Remarks on Next Generation (NG) of Cyberinfrastructure for BDEC

Geoffrey Fox, Indiana University and Shantenu Jha, Rutgers University
September 8, 2019

Goals of NG Cyberinfrastructure

- We want to discover a very small number of classes of Hardware-Software systems that together will support all major Cyberinfrastructure (CI) needs for researchers in the next 5 years. It is unlikely to be a single class but a **small** number of shared major system classes will be attractive as it
    - Implies not many distinct software stacks and hardware types to support
    - The size of resources in any area goes like Fixed Total Budget/Number of distinct types and so will be larger if we just have a few key types.
- Note projects like BDEC2 are aiming at new systems and it is not clear how important constraints from continuity to the past should be.
- Almost by definition, any big data computing must involve HPC technologies but not necessarily classic HPC systems.
- The growing demand from Big Data and from the use of ML with simulations (MLAutotuning, MLaroundHPC) implies new demands for new algorithms and new ideas for hardware/software of the supporting cyberinfrastructure CI. This note is a start to address this.
- The AI for science initiative from DoE will certainly need new CI and implicitly it is contained in the discussion below.
- We will call the systems HPC Clouds, which are linked to HPC Edge computing as seen in hardware like the Google Edge TPU. Both the Cloud and Edge systems can be considered Intelligent

Application Requirements for NG Cyberinfrastructure

Four clear classes of applications are

a) **Classic simulations** which are addressed excellently by DoE exascale program and although our focus is Big Data, one should consider this application area as we need to integrate simulations with data analytics and ML (Machine Learning) -- item d) below
b) **Classic Big Data Analytics** as in the analysis of LHC data, SKA, Light sources, health, and environmental data.
c) **Cloud-Edge** applications that certainly overlap with b) as data in many fields come from the edge. However, streaming and Edge applications have special requirements.
d) **Integration of ML and Data analytics with simulations**. Under the rubric “learning everywhere”, this has been discussed in detail by Fox and Jha.

Remarks on Deep Learning in NG Cyberinfrastructure

- We expect growing use of deep learning (DL) replacing older machine learning methods and DL will appear in many different forms such as Perceptrons, Convolutional NN's,

Recurrent NN's, Graph Representational NN's, Autoencoders, Variational Autoencoder, Transformers, Generative Adversarial Networks, and Deep Reinforcement Learning.
- For industry, growth in reinforcement learning is increasing the computational requirements of systems. However, it is hard to predict the computational complexity, parallelizability, and algorithm structure for DL even just 3 years out.
- Note we always have the training and inference phases for DL and these have very different system needs. Training will give large often parallel jobs; Inference will need a lot of small tasks.
- Note in parallel DL, one MUST change both batch size and # training epochs as one scales to larger systems (in fixed problem size case) and this is implicit in MLPerf results
- For DL as well as other forms of machine learning, there are important special considerations for Edge and Streaming use cases.

Compute and Data Patterns

The application classes listed in “Application Requirements for NG Cyberinfrastructure” will all require aspects of the four capabilities below.

1) Parallel simulations;
2) Data Management;
3) Data analytics;
4) Streaming

but the proportion of these will differ in the four classes and application class d) (“Integration of ML and Data analytics with simulations”) is particularly hard as it intermixes all of them in a dynamic heterogeneous fashion.

- A system for class d) will be able to run the other classes as it will have “everything” but a “d” system may not be the most cost-effective for a pure class a) or b) application.
- The growing complexity of work implies that dynamic heterogeneity will characterize all major problems of each class.
- In the absence of consensus application requirements and software systems to support ML-driven HPC, it is unclear whether scaled up machines should have the same design but larger sizing or have a radically different design and performance point.

Cloud-Edge computing

Here the fog concept suggests that the Cloud model extends to compute resources near the edge as in AWS Greengrass but doesn’t affect the “central” HPC Cloud much except to insist that cloud support streaming systems such as the Apache Storm/Kafka model. This does not seem difficult to support as a modest upgrade to any design aimed at classes a), b), d). Real-time machine learning is important and discussed later.

Common Practices underlying HPC Cloud

Low level structure -- Industry important leader

- Docker/Openshift/Singularity containers

- Kubernetes container orchestration/management with Slurm supporting high performance
- Build everything as services OpenAPI REST standard
- DevOps
- Possibly Bare metal or hypervisor-based nodes (but always containers)
- Use MLAutotuning or Dean's Systems for ML

Middleware

- Service Mesh to link services e.g.Openshift offers Docker Kubernetes Service mesh
- HDFS based on node storage with the growing use of high-speed storage (NVMe, Intel Optane)
- Variety of databases as a service
- Data Management: Hadoop, Spark, Flink. Twister2 offers a high-performance version

Edge-Cloud Link

- Storm/Heron-Kafka style streaming -- dataflow compute engine plus buffered edge messages. Twister2 offers a High-performance version of Storm API.
- Fog (cloudlets) implies environment should run on on small systems near the edge. See AWS Greengrass. Industry must solve this anyway
- Real-time machine learning is an important part of many edge problems and here one either needs just real-time inference but sometimes real-time inference and training. Supporting real-time ML is more dependent on the algorithm used and edge systems than on the HPC Cloud.

Languages

- Different Languages inevitable and Industry needs to alleviate as not specific to science
- Python as either frontend (~universal) or main tool (several important libraries); MDAnalysis and Keras as examples
- Jupyter notebooks and environments like Google Colab built on them are growing in importance,
- C or C++ Best performance
- Java -- most data management

Simulation

- Parallelism is very important in application class a)
- Simulation has important issues where there is typically agreement among the experts on the issues, if not the solutions. Obviously these issues underlie the exascale initiatives, which will produce great hardware and software
- MPI remains important
- Science is probably lead stakeholder in

Data analytics

- Parallelism is of growing importance in data analytics especially in deep learning

- Data analytics is of great importance to industry and of course, science is smaller than industry but data analytics is of central importance to science
- Keras, Tensorflow, MXNET, PyTorch are key tools for deep learning
- Graph databases and graph analytics key in some important areas and are perhaps the hardest parallel algorithms
- Decreasing but still nontrivial interest in general Machine learning such as Scikit-Learn
- Note the different communication needs of different classes. DL is dominantly AllReduce and Broadcast. Graph analytics is Scatter and Gather. The local communication structure of most simulations is not seen in data analytics. There is a lot of study of this but could affect optimal interconnect and messaging software.
- See interesting MLPerf benchmarks
- See results from SPIDAL project http://dsc.soic.indiana.edu/publications/FormattedSPIDALPaperJune2019.pdf for a survey of HPCforML
- Real-time and online algorithms and systems are critical in some applications such as speech recognition on edge, autonomous driving, etc.
- Link to simulation with MLaroundHPC integrating ML with the simulation.
- Link to simulation with MLafterHPC; data analytics processing results of a simulation.

Scheduling

- In Slurm, the unit is a node but in Mesos or Kubernetes it is a container. This needs to be resolved. Large scale data analytics will need to be allocated at the node level as for example parallel deep learning will use classic synchronizing algorithms.
- HPC tends to emphasize single job performance whereas industry focus is throughput i.e.total system performance
- Need Resource management for dynamic heterogeneity,
- Rutgers Radical Pilot addresses some of these issues

Not necessary but could be important for HPC Cloud -- need to watch Industry

Here are four trends that are important in today's cloud computing but may not have huge impact on BDEC's next-generation cyberinfrastructure

- Serverless
- Microservices
- Event-based computing
- Cloud-native for fault tolerance and microservices

High-performance System Principles

- Accelerators that have commonality across simulation and ML, allow one to straightforwardly deploy a system good at all four application classes. This could be true for GPUs and TPU's as accelerators but the more exotic ANN accelerators may not be very useful in simulations. In fact, some accelerators may not be good at all variants of DL/ML. For example, in some cases, RNN and CNN behave differently in performance studies

- High speed interconnect will always be useful
- Always minimize data transfer; bring compute to data

Workflow

- There is little overlap between approaches to orchestration and workflow in simulation and big data fields. This is perhaps historical as the requirements are not very different. Probably one could support either the HPC or the Big Data approaches to workflow on most system designs.
- In more detail, Big data systems like Spark or Flink support a dataflow computing model that is at a finer grain size than that supported by HPC systems such as Pegasus or Kepler. Developments such as Function as a Service and event-driven computing also suggest a finer grain size (microservices) computing model.
- NiFi is a big data workflow system that is similar to HPC approaches.

References

The material in Section IV of “Understanding ML-driven HPC: Applications and Infrastructure” https://arxiv.org/abs/1909.02363. Addresses the issues described above.

Acknowledgments

Partial support by NSF CIF21 DIBBS 1443054, NSF nanoBIO 1720625, and NSF BDEC2 1849625 is gratefully acknowledged.