Potential Ideas for an El Capitan Center of Excellence (COE) Around Intelligent Simulation

Proposing a Center of Excellence under the CORAL-2 NRE is a mandatory requirement. The current *Sierra COE* (CORAL-1) is largely focused on helping LLNL applications make the disruptive transition to a heterogeneous GPU-based system by 2018. We expect a continuation of COE activities around optimizing our broad and diverse application base (of so-called "traditional" simulation codes) optimized for the El Capitan architecture, as well as supporting the underlying software stack (compilers, tools, programming models, etc.) – but do not expect this to require as much effort in the *El Capitan COE* (assuming a heterogeneous node architecture).

What follows are some thoughts on 3 potential topics of interest for an El Capitan COE at LLNL – largely focused around AI and machine learning and the concept of advancing our goal of *intelligent simulation* or *cognitive computing* in the timeframe of deployment and production use of El Capitan (2023-2038). We believe vendor engagement through a COE in this area will provide a natural point-of-interest between LLNL and our vendor partner for common advancement of machine learning capabilities focused on scientific data – potentially greatly broadening the ecosystem around HPC architectures and the supporting software stack for scientific simulation-based AI.

Use of AI and Machine Learning to Amplify Traditional HPC Applications and Workflows

The ASC Program at LLNL has recently stood up *Cognitive Computing Initiative*, which aims to coordinate and accelerate research in the area of using machine learning as a means for making our existing "traditional" HPC simulation codes more intelligent, more productive, and more robust. What follows are 3 general areas we would like to advance the state-of-the-art over the next decade in a collaborative effort with the COE:

1) **Using AI and machine learning to improve physical models by mimicking sub-grid physics or providing in-situ tuning of an algorithm**. While a major driver of exascale computing is to enable the ability for higher-fidelity physics models to be incorporated into our continuum codes, the other axis of predictive science is the ability to run large ensembles of 3D calculations which will require faster turnaround time than even an exascale system can provide if those models are to be used in day-to-day workloads. One idea is to use high-fidelity calculations as training data for models at the continuum – building on our multi-scale modeling approach by helping to automate the process of bringing sub-grid physics phenomena to extreme scale calculations. In general, these types of applications will need to be accessed on a cycle-by-cycle basis and are local in nature. Thus, quick turnaround and the ability to reside in fast local memory are important. Examples include:

    - Replacing DCA Non-local thermodynamic equilibrium in line calculations (NLTE) of opacities and EOS (i.e. Cretin) with ML algorithms in our multi-physics apps
    - Adding microscopic information, such as fracture mechanics at the grain scale, with ML techniques to inform constitutive models used in our multi-physics apps

- Supplementing heuristic models, such as Reynolds Averaged Navier Stokes (RANS) turbulence models with additional terms provided by ML techniques that improve accuracy based on instantaneous conditions

2) **Providing in-situ performance or robustness tuning, performance optimization, or analysis of an application**. As our codes become more complex, the expertise required to effectively run these applications puts a larger burden on the expertise of the user to make intelligent choices of user inputs and heuristics built into the code. The goal is to enable the codes with the ability to make more intelligent decisions without user-intervention. In general, these use cases are embedded in the application, are non-local in nature, but do not need to be called every cycle. Examples include:

   - ALE mesh management where ML techniques are used to anticipate mesh tangling and relax the mesh before such tangling occurs
   - Load balancing among processors where ML techniques go beyond simple metrics of work load to determine load balancing and include things like expected integer vs floating point work load, thermal conditions of the processor, locality of communication, etc.
   - Detection of features that could be of interest to human analysts, thus triggering a data dump of some kind for future analysis.
   - Use of machine learning to select execution policies for our numerically intensive kernels, both at compile time and dynamically during runtime

$$\{\sigma_1^{n+1}, \sigma_2^{n+1}, \sigma_3^{n+1}\} = f(\sigma_1^n, \sigma_2^n, \sigma_3^n, L_{11}\delta t, L_{22}\delta t, L_{33}\delta t)$$

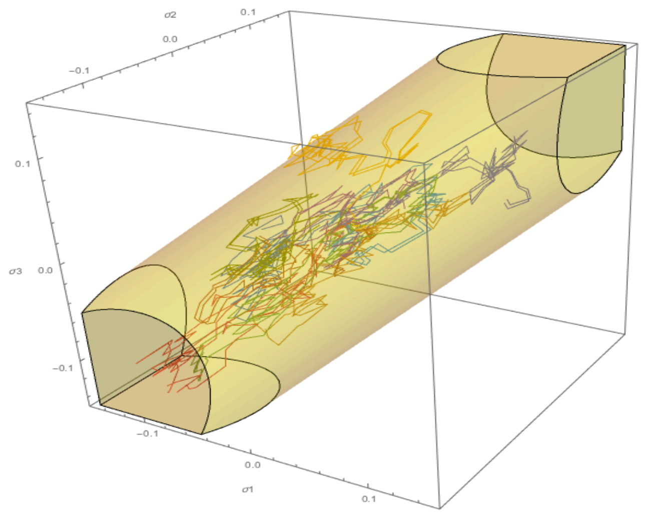


Figure 1 Using a Von Mises Plasticity model, this figure shows results in a phase space for analytically calculated data overlaid with data calculated from a trained data set. Differences of 2-3% seen here may fall into acceptable uncertainty bounds for other more complex constitutive models, but significant research is required

3) **Post processing simulation results or provide some kind of analysis of a large data set**. The "big data" issue in traditional HPC lies largely in our ability to take the output generated by our simulations and turn reams of data into actionable information that can in turn inform the decision-makers what the simulations are telling them. Traditionally, it has been the role of visualization to provide the user with insight – but as data sets become even larger and the ability to consistently monitor the state of an application in real-time is driving us both toward the need for in-situ analysis

(alongside visualization) that can inform the user of “interesting” features emerging in the calculation, as well as in-depth post-processing of raw data and image analysis of the reams of data being generated – not all of which can be saved or archived, and thus must be analyzed with expedience. In general, these use cases do not require tight coupling of the ML engine with an application when post-processing is enabled, offering a quick path to developing the algorithms and techniques required. Over time, those techniques will need to be incorporated into the running code as in-situ analysis, alongside in-situ visualization, code steering, and general workflow optimization for the user. Examples include:

- Use of ML techniques to analyze micrographs of high explosive mixtures to predict performance characteristics.
- Use of ML techniques to build a response function, with probabilities, from a large data set for uncertainty quantification analysis.

Employing accelerated systems to optimize cognitive simulation workflows

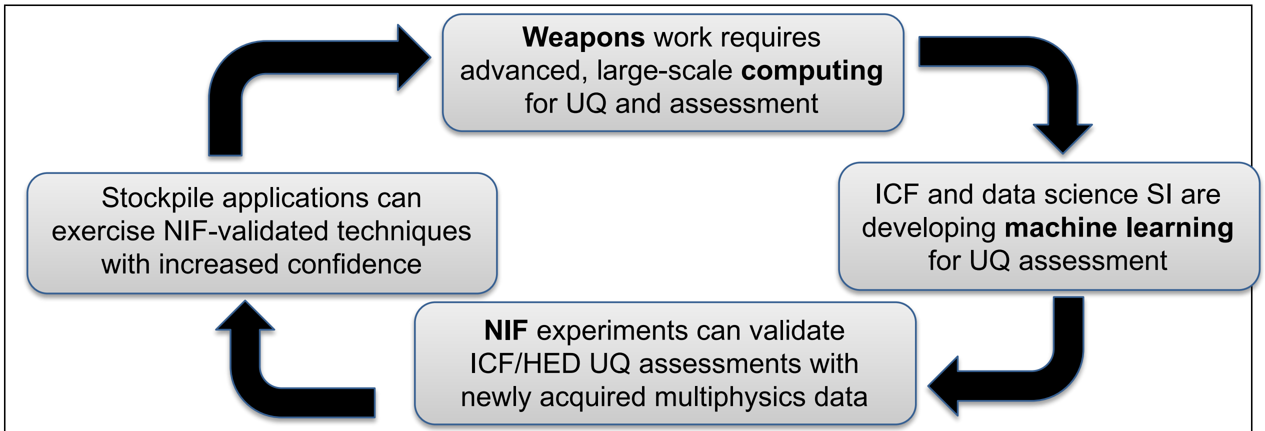
Predictive simulations are constantly challenged and improved by comparison with experimental data. The experimental observations and simulations of those observations have grown data-rich, including collections of images, vector-valued data, and scalars. However, the traditional approaches for comparing simulation and experiment omit much of this data, leaving predictive models less accurate than they could be.

We aim to use modern machine learning techniques to develop improved predictive models. These learned predictive models will begin with deep neural networks that are trained on the entire collection of simulated observations, using ICF implosions as a testbed. The trained network will then be elevated to reflect experimental data trends from the National Ignition Facility (NIF).

A central challenge in this effort is optimizing a complicated, integrated workflow that can:

- asynchronously execute scientific simulation
- continuously evolve enormous simulated training databases
- frequently train multi-modal deep learning models
- quickly perform high-speed inference, both for feedback to intelligent parameter sampling systems and for uncertainty quantification of predictions.

Why is it important for national security?
Stockpile stewardship faces the challenging task of maintaining the nation’s nuclear weapons arsenal through the application of both advanced computing and state-of-the-art experimentation. Machine learning is becoming a key enabling technology that unifies our high fidelity simulations with experimental data, both historical and newly gathered. This deeper, more comprehensive incorporation of experimental data into predictive models can make our stewardship efforts more flexible and responsive to evolving deterrent needs.


Weapons work requires advanced, large-scale **computing** for UQ and assessment
ICF and data science SI are developing **machine learning** for UQ assessment
NIF experiments can validate ICF/HED UQ assessments with newly acquired multiphysics data
Stockpile applications can exercise NIF-validated techniques with increased confidence

Robust machine learning can strengthen the bonds between weapons, advanced computing, and NIF experimentation

What is novel and compelling about your project?
To pursue these transformational stewardship tools, we require computational platforms that are themselves more flexible and differently capable than past HPC machines. In particular, the machine learning operations at the center of improved prediction benefit enormously from the highly parallel and often reduced precision computations made by general-purpose graphics processing units (GPGPUs). Our experimental workflows also feature high-frequency evaluation of learned models to guide iterative online learning and retraining. Such inference loads potentially require hardware dedicated specifically to evaluating trained models. While the potential list of candidate processors -- CPUs, GPUs, inference accelerators -- is clear, much remains to be explored. Principally, we are concerned with five questions:

- Can we identify and mitigate workflow bottlenecks that result from an interacting computational loads across simulation, training, and inference tasks?
- Can we identify and mitigate workflow bottlenecks that result from data motion amongst heterogeneous processors?
- Can we optimize heterogeneous processor performance to accommodate increasingly large data sets?
- Can we produce workflow and computational platform requirements that help inform a template for future cognitive (2023) machines?
- Can we use machine learning to drive the simulation process, using knowledge discovered during simulation to modify growing training data sets?

How do we envision vendor engagement in our existing straegy?
Examples of work items might include:

1. Implement a mini-app that emulates the physics simulation, ML training, and ML inferencing expected in our asynchronous workflows.
2. Test new strategies for optimizing training and inference computational workloads, preferably on the same card with support for existing ML frameworks (TensorFlow, PyTorch).
3. Develop strategies for managing data motion, including movement of both training data and learned models, between GPGPUs and inference accelerators.

4. Develop hardware and software strategies for training on increasingly large data sets while steering the simulation process using machine learning techniques.

Engaging in research & development along these lines has the potential to substantially impact the delivery of intelligent simulation platforms in the 2023 timeframe. This work would directly inform design requirements on the type and number of heterogeneous processors. Furthermore, it would clarify the demands for memory and networking necessary to support asynchronous simulation, learning, and intelligent sampling. The interaction would allow us to investigate, at the prototype level, varieties of schemes and architecture configurations that we could not otherwise explore on static production or large-scale Laboratory platforms.

Developing and implementing a detailed research plan to tackle these issues would require 2-5 vendor partner experts capable of engaging in co-design with a multidisciplinary Laboratory research team. We need vendor colleagues capable of supporting current hardware, modifying hardware configurations, and assisting in implementation of learning frameworks and mini-apps on model systems.

Foundational research in machine learning at extreme scale

Developing the neural networks for cognitive simulations is a computationally demanding task that requires training novel networks on massive, high dimensional, data sets. To support and accelerate these learning efforts we have developed the Livermore Big Artificial Neural Network training toolkit (LBANN) that is optimized for executing multiple levels of parallelism on HPC systems. LBANN composes model-, data- and ensemble-level parallelism to strong and weak scale the training of deep neural networks, as illustrated in Figure 2. LBANN is optimized for tightly-coupled GPU accelerators, node-local NVRAM, and low-latency, high-bandwidth interconnects. The core of LBANN is a GPU-accelerated distributed linear algebra library that allows domain scientist to seamlessly distribute the training and inference of neural networks across multiple HPC nodes. LBANN has implemented custom support for asynchronous communication algorithms that are optimized for deep learning training and novel methods for parallelizing convolutional networks. LBANN has support for state of the art learning algorithms and takes advantage of multiple programming models, including MPI+OpenMP and CUDA, as well as state of the art DNN libraries such as cuDNN and NCCL, plus accelerated BLAS libraries.

Direct vendor engagement for the continued development of LBANN would provide a unique opportunity to continue to improve the training and scalability of deep learning for cognitive simulation algorithms. Opportunities for specific engagement are listed below:

1) Integration of LBANN into Vendor specific deep learning workflow. Specifically, vendors are now providing optimized builds of standard deep learning toolkits that are tightly integrated into a pipeline that support tasks such as: 1) data acquisition, curation, and augmentation, 2) management of trained networks, 3) easy containerized distribution, 4) visualization and job scheduling / management. Providing support for LBANN in an environment such as this would improve the deployability and usability of our HPC-centric toolkit.

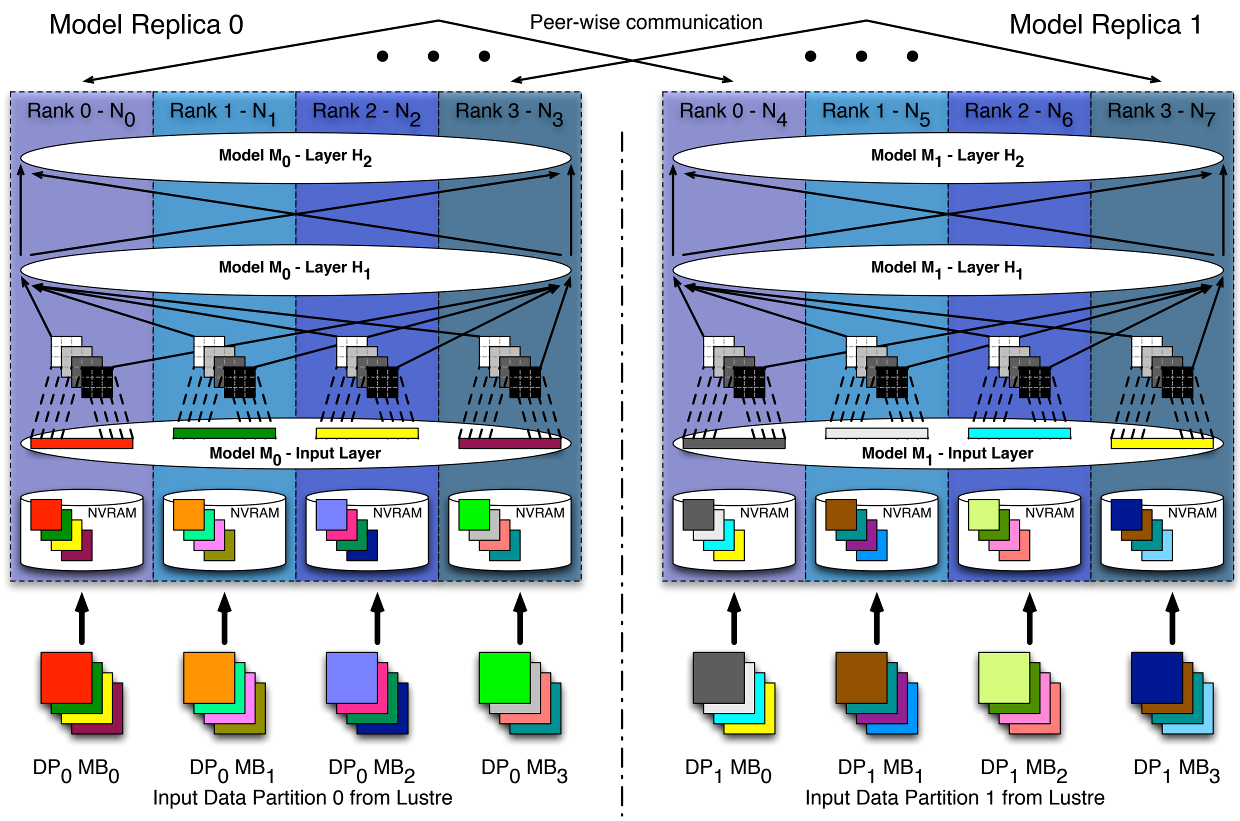


Figure 2: Block diagram of LBANN execution on HPC nodes, illustrating composition of model and data parallelism.

2) Provide optimized DNN library routines for model parallel execution of convolutional subroutines. LBANNs current implementation for parallelizing convolutional layers is limited by the support provided in existing DNN library routines. Providing general parallelization strategies will require a more flexible interface into a DNN library, with certain corner-cased optimized data distributions.

3) Optimizing LBANN for low precision computing (<FP16) with support for densely packed "DNN vector" optimizations. Current roadmaps for LBANN include adding support for FP16 data representation. Direct vendor interaction on developing training methods for FP16 and other smaller data types, as well as DNN library support for such low-precision representation would allow better utilization of emerging hardware architectures. Furthermore, optimization for vendor specific short vector operations within LBANN and associated DNN libraries would be a great opportunity.

4) Optimization of vendor BLAS libraries for low-precision data types and skewed matrix-matrix multiplications. Model parallelism in LBANN of large fully connected layers is implemented by distributed algorithms such as SUMMA with multiple smaller GEMMs per node. Frequently these smaller GEMMs can have non-standard aspect ratios due to the imbalance between number of neurons per layer and the mini-batch size. Additionally, current efforts for low-precision computing support in vendor libraries is optimized for DNN-specific libraries, however,

our approach heavily utilizes standard BLAS libraries and thus would take advantage of low-precision BLAS routines.

References:

Van Essen, B., Kim, H., Pearce, R., Boakye, K., & Chen, B. (2015). LBANN: Livermore Big Artificial Neural Network HPC toolkit (pp. 1–6). Presented at the Workshop on Machine Learning in High Performance Computing Environments, New York, New York, USA: ACM Press. http://doi.org/10.1145/2834892.2834897

Dryden, N., Jacobs, S. A., Moon, T., & Van Essen, B. (2016). Communication quantization for data-parallel training of deep neural networks (pp. 1–8). Presented at the Workshop on Machine Learning in High Performance Computing Environments, IEEE Press. http://doi.org/10.1109/MLHPC.2016.4

Auspices and Disclaimer