ZettaFlow with machine learning frameworks.

5) *As far as one can tell at this early date, to what extent can this be done with existing and/or otherwise available hardware/software/human resources?*

Building an operational AI demonstrator requires interaction with industrial and research partners to identify machine learning access patterns and develop AI models that will help ZettaFlow self-optimize and reach bare-metal performance, while considering the variety of use cases, effectively optimizing the critical machine learning aspects. We are also in the process of building a startup that will leverage the ZettaFlow platform for real-time edge/cloud data analytics.

References

[Antoniu2019] Gabriel Antoniu, Alexandru Costan, Ovidiu Marcu, Maria S. Pérez, Nenad Stojanovic: Towards a demonstrator of the Sigma Data Processing Architecture for BDEC2, BDEC2 workshop, Poznan, May, 2019. White paper.

[Gartner2018] Smarter with Gartner: What Edge Computing Means for Infrastructure and Operations, 2018.

[Ericsson2019] 5G use cases, Ericsson, https://www.ericsson.com/en/5g/use-cases.

[Marcu2018] Ovidiu Marcu, KerA: A Unified Ingestion and Storage System for Scalable Big Data Processing, PhD thesis, 2018. https://tel.archives-ouvertes.fr/tel-01972280.

[Ousterhout2015] John Ousterhout, Arjun Gopalan, Ashish Gupta, Ankita Kejriwal, Collin Lee, Behnam Montazeri, Diego Ongaro, Seo Jin Park, Henry Qin, Mendel Rosenblum, Stephen Rumble, Ryan Stutsman, and Stephen Yang. "The RAMCloud Storage System". In: ACM Trans. Comput. Syst. 33.3 (Aug. 2015), 7:1–7:55

[Arrow2019] Apache Arrow, http://arrow.apache.org/

[Zaharia2019] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, Corey Zumar: Accelerating the Machine Learning Lifecycle with MLflow, USENIX'19, Santa Clara.

[Dorier 2013] M. Dorier, R. Sisneros, T. Peterka, G. Antoniu, D. Semeraro, “Damaris/Viz: a Nonintrusive, Adaptable and User-Friendly In Situ Visualization Framework”, Proc. LDAV – IEEE Symposium on Large-Scale Data Analysis and Visualization, Oct 2013, Atlanta, USA. URL: https://hal.inria.fr/hal-00859603

[Damaris] The Damaris project. https://project.inria.fr/damaris/

Self-improving extreme-scale systems with hierarchical reinforcement learning

Prasanna Balaprakash (corresponding author, pbalapra@anl.gov)
Mathematics and Computer Science Division & Leadership Computing Facility
Argonne National Laboratory

Heterogeneous nodes, many-core processors, deep memory hierarchies, power and energy walls, and resiliency concerns make application and system management on high-performance computing systems an increasingly daunting task [1]. Current strategies provided by application development tools, compilers, profilers, operating, and runtime systems (OS/R), and schedulers are mostly static and unlikely to be successful for porting and running traditional and emerging applications on future extreme-scale systems [2]. To overcome this challenge, we need to develop machine-learning-enabled software stack that can surpass human capability to port and run applications on extreme-scale computing platforms. The objective here is to enable extreme-scale systems to observe themselves and to predict and optimize the overall performances of the applications and system automatically over time.

A promising approach to design extreme-scale systems consists in hierarchical reinforcement learning (RL), in which, at each level of the hierarchy, several agents try to optimize their level-specific reward functions in the extreme-scale systems. At the lower, node level, agents will optimize performance, power, energy, and memory footprint based on the available OS/R and application parameters. At the middle, system level, agents will learn to optimize communication, load balancing, and I/O for node boards, midplanes, and racks. At the upper, facility level, agents will account for resource utilization; facility-wide application performance, power, energy, and resilience; shared I/O usage; and mapping of jobs to heterogeneous resources. At each level, the data required for computing the reward function will be collected from OS/R instrumentation infrastructure, such as performance counters, data logging, and sensors or custom tools. Since many upper-level reward functions are derived quantities of middle- and lower-level reward functions, we also will require hierarchical reward functions. Furthermore, we need to develop Pareto dominance as an optimality criterion to handle those reward functions that are likely to be conflicting but need to be optimized simultaneously.

To develop the proposed hierarchical RL, three broad research areas need to be explored: (1) machine-learning-based multiobjective modeling that captures the relationship between controllable parameters/knobs and various objectives at different levels; (2) fast search methods for decision making; and (3) reinforcement learning for online adaptation.

Multiobjective modeling

Modeling level-specific objectives such as runtime, failures of critical subcomponents, power, and energy as functions of application and platform characteristics will play a central role in the proposed self-improving extreme-scale systems. These models will be used to quantify meaningful differences across the decision space and to offer a convenient mechanism for exposing near-optimal spots in the decision space. Analytical performance models based on first principles may not be sufficiently accurate for all objectives of interest. At extreme scale, data-driven modeling is expected to bridge the gap. In this approach, application and platform characteristics and their corresponding metrics are collected directly on the target platform, and a predictive model is built for multiple objectives using statistical machine learning approaches. The key requirement here is to develop a fast and practically viable, completely automated approach that adaptively selects the most suitable learning algorithm to model the required objectives based on hardware and application signatures.

Multiobjective search algorithms for offline tuning

Even when models for multiple objectives are available, finding the parameter configuration that can optimize those multiple (conflicting) objectives is a difficult task. Existing optimization studies have several shortcomings. First, the multiobjective algorithms adopted so far are not tailored to exploit tuning problem characteristics. Second, it is not clear whether local search or global search is better suited for multiobjective performance tuning and, in particular, under what conditions one class may be better than another. Third, search algorithms have not been tightly integrated with multiobjective performance models. To that end, we need to focus on approximation methods with guaranteed bounds and global-local search hybrids that can obtain good solutions rapidly and refine them rigorously. We can exploit level-specific and metric-specific information to prune the search space and balance the tradeoff between exploration and exploitation. We need to develop subsidiary methods based on analytical models and expert driven heuristics to detect and prevent unintended agent actions.

Reinforcement-learning-based online adaptation

Since offline models do not capture all possible system state, the adaptation of parameters is required at runtime to take into account the running state of the system and possible errors in the offline models. The success of online adaptation approaches will be determined by tradeoffs between online exploration of the search space and exploitation of the accumulated search history via offline multiobjective models.

We can formulate the online adaptation problem as a computational reward maximization problem using reinforcment learning (RL) [3]. At any discrete time step, a learning agent performs an action and receives a response from the unknown environment. Given a cumulative reward function defined over responses, the goal of the agent is to interact with the unknown environment by selecting actions that maximize future rewards. Model-based RL requires an explicit model of the unknown environment. It is defined as a probability distribution over future responses conditioned on the past actions and responses, and it is updated whenever a new response becomes available. Model-free RL tries to learn the relationship between the action and reward directly. Both model-based and model-free RL provide promising avenues for designing self-improving extreme-scale systems. However, given the complexity and the degrees of freedom in extreme-scale systems, a single agent is unlikely to be able to optimize all the OS/R and application parameters. Scaling RL for such complex tasks is an ambitious goal and will be a promising area of research in AI for high performance computing.

The hierarchical RL must be carefully implemented within software stack. A straightforward approach consists in implementing RL agents as components of OS/R at various levels. Despite the data required for the agent being available locally, runtime and memory overheads of RL could be detrimental to overall performance. Moreover, there is a risk of making the system software on a node smarter, which consumes more processing, memory, and bandwidth. Therefore, a tradeoff must be made between what can be done versus what should be done; this tradeoff needs to be examined carefully. We therefore need to decouple the agents from the compute-/memory- intensive components of hierarchical RL through control software running on a dedicated partition of the system. Consequently, RL agents will send the responses and receive actions from the dedicated control-software partition. Although this approach introduces extra messages, the sizes of the messages will be small, and their frequencies will be optimized accordingly.

To improve the overall productivity of the DOE leadership facilities and to accelerate scientific discovery, we need to automate the process of porting and mapping scientific applications to extreme-scale systems. We can accomplish this goal by developing self-improving features for extreme-scale systems, where new machine learning approaches are used to model, predict, and optimize important objectives in ways not previously possible.

References

[1] Exascale operating systems and runtime software report. DOE ASCR Workshop Report, 2012.

[2] Machine learning and understanding for intelligent extreme scale scientific computing and discovery. DOE ASCR Workshop Report, 2015.

[3] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT Press Cambridge, 1998.

AI aided Scientific Computing Applications on Large Scale computing platform

Rongqiang Cao*, Yangang Wang

Computer Network Information Center, Chinese Academy of Sciences

P.O. Box 349，Bejing 100190, China

E-mail: caorq@sccas.cn

Track classification: ***CI for the AI***

We built Artificial Intelligence Computing and Data Platform (AICDP) in February 2018 and provides massive computing capability for large-scale deep learning training and big data processing, as shown in figure 1. The platform is built on heterogeneous HPC supercomputers,clusters and cloud virtual clusters, which consist of K80, V100 and other GPU accelerators. There are a dedicated computing cluster and shared High Performance Computing resources in the platform. The dedicated computing cluster is composed of 48 computing nodes, 380 NVIDIA P100 GPUs, whose double-float performance is 1.8PF and the single-float performance is 3.6PF. Besides, the shared HPC resources are acquired from national HPC environment by RESTful APIs where double-float performance is 200PF and the storage capacity is 200PB. Base on the massive computing resources encapsulated by virtual machines and containers, the platform provides one-stop and full life-cycle services for AI computing jobs. Several interactive interfaces are provided in the platform, including command lines embedded in WEB fixed and graphical workflows in WEB services.

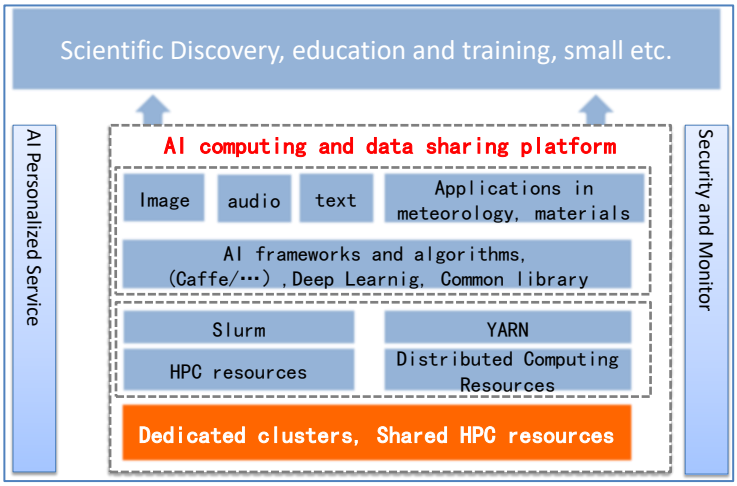
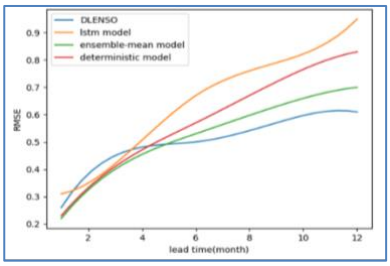


Figure 1. the architecture of the AI computing and data sharing platform

At present, the platform has more than 200 users from universities, institutes and companies. There are lots of AI frameworks and applications in the platform, such as TensorFlow, Caffe, Pytorch, mxnet, keras, spark, hadoop and so on. The platform provides computing service for a variety of research fields, including but not limited to meteorology, biology, energy, finance, health.

ENSO（El Niño-Southern Oscillation）is a bi-modal variation in sea level barometric pressure between the western and eastern Pacific. A Deep Learning ENSO Forecasting application is proposed based on the ConvLSTM and Seq2seq models, as shown in figure 2(a). The result shows

that the prediction accuracy is higher than the deterministic forecast model in the medium and long-term forecasts. In addition, a Deep Learning Air Pollution Forecasting application is proposed to predicate PM2.5 concentration （unit： μg/m³）of grid regions and its results are as shown in figure 2(b).



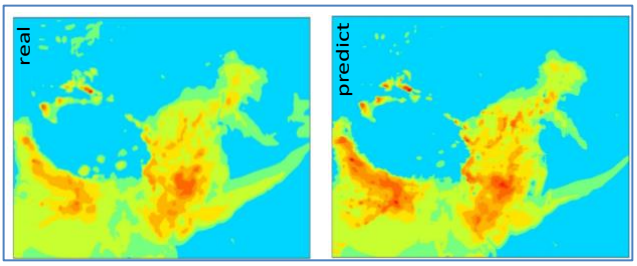


Figure 2(a) Comparison of RMSE　　Figure 2(b) PM2.5 comparison between real and predict

Fingerprint recognition is an AI application based on the platform. Automatic fingerprint recognition is one of the most widely studied topics in biometrics over the past 50 years. If the database is large, this approach may be problematic for real-time applications due to the high number of comparisons needed. As a result, AI algorithms are used to extract features from fingerprint images to accelerate the speed of fingerprint recognition in the fingerprint recognition application, as shown in figure 3.

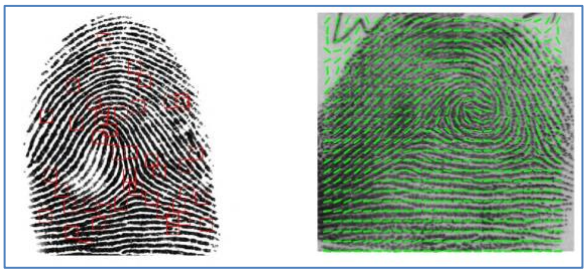


Figure 3 the features extracted from fingerprint images by AI algorithms.

Short-term LSTM PV power forecasting models based on seasons and weather types are established to further improve efficiency and accuracy, as shown in figure 4. the result in table 1 shows that Different seasons exhibit different forecasting effects. Under different weather types, sunny days have the best fitting effect.

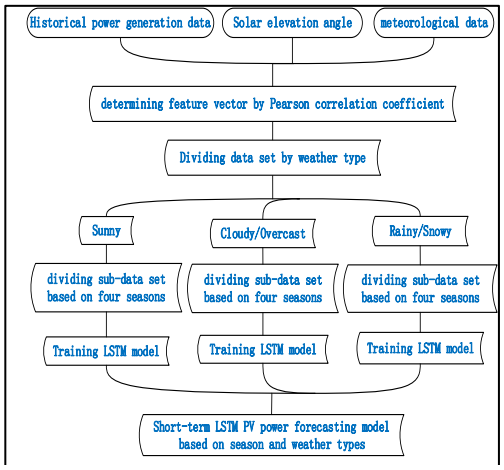


Figure 4 the workflows of Short-term LSTM PV power forecasting models

Table 1 results of Short-term LSTM PV power forecasting models

Weather types	Seasons	RMSE	MAE
Sunny	Spring	5.78	2.34
	Summer	2.56	1.05
	Autumn	5.90	2.44
	Winter	4.78	2.13
Cloudy /Overcast	Spring	6.92	3.54
	Summer	8.02	3.99
	Autumn	8.74	4.73
	Winter	6.04	2.75
Rainy /Snowy	Spring	7.72	3.46
	Summer	7.07	3.66
	Autumn	8.96	4.99

Lossy compression and AI for scientific data: how do they interplay?

Franck Cappello[#], Robert Underwood*, Sheng Di[#], Justin M. Wozniak[#], Jon C. Calhoun*
[#]ANL, *Clemson University

Numerical simulations and instruments are already generating very-large, high-velocity data sets that users need to reduce for different use cases: reduction of storage footprint, reduction of streaming intensity (physics instruments), reduction of memory footprint, reduction of I/O time, etc. This problem will worsen on Exascale systems and updated physics instruments. For example, the SKA (Square Kilometer Array) Phase 1 will generate ~300 PB/year of science data in 2023. It is estimated that 1 EB of science data will be generated from HL-LHC (High Luminosity-Large Hadron Collider) experiments in 2026 [1].

This over-production of scientific data not only raises problems in terms of data management (storage space and I/O bandwidth, etc.), it also raises serious problems on data analysis. In many cases, classic analysis algorithms are too computationally complex to analyze data coming at very high velocity. Or analysis involves human-in-the-loop that slows down the analysis process. One of the solutions explored to analyze scientific data at very high velocity is using ML/AI based algorithms. Filters (noise reduction), classifiers (identification), predictors, detectors (including for pattern), feature finders, surrogate models are designed considering ML or DL algorithms. For example, teams at LCLS are testing using ML for detecting/predicting unwanted changes or failures, getting more useful information out of complicated machine signals *(e.g.images, waveforms)*, system control and optimization, designing accurate system models, facilitating improved physics, and understanding of machine behavior [2]. DNNs are tested for the trigger system of the HL-LHC (phase II) [3] and investigated in other areas for local simulation by generative models, physics mining, reconstruction, particle tracking, analysis, instrument operation, anomaly detection, etc. [4]. In another domain, lossy compression is tested for the ECP CANDLE project NT-3 datasets. This dataset is used to design a DNN capable of assessing tumor response to the drug, given information about the tumor and potential drug treatment. Lossy compression here is used to accelerate training [5].

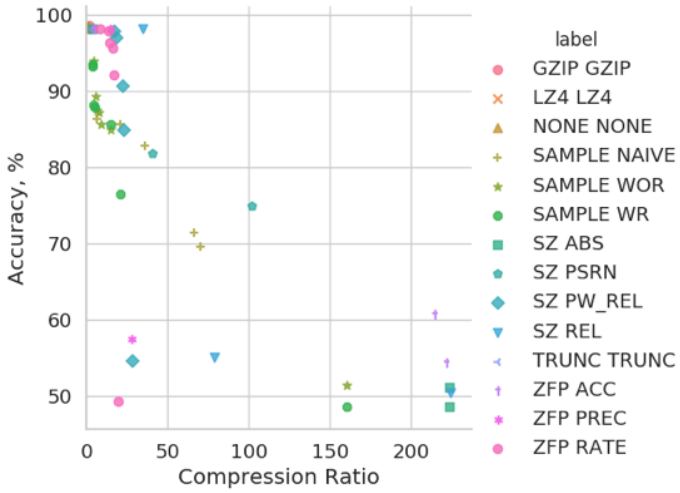
In short, both lossy data compression/reduction and ML/DL based analysis algorithms are required and they must be combined efficiently to address the big scientific data challenges coming with the next generation of supercomputers and physics instruments. However, the effective and efficient combination of lossy compression and AI algorithms is an open research problem that raises several non-trivial questions:

1) How ML/DL algorithms react to lossy compressed/reduced data compared to non-reduced data? This question is already explored for image processing/analysis (AlexNet, VGG, ResNet) [6] but not for scientific data.
2) How shall we design lossy compression/reduction techniques if we know that ML/DL based algorithms will be used for analysis?
3) Are there ML/DL based algorithms that are more resistant/resilient to lossy compression/reduction errors than others in the context of scientific data?
4) How to dynamically control lossy compression settings based on analysis done on the ML/DL results?
5) Is this a co-design issue?

In the context of the first question, we investigated the impacts of compression (lossless and lossy) on several ML and DL algorithms. For each of them, we considered a relevant application. Due to space limitations, we only present here early results related to the ECP CANDLE NT3 problem. To evaluate how lossy compression affects the results of the DNN used for the CANDLE NT3 problem, we apply lossy compression (with a large set of different error bounds for each lossy compressor) and decompression to the dataset and then train the DNN from the decompressed data. To assess the distortion on results caused by lossy compression of the training

set, we computed a number of summary statistics from the CANDLE NT3 test set obtained with and without lossy compression. In this white paper, we only focus on the pareto-optimal points (points from which there is no improvement in compression ratio without a decrease of accuracy). Figure 1 presents the pareto-optimal points (samples from the pareto optimal curve) for the CANDLE NT-3 problem. The accuracy of the DNN without compression (NONE) or with lossless compression (GZIP, LZ4) is 98%. Points closest to the top right (greatest accuracy and greatest compression ratio) would be considered optimal.

Figure 1: Pareto optimal points between DNN accuracy and compression ratio for the CANDLE NT-3 problem.



The results of this empirical study show that some lossy compression algorithms (SZ REL) maintain the accuracy of the ML/DL algorithm trained with uncompressed data while providing compression ratios of ~30. Other compression algorithms (SZ PW_REL, SZ ABS, ZFP RATE) are degrading significantly the response of the ML/DL algorithms to the point where the use of these compressors with ML/DL is not recommended. An important element here is that the DNN accuracy is highly sensitive to the type of error bound used for lossy compression. The SZ compressor provides the best overall performance with the REL error mode while it degrades the accuracy drastically in the PW_REL error mode.

These early results show that the combination of lossy compression and ML/DL algorithms is not trivial and can dramatically degrade the performance of ML/DL algorithms. The scientific community needs careful analysis of the lossy compression and ML/DL algorithms interplay to better understand of how to combine them effectively.

References:

[1] Mark Neubauer, IRIS-HEP Blueprint: Concepts and Process, FAST ML Workshop, Fermilab, Sept 2019.
https://indico.cern.ch/event/822126/contributions/3500165/attachments/1905471/3146855/IRIS-HEP_BluePrint_FML.pdf
[2] A. Edelen, et al., Demonstrations of Machine Learning on Particle Accelerators, FAST ML Workshop, Fermilab, 2019
https://indico.cern.ch/event/822126/contributions/3500175/attachments/1905503/3146921/201909010_FAST_ML_Edelen.pdf
[3] I. Ojalvo, Overview of triggering and real-time systems, FAST ML Workshop, Fermilab, Sept 2019.
https://indico.cern.ch/event/822126/contributions/3500166/attachments/1905489/3146910/ojalvo_Trigger_overview_FNAL_Sept_2019_Low_res_copy_v2.pdf
[4] Jean-Roch Vlimant, (Fast) Machine Learning at the LHC, Fermilab, Sept 2019.
https://indico.cern.ch/event/822126/contributions/3500173/attachments/1905507/3146925/vlimant_FML_MLLHC_Sept19.pdf
[5] J. M. Wozniak, et al.. CANDLE/Supervisor: A workflow framework for machine learning applied to cancer research. BMC Bioinformatics, 19(S18):491, 2018.
[6] Y. Li et al., A Network-Centric Hardware/Algorithm Co-Design to Accelerate Distributed Training of Deep Neural Networks, IEEE/ACM MICRO, Fukuoka, 2018, pp. 175-188.

Feeding the Beast: High Performance Data Pipeline for Large-Scale Deep Learning

Cong Xu, Antonio Lain, Paolo Faraboschi, Nic Dube, Dejan Milojicic
Hewlett Packard Enterprise
{cong.xu,antonio.lain,paolo.faraboschi,nic.dube,dejan.milojicic}@hpe.com

Abstract

The upcoming generation of Deep Neural Network (DNN) training accelerators with orders of magnitude higher performance enable much larger datasets and models. While weights and activations will be resident in accelerator memory, the memory required for hosting the training dataset becomes large and expensive, and caching will not always work effectively. Meeting the extreme bandwidth requirement for the training data pipeline will become a key challenge. This whitepaper analyzes training I/O requirements of commonly used DNNs, highlights challenges, and proposes some directions to be explored.

Problem statement

Accelerators for DNN training are quickly improving their performance, putting pressure on other system components. The next generation of training accelerators [3][5][6], is expected to be orders of magnitude faster than today's GPUs. At the same time, memory and storage technology are not accelerating at the same pace. While we expect that through software (compression) and hardware (local memory) improvements, weights and activations can remain within the fast accelerator memory (SRAM or HBM), feeding the training data will become the fundamental bottleneck for machine learning at scale.

The amount of training data to feed the accelerators, even after partitioning across servers, grows linearly with the processing speed, since non-trivial training sets require hundreds seconds of computation per iteration. Additional augmentation steps to enhance the robustness of the model, reduce bias and resilience to adversarial patterns, further increase the training data requirements.

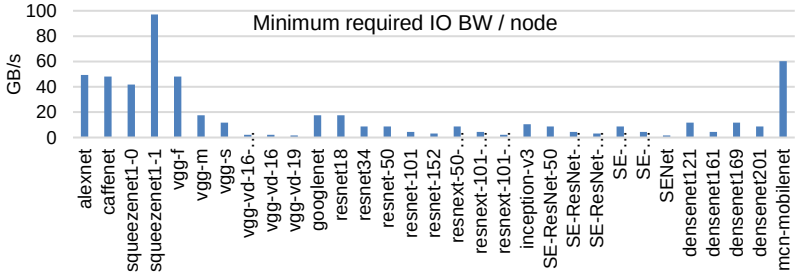


Figure 1. IO bandwidth required to feed 8 Turing GPUs for CNNs

While benchmarking datasets (like ImageNet, ~250GB) can fit in a single node memory, real-world applications will require traversing PBs of data by training over billions of samples (like Google TPUs [1]), and at a sufficient speed not to be a bottleneck for the accelerators. Today's approaches use DRAM to cache the training set, but soon it will become economically infeasible to keep the whole dataset in DRAM, even across distributed servers, during training. A scalable *"training data pipeline"* architecture that optimizes performance and cost using a combination of technologies, including DRAM, Flash, or new non-volatile memories, is necessary.

The key challenge for this data pipeline is to provide enough bandwidth without overprovisioning capacity. For example, training different CNNs on 8 NVidia Turing GPUs in a single compute node requires up to 100 GB/s of IO bandwidth to keep the GPUs busy (Figure 1). Luckily, modern server processors have enough internal I/O bandwidth to approximately match DRAM memory bandwidth. For example, an AMD EPYC CPU has 128 PCIe gen 4.0 lanes providing a peak of 256 GB/s of unidirectional I/O bandwidth, that roughly matches the 204 GB/s of peak DDR4-3200 bandwidth per socket (8 channels).

Challenges and Opportunities

Figure 2 describes a typical workflow for supervised DNN training. The dataset lifecycle starts with labelling using manual annotations, or other semi-supervised approaches. To train the model, a subset of the dataset is extracted, and preprocessed to fit the model requirements. Data is then "augmented" (i.e., replicated and slightly modified) across multiple server groups. Within each group, the dataset is partitioned across servers, and locally cached in memory for the entire training.

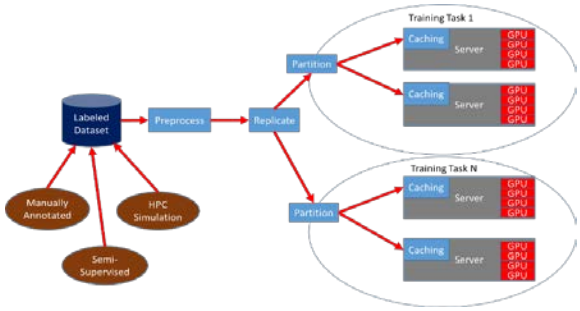


Figure 2. Dataset life-cycle for supervised training

Figure 3 shows the data flows within a node for one of the training tasks in Figure 2. Training iterates over the cached dataset hundreds of times, with some optional preprocessing within the accelerator to randomize the samples. The model weights are reduced and shared across all nodes regularly to speed up convergence.

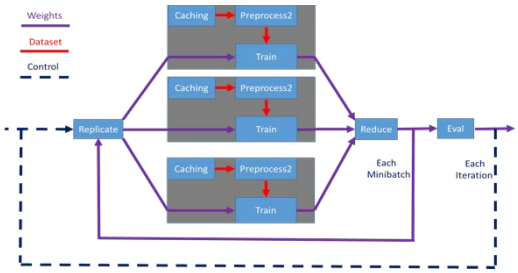


Figure 3. Internal data flow for a training task

We built an analytical model to estimate the I/O bandwidth for training different DNNs. The model takes the CNN architecture, batch size, input image size, hardware specification, effective utilization, and node configuration as input and output the required I/O bandwidth per node. It shows that scaling the CNN training over thousands of compute nodes would require tens of TB/s aggregated I/O bandwidth and PB capacity.

Model complexity also plays a fundamental role in the I/O bandwidth requirements, as we can see from the I/O arithmetic intensity roofline graph of Figure 4. Here, we hypothesize an acceleration cluster with an aggregated computational power of 100 16b PF/s (the horizontal roofline), and we plot a few I/O rooflines at 1, 2 and 4 TB/s. Ideally, we would like all models to fall into the "happy models" group, where they are compute bound. As we can see, the size and stride of convolution kernels has a significant impact on I/O arithmetic intensity. For example, Squeezenet with many 1x1 convolution kernels has orders of magnitude lower I/O intensity than other models with same-level of accuracy and would require over 50 TB/s not to be checked by I/O. If we scale this projection to a hypothetical 100 EF (16b) in the post-exascale era, the I/O requirements skyrocket to > 1 PB/s which is a very challenging target (considering that Exascale-class machines are expected to be in the 10-2 TB/s I/O range). The evolution of model complexity is a very important trend to track (and possibly influence) to avoid reaching the I/O bottleneck prematurely.

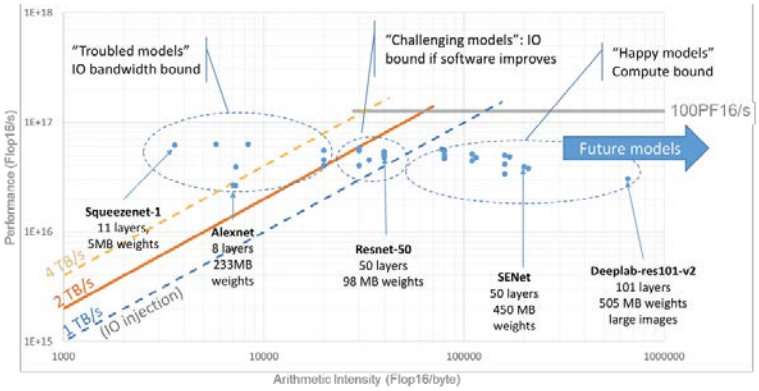


Figure 4. I/O Arithmetic Intensity of DNN models

Caching the dataset in local DRAM of each compute node would be a mitigating solution, but it is prohibitively expensive. And the cached dataset adds on top of the memory consumption because they cannot share the memory space with intermediate results during data preprocessing and neutral net training. The common approach for distributed DNN training is to partition the dataset across nodes, and host datasets in DRAM for the duration of the training. The limitations of that approach were described in the introduction. Google suggests using a double buffering technique for distributed training over TPU pods [1], but it does not reduce the DRAM capacity requirement if the I/O bandwidth does not match the training throughput.

We need solutions that architect a data pipeline to match the training performance at an optimal cost. Neither caching everything in DRAM or overprovisioning a standard cluster parallel file system (like Lustre) works. Interesting approaches to be studied include tiered centralized data systems, for example adding a performance tier in the storage hierarchy provides a great flexibility in configuration, allocation of resources, provisioning of capacity and bandwidth, and ease of data management and sharing across nodes. However, the I/O traffic between accelerators and I/O nodes or the all-reduce operations at the end of each training iteration can cause network interference and long latency tails that negatively impact the achievable I/O bandwidth. Alternatively, localized data systems where accelerators have exclusive access to local NVMe devices can scale more easily because of the lack of network traffic during the training phase. However, the ratio of I/O capacity/bandwidth to accelerators is fixed and may lead to overprovisioning if bandwidth is the limiter. Finally, we can think of workflow changes that can mitigate the I/O arithmetic intensity: for example, reusing the same data to train multiple parallel models. All of these are important directions that need to be extensively investigated.

References

[1] Effective machine learning using Cloud TPUs (Google I/O '18)
[2] Burst buffer architecture. https://www.nersc.gov/users/computational-systems/cori/burst-buffer/burst-buffer/
[3] S. Lie, Cerebras: Wafer Scale Deep Learning, HotChips32, Aug 2019
[4] NVidia GPU Direct technology. https://devblogs.nvidia.com/gpudirect-storage
[5] E. Medina. Habana Labs Approach to Scaling AI Training. HotChips32, Aug 2019
[6] J. Silver. Graphcore: Inside the UK unicorn that's about to become the Intel of AI. Wired UK

Towards Intelligent Management of Heterogeneous Memories with Deep Reinforcement Learning

Balazs Gerofi <bgerofi@riken.jp>
RIKEN, Center for Computation Science, Japan

The past decade has brought an explosion of new memory technologies. Various high-bandwidth memory devices, e.g., 3D stacked DRAM (HBM), GDDR and multi-channel DRAM (MCDRAM), together with storage class memories (SCM) such as PCI-E non-volatile memory (NVRAM), as well as byte addressable non-volatile memories, e.g., phase-change memory (PCM), resistive RAM (ReRAM), and the recent 3D XPoint, are already in production or expected to become available in the near future. Additionally, network attached memory technologies and PCI-switch accessible memory have been also proposed for future large-scale systems. Moreover, many of these technologies are available in multiple configurations. For example, recent server class Intel CPUs support multiple non-uniform memory access (NUMA) configurations (e.g., Quadrant, SNC-2, SNC-4, etc.), which further complicates the issue of efficiently utilizing the hardware.

Management of such complex memory hierarchy is a major challenge for application developers, not only in terms of memory allocation (i.e., placing data structures into the most suitable memory device), but also to adaptively move content as application characteristics changes in time. System level solutions that optimize memory allocations and data movement in an intelligent fashion by transparently mapping application behavior to the underlying hardware are thus highly desired. With the recent proliferation of artificial intelligence techniques, the questions naturally arise:

- Can AI/ML techniques be combined together with operating system level mechanisms to provide a general solution that is applicable to a wide range of applications and memory technologies?
- Is such an approach adequate for parallel applications running on large-scale systems?

Future applications will need adaptive system software that can dynamically manage hierarchical, heterogeneous memory devices. The operating system may be the most suitable place to address this problem as it has access to privileged mode, low-level hardware capabilities. For example, it can provide features such as page remapping, copy-on-write (COW), and write protection during page migration. It can also transparently utilize idle resources, e.g., taking advantage of idle CPU cores of multi/many-core systems for parallel data copy. These features, together with AI/ML driven policies can be used to improve application data movement between memory types; however, they must be adapted and optimized for HPC applications. Our approach is built upon three essential pillars detailed as follows.

First, automatic discovery of memory device attributes and topology information is needed. In order to make any informed decision on mapping application behavior to underlying hardware characteristics, there is a need for the system software to establish an accurate of view of the properties of available memory devices and their relation to topology information. These properties include capacity, latency, bandwidth, and volatility. While operating systems usually expose certain description of such properties, they often rely on vendor provided information as opposed to a method of discovery. As a result, such information tends to mismatch the real properties of the hardware. For example, the Linux operating system provides various information on NUMA properties in the `/sys` pseudo file system. The NUMA distance between NUMA nodes is one piece of information provided. While this attribute is often interpreted as the inverse proportion of available bandwidth, certain architectures with heterogeneous memory devices (e.g., the Xeon Phi Knights Landing) intentionally misreport these properties so that the operating system kernel does not utilize high bandwidth memory for kernel allocations. Unfortunately, this also implies that user-level applications and/or runtime systems are unable to trust such information and thus must make architecture specific exceptions. In contrast to current practice, we envision automatic discovery of

characteristics of memory devices through measurements during initialization so that an accurate view of the system can be provided to higher-level components. Some of the questions here to examine are: how general this approach can be? What exact information should be further exposed to upper layers of the system?

Second, low overhead application memory access tracking and performance profiling is necessary. A critical component for system level memory management is the ability to real-time monitor memory access behavior of applications. Existing solutions for memory access pattern tracking primarily rely on dynamic instrumentation (e.g., the widely used PIN library), which introduce very high runtime overhead. On the other hand, recent advances in hardware performance monitoring have made a wide range of hardware facilities available that can be utilized to obtain an accurate profile of application behavior while imposing very low runtime overheads. For example, hardware performance counters in the CPU provide information on stalled CPU cycles, cache behavior, etc. Also, with the introduction of un-core performance counters it is now possible to gather information on memory controller transactions, which can be directly related to memory bandwidth usage. Additionally, event based sampling facilities (e.g., Intel's PEBS) report virtual addresses associated to processor events (e.g., misses in the last level cache) that are statistically sampled according to hardware configuration. This mechanism enables software to characterize memory access patterns with respect to specific ranges of the virtual address space. We will combine information on different hardware events to obtain a real-time profile applications' memory access behavior. We are currently exploring what performance events to combine to obtain the right representation for memory access profiles of applications.

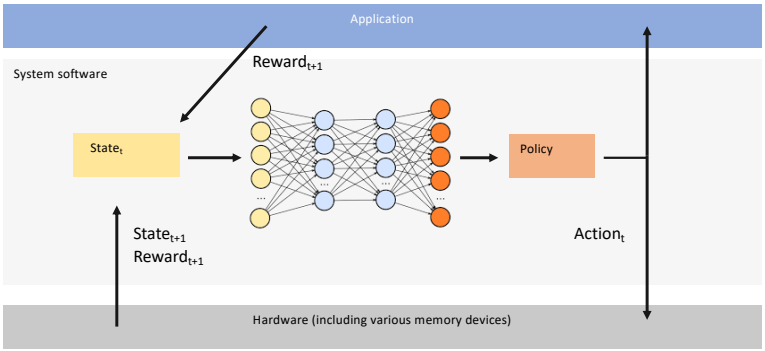


Figure 1.: Reinforcement learning based memory management

Finally, the system software will utilize AI/ML techniques that can enable mapping application memory access profiles to optimal memory layouts. Recent advances in AI/ML demonstrated the superiority of these techniques compared to hand crafted heuristics in a wide range of scientific/industrial domains. In particular, reinforcement learning (RL) seems highly promising in the context of mapping memory access profile information to memory layout suggestions. A basic overview of the envisioned system is shown in Figure 1. The OS/runtime obtains real-time profile of the application via hardware performance counters together with memory access patterns. This information combined with the virtual memory layout of the application represents the state of the reinforcement learning agent which is then fed into a deep neural network to control action regarding memory layout modification. RL maximizes cumulative reward over time which can be a combination of application provided feedback on its progress as well as the observed characteristics of the hardware. Some of the challenges we foresee are: state representation must be kept minimal to avoid explosion of the state space and thus we will focus on memory areas that are being addressed by the application. The cost of actions (e.g., to move memory pages between different devices) must be part of the reward scheme so that overhead does not outweighs benefits. Finally, significant amount of compute power must be available as RL techniques require extensive experimentation.

AI and ML for High Energy Physics
Maria Girone, Viktor Khristenko, CERN
BDEC October 15-17

The High Energy Physics community (HEP) is investigating machine learning techniques across data collection, processing, classification and analysis workflows. The ML techniques have the potential to replace resource intensive reconstruction and simulation algorithms and help close the gap in computing resources expected during the next phase of the LHC program. AI and ML in the scientific workflows will require not only technical elements like cyber infrastructure to support them, but also and equally importantly experience and validation to be fully adopted. On the other hand, AI and ML techniques to optimize distributing computing “AI in the CI” have the potential to make a significant impact and to improve efficiency, as they are seen as less risky to the mission, thus easier to adopt.
We will describe examples of projects in “AI in the CI” that we believe could be adopted for rapid efficiency gains. Additionally we will discuss “CI for AI” projects that could facilitate AI development in the data intensive science community.

“AI in the CI”

The Worldwide LHC Computing Grid (WLCG) has been the computing system for the LHC experiments since the beginning of the program. Due to the complexity of the system and scale of the problem a lot of effort was devoted in the beginning to both health and activity monitoring. Initially there were also ambitious services planned that were intended to make automated decisions for job scheduling and data movement. Most of the automated systems were descoped as the state of the system could never be sufficiently up-to-date to make good scheduling decisions. The advent of AI and ML techniques gives the hope that automated systems have some predictive power and can learn from their activities. Some of the capabilities that were envisaged for the WLCG more than a decade ago may now be possible.

HEP datasets are already hundreds of petabytes in size and will be several exabytes during the High Luminosity LHC program (HL-LHC). Two hundred globally distributed processing facilities are used to store, process, and distribute the data. The storage mix is roughly equally divided between large scale disk systems and active archival tape libraries. The LHC experiments have instrumented their data systems to measure how frequently samples of data are accessed, how long it takes to recover data from tape and to move between locations, and what fraction of the data, stored in files, is required by the applications. There is sufficient information to train AI systems to schedule data transfer from tape and data movement into local caches based on access patterns and job scheduling.

HEP workflows are comprised of hundreds or thousands of individual pieces with well defined dependencies. The processing needs, the input data required, and the progress and status of each piece is monitored. Using machine learning systems, job scheduling combined with

intelligent data placement would increase the efficiency and could be tuned to the needs of the researchers.

Much like a complex industrial system, the computing system for HEP is made up of infrastructure and services. The system was designed with fault tolerance, and the health of components is monitored by operations teams. AI systems could significantly reduce the effort and improve uptime. Predicting faults, scheduling maintenance, adjusting workflows to avoid overloads are all potentially possible.

Looking aside towards capabilities in terms of “AI for S/W” this time, AI has been very effective in developing translations between human languages. We would be interested in facilitating code porting between computing languages as well using AI techniques. HEP applications are millions of lines written by a thousand contributors. Automating code portability and maintenance could be a big benefit as human effort for code optimization for new architectures is a huge effort.

We think there is tremendous potential for AI to improve the cyberinfrastructure. The HEP distributed computing system is complicated and well instrumented and would benefit from ML techniques. The development and techniques needed to implement AI and ML systems into HEP should be applicable to more centralized systems also.

CI for the AI

In order for AI and ML techniques to make an impact in scientific discoveries, and not just technical efficiency improvements, we need to build a cyber infrastructure designed for training and development that matches the organizational and data requirements of the scientific community.

Many HPC systems have adopted accelerated hardware which run optimized code for AI and ML training. Being able to open these resources to the broader user community to support Training as a Service has the potential to expand the adoption of these powerful tools. In order to use HPC resources for training and validation, large samples of data will need to be moved and accessed. The HEP community is working on systems to deliver the data efficiently even to external computing resources.

Within the context of HPC, whenever AI comes into play, we typically talk about computing infrastructure rather than the delivery/flow/management of enormous amounts of data required to make full use of data-driven techniques and provided hardware. Coming from the LHC experience, HEP reconstruction workflows are not able to utilize HPC facilities to their full potential, with one of the many reasons being inability to efficiently ingest/deliver and stage out data to/from the compute nodes. Existing infrastructure needs to be adapted in order to provide simple and efficient interfaces for researchers to be able to ingest, reuse, and store large

datasets, which is one of the primary requirements for successful use of data-driver AI-based algorithms.

Scientific Machine Learning Benchmarks and Datasets

Tony Hey

Chief Data Scientist

Rutherford Appleton Laboratory

Didcot OX11 0QX, UK

At the Rutherford Appleton Laboratory (RAL) in the UK, we have been investigating the applicability of Machine Learning (ML) algorithms in the production and analysis of scientific data generated at the LHC and by the large-scale experimental facilities at these laboratories. These facilities include the Diamond X-ray synchrotron, the ISIS neutron and muon source and cryoEM electron microscopy services. In addition, the JASMIN High Performance Data Analytics system hosts environmental data for the Centre for Environmental Data Analysis.

The applicability of Deep Neural Networks to the large datasets being produced by these facilities is of particular interest, given the breakthroughs in image recognition using Deep Learning (DL) neural networks with the Imagenet labeled dataset [1]. However, in the case of experimental data generated at the facilities, the application of DL networks is usually complicated by the lack of labelled ground truth data for training. Although simulated experimental data can provide an alternative basis for obtaining a labeled dataset for training, the use of simulated data comes with its own set of concerns and limitations.

The goal of the Scientific Machine Learning group (SciML) at RAL is to produce an interesting collection of curated scientific datasets which can be shared openly with the scientific and computer science communities as an 'experimental Machine Learning laboratory'. In particular, such scientific datasets can be used to benchmark the performance of different ML algorithms on a range of different hardware platforms. Our SciML benchmark suite is intended to span multiple scientific domains, and cover several of the different types of problems arising in each domain [2]. We are therefore aiming to provide a number of reasonably large and complex datasets specific to each domain, together with one or more baseline models that address particular domain-specific problems. The evaluation metrics for our SciML suite go beyond simple runtime performance. Our goal is to capture the overall performance of a given scientific application by assessing both the training and inference times per sample, as well as classification accuracy using several appropriate metrics. While it can be taken as a given that GPU implementations of these neural networks will perform well, a range of new architectures and chips are now emerging. The availability of these curated datasets and ML benchmarks will allow the evaluation of these architectures for scientific ML.

Our initiative is similar in spirit to two other scientific ML benchmarking efforts. The MLPerf initiative has begun with applications of direct interest to the commercial IT community in areas such as object detection, speech to text and machine translation which have been revolutionized

by Deep Learning technology [3]. However, there is now a working group, MLPerf HPC, that is exploring the development of a set of 'Scientific Machine Learning' benchmarks similar to those or the RAL SciML initiative. Similarly, the CANDLE project at Argonne Nation Laboratory has developed benchmarks for Deep Learning methods applied to the problem of cancer diagnosis and treatment [4]. We believe that all these groups need to work together to create a compelling and useful suite of datasets and ML benchmarks of direct relevance to the scientific community. The common overall goal is to maximize the discovery of new science, be it in astronomy and particle physics or material and molecular sciences.

However, note that for both scientists and computer scientists, there is an important research agenda that can be explored with such SciML datasets. What is the robustness of the ML algorithms to adversarial noise for example? There are now many examples of the brittleness of Deep Learning solutions and their vulnerability to adversarial noise [5]. In addition, uncertainty quantification for these methods is needed to give scientists confidence in the accuracy of their results. Finally, more research on the 'understandability' of Deep Neural Networks will be important to convince scientists of the validity of their results.

A final possible research topic is how physics, chemistry and biology constraints can be incorporated into machine learning methods. Google DeepMind's AlphaFold has demonstrated that such scientific constraints can be combined with DL networks to produce very accurate results for protein folding [6]. Can such techniques be generalized to other research areas?

References and URLs

[1] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 2012;1097-1105.

[2] Hey T, Butler K, Jackson S, Thiyagalingam J. Machine Learning and Big Scientific Data. To be published in Phil. Trans. Roy. Soc. A, 2019.

[3] https://mlperf.org/

[4] https://candle.cels.anl.gov/

[5] https://www.nature.com/articles/d41586-019-03013-5?sf221156507=1

[6] Service R. Google's DeepMind aces protein folding. Science. 2018 Dec 6; Available from: http://dx.doi.org/10.1126/science.aaw2747

Edge-to-Cloud Processing with InLocus and INDIANA

Overview

The Data Logistics Toolkit (DLT)[1] is a software distribution that incorporates and extends the software components produced by the research program in Logistical Networking[2]. The *depot* service is a central component of the DLT that exposes raw storage as variable-size data allocations accesible in a distributed networking environment. Allocations and associated metadata are managed by a policy engine to meet geographic and temporal locality demands of the higher-level services and applications making use of DLT infrastructure. With the addition of service discovery and management (UNIS), we have the necessary supporting elements to bring computation to data at the periphery, or edge, of the network.

The InLocus architecture[3] operates on streams of data and represents an effort to generalize the ability to process data from a variety of sources. This includes both edge-generated collections as well as pre-staged data that may be cached in nearby depot services (Figure 1). Due to its limited execution model and semantics, it is suitable for processing streams of e.g. sensor data on a message by message basis. This primitive execution allows InLocus to execute on devices at the *nano* or *micro* end of the computing continuum.

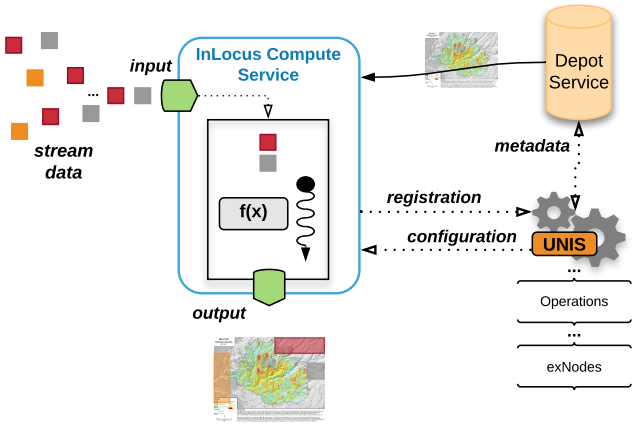


Figure 1: InLocus compute service fusing stream data with latency-sensitive data stored within depots.

A key observation with InLocus is that network packet processing overheads consume significant computational resources on constrained, edge-class (e.g. system-on-chip) platforms that limit the useful work such devices can perform. With the rise of low-cost, low-power SoC+FPGA devices, InLocus implements network protocol offload using the available programmable logic (PL) to scale such systems for data-intensive workloads.

Exploring Machine Learning with INDIANA

As data volumes from the edge continues to increase, there is a need to reduce bandwidth entering the cloud, improve data acquisition latency, and improve the ability of AI algorithms to further classify and process collected data sets. As one pertinent example, machine-assisted classification of remote sensing data (e.g. imagery and video from spaceborne or airborne systems) has the potential to significantly

[1] M. Beck, T. Moore, N.H. French, E. Kissel, M. Swany. Data Logistics: Toolkit and Application, GoodTechs '19 Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good, 2019

[2] M. Beck, T. Moore, J. Plank, and M. Swany. Logistical Networking, pages 141–154. Springer US, Boston, MA, 2000.

[3] L. Brasilino, A. Shroyer, N. Marri, S. Agrawal, C. Pilachowski, E. Kissel, and M. Swany. Data Distillation at the Network's Edge: Exposing Programmable Logic with InLocus. In IEEE International Conference on Edge Computing, July 2018. https://doi.org/10.1109/EDGE.2018.00011

impact the operations of first responders and other communities that rely on accurate and up-to-date visual information in their operating theatres.

Within the InLocus framework, the In-Network Distributed Infrastructure for Advanced Network Applications (INDIANA) considers lightweight computation at the intersection of mobile-edge and the data center. With INDIANA, lightweight (RISC-V) cores are integrated into the PL to perform simple functions applied to streams of data traversing the network, rather than virtual machines, OS-containers, or otherwise fixed-function, hardware-optimized implementations. Extending this concept, an aggregate of soft-cores (configured as an overlay) within the PL allows for the development of neural networks to perform more involved operations such as the classification of imagery data.

The growing market of edge-targeted FPGAs with such potential presents an opportunity to develop binary neural networks (BNNs) that can be implemented on constrained devices and in a power-efficient manner[4]. With the ability to reconfigure and deploy (via InLocus, INDIANA) the operations performed by the FPGA-enabled devices, programmable logic for AI at the edge becomes a viable option with the potential to accelerate existing approaches by orders of magnitude[5]. Another potential benefit is that security and privacy concerns may be mitigated by processing data close to the source without requiring transfer to centralized, persistent infrastructure.

A number of open questions do remain: *i*) Can edge-deployed neural networks be reconfigured and trained rapidly enough to meet changing application demands? *ii*) What are the practical limitations (e.g. image resolution and tiling constraints) for the size of BNNs possible in current PL silicon? *iii*) Is the complexity of development across a variety of SoC+FPGA platforms a barrier to forming a general model? The ever-increasing size and efficiencies of deployable PL may provide one answer, and FPGA vendors are now releasing SDKs to develop on their platforms at a level above the hardware description language. The INDIANA research project aims to incorporate these developments and provide a model and prototype for orchestrating ML designs.

Finally, developing and demonstrating SoC+FPGA AI capabilities at the edge allows machine-learning techniques to become possible in situations where fixed infrastructure may not be available, or bandwidth constraints limit communication with remote cloud resources. This is of particular importance in situations of natural disaster or in emergency management scenarios, and recent efforts in the DLT have enhanced the software stack to be resilient in the face of frequent network disruption.

To summarize, the DLT, InLocus architecture, and INDIANA represent the data movement, execution model, and hardware design components, respectively, of an ecosystem for edge-to-cloud processing. Demonstrations of stream processing are possible with the existing tools with the reconfigurable neural network support in early development.

[4] iCE40 UltraPlus. `http://www.latticesemi.com/en/Products/FPGAandCPLD/iCE40UltraPlus`

[5] Zynq UltraScale MPSoC Accelerated Image Classification via Binary Neural Network TechTip `https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841949/Zynq+UltraScale+MPSoC+Accelerated+Image+Classification+via+Binary+Neural+Network+TechTip`

Large-Scale Optimization Strategies for AI&HPC Workloads

Yu Liu*

AI&HPC Department, Inspur Electronic Information Industry Co.,Ltd

2F,Tower C, No.2 Xinxi Rd. Shangdi Haidian District, Beijing, P.R.C. 100085

***E-mail:** liuyubj@inspur.com*

Track classification: ***CI for the AI***

With AI pushing the boundaries of innovation and technology, growingscientific research fields are adopting AI approaches to accelerate traditional large scale simulation workload or explore the unknown. On the one hand, AI promotes the development of traditional computing facilities, from the design suitable for traditional scientific computing to the hybridization of science and AI computing at the same time as a shared CI. On the other hand, the hybridization of computing facilities also puts forward high requirements for softwares or applications. Because the hybrid computing architecture usually contains a large number of acceleraters or dedicated computing cores, it requires not only that traditional scientific computing softwares can run efficiently on the hybrid computing platform, but also that future applications can make full use of the hybrid computing architecture to achieve ultra-large-scale computing. Currently, the developmentof data center software, especially HPC software, is dramatically outpacing that of hardware. As a result, how to make softwares run efficiently on large-scale computing platform is now a frontier subject of common concern both in AI and HPC.

As the world's leading computing platformprovider, Inspur has provided a large number of advanced AI&HPC computing platforms (Figure 1, A 16 GPU card computing platform based on NVSwitch) for many customers around the world. At the same time, Inspur has done a lot of work on software optimization, make it well able to efficiently run on such a leading platform, and has summarized and formed unique large-scale AI&HPC application optimization strategies.

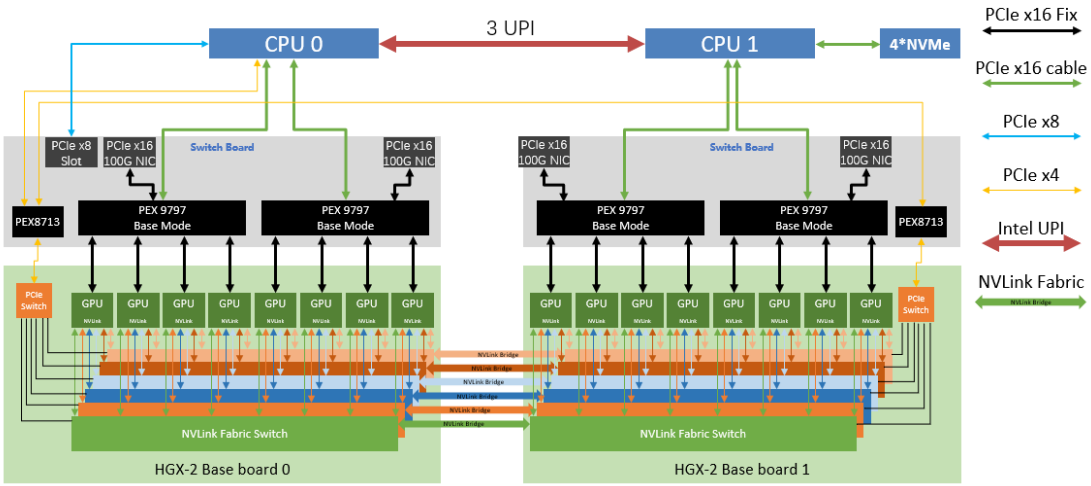


Figure 1, A 16 GPU card computing platform based on NVSwitch

Our strategies are based on performance analysis and optimization for applications in different fields of research using large-scale AI&HPC clusters. Our strategies are designed to comprehensively analyze runtime micro-architechture features of applications by using an efficient tool - Teye (Figure 2, Large-scale application feature analysis tools - Teye), and optimize application to novel computing platforms. This two levels strategies cover platform optimization, technological innovation, and model innovation, and targeted optimization based on these features. State-of-the-art micro-architectures, network communication and other modules, and innovative parallel mode of some applications have been optimized. After optimization, these applications will show greater performanece when running up to 8000 cores, compared to the non-optimized versions.

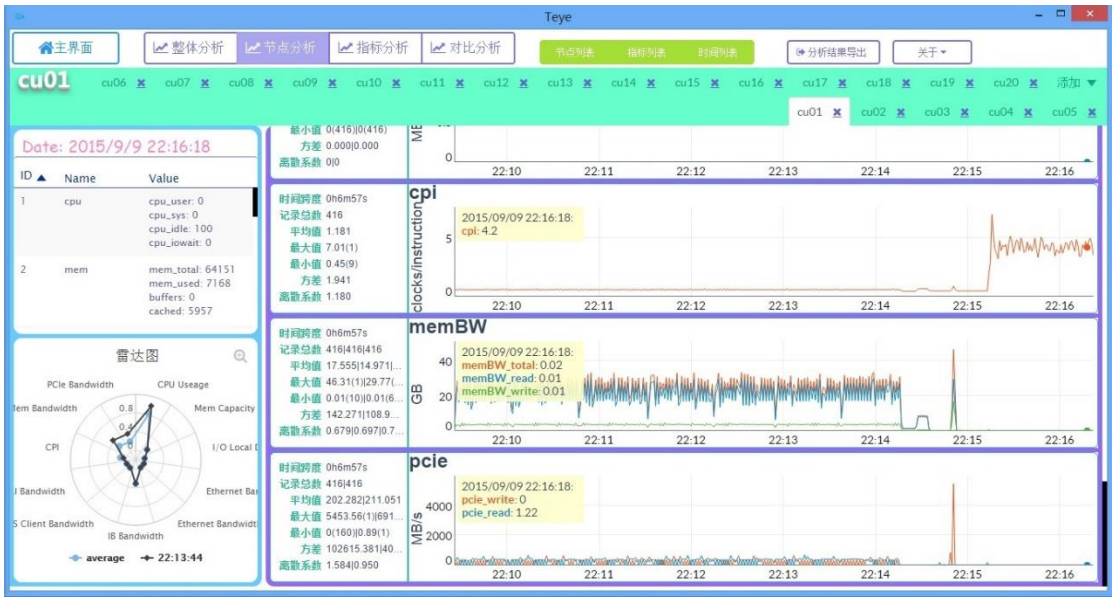


Figure 2, Large-scale application feature analysis tools - Teye

A Community Machine Learning Commons for Advancing Machine Learning in Earth System Science

Dr. Richard Loft, National Center for Atmospheric Research

The National Center for Atmospheric Research (NCAR) is a Federally Funded Research and Development Center, sponsored by NSF. Located in Boulder, Colorado, NCAR is a focal point for research in the field of atmospheric sciences and, in the face of climate change, for advancing the broader field of Earth system science. Recently, NCAR has begun to delineate roles and a strategy for advancing the scientifically rigorous application of machine learning (ML), and indeed, data-driven science as a whole, across the broad front of possible applications. This strategy has three elements.

First, NCAR, needs to develop an in-house, cross-cutting scientific research capability in ML. Establishing such a capability across NCAR laboratories has been a major focus of the Analytics and Integrative Machine Learning (AIML) group, which I manage, as well as other internal laboratory investments, since 2018.

Second, NCAR recognizes that it must build a new cyberinfrastructure (CI) to support data-centric science, including ML. This strategy, called Science at Scale (S@S), envisions end-to-end software and hardware changes throughout the science workflow. The CI elements of S@S include:

- **HPC:** design includes integrated, large memory, multi-GPU nodes designed to support machine learning use cases. ML benchmarks are baked into next procurement (a.k.a. NWSC-3 in 2022).
- **Storage:** include traditional highspeed disk tiered with on-prem object store and off-prem cloud.
- **System software:** including containers and container orchestration
- **Tools:** Jupyter, and ML frameworks (e.g. Tensorflow, Keras, Horovod); and Pangeo, a ML-compatible analysis system built on DASK and xarray.

As part of the effort to prototype the S@S CI, NCAR has deployed a Cloud Commons (off-prem object store). Cloud Commons will host selected datasets in the cloud for public access. Amazon Web Services (AWS) has been selected as the commercial cloud partner. The Cloud Commons pilot project will leverage, where possible, AWS Public Datasets Program, which provides free S3 hosting and egress. The first dataset to be deployed on the Cloud Commons will be the 100 TB Community Earth System Model (CESM) Large Ensemble (LENS) dataset[2]. LENS is a valuable and often-cited dataset used to the climate research community, particularly those studying climate extremes.

Third, NCAR must help community in evaluating and adopting ML techniques in their research and help them learn to effectively use the new CI. NCAR's ML capacity-building efforts will leverage S@S training and outreach plans to accomplish this goal. A centerpiece of NCAR's strategy here to leverage the Cloud Commons to create a collection of both observation-based and modeling-based ML reference problems, with the training datasets packaged with the necessary supporting notebooks, frameworks and tools. A core set of ML reference problems have been defined, but may be expanded over time. NCAR is working through the American Meteorological Society (AMS) AI Working Group to contribute this reference collection to EnviroNet, a proposed atmospheric science analog to the successful ImageNet database. We hope that, like ImageNet, EnviroNet can become a resource for researchers and students to advance the state of the art in ML in our discipline.

References

1. For more information on Pangeo, see: https://pangeo.io
2. J.E. Kay, et al., 'The Community Earth System Model (CESM) Large Ensemble Project: A Community Resource for Studying Climate Change in the Presence of Internal Climate Variability', *Bull. Amer. Meteor. Soc.*, **96**, 1333–1349, doi: https://doi.org/10.1175/BAMS-D-13-00255.1.
3., S. K. Mukkavilli, et al., 'EnviroNet: ImageNet for the Environment', AMS Annual Meeting, January 8, 2019. See https://ams.confex.com/ams/2019Annual/webprogram/Paper353532.html

LUMI, the pan-European pre-exascale supercomputer – designed for the converge of AI and HPC

Pekka Manninen and Sebastian von Alfthan
CSC – IT Center for Science
Finland

Abstract

In this contribution, we will present the concept plan one of the upcoming European flagship supercomputers, designed from the ground-up as a platform for workloads featuring converged simulations and artificial intelligence methods.

AI as an HPC workload

Artificial intelligence (AI), especially machine learning methods are applied to many kinds of scientific challenges, and their use is rapidly expanding to various scientific disciplines, including life sciences, humanities and social sciences. Especially deep neural networks' ability to learn from very complex data is revolutionizing data analysis by taking over repetitive tasks and finding new patterns in data. For machine learning high performance access to large amounts of data, like microscope images and data repositories, is crucial. AI is also disrupting traditional HPC simulation workflows, and finding use in all stages of work. Many groups are also experimenting in using deep neural networks for replacing parts of the simulation models, to speed up the simulation or to improve the model by bringing in additional physics. It has even been shown the ability to learn a full simulation model, and enable approximate solutions to, e.g., fluid dynamics simulations. AI is also an integral part in unraveling insight from the data produced by massive simulations on a pre-exascale machines.

A key use case of the major supercomputer installations of the 2020's will be the AI-accelerated data analytics. This refers to the use of AI for analyzing data sets, enriching them and learning from them to generalize to new sets of parameters. Just a few examples:

- Climate modelling: Identification of extreme weather patterns from high-resolution climate simulations.
- Materials science: Constructing databases that can be utilized as training sets for AI based methods.
- Linguistics: Massively multilingual neural machine translation models based on large data sets of previously translated documents.
- Medical biotechnology: Molecular phenotyping has emerged as the natural step forward to make diagnosis more precise and comprehensive. Determine to what extent morphological patterns in medical imaging data can be utilised to predict molecular phenotypes using deep learning.

The overall observation is that AI training workloads need "traditional supercomputers" more than simulation science ever: an extreme floating-point capability combined with extreme interconnect capability and I/O performance. At the same time the new workloads are driving fundamental changes in the design of systems. In AI training workloads floating point accuracy down to 16-bit in the form of FP16 or Bfloat16 can be used, especially when mixed with 32-bit accurate aggregation. The operations themselves are also mostly matrix operations which in turn has led to the emergence of computational units specialized for those. These new capabilities are also leading to new opportunities or speeding up traditional workloads. In fact, the first Exaops computations have already been performed.

The I/O pattern of AI workloads place a huge demand on IOPS in addition to bandwidth, to feed the massive data sets to the training task. These have led to the emergence of a number of non-volatile memory based storage and persistent memory layers.

While training workloads benefit from large single-site installations, the inference workloads are for the most part easier to distribute, and decentralize. Hence, these workloads are run on centralized systems, cloud, and at the edge.

EuroHPC program

The EuroHPC initiative[1] is a joint effort by the European Commission and 29 countries to establish a world-class ecosystem in supercomputing to Europe. One of its first concrete efforts is to install the first three "precursor to exascale" (or pre-exascale) supercomputers. Finland, together with 8 other countries from the Nordics and central Europe, will collaboratively host one of these systems in Kajaani, northern Finland. This system, LUMI, will be one of the most powerful and advanced computing systems on the planet at the time of its installation at the end of 2020.

LUMI design

LUMI will be a leadership-class pre-exascale accelerated supercomputer of over 200 Petaflop/s peak performance, fostering the convergence of high-performance computing (HPC), artificial intelligence (AI), and high-performance data analytics (HPDA). The system will be a true precursor to exascale, spearheaded by a large accelerated partition utilizing the latest generation graphics processing units (GPUs). Accelerated computing, backed up by extreme-performance interconnect and storage, is required both for exascale HPC and AI workloads. The system will consist of three islands/partitions that share the same login nodes and storage solutions:

- The Accelerated partition represents the Tier-0 floating-point workhorse of the system, and excels both at HPC and high-performance AI workloads. At the convergence of AI and HPC the partition will be designed with a high-performance and low-latency interconnect, high-performance storage providing bandwidth and IOPS, as well as accelerated compute nodes providing very good floating point performance across half, single and double precision compute.
- The Tier-1 CPU partition targets the wide base of scientific applications and workflows which have not been ported to accelerators, or which do not benefit from accelerators due to lack of inherent parallelism in the problem. This enables scientists to utilize the best tool for each step in their workflow, and expands the impact of the whole system.
- Pre/post-processing & data analysis partition is a special resource for interactive usage, featuring a couple of fat GPU nodes and a couple of SMP-like fat CPU nodes. These are geared for machine learning tasks, data analytics, meshing, interactive visualization and other non-batch and non-MPI-parallel workloads, and in general GPU focused applications, which do not scale well across the interconnect and make extensive use of the large shared memory on the GPU and CPU side. Further use cases include in-memory analytics and databases.

The storage solution features the following components, with volumes cost-optimised for supporting the anticipated use cases, avoiding excessively large solutions:

- A highly capable parallel file system. Anticipated volume is 60 PB or more.

[1] Read more at https://eurohpc-ju.europa.eu

- An accelerated I/O (SSD/NVMe) layer, providing automated tiering capability to and from the parallel filesystem, acting as a cache layer. The target is to achieve more than 1 TB/s sustained bandwidth and an extreme IOPS capability. Anticipated volume is around 5 PB.

A key part of the infrastructure is an object storage service for project-time storage and for convenient data management. This is a central hub for managing data in the Lumi systems, and enables sharing datasets and other results to the world with fine-grained authorization and support for metadata. In addition to the HPC system, this object storage service is also tightly connected to an IaaS solution as well as Kubernetes style container cloud. This allows scientific communities and users to develop services and solutions for accessing, analysing and sharing large sets of data computed on the HPC system.

The LUMI system,is committed to providing the best possible user experience, featuring emerging capabilities such as interactive supercomputing and user-friendly interfaces to the HPC resources as well as support for containers. The system is designed for advanced virtualization capabilities (batch job containers, supporting IaaS cloud and container cloud capabilities and user-friendly interfaces like Jupyter Notebooks) with a pronounced support for data management challenges.

One important value-added service for the LUMI system lies in the concept of "datasets as a service". In analogy with pre-installed scientific software, the system will avail regularly updated and curated (e.g. checksum-verified) versions of reference datasets, for example important reference genomes and language corpora. For AI training tasks, this means the availability of the industry-standard reference training and testing datasets readily available and up-to-date.

Big Data Assimilation Incorporating Deep Learning with Phased Array Radar Data and Numerical Weather Prediction

by
Takemasa Miyoshi
RIKEN Center for Computational Science, Kobe, Japan
takemasa.miyoshi@riken.jp

The Japan's Big Data Assimilation (BDA) project started in October 2013 and ended its 5.5-year period in March 2019. The direct follow-on project was accepted and started in April 2019 under the Japan Science and Technology Agency (JST) AIP Acceleration Research, with emphases on the connection with AI technologies, in particular, an integration of DA and AI with high-performance computation (HPC).

The BDA project aimed to fully take advantage of "big data" from advanced sensors such as the phased array weather radar (PAWR) and Himawari-8 geostationary satellite, which provide two orders of magnitude more data than the previous sensors. We have achieved successful case studies with newly-developed 30-second-update, 100-m-mesh numerical weather prediction (NWP) system based on RIKEN's SCALE model and local ensemble transform Kalman filter (LETKF) to assimilate the big data from PAWR in Osaka and Kobe. We also developed two precipitation nowcasting systems with the every-30-second PAWR data: one with an optical-flow-based system, the other with a deep-learning-based system. We chose the convolutional Long Short Term Memory (Conv-LSTM) as a deep learning algorithm, and found it effective for precipitation nowcasting. The results show that the Conv-LSTM-based nowcasting outperforms the optical-flow-based nowcasting.

The Conv-LSTM takes inputs from most recent PAWR data and produces future data as the prediction. For further improvement, the Conv-LSTM is designed to take inputs from future data. Here, we use prediction data from 30-second-update NWP. The Conv-LSTM system taking inputs from both past PAWR data and NWP forecast data can be considered as a step toward the integration of DA and AI with HPC. Here, the HPC simulations with DA and NWP will provide data for Conv-LSTM. In terms of the *CI-for-the-AI* perspective, the input data to the Conv-LSTM will not be big, so that the real-time operations would be straightforward. However, training the Conv-LSTM would be challenging, since we need a large number of past cases of NWP forecasts and PAWR data.

Another AI-related focus in the BDA project is to use AI for accelerating NWP model computations. NWP is expensive and requires HPC platforms, but we aim to develop an AI-aided NWP system that does not require an HPC but still provides as precise and accurate forecasts as the expensive NWP models. Here, we run expensive, high-resolution NWP models for various cases offline using HPC platforms and use the huge amount of data for training AI. We would not expect to replace the whole complex NWP model with the AI, but instead, we run a simplified (such as low-resolution) version of the NWP model with a small, feasible computer, and hope that the trained AI can fill in the gap to the expensive high-precision simulation. I believe that the process-driven approach is still useful

even with a simplified version. When we train the AI with the tremendous amount of simulation data from the expensive NWP models, it would be prohibitive to move around the data. Therefore, we need to run the AI training on the same HPC platform as where we run the large-scale NWP simulations. We may not even save the data from the HPC simulations, and direct streaming to AI learning processes may be necessary. This may be even more challenging than the Conv-LSTM case in terms of the *CI-for-the-AI* perspective.

"Fugaku", the successor of the K computer, will be a good platform to test these ideas.

Learning at Scale and Learning for Batch Scheduling

Bruno Raffin, Olivier Richard, Denis Trystram,

Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France

1 Introduction

The thoughts presented in this paper are based on results and discussions that emerged from the INRIA team DataMove and the INRIA Project Lab HPC-BigData (focused on the convergence between HPC, AI and Big Data).

2 AI-in-the-CI: ML-based Batch Scheduling Challenges

Resource allocation is a critical component on large scale distributed platforms such as supercomputers. The Resources and Jobs Management System (RJMS), also commonly called batch scheduler, is the component in charge of assigning when and where the applications are executed on the machine. The strain on this component is growing with the complexity of supercomputers, becoming larger and more heterogeneous. The profile of the workload supported is also gaining in diversity, making it more difficult for the job scheduler to comply with the multiple objectives to be optimized. Supercomputers workload now integrates Big Data and AI jobs. Supercomputers are also becoming integrated into larger workflows, typically for Digital Twins applications, imposing new constraints.

But the RJMS decisions are often based on uncertain data. For instance the user needs to supply the expected execution time of the job. However this information is at best roughly overestimated or simply missing. The flow and profile of job submissions is also not regular, influenced by multiple factors that are difficult to capture with traditional algorithms. For instance users tend not to submit the same type of jobs over the week-end or during the work days.

In the Datamove team we have investigated different approaches for using Machine Learning (ML) for resource allocation. First to get a better estimate of the job execution time [1]. We also investigated how to use multi-armed bandit technics to select a job priority amongst the range of priorities computed from 12 scheduling policies [2,3]. One issue with ML algorithms is the lack of performance guarantee. Using ML for choosing between different policies is one way to combine ML with traditional algorithms and avoid this limitation.

Beside the algorithmic challenge, one difficulty is to have access to sufficient and hopefully reasonably unbiased data sets for training these algorithms. Having access to recent traces is difficult. They require a significant work to be curated, before being usable by learning algorithms. One interesting approach

could be to rely on generative machine learning to learn the probability distribution of existing traces. Taking benefit of their generalization capabilities they could enable to generate on-demand large plausible traces.

This needs to be coupled with simulation frameworks to develop, experiment and validate new policies at scale but without having to actually deploy it on a production supercomputer. For that purpose we have been working on developing the BatSim simulator based on SimGrid [4]. BatSim is an extendable, language-independent and scalable RJMS simulator. It allows researchers and engineers to test and compare any scheduling algorithm, with reproducibility guarantees, using a simple event-based communication interface.

3 AI-in-the-CI: Learning at Scale

Parallelization of deep learning is split in between two main approaches: data parallelism and model parallelism.

With data parallelism the neural architecture is duplicated and trained on different examples with periodic averaging of the weights to exchange the "knowledge". Tremendous progressed have been made recently for this approach combining a better way to handle the learning rate for enabling learning with large batches and efficient reduction operations directly coming from the HPC world (MPI_allreduce). For instance the ResNet-50 network, a standard benchmark, progressed from learning in 29 hours in 2016 on 8 GPUs, down to 224 seconds in 2018 with 2176 GPUs. There are still progress to be made, in particular to balance the I/O performance with the compute ones when the training data is read from files.

But data parallelism shows one strong limitation: the size of the network is limited by the memory capability of the device it runs on (often a GPU). Model parallelism consists in partitioning the neural network to distribute it on different devices. This is way more challenging than data parallelism as the dependencies between neurons are tight. This is today a research challenge. In the context of the IPL HPC-BigData we are exploring intermediate approaches to control the memory footprint during the training phase, as well as domain specific neural network splitting strategies.

But needs for scale in the learning go beyond the parallelization of the neural network itself. Reinforcement learning goal is to self-learn a task trying to maximize a reward (a game score for instance) interacting with simulations. Researchers have successfully introduced deep neural networks enabling to address more complex problems. This is often referred as Deep Reinforcement Learning (DRL). The most visible success of DLR is probably AlphaGo Zero that outperformed the best human players (and itself) after being trained without using data from human games but solely through reinforcement learning. The process requires an advanced infrastructure for the training phase. For instance AlphaGo Zero trained during more than 70 hours using 64 GPU workers and 19 CPU parameter servers for playing 4.9 million games of generated self-play, using 1,600 simulations for each Monte Carlo Tree Search. The general workflow is

the following. To speed up the learning process and enable a wide but thorough exploration of the parameter space, the learning neural network interacts in parallel with several instances of actors, each one consisting of a simulation of the task being learned and a neural network interacting with this simulation through the best wining strategy it knows. Periodically the actor neural networks are being updated by the learned neural network. This workflow has evolved through various research works combining parallelization, asynchronism and novel learning strategies (GORILA, A3C, IMPALA,...). But if we look at these results from a HPC point of view the scale is still modest. Very large scale is a challenge as requiring adapted frameworks as well as compliant learning policies, but it could be disruptive, opening the way to new applications of DRL.

Auto deep Learning is an other very compute intensive task that is getting traction. Deep neural networks architectures are defined through many hyper-parameters, i.e. human defined parameters (number and size of layers, choice of activation function, etc.). Current trend is to develop algorithms to explore and define these hyper-parameters. Todays most used solutions rely on genetic algorithms and reinforcement learning (see the Ray Tune library for instance). The challenge is here to develop full features environment that enable users to easily control the full workflow of hyper-parameter setting and their deployment on supercomputers, with in sight the emergence of *neural factories.*

References

1. D. Carastan-Santos and R. Y. de Camargo, "Obtaining Dynamic Scheduling Policies with Simulation and Machine Learning," in *SC'17 -2 International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing)*, Denver, United States, Nov. 2017. [Online]. Available: https://hal.inria.fr/hal-01618940
2. É. Gaussier, J. Lelong, V. Reis, and D. Trystram, "Online Tuning of EASY-Backfilling using Queue Reordering Policies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 10, pp. 2304–2316, Oct. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01963216
3. D. Carastan-Santos, R. Y. De Camargo, D. Trystram, and S. Zrigui, "One can only gain by replacing EASY Backfilling: A simple scheduling policies case study," in *CCGrid 2019 - International Symposium in Cluster, Cloud, and Grid Computing*. Larnaca, Cyprus: IEEE, May 2019, pp. 1–10. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02237895
4. P.-F. Dutot, M. Mercier, M. Poquet, and O. Richard, "Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator," in *20th Workshop on Job Scheduling Strategies for Parallel Processing*, Chicago, United States, May 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01333471

Cyberinfrastructure for AI: Rapid response and recovery after hurricane with AI and UAS

Maryam Rahnemoonfar[1], Robin Murphy[2]

1) Computer vision and Remote Sensing Laboratory (Bina Lab), University of Maryland, Baltimore County
2) Humanitarian Robotics and AI Laboratory, Texas A&M University

For quick response and recovery on a large scale after a natural disaster such as a hurricane, access to aerial images are critically important for the response team. The emergence of small unmanned aerial systems (UAS) along with inexpensive sensors presents the opportunity to collect thousands of images after each natural disaster with high flexibility and easy maneuverability for rapid response and recovery. Moreover, UAS can access hard-to-reach areas and perform data-gathering tasks that are otherwise unsafe or impossible for humans. Despite the ease of data collection, data analysis of the big datasets remains a significant barrier for scientists and analysts. While traditional analyses provide some insights into the data, the complexity, scale, and multi-disciplinary nature of the data necessitate advanced, intelligent solutions.

This whitepaper presents some of our preliminary results for detecting flooded areas after Hurricane Harvey by developing novel artificial intelligence techniques for imagery collected by UAS. We will also discuss some of the challenges and cyberinfrastructure gaps to use AI from Edge to HPC to be able to detect flooded houses and roads, and also the degree of damage to houses from high winds in a timely manner.

Spatial Sensor Data, Effective Training and the Computing Continuum

Joel Saltz
Stony Brook University

Sensors of light and other electromagnetic radiation – cameras and other photodetectors – have dramatically diminished in cost and become ubiquitous. This has led to dramatic increases in the availability and use of spatial and spatio-temporal data at multiple scales. Examples include digital microscopy applications such as digital Pathology, analyses of satellite and aerial image data and analyses of microscopy modalities such as spinning disk confocal, two photon and light sheet microscopy. These seemingly diverse application areas share a need for analyzing increasingly massive sets of spatial or spatio-temporal data. In a large fraction of application domains the data is distributed and cannot easily be centralized for logistical, commercial or legal reasons.

The details of required analysis vary, but generally have in common the need to carry out some form of object detection, object classification and/or semantic image segmentation. The objects detected and classified may be natural objects such as cells, glands, ducts, trees, wildlife or man-made objects. Examples of segmentation include identification of geographic regions with various classes of land use, e.g. forest, lakes, pastures, urban areas and regions of digital Pathology images with tumor, immune cell infiltration or necrotic tissue.

In dynamic (or multi-image) scenarios, there exist multiple high-resolution images of the same area, either acquired at different times, e.g. satellite images across seasons, or a series of 2D slices obtained from a three dimensional tissue block that may be stained in different ways to reveal morphological structures. These images are closely inter-related due to temporal or spatial adjacency, and this requirement for spatio-temporal coherence provides auxiliary information that can greatly assist in classification. In this case, the prediction result or ground truth segmentation label for one of the images imposes a coarse-grained prior to other images of the same object. Importantly, the prior may lack high resolution details, either because it comes from a different sensor/stain or because of uncertainty in pixel-wise mappings across two images.

Deep learning methods have proved to be extremely effective in carrying out the types of tasks described above. However, deep learning techniques for spatio-temporal analysis applications require vast amounts of training data - a crucial challenge is to develop efficient strategies that make highly effective use of limited and strategically focused human input. A variety of methods - individually or in combination - can be used to tackle this challenge. These include 1) generative adversarial networks employed to generate synthetic imagery that can be used to train object detection, classification or semantic segmentation models, 2) super-resolution methods able to generate high resolution results from coarse grained regional labels, combined with limited high resolution labels and 3) methods that systematically combine

labeling data of varying levels of quality, 4) active learning or adaptive methods that iteratively assess and systematically target the need for additional training data.

A major continuum challenge consists of implementation of both algorithms and infrastructure needed to 1) support efficient distributed spatio-temporal model training and 2) carry out ongoing assessments of model accuracy as new sensor data is acquired. The kinds of model training algorithms described above have complex and likely irregular data access patterns. Model training will need to make use of data from multiple sites, active learning methods will need to obtain ongoing distributed human input. Ongoing accuracy assessments are essential as statistical properties of sensor data change over time for myriad reasons – e.g. change in camera technologies, upgrades in whole slide imaging scanners, changes in how specimens processed. In sum, large scale deep learning based analyses of spatial and spatio-temporal imagery will be typically carried out in a continuum setting and there is a tremendous to support methods needed for carrying out sophisticated training strategies in a continuum setting.

White paper: BDEC2, Oct 2019

Inline processing with FPGAs for Edge-to-HPC AI Workflows

Kentaro Sano

Team Leader, Processor Research Team, Riken R-CCS

1. Background

New approach of "AI-for-Science" requires the application of Artificial Intelligence in Cyberinfrastructure (AI-in-the-CI) and Cyberinfrastructure that supports Artificial Intelligence (CI-for-the-AI). For CI-for-the-AI, we need higher processing performance as well as much wider bandwidth for network interfaces to take inputs of large amount of data incoming from the edge. To satisfy these requirements, we need to introduce another computing model which is delivered "near" the network interface so that CPU's load can be effectively used for higher-level tasks instead of processing the incoming data.

2. Proposal: Inline processing with FPGAs

For the data incoming from the edge, we need to perform high-throughput but regular processing for low-level processing such like statistical analysis, compression and encoding/decoding, and preprocessing of AI workloads, as well as packet processing through protocol stacks. However, the performance of general-purpose CPU and the bandwidth between the network interface and CPU are usually not sufficient for processing the incoming data at a required processing rate. Not software-based processing with a CPU, but hardware-based processing with a deeply pipelined customized circuits is suitable for such high-throughput regular processing.

We propose to introduce inline processing with FPGAs (field-programmable gate arrays) into a supercomputing system to support data logistics from the edge together with high-throughput processing near the network interface. FPGA is a reconfigurable semiconductor device where you can repeatedly program your own designed circuit on it. We can implement a network interface, on-chip interconnects, and inline processing hardware modules according to required tasks.

Figure 1 shows the concept of inline processing with FPGAs in a supercomputing system for edge-to-HPC AI workflows. The left-side part of the figure is a supercomputer with FPGAs, where CPUs are connected by the system network to form a massively parallel computer, and FPGAs are attached to the CPUs via the CPU-FPGA network. The example of the CPU-FPGA network is a PCI-express switch while we can also consider more flexible network to connect FPGAs to CPUs. Each FPGA has its own high-speed network ports so that we can connect them to the FPGA-external network such as commercially standardized Ethernet network, which is also connected to IoT devices and machines in Fogs and Edges.

So far, we have designed and prototyped a small experimental system with CPUs and sixteen FPGAs, where an EDR Infiniband network is adopted for the CPU-FPGA network. Figure 2 shows the system-on-chip (SoC) design for FPGA that we are developing for the system. We use Intel's programmable acceleration card (PAC) with Intel Stratix10 FPGA which is fabricated in 14nm. The card also has four DDR4 memories, PCIe gen3 x16 interface, and two QSFP28+ ports each of which supports 100Gbps connection. The SoC contains the FPGA interface manager, FIM, where the fixed hardware modules such like memory interfaces, PCIe interface, on-chip interconnects, and high-speed serial network interfaces are implemented. Once FIM is programmed, we usually don't change this region even when we execute different computation.

The SoC also contains the acceleration function unit, AFU, which is a dynamically reconfigurable region. In AFU, we implement an interconnect, direct memory access controllers (DMACs), and a custom computing core.

We switch AFU region for different processing to be executed on FPGA. Before programing AFU region, we need to design, implement, and compile the hardware of AFU. For a custom computing core, we can use our own data-flow compiler, Stream-processor generator (SPGen), and a high-level synthesis (HLS) compiler to generate RTL codes for the core. Once a core is generated, we embed it to the pre-designed AFU system by using a platform design tool. So far, we have implemented and demonstrate stream computation for Tsunami simulation with the shallow water equation solver [1] and 2D Fluid dynamics simulation based on the lattice Boltzmann method [2]. Now we are designing and implementing the network subsystem on FPGA which is connected to the custom computing core for processing the incoming data.

References

[1] Kohei Nagasu, Kentaro Sano, Fumiya Kono, and Naohito Nakasato, "FPGA-based Tsunami Simulation: Performance Comparison with GPUs, and Roofline Model for Scalability Analysis," Journal of Parallel and Distributed Computing, Vol.106, pp.153-169, DOI:10.1016/j.jpdc.2016.12.015, 2017.

[2] Kentaro Sano and Satoru Yamamoto, "FPGA-based Scalable and Power-Efficient Fluid Simulation using Floating-Point DSP Blocks," IEEE Transactions on Parallel and Distributed Systems, Vol.28, Issue.10, pp.2823-2837, DOI:10.1109/TPDS.2017.2691770, Oct 2017.

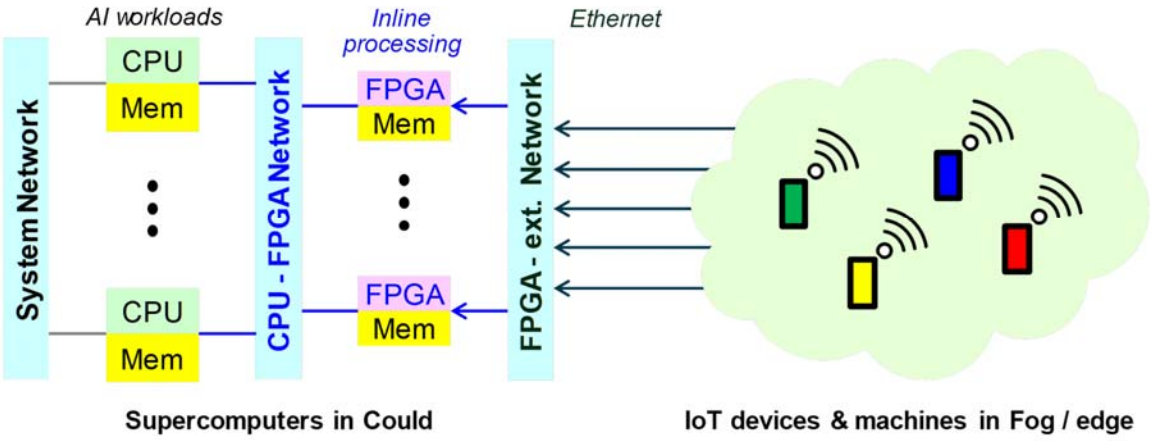


Figure 1: Overview of the proposal.

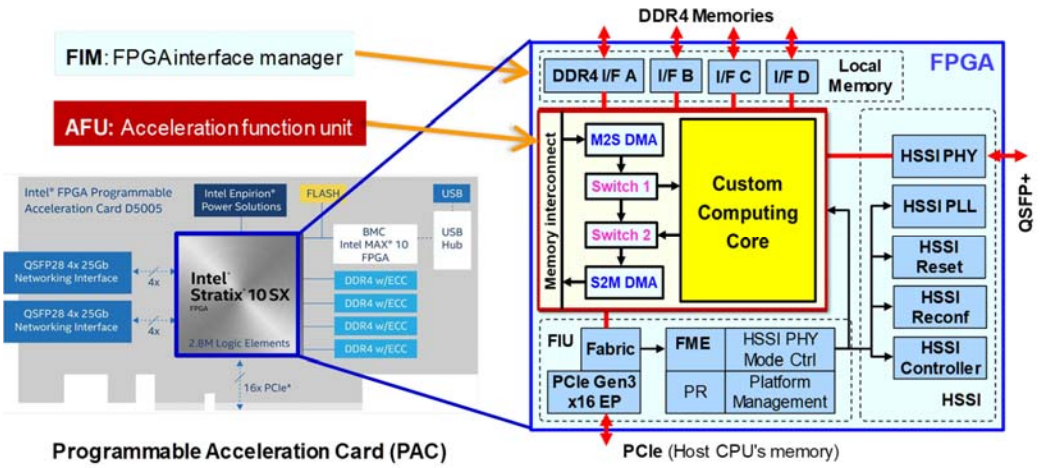


Figure 2: System-on-chip design for FPGA.

Big Data and Extreme Scale Computing, 2nd Series, (BDEC2)
Workshop 4: San Diego, California,
October 15-17, 2019

Data Compression with Deep Predictive Neural Network

Rupak Roy[†1], Kento Sato[†2], Jian Guo[†2], Jen s Domke[†2], Weikuan Yu[†1],
Takaki Hatsui[†3] and Yasumasa Joti[†4]

[†1] Florida State University, [†2] RIKEN R-CCS,
[†3] RIKEN SPring-8 Center, [†4]Japan Synchrotron Radiation Research Institute

A lot of intermediate data has to be generated and transferred for further analysis in data science. A Large Hadron Collider (LHC) in CERN generated about 88PB of data in 2018 and foresees "*Data archival is expected to be two-times higher during Run 3 and five-times higher or more during Run 4 (foreseen for 2026 to 2029).* " [1]. RIKEN also has a large synchrotron radiation facility (SPring-8). Spring-8 public beamline generated 0.32 PB/year in 2017. In 2025, with the next generation detector (CITISU), it is projected that a single beamline will generate 1.3 Exabytes of data per year. For the data analysis, checkpointing, debugging, visualization, etc. ,the data generated by the scientific applications or simulations must be transferred from the sensors to computer systems. Fast transfer of such huge scale data from the sensors in edge-side to computer systems is critical.

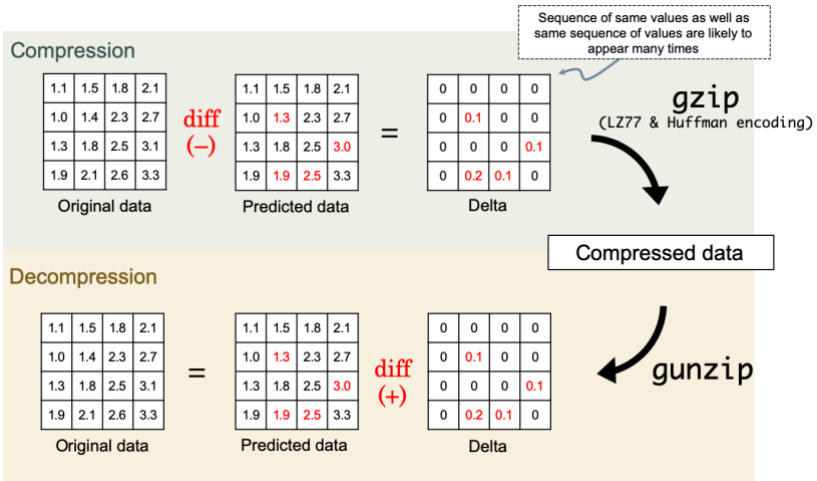


Figure 1: Overview of predictive delta compression

One of the approaches to accelerate data transfer is to reduce data size, i.e., data compression. However, existing compression algorithms show a low compression ratio for such kind of random evenly distributed floating point data. As a result, achieving significant I/O acceleration is not possible with existing compressors. One of the promising approaches is predictive delta compression. Predictive delta compression is a technique to store only difference between original data and

predicted data or the difference between consecutive predicted frames. Therefore, accurate prediction to the original data, which is data to compress, is important (Figure 1), both for increasing the compression ratio by making the delta values very small. Because the image data from sensors are time-evolutinal images, we need a technique to acculatery predict future image frames.

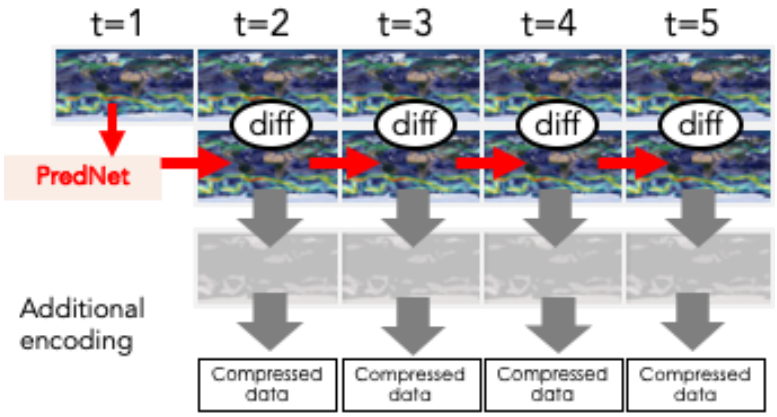


Figure 2: Data compression with predictive neural network

Predictive neural network is a predictive coding based deep convolutional neural network which learns to predict future frames of a video sequence. PredNet [2] is such an architecture which is trained to predict the future movement of objects. We use PredNet to predict the future frames from the given time evolutional frames. First, we train PredNet to learn movement of pixels by giving a number of time-evolutional frames generated from the sensor. In the example, in Figure 2, when we compress frames from t=2 to t=5, we predict future frames from original frame (t=1), we compute the difference and then apply conventional compressors such as gzip. Since we can always generate the predicted frames with the original frame at t=1 with the help of trained neural network. We can restore the original frames by only storing (1) the original frame; (2) the trained neural network; (3) compressed frames from t=2 and t=5. For compressing the frames we pass the delta data through two steps. The steps are modified spatial delta encoding and modified entropy coding. In the evaluation, we observe that we can compress SPring-8 data by a factor of 40 compared to original size. Our approach shows 2.8x time better compression ratio compared to a recent compression algorithm SZ[3].

REFERENCES

[1] Esra Ozcesmeci, “LHC: pushing computing to the limits”, https://home.cern/news/news/computing/lhc-pushing-computing-limits, March 1st, 2019
[2] WilliamLotter,GabrielKreiman,andDavidCox.Deeppredictivecodingnetworks for video prediction and unsupervised learning. arXiv preprint arXiv:1605.08104, 2016.
[3] S. Di and F. Cappello. Fast error-bounded lossy hpc data compression with sz. In2016 ieee international parallel and distributed processing symposium (ipdps), pages 730–739. IEEE, 2016.

AI Feedback Loop for Redshift Surveys in Astronomy

Alex Szalay, The Johns Hopkins University

The Prime Focus Spectrograph and Wide-Field Spectroscopy

The Prime Focus Spectrograph (PFS) is a major new multi-object fiber-fed spectrograph currently under construction for the Subaru 8.2-meter telescope on the summit of Mauna Kea, Hawai'i. PFS will be the largest and most powerful multi-object spectrograph in the world when it sees first light in late 2019. It has 2394 optical fibers deployed over a field of view of 1.3 deg^2 on one of the largest optical telescopes in the world, allowing ~2000 spectra to be measured simultaneously. One of the great challenges of designing the survey is determining exactly which objects will be observed spectroscopically, and how long each should be exposed. An on-going imaging survey using a camera called Hyper Suprime-Cam (HSC), also on the Subaru telescope, is providing the raw material for the PFS: it is identifying the faint stars and galaxies that PFS will obtain spectra of. However, there are orders of magnitude more objects in the imaging data than we can obtain spectra for in a 300-night survey, despite the multiplexing power of PFS.

Optimizing Target Selection for the Galaxy Evolution Survey

We seek to optimize the target selection for PFS such that we can maximize the scientific utility of the entire survey given its observational constraints. This task is complicated by our not knowing in advance what the data will show, and what structures and correlations it will find. We therefore seek to devise a scheme that allows us to update the target selection on-the-fly such that we get an optimal overall yield. The science cases of PFS are complex and multi-pronged, and one of the challenges of optimizing the targeting for them is that the science return as a function of the nature of the data are highly non-linear functions that may not be fully known in advance. The Galaxy Evolution component of the survey, requiring roughly 100 nights of telescope time, will be carried out in special regions of the sky covering 15 square degrees (roughly sixty times the area of the full Moon on the sky), in which HSC has obtained particularly deep imaging. There is a broad range of scientific goals for this component:

- Measure the distances and physical properties of typical galaxies in the range of cosmic time (between 6 and 10 Byears ago) when most of the stars in galaxies formed.
- Measure the large-scale distribution of galaxies at those epochs, to understand how the physical properties of galaxies have evolved and been influenced by their environments.
- Map the distribution and trace the physical nature of galaxies at the earliest epochs, less than 1B years after the Big Bang, to gain new insights on how the first galaxies formed.

Identifying the galaxies to be observed and determining how long each exposure should be, is a challenge. For each galaxy in the images, we can make a first guess as to its redshift, as well as its mass, using the broad-band colors determined from the imaging data, a technique termed "photometric redshift" (Budavari 2009). Given the galaxy's brightness, we can then estimate (i) roughly how long an exposure is needed to get a spectrum good enough to measure a precise redshift, and (ii) how much longer we would have to integrate in order to determine the stellar makeup of the galaxy and other internal physical properties.

The challenge is how to do this in an optimal and dynamic way. One of our key science goals is to map the large-scale distribution of galaxies. Given the large number of targets in any given field, the long (and varying) exposure times for each one, and the complexity of the "cosmic web" of filaments, clusters and voids which galaxies trace out, we need to make a decision galaxy by galaxy and exposure by exposure, to observe the highest priority objects to maximize our science.

We intend to perform a Bayesian reconstruction of the cosmic web using a technique termed

"Bayesian Origin Reconstruction from Galaxies" or BORG, (Jasche & Wandelt 2013, Leclercq et al 2015), conditioned on all galaxy spectra available at a given time. The BORG algorithm estimates the set of cosmological initial conditions consistent with a given set of measured galaxy positions and redshifts, using our knowledge of the growth of structure under gravity to evolve from the initial conditions to the observed data. The Bayesian approach yields samples of the initial and final positions and velocities of galaxies, from which one can determine the most probable initial density field, as well as its uncertainty. To improve the reconstruction of the cosmic web, we can keep running the BORG algorithm as the survey continues and obtain spectra for target galaxies in regions of high variance of the reconstruction. Similarly, for those galaxies whose spectra yield the absorption lines that trace the cosmic web in neutral hydrogen, we can quantify how higher S/N per galaxy can refine the mapping of the web.

The utility for this LSS program is therefore defined by reducing the reconstruction uncertainty of the cosmic web and, by extension, the classification uncertainty for the environment. This is a highly non-linear and evolving utility function, for which there is no analytic prescription. We plan to *learn* that decision from the data by running BORG with randomly selected subsets of the data, withholding some samples, and testing the improvement from the withheld samples as a function of their position and signal-to-noise ratio. This process will be run as the actual observations are underway, allowing us to train a neural network that predicts the optimal targets for the next set of observations, a regression scheme for the complex utility function. Prior work (e.g. Pedro & Takahashi 2011a,b) provides a viable basis for starting our investigations. How those various, and sometime competing utilities are weighed will have to be determined according to the preference of the survey scientists towards specific science goals. The approach outlined above will give us a powerful and transparent approach for optimal target selection and the flexibility to shift it over time if priorities change.

Translating to Machine Learning Challenges

Our goal is to define and implement a machine learning strategy that will maximize the information gain from spectroscopy during a survey using a rapid feedback loop between the predictions of properties of our objects of interest and their uncertainties, given our (evolving) models and the spectroscopic target selection and subsequent observations. Doing so requires:

- The use of models and the ability to improve them given newly acquired information.
- An objective and optimal selection of targets to sample the space of model parameters.
- Rapid evaluation of the new spectroscopic observations

References

Budavari, T. A Unified Framework for Photometric Redshifts, Astrophys. J., 695, 747, (2009).

Jasche, J., Wandelt, B.D., Bayesian physical reconstruction of initial conditions from large-scale structure surveys, M.N.R.A.S., 432, 894, (2013).

Leclercq, F., Jasche, J.Lavaux, G. Wandelt, B.D., Probabilistic cartography of the large-scale structure, arXiv:1512.02242v (2015).

Pedro, L. R. & Takahashi, R. H. C., Modeling Decision-Maker Preferences through Utility Function Level Sets, in Evolutionary Multi-Criterion Optimization, Takahashi, Deb, Wanner, Greco (eds.), Springer, (2011)

Pedro, L.R., and Takahashi, R.H.C., Modeling decision-maker preferences through utility function level sets, In 6th International Conference on Evolutionary Multicriterion Optimization, volume 1, (2011).

BDEC-2 White Paper: AI/Deep Learning Computing at the Edge Exemplar: Fusion Energy Sciences

W. M. Tang, Princeton University/PPPL

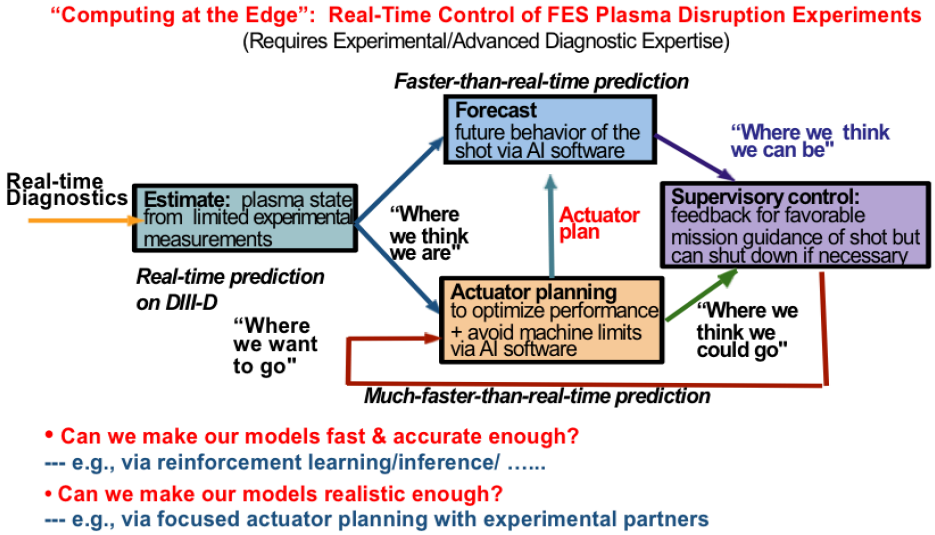
Accelerated progress in producing accurate predictions in science and industry have resulted from recent engagement of big-data-driven statistical methods featuring artificial intelligence/deep learning/machine learning (AI/DL/ML). Such techniques have enabled new avenues of big-data-driven discovery in key scientific application areas, including the quest to deliver Fusion Energy. Tokamaks currently represent the closest magnetic fusion energy (MFE) approach to satisfying the basic Lawson Criterion or "triple product" requirement of density, temperature, and confinement time and the associated ignition requirement for delivering fusion energy[1] However, an especially time-urgent and challenging problem facing the development of a fusion energy reactor following this approach is the need to reliably predict and avoid large-scale major disruptions -- powerful instabilities that result in a loss of plasma confinement which if unmitigated can result in significant damage to the plasma facing components and structural integrity of the whole device. As these instabilities become more severe for larger tokamaks, their prediction and mitigation is important in tokamak systems such as the EUROFUSION Joint European Torus (JET) today and will be crucial for the burning plasma ITER device in the near future.

High performance computing (HPC) advances in the deployment of AI-enabled software is increasingly visible today. For example, Princeton's Fusion Recurrent Neural Network code (FRNN) uses convolutional & recurrent neural network components to integrate both spatial and temporal information for predicting disruptions in tokamak plasmas with unprecedented accuracy and speed on top supercomputers worldwide – including SUMMIT – the current #1 supercomputer. This R&D accomplishment was recently published in April 25, 2019 in the top scientific journal NATURE, DOI: 10.1038/s41586-019-1116-4, p. 526-531. In this exemplar, statistical deep learning/AI software trained on very large observational data sets (such as JET's huge disruption-relevant database) demonstrates exciting promise for delivering much-needed predictive tools capable of accelerating scientific knowledge discovery. The associated creative methods being developed here also have significant potential for cross-cutting benefit to a number of important application areas in science and industry. Key Achievements featured in NATURE paper include: (1) First demonstration of crucially-needed ability for predictive software trained on one experimental device (e.g., DIII-D tokamak) to make accurate predictions on another (e.g., the much larger, more powerful JET system) –> a key requirement for ITER relevance; (2) Unique demonstration of AI/DL software capability to efficiently utilize leadership class supercomputers -- e.g., Titan, Summit in US; Tsubame-3 in Japan, etc. — and exciting powerful systems in near future such as in (US) AURORA-21 (ALCF), Perlmutter (NERSC), Frontier (OLCF) and in (JAPAN): ABCI & POST-K "FUGAKU."

The key outstanding AI/DL challenge to be met in FES is to integrate unprecedented predictive accuracy to real-time control. In BDEC-2 terminology, this is basically a "Computing at the Edge" problem to deliver AI/HPC for real-time control of "live" experiments in FES – as illustrated here,

[1] Lawson, J. D. (December 1955). "Some Criteria for a Power Producing Thermonuclear Reactor". Proceedings of the Physical Society, Section B. 70 (1): 6-10, doi:10.1088/0370-1301/70/1/303

Also: Wikipedia: https://en.wikipedia.org/wiki/Lawson_criterion & https://en.wikipedia.org/wiki/Fusion_energy_gain_factor.



Associated efforts will demand enhanced cross-disciplinary exploration with novel real-time control methods learned from expertise residing, for example, in application areas such as robotics and self-driving cars. This can enable leveraging vast experience from well-established institutions, including (e.g.,Alan Turing Institute in UK) and active industrial engagement, such as Microsoft, NVIDIA, INTEL,..). Moreover, it can build on the proven beneficial experiences from cross-disciplinary HPC programs such as SciDAC, CoDesign, ECP, …
Example: NIH's "Cancer Moonshot," including collaboration with DOE's Exascale Computing ***"Candle Project"*** *to identify optimal cancer treatment strategies, by building a* ***scalable deep neural network code called the CANcer Distributed Learning Environment (CANDLE)*** → *development of predictive models for drug response, and automation of the analysis of information from millions of cancer patient records -- via developing, implementing, & testing DL/AI algorithms and their benchmarks*

.

Required Investmentments will include:

- R&D in people; especially young talent
- New diagnostics & actuators in experimental facilities leading to ITER
- Access with support for most powerful supercomputing resources to enable rapid training of huge amounts of experimentally measured complex data
- Accelerated development of cross-disciplinary strategies:
  --- Example 1: Optimize hyper-parameter tuning workflows – such as FRNN in FES & CANDLE in Cancer precision medicine with identified similarities to collaboratively exploit;
  --- Example 2: Performance optimization and scalability improvement of deep learning algorithms via Applied Math and Computer Science innovations.

The focus of this Exemplar is on the main FES Tokamak line because this approach represents the closest MFE track toward actually achieving the Lawson criterion "triple product" and associated ignition requirement for delivering fusion energy. For risk mitigation purposes, alternative confinement concepts (stellarators, FRC's, etc.) can of course also be pursued with AI-based approaches. In order to meet the goals of the $25B international burning plasma ITER tokamak experiment, AI-enabled capabilities will need to be delivered in a timely way to help ensure real-time control of the plasma state in burning plasmas. Overall, risk mitigation strategies must focus on "big data from observations/measurements" needed for the rigorous validation and uncertainty quantification analysis featuring realistic sensitivity studies.

The specific requirement here is to develop and implement a true AI/DL FRNN-based prediction capability within an actual tokamak plasma control system – such as the DIII-D tokamak facility in S. Diego, CA. Here, it is necessary not only to accurately predict disruptive activity but also to categorize the primary causes. From a control perspective, the myriad (over 200 !) actuator possibilities make it essential to be able to properly design and effectively utilize the advanced AI/DL capabilities to automatically make the best choices.

PRIONN: Predicting Runtime and IO using Neural Networks

Michael R. Wyatt II and Michela Taufer
University of Delaware and University of Tennessee Knoxville

Todd Gamblin, Stephen Herbein. Adam Moody, and Dong H. Ahn
Lawrence Livermore National Laboratory

In this presentation we want to tackles the problem of how AI can support current batch schedulers on HPC systems to predict resource usage, such as IO and network bandwidth, when allocating resources to jobs. Currently, when submitting HPC jobs, users submit job scripts to a batch system with requests for compute resources (i.e., number of nodes and cores) for a period of time. Consequently, current batch schedulers have access to and use only information on the number of nodes and time for their job allocation decision, omitting the fact that jobs still contend for other resources. For example, co-scheduling many IO-intensive jobs can cause IO contention and underutilization of other resources, ultimately degrading performance (i.e., execution slowdown) as the jobs compete for access to the parallel filesystem. In general, including resource-awareness in schedulers, by considering a broader range of resources, such as IO and network bandwidth, can solve contention and underutilization problems: jobs can be scheduled so that resource usage is known and balanced [1, 2].

Clearly, current resource usage requests (i.e., number of nodes and time) are insufficient to achieve resource-aware scheduling. Delegating the specification of resource usage to users is not a feasible solution either. Analysis of job traces in HPC centers shows how user-requested runtimes are often overestimated. For example, user-requested runtimes for nearly 300,000 jobs on the Cab cluster at Lawrence Livermore National Laboratory in 2016 had a mean error of 172 minutes (24% relative accuracy). Current batch scheduler policies that terminate jobs when they exceed the user-requested time are often the cause for runtime overestimation. Furthermore, when extending the type of resources considered in scheduling decisions to include IO and other resources, users in scientific computing do not have an accurate understanding of jobs' requirements for a given HPC system, and cannot be expected to learn how to accurately estimate the associated resource usage. If schedulers had access to accurate estimates of jobs' resource usage, they would better deal with system resource contention and increase scientific throughput. This need can be addressed by predictive methods that can accurately obtain the per-job resource usage information necessary for resource-aware scheduling.

Our work has built a tool to accurately predict per-job runtime and IO bandwidth called PRIONN (Predicting Runtime and IO using Neural Networks). We use the predictions to enable IO-awareness in a real HPC scheduler when it allocates resources to jobs. The need for accurate resource usage estimates was addressed in previous efforts with traditional machine learning models (e.g., Decision Tree and k-Nearest Neighbors) that were mainly used to predict runtime [3, 4, 5, 6, 7] and, in some cases, more general resource usage [6, 7]. This previous work has shown that machine learning models can provide more accurate runtime estimates than users [4]. However, previous work also required development and maintenance of parsers to extract features from diverse sets of jobs scripts for input into machine learning models. Therefore, when relying on simple parsers, the previous efforts have failed to generalize (i.e., different types of scripts require different parsers). Moreover, any effort to write general parsers incurs the cost of truncating and removing unique information present only in subsets of job scripts. In other words, not all information from job scripts can be captured by a parser.

Contrary to the traditional ML techniques listed above, deep learning models can automatically detect important patterns in data without truncating and removing text specific to a given script. Therefore, deep learning models remove the need for any manual parsing and feature extraction. This capability has been demonstrated thoroughly in the past decade with the application of Convolutional Neural Networks

(CNNs) where traditional models have failed, such as in image recognition and text classification [10, 11, 12]. The challenge of using deep learning models, such as CNNs, when dealing with predicting resource usage from job scripts is how to transform non-image data (i.e., the job scripts) into image-like data suitable for CNNs. We tackle and solve this challenge with PRIONN, our automatic, general, and scalable tool for Predicting Runtime and IO using Neural Networks. PRIONN relies on a novel, direct, and quick data mapping method for job scripts that allows us to exploit the power of deep learning models to provide accurate per-job runtime and IO resource (i.e., per-job IO bandwidth usage) predictions. PRIONN is unique in that it concurrently transforms entire job scripts into image-like data, and by doing so it removes the need for script-specific parsing and feature extraction. We feed the image-like representation of job scripts into a 2D CNN, and by doing so we unleash the power of deep learning models to accurately estimate job resources from whole job scripts. The PRIONN automatic workflow (i.e., mapping and feeding into CNN), when integrated into HPC schedulers, provides us with a general and scalable solution for runtime and IO resource predictions.

The contributions of this work are as follows:

- The PRIONN components used to leverage a neural network to interpret whole job scripts and predict per-job runtime and IO resource in HPC systems;
- The evaluation of PRIONN's ability to outperform traditional machine learning techniques in accurately predicting per-job runtime and IO resources; and
- The application of PRIONN's predictions (i.e., per-job runtime and IO resources) to capture system IO and IO bursts for an IO-aware scheduler.

We will show results on how PRIONN can achieve over 75% mean and 100% median accuracy for predictions of per-job runtime and IO resources across nearly 300,000 jobs from a real HPC machine. We inject PRIONN's predictions into an IO-aware scheduler that manages a real HPC system. Because of PRIONN, the system's scheduler becomes aware of system IO and predicts over 50% of IO bursts (i.e., times of IO contention) in advance. This work was initially presented at ICPP 2018 [13].

References:

[1] Stephen Herbein, Dong H. Ahn, Don Lipari, Thomas R.W. Scogland, Marc Stearman, Mark Grondona, Jim Garlick, Becky Springmeyer, and Michela Taufer. 2016. Scalable I/O-aware job scheduling for burst buffer enabled hpc clusters. In Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC '16). ACM, New York, NY, USA, 69–80.

[2] Jay Lofstead, Fang Zheng, Qing Liu, Scott Klasky, Ron Oldfield, Todd Kordenbrock, Karsten Schwan, and Matthew Wolf. 2010. Managing Variability in the IO Performance of Petascale Storage Systems. In Proceedings of the 22nd ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC). 1–12.

[3] Warren Smith, Ian Foster, and Valerie Taylor. 1998. Predicting application run times using historical information. In Workshop Job Scheduling Strategies for Parallel Processing (JSSPP). 122–142.

[4] Shonali Krishnaswamy, Seng Wai Loke, and Arkady Zaslavsky. 2004. Estimating computation times of data-intensive applications. IEEE Distributed Systems Online 5, 4 (2004).

[5] Dan Tsafrir, Yoav Etsion, and Dror G Feitelson. 2007. Backfilling using systemgenerated predictions rather than user runtime estimates. IEEE Transactions on Parallel and Distributed Systems 18, 6 (2007).

[6] Renato LF. Cunha, Eduardo R. Rodrigues, Leonardo P. Tizzei, and Marco AS. Netto. 2017. Job placement advisor based on turnaround predictions for HPC hybrid clouds. Future Generation Computer Systems 67 (2017), 35–46.

[7] Andréa Matsunaga and José AB Fortes. 2010. On the use of machine learning to predict the time and resources consumed by applications. In Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid). 495–504.

[8] Ryan McKenna, Stephen Herbein, Adam Moody, Todd Gamblin, and Michela Taufer. 2016. Machine Learning Predictions of Runtime and IO Traffic on High- End Clusters. In 2016 IEEE International Conference on Cluster Computing (CLUSTER). 255–258.

[9] Eduardo R. Rodrigues, Renato LF. Cunha, Marco AS. Netto, and Michael Spriggs. 2016. Helping HPC users specify job memory requirements via machine learning. In Proceedings of the Third International Workshop on HPC User Support Tools. 6–13.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th Neural Information Processing Systems Conference (NIPS). 1097–1105.

[11] Patrice Y Simard, David Steinkraus, John C Platt, et al. 2003. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR), Vol. 3. 958–962.

[12] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI), Vol. 333. 2267–2273.

[13] R. Wyatt II, S. Herbein, T. Gamblin, A. Moody, D. Ahn, and **M. Taufer**. PRIONN: Predicting Runtime and IO using Neural Networks. *In Proceedings of the International Conference on Parallel Processing (ICPP)*, pp. 1 – 12. Eugene, OR, USA. August 13-16, 2018.

Rethinking Scalable Services for the Internet of Things

Rich Wolski, Chandra Krintz, Fatih Bakir, Wei-tsung Lin, Gareth George
University of California, Santa Barbara

1 Introduction

As the Internet of Things (IoT) grows in size and ubiquity, it is becoming critical that we perform data-driven operations (i.e. analytics, actuation, and control) at the "edge" to reduce the latency, response time, cost, and energy use for IoT applications. As such, edge systems increasingly co-locate data management and analysis services with sensing, instead of requiring that devices ship their data over long-haul networks for remote processing using the traditional "cloud" model.

Edge systems [3, 5, 8–10, 19, 20] typically implement a publish/ subscribe (pub-sub) model in which devices publish streams of data (often via a nearby broker) that are analyzed by applications that, today, run in the cloud. Indeed, the technologies that are provided by commercial public cloud vendors (based on MQTT [5]) tacitly assumes that devices are sensors that actuate in response to application behavior. That is, sensors are publishers in the typical pub-sub deployment and applications are subscribers.

However, in our view, a client-server model is well suited to emerging IoT applications in which resource constrained edge devices provide *nanoservices* – data-driven actuation and control, data analysis, processing, *and* sensing. That is, the current cloud-based approach that places services in the clouds and clients on sensors *will need to be reversed* to accommodate the upcoming technological demands of IoT.

Further militating for this disruptive shift, IoT deployments in the not-too-distant future will include multi-function devices. As a result, a services model in which the specific function (including data publication) can be requested from each device when it is needed is more appropriate. Further, because devices will continue to be resource constrained, they will require "helper" services at the edge that augment device capabilities and enable scale. In this paper, we outline this approach to implementing *Devices-as-Services* and describe some of the capabilities of an early prototype.

Summarizing the motivation for a new, "flipped" internet architecture:

- IoT applications can and will likely be structured as collections of services that require functionality from a device tier, an edge tier, and a cloud tier
- in-network data processing can significantly reduce response time and energy consumption [23],
- edge isolation precludes the need for dedicated communication channels between application and devices, and facilitates privacy protection for both data and devices,
- actuation in the device tier will require some form of request-response protocol where the device fields the request,
- the heterogeneity of devices (e.g. architectural heterogeneity, capability heterogeneity, etc.) argues for a single programming paradigm and a single distributed interaction model, and
- multi-function devices can and will be able to perform their functions (including data publication) conditionally based on application needs (e.g. as a power optimization).

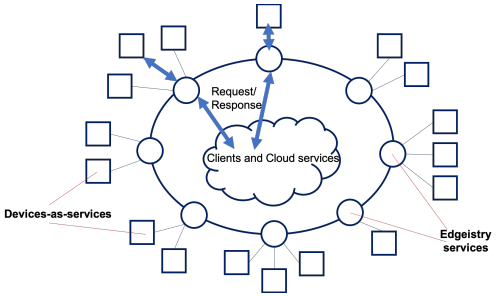


Figure 1: *A new distributed services model for IoT: "client" applications in the cloud compose services exported by resource-constrained devices at the edge via Edgistries – edge nodes that facilitate device discovery/registration, privacy mediation, and optimization.*

Thus we propose a new model for distributed IoT applications in Figure 1. We *flip* the client-server model such that devices at the edge are "servers" that although resource constrained, service multiple, scalable applications (i.e. clients) deployed in the cloud. Such a model requires a new distributed architecture that leverages edge resources in these ways.

The architecture is based on the functions as-a-service (FaaS) programming and deployment model (the execution

engine that underpins serverless computing for clouds [13, 15–17]), and integrates a new approach for efficient client and server discovery, authentication, and authorization via a combination of capability-based security [7, 11, 18] and a set of edge services for service registry, privacy mediation, and optimization called an *Edgistry*.

2 Devices-as-services – a New Approach

Our view is that the current Internet and cloud architecture should "reverse" to accommodate scalable IoT applications. Rather than hosting services in the cloud (logically making IoT devices clients of those services), we view devices as "servers" and applications running in the cloud as "clients".

This viewpoint is supported by several observations. First, devices are resource restricted making them capable of delivering limited and relatively fixed functionality. This functionality is naturally described as an enumerable set of services (responses to requests) that are composed by applications. These services are necessarily "small" but even fully resourced cloud services are now being developed as microservices [14]. Devices are the "nanoservices" in an IoT setting.

Secondly, applications must compose device capabilities from vast collections of devices into meaningful functionality for users, and in an IoT setting, the user scale will be substantial. The cloud is where this scaling will necessarily take place, both in terms of aggregating device functionality, and matching these aggregations to user demand for them.

Thirdly, much of the current technological development for IoT [3, 10] is focused on bringing devices to the cloud. Many of the commercial offerings provide an SDK for using MQTT [5, 20] or AMQP [1] to publish telemetry, events, etc. to objects (which subscribe to device channels) in the cloud that represents each device. For actuation, each device must separately subscribe to such a channel to receive asynchronous commands (although for low-power applications that use MQTT-SN, this option is not possible). With our system, the same server code runs at all tiers – device, edge, and cloud – unifying the device API as a first-class services API.

The currently prevalent commercial IoT/cloud APIs require devices to act separately as "publishers" or "subscribers" or both (the last to implement "closed-loop" actuation). We believe that IoT infrastructure must accommodate a richer model in which devices (however resource restricted) are capable of actions as well as event telemetry. Thus, logically, IoT devices are better modeled as servers that provide services to applications.

2.1 SPOT – A Serverless Platform of Things

To address these challenges requires a universal and consistent set of abstractions that tames the vast device heterogeneity and makes it possible to write application code capable of running at *any* resource scale – from resource restricted microcontrollers to the public clouds. To this end, we have developed SPOT (Serverless Platform of Things [22]) as a portable, high-performance, and multi-scale runtime system for IoT. SPOT is event-based and is similar in conception to cloud "Functions as a Service" [2, 12] (FaaS) but with some key differences. First SPOT defines a portable, append-only, and efficient storage abstraction so that it does not need to rely on external storage services for function.

Secondly, the SPOT API is simple enough yet rich enough to make it possible to run it as the native operating system on microcontrollers. However, it is "serverless" (using Linux containers for configuration isolation) when run on systems that support Linux OS functionality. Thus, SPOT is multi-scale – capable of running *exactly* the same code on all devices (resource restricted, edge, cloud) in an IoT setting.

Thirdly, SPOT is high performance. The current version of spot is between *one and two orders of magnitude* faster that either AWS or Azure commercial FaaS systems Thus, SPOT is multi-scale – capable of running *exactly* the same code on all devices (resource restricted, edge, cloud) in an IoT setting. Thirdly, SPOT is high performance. The current version of spot is between *one and two orders of magnitude faster* than either AWS or Azure commercial FaaS systems. In this way, SPOT provides a common, unifying platform for IoT applications running at all scales.

2.2 The Edgistry

Our approach splits the provisioning of services between devices and a common set of management services located at the edge, termed *The Edgistry*. Using SPOT, devices host small, resource-light services that field requests from, and respond to, client applications (hosted in the cloud or edge).

The purpose of the Edgistry is threefold. First, it provides a common platform for device *commissioning*. When devices are integrated into an IoT deployment, they must be securely registered with the other deployment components. Currently, this registration takes place in the public clouds using public-key encryption. We advocate using a capability-based system based on cryptographic hashing. Our initial implementation is between *one and three orders of magnitude faster* than public key approaches when implemented on several popular microcontrollers [4]. Not only does this improve the power efficiency associated with commissioning, we use the same capability scheme to perform message authentication. Again, the power savings for this approach are substantive which is important when the devices are battery powered.

Secondly, the Edgistry implements access control policies for device-hosted services. Using capability attenuations [6], an Edgistry node can be delegated to implement access control polices on behalf of a device in a way that the device can check using with the efficiencies that the capability system provides.

Thirdly, the Edgistry implements a distributed "pub-sub" system for device metadata. In an IoT context, device discovery must be scalable and tamper proof. Rather than employing a centralized broker (as with MQTT and AMQP) the Edgistry implements this functionality using cryptographic verification and distributed hashing.

3 Conclusions

We propose a new "flipped" client-server model for IoT in which devices at the edge are servers that provide nanoservices, which applications in the cloud (the clients) compose for their implementations. We contribute a novel approach to distributed service design based on "FaaS everywhere," edge-level support, and a novel capability mechanism for distributed policy implementation, a fuller exposition of which is available from [21]. Our empirical evaluation shows that this approach is feasible and introduces very little overhead and power consumption.

This research is supported in part by NSF (CNS-1703560, OAC-1541215, CCF-1539586), ONR NEEC (N00174-16-C-0020), AWS Cloud Credits for Research, and the USC Viterbi Center for Cyber-Physical Systems and the Internet of Things.

References

[1] Amqp home page. https://www.amqp.org, 2019. [Online; accessed 2-May-2019].

[2] AWS Lambda. https://aws.amazon.com/lambda/. [Online; accessed 15-Nov-2016].

[3] Azure Internet of Things. https://www.microsoft.com/en-us/cloud-platform/internet-of-things-azure-iot-suite. [Online; accessed 22-Aug-2016].

[4] F. Bakir, R. Wolski, C. Krintz, and G. Sankar Ramachandran. Devices-as-Services: Rethinking Scalable Service Architectures for the Internet of Things. In *USENIX HotEdge*, 2019.

[5] A. Banks and R. Gupta. Mqtt v3.1.1 protocol specification, 2014.

[6] Arnar Birgisson, Joe Gibbs Politz, Ulfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In *Network and Distributed System Security Symposium*, 2014.

[7] Arnar Birgisson, Joe Gibbs Politz, Úlfar Erlingsson, Ankur Taly, Michael Vrable, and Mark Lentczner. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In *Network and Distributed System Security Symposium*, 2014.

[8] Andy Rosales Elias, Nevena Golubovic, Chandra Krintz, and Rich Wolski. Wheres the bear?–automating wildlife image processing using iot and edge cloud systems. In *ACM Conference on IoT Design and Implementation*, 2017.

[9] Fog Data Services - Cisco. http://www.cisco.com/c/en/us/products/cloud-systems-management/fog-data-services/index.html. [Online; accessed 22-Aug-2016].

[10] GreenGrass and IoT Core - Amazon Web Services. https://aws.amazon.com/iot-core,greengrass/. [Online; accessed 2-Mar-2019].

[11] S. Gusmeroli, S. Piccione, and D. Rotondi. A Capability-based Security Approach to Manage Access Control in the Internet of Things. *Mathematical and Computer Modelling*, 58(5-6), September 2013.

[12] Joseph M. Hellerstein, Jose Faleiro, Joseph E. Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov, and Chenggang Wu. Serverless computing: One step forward, two steps back. In *Conference on Innovative Data Systems Research (CIDR)*, 2019.

[13] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. Serverless computation with openlambda. In *HotCloud*, 2016.

[14] M. Jung, S. Mollering, P. Dalbhanjan, P. Chapman, and C. Kassan. Microservices on AWS. https://docs.aws.amazon.com/aws-technical-content/latest/microservices-on-aws/introduction.html, September 2017. [Online; accessed 2-Mar-2019].

[15] H. Lee, K. Satyam, and G. Fox. Evaluation of production serverless computing environments. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, July 2018.

[16] T. Lynn, P. Rosati, A. Lejeune, and V. Emeakaroha. A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms. In *IEEE International Conference on Cloud Computing*, Dec 2017.

[17] G. McGrath and P. R. Brenner. Serverless computing: Design, implementation, and performance. In *International Conference on Distributed Computing Systems Workshops*, June 2017.

[18] S. Mullender, G. van Rossum, A. Tanenbaum, R. van Renesse, and H. van Staveren. Amoeba – A distributed Operating System for the 1990's. *IEEE Computer*, 23(5), May 1990.

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The Case for VM-based Cloudlets in Mobile Computing. *IEEE Pervasive Computing*, 8(4), 2009.

[20] A. Stanford-Clark and H. Truong. Mqtt for sensor networks (mqtt-sn) protocol specification, 2013.

[21] R. Wolski and C. Krintz. CSPOT: A Serverless Platform of Things. Technical Report 2018-01, UC Santa Barbara, 2018. https://www.cs.ucsb.edu/research/tech-reports/2018-01.

[22] R. Wolski, C. Krintz, F. Bakir, G. George, and W-T. Lin. CSPOT: Portable, Multi-scale Functions-as-a-Service for IoT. In *ACM Symposium on Edge Computing*, 2019.

[23] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3), September 2002.