ARGONNE
ATPESC2025
EXTREME-SCALE COMPUTING

Workflow Management Tools to Couple Simulation and AI

Christine Simpson
(Riccardo Balin)
Data Services & Workflows, ALCF

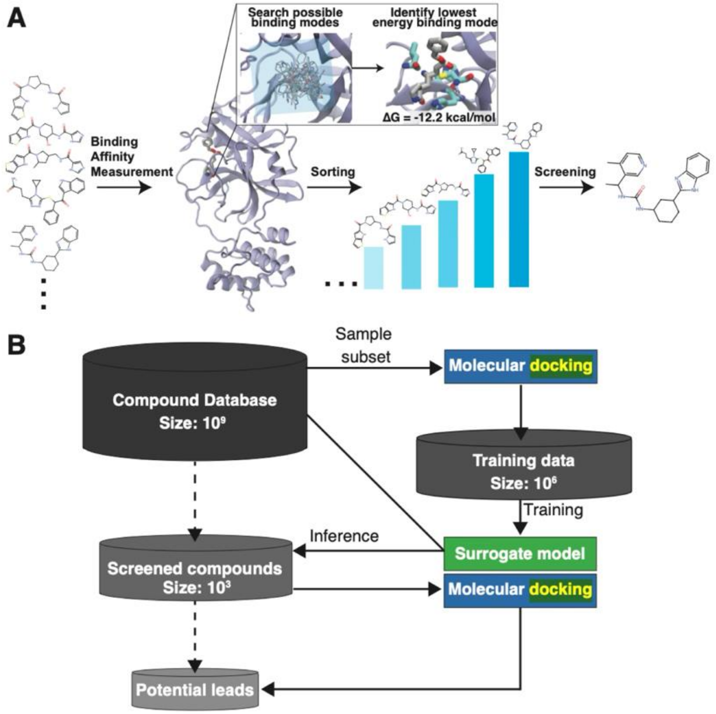
Why Couple Simulations to AI?

Substitute inaccurate or expensive components of simulation with ML models
e.g. closure or surrogate modeling

Control simulations with ML
e.g. select numerical scheme or input parameters, e.g. AI-in-the-loop

Avoid IO bottlenecks and disk storage issues

Active learning
e.g. continuous improvement of ML model training as simulation progresses

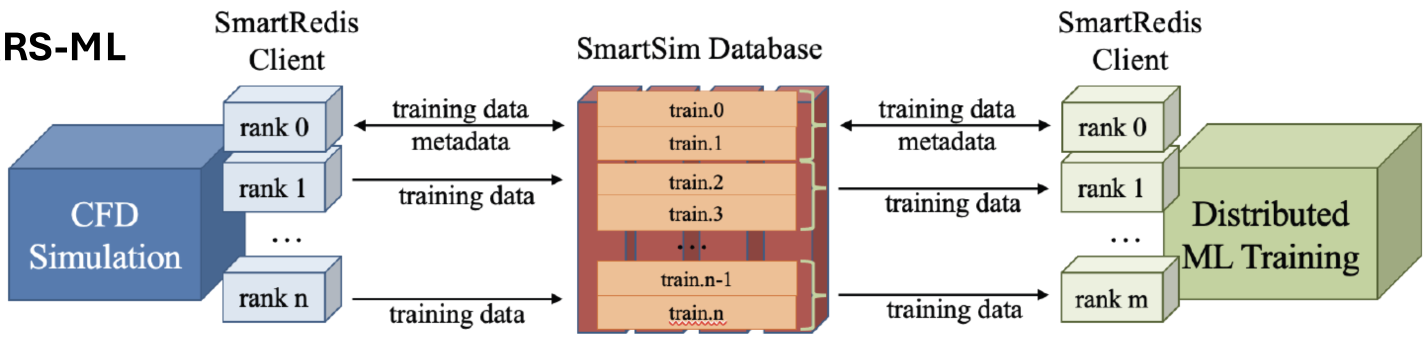


Vasan et al. 2024 IEEE International Parallel and Distributed Processing Symposium Workshops

Online vs. Offline Training

- Increasingly common for research groups to use simulated datasets for model training
- Sometimes groups tackle this in two stages run separately by human researchers, e.g. ***Offline Training***
- However, more groups are now using workflow patterns where simulation and training (and/or inference) are coupled to the simulation during runtime, e.g. ***Online Training***
- ***Today, we'll focus on Online Training or Online Inference workloads where AI/ML are coupled with Simulations in an automated fashion***

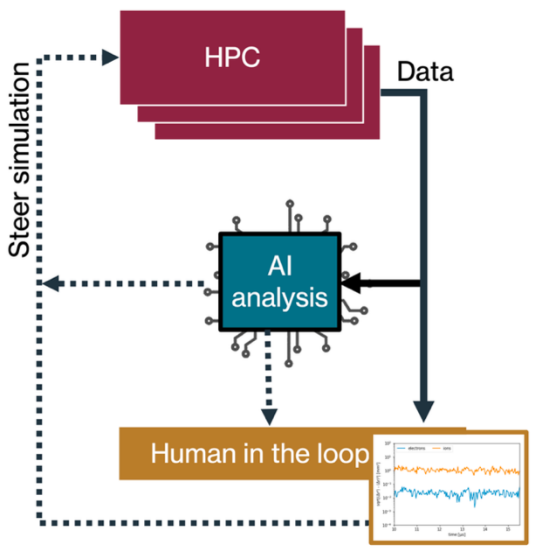
NekRS-ML

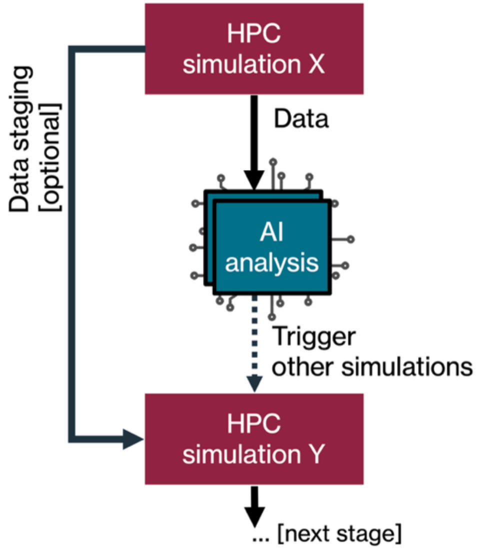


Balin et al., arXiv:2306.12900, 2023.

Patterns of AI-Simulation Coupled Workflows

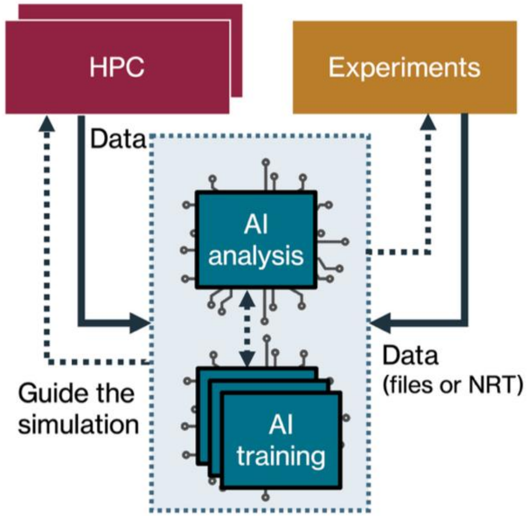
Steering workflow

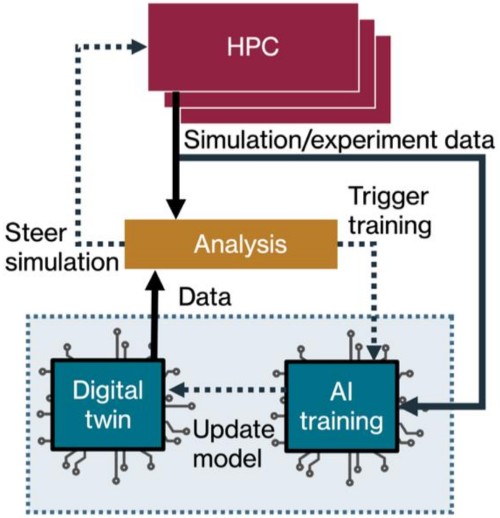




Sequential workflow

Inverse problem workflow





Digital twin workflow

Brewer, Wes et al. "AI-coupled HPC workflow applications, middleware and performance." *arXiv:2406.14315* (2024)

Elements of an AI-Simulation Coupled Workflow

Task Launching

How to launch processes running different applications? How are they coupled?

Do your applications need to be launched with MPI?

Data Sharing

How to share data between components?

Process placement

Are the Simulation and AI components run by the same processes? or different ones?

Do Simulation and AI share nodes or are they placed on different nodes?

Elasticity

Do you need to extend your workflow over many PBS/Slurm jobs? How to automate this?

Multi-machine

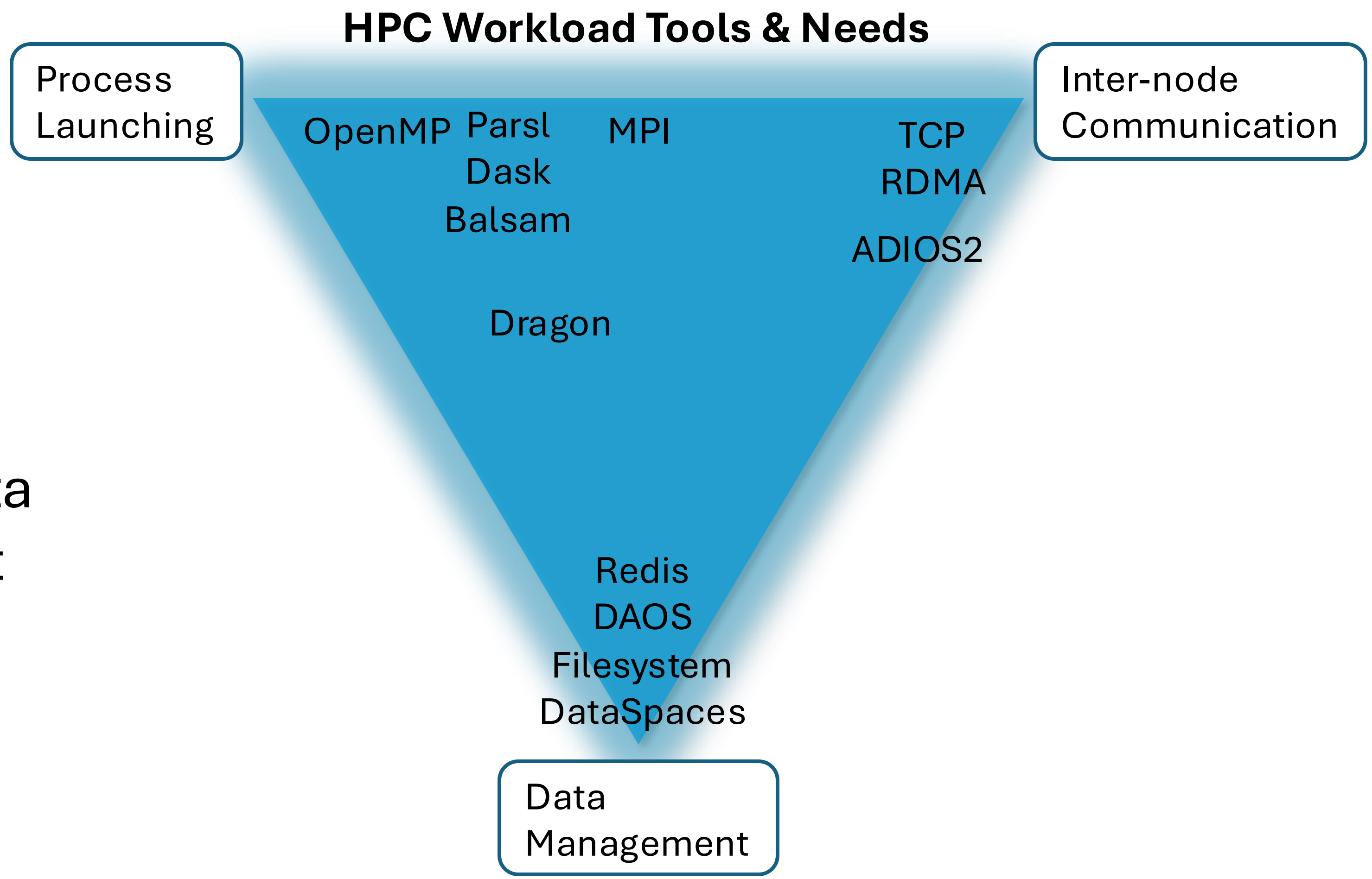
Do you need different resources for different components of the workflow?

- E.g. Simulations on Aurora and LLM inference on Sambanova

How to couple cross machine workflows?

Processes, Data & Communication

- Traditional ModSim (and stand-alone AI) applications often lie along the Process Launching – Communication axis in their functionality and needs
  - e.g. CFD, Cosmology, MD
- With the introduction of AI/ML to the picture, a third dimension, Data Management, becomes important
- Data often needs to be available for longer and in larger quantities to feed online training and inference



Tight vs. External coupling

There exist many patterns with specific nuances, but we will consider two broad cases for this talk

Tight Coupling: a kernel in the simulation has been replaced by an AI-surrogate. Inference results from the surrogate couple tightly to traditional parts of the simulation code; periodic retraining may also occur

External Coupling: AI components interact with the outputs or results of simulations. Can be used to create a surrogate for the simulation/system (e.g. digital twin) or for steering purposes (e.g. AI-in-the-loop or hyperparameter searches)

Software for Coupling Simulations and AI/ML

- Tight coupling
  - Python and ML frameworks embedding into simulation code
    - PythonFOAM, TensorFlowFOAM and HONEE (by Romit Maulik, Saumil Patel, Bethany Lusch at ALCF)
  - Linking to LibTorch or ONNX Runtime libraries for ML inferencing from C, C++ and Fortran
    - Aurora will support LibTorch and Intel's OpenVINO inference library
- External coupling
  - Parsl
    - Workflow tool for distributed, parallel task execution
  - SmartSim / SmartRedis
    - Workflow manager and client libraries for in-situ workflows by sharing data across a database
  - ADIOS2
    - Same I/O API to transport data across different media (file, wide-area-network, in-memory staging, etc.), favoring asynchronous streaming
  - Dragon
    - Run-time library for managing dynamic processes, memory, and data at scale through high-performance communication

Using AI/ML Frameworks in C++ for Tight Coupling (libTorch)

- Tight coupling requires integration of ML code within simulation code, however most common languages for traditional simulation codes are C/C++ and FORTRAN and not python
- libTorch is one example of a package that bridges this gap for Torch and C++
- libTorch has most of the functionality of Torch's python API, pytorch
- libTorch is included in the Auora frameworks module, details on compiling and linking with libTorch libraries are in the Aurora documentation

```cpp
#include <torch/torch.h>
#include <torch/script.h>

int main(int argc, const char* argv[]) {
  torch::jit::script::Module model;

  model = torch::jit::load(argv[1]);
  std::cout << "Loaded the model\n";

  model.to(torch::Device(torch::kXPU));
  std::cout << "Model offloaded to GPU\n\n";

  auto options = torch::TensorOptions()
                 .dtype(torch::kFloat32)
                 .device(torch::kXPU);
  torch::Tensor input_tensor = torch::rand({1,3,224,224}, options);
  assert(input_tensor.dtype() == torch::kFloat32);
  assert(input_tensor.device().type() == torch::kXPU);
  std::cout << "Created the input tensor on GPU\n";

  torch::Tensor output = model.forward({input_tensor}).toTensor();
  std::cout << "Performed inference\n\n";

  std::cout << "Slice of predicted tensor is : \n";
  std::cout << output.slice(/*dim=*/1, /*start=*/0, /*end=*/10) << '\n';

  return 0;
}
```

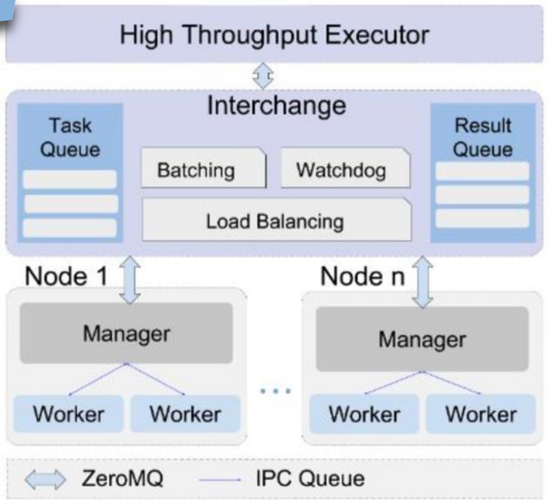
Process Launching for External Coupling

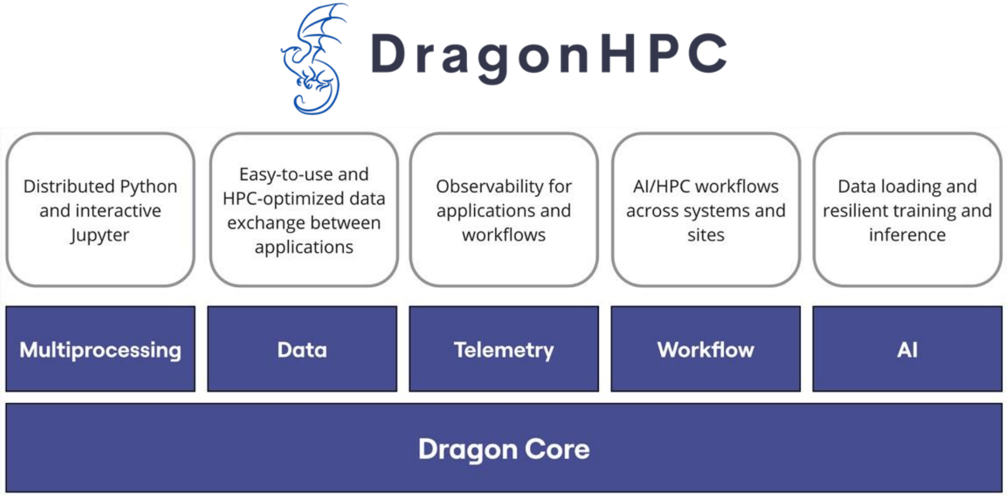
- **mpiexec** can be used to launch application on specific nodes with the **--hosts** or **--hostlist** options, however, can't refill hardware with new tasks once applications complete

```
mpiexec -n 24 —ppn 12 --hosts $HOST_NAME1 $HOST_NAME2  ./hello_affinity &
```

- Workflow tools (parsl, dragon, dask, balsam, etc.) can be used to pin applications to specific hardware but also refill hardware when tasks complete and manage task dependencies

Parsl HTEX Interchange





Parsl:* *a parallel programming library for Python*

- Simple installation with pip
- Workflow contained within memory
- Can orchestrate work from login node and submit jobs to PBS/Slurm or can orchestrate within jobs
- Configuration (assignment of tasks to hardware) set by user, separate from workflow logic and application definitions
- Apps define how to run tasks
  - Python apps call python functions
  - Bash apps call external application
- Apps return futures: a proxy for a result that might not yet be available
- Apps run concurrently, respecting dependencies
- Community of 70+ developers, several at UChicago & ANL, part of Globus Labs

```
@python_app
def hello ():
    return 'Hello World!'

print(hello().result())
```

```
Hello World!
```

```
@bash_app
def echo_hello(stdout='echo-hello.stdout'):
    return 'echo "Hello World!"'

echo_hello().result()

with open('echo-hello.stdout', 'r') as f:
     print(f.read())
```

```
Hello World!
```

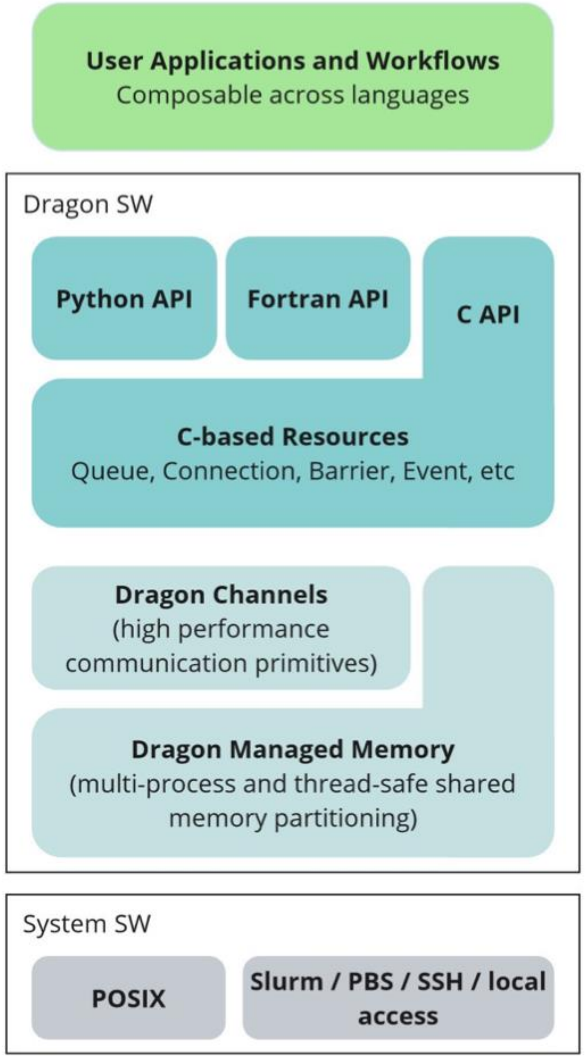
*[demo materials link](#)

Dragon*

- Composable distributed run-time for managing processes, memory, and data at scale through high-performance communication objects
- Open source project developed by HPE
- Key features:
    - Multi-node extension to Python multiprocessing (mp.Process, mp.Pool, ...)
    - C API included, Fortran API in development
    - Managed memory through sharded dictionary objects
    - Parallel process launching, including PMI enabled for MPI applications with fine-grained control of CPU/GPU affinity
    - PMIX support in development (for use on Aurora)
    - High-speed RDMA transport agents for off-node communication on Slingshot and Infiniband networks (TCP for other networks)
    - Interfaces for higher-level workflow tools, e.g. SmartSim & Parsl

DragonHPC

Hewlett Packard Enterprise



*demo materials link

External Coupling Example: AI/ML-in-the-loop with Parsl*

Science Problem: identify high value molecules (i.e. molecules with high ionization energy) among a search space of billions of candidates

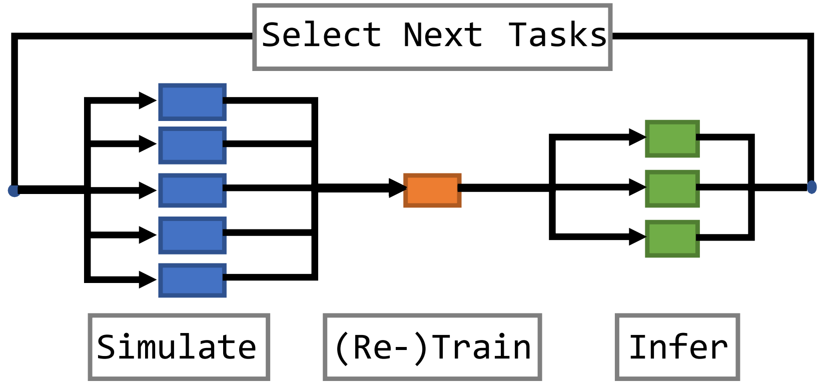
Challenge: The simulation is too computationally expensive to run for every candidate molecule

Approach: Create an active learning loop that couples simulation with machine learning to simulate only high value candidates

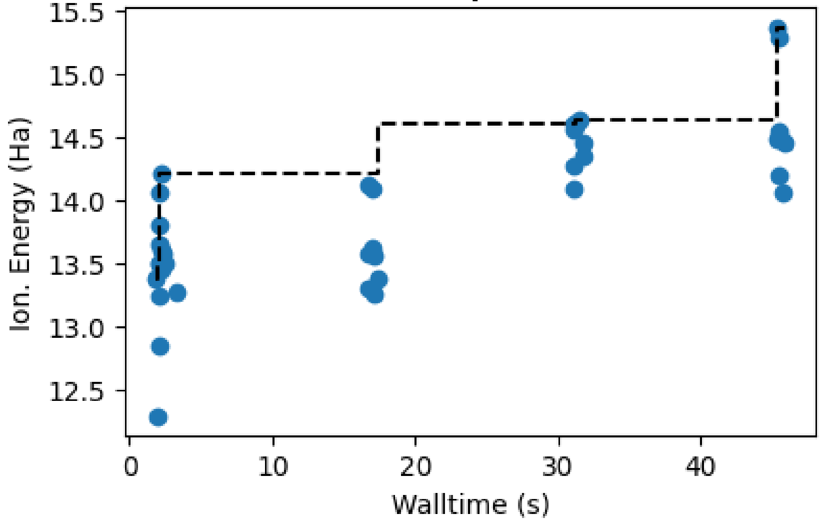
Tools:

- **Parsl** is used for task launching and integration
- Use RDkit and scikit-learn to train a k-nearest neighbor (knn) model
- Simulations done with MD package xTB

Workflow Pattern: AI/ML Components steer Simulations



Simulation Results produced over time



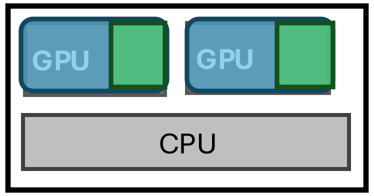
[* Demo materials link]

Distributing Processes in Space and Time

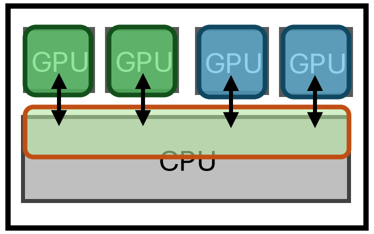
Execution Management

- **Time division (tight coupling)**
  - Components run on same compute resources (may even use same processes)
  - Staggered in time, execution of one component halts the other
  - May allow for direct memory access and no data copy/transfer
  - Idle time of individual components may be significant
- **Space division (external coupling)**
  - Components run on separate compute resources
  - Concurrent in time, both components run simultaneously
  - Minimal idle time of components for fast data copy/transfer
  - Usually requires indirect memory access with data copy/transfer

Time Division: Same Compute Resource



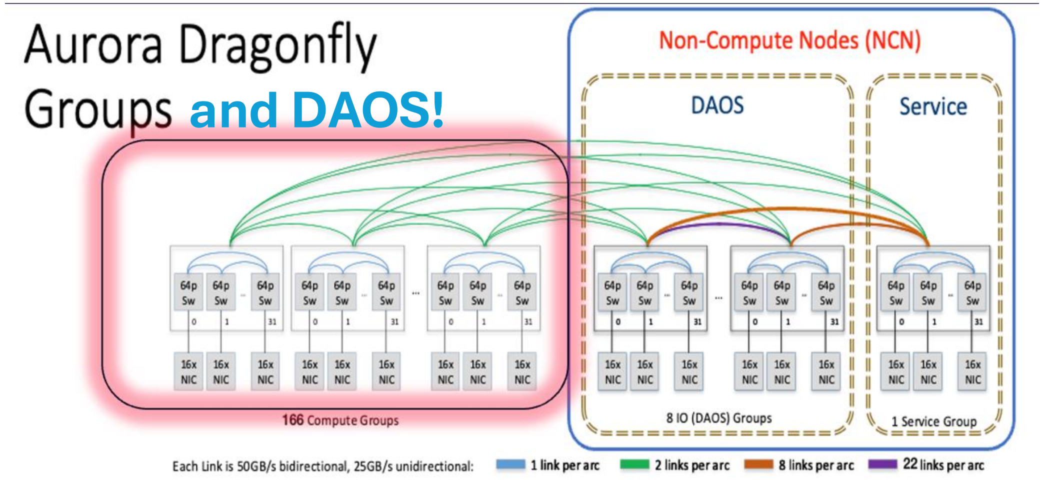
Space Division: Same Node



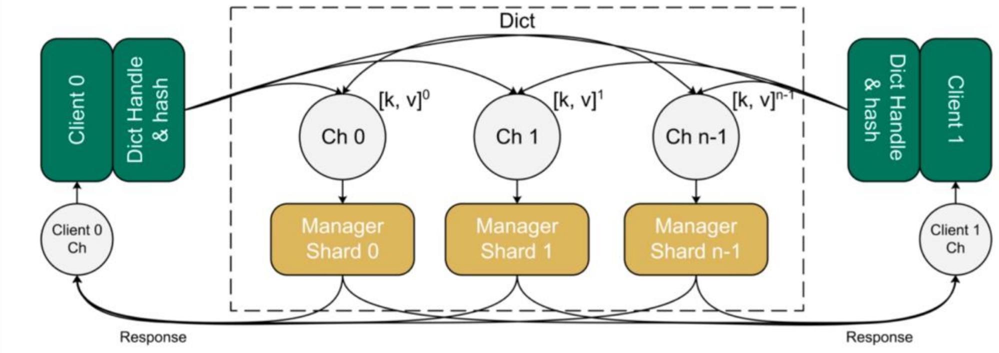
Childs et al., “A terminology for in situ visualization and analysis systems”, Intl. Journal of High Performance Computing Applications, 2020

Tools for Data Sharing

- Tools for sharing data between Simulation and AI/ML components:
  - Parallel Filesystem (e.g. Lustre, on Aurora >650 GB/s bandwidth)
  - DAOS (object datastore, bandwidth >25 TB/s on Aurora)
  - Dragon Dictionary
  - Redis
  - ADIOS2
  - Node local I/O, i.e. read and write directly to DRAM (e.g. /tmp)
- Typically, the least performant of these approaches will be the parallel filesystem, and the most performant the node local I/O
- Node local I/O for both reads and writes may be particularly useful for "co-located" patterns where AI/ML and Simulation processes share the same nodes (but split on-node GPU/CPU resources)
- Intermediate solutions for non co-located (clustered) workflows may use data staging layers in memory like Redis or Dragon Dictionaries
- ADIOS2 is a tool for data streaming between nodes, but does not provide a persistent data staging layer

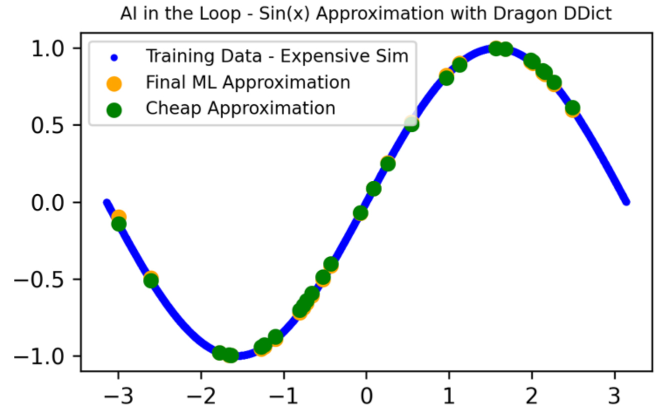


Dragon Dictionary Architecture

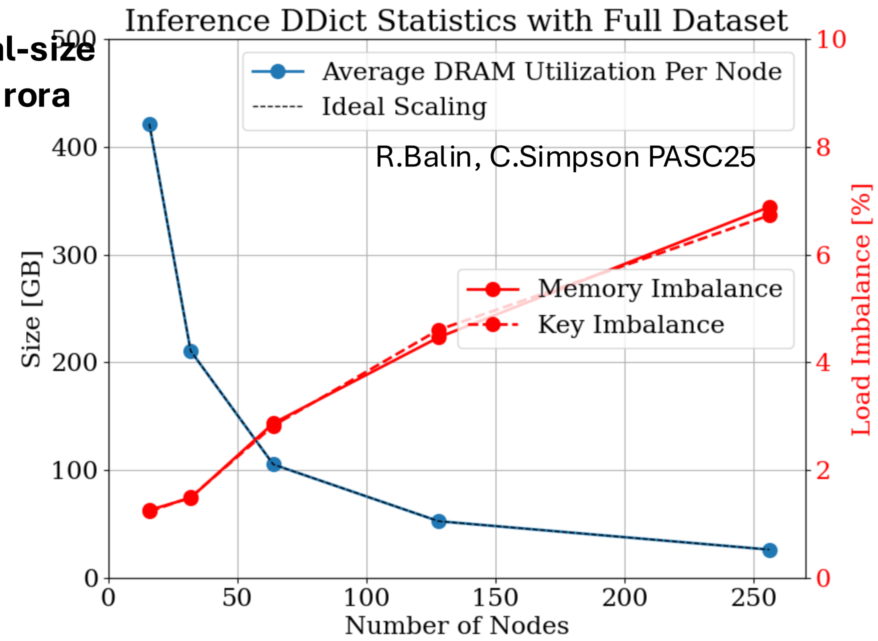


Dragon Dictionaries*

- This demo is another AI-in-the-loop case, this time using Dragon Dictionaries as a data layer between components
- It trains a model to predict the value of sin(x)
- Dragon Dictionaries take in data from client processes in the form of key-value pairs
- Data are sharded across nodes through channels by Memory Pool managers that sit on each node
- Dragon Dictionary Managers dynamically load balance key-value pairs across managers
- Transfers are done with RDMA (slingshot networks) or TCP (non-slingshot networks)



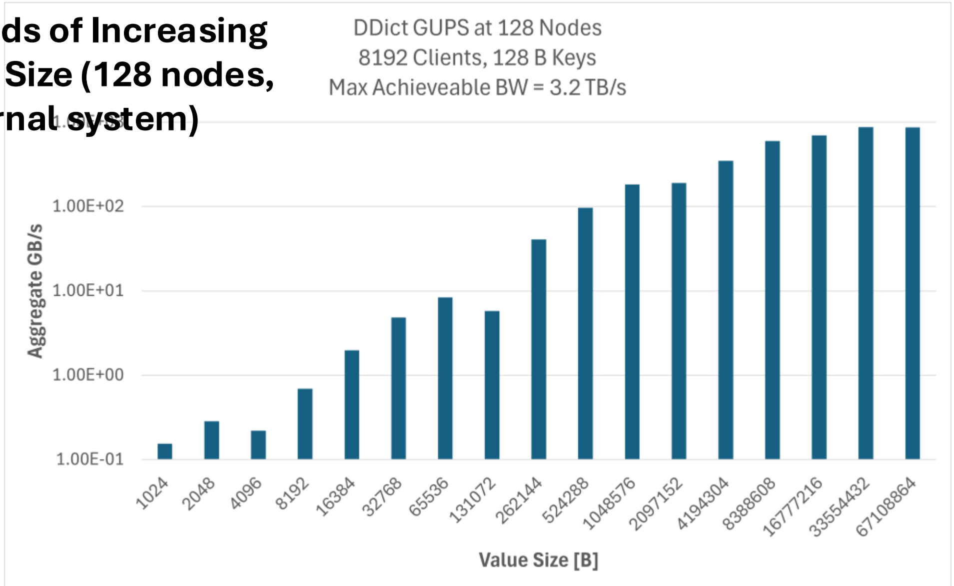
Load balance of equal-size key-value pairs on Aurora



*demo link

ATPESC20

Aggregate Bandwith of Data Reads of Increasing Message Size (128 nodes, HPE internal system)



Summary

Many reasons to couple AI/ML to Simulations

Utilize fast AI surrogates, Efficiently steer simulation ensembles, Avoid I/O Bottlenecks, Active Learning

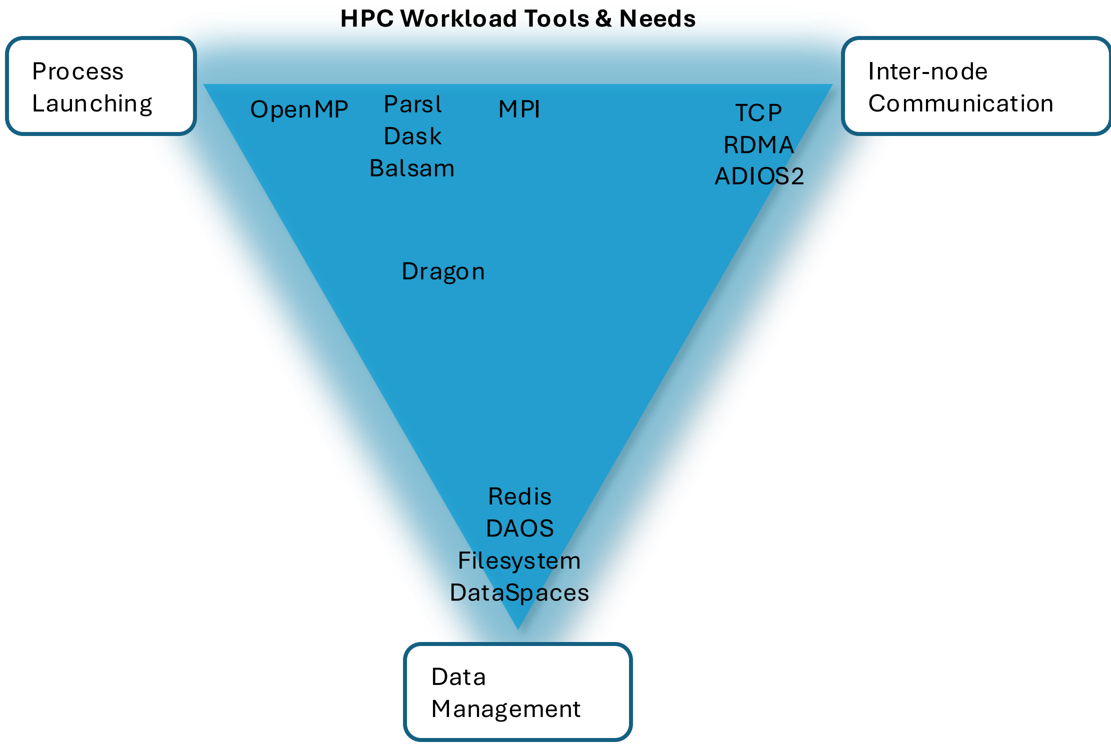
Two broad types of coupling: **tight coupling** and **external coupling**

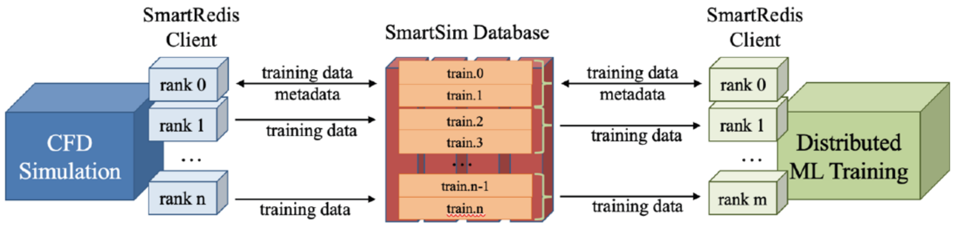
Tight coupling often needs a package to embed ML code in simulation code (e.g. PythonFOAM, libTorch, OpenVINO)

External coupling often needs a workflow tool like Parsl, Dragon or SmartSim

Distributing processes across time and compute resources involves trade-offs and may be different for each workload

AI/ML-Simulation coupled workloads involve a balance of process launching, communication, and data management that is different from how these application run in a stand-alone way





nekRS-ML, R. Balin

ARGONNE
ATPESC2025
EXTREME-SCALE COMPUTING

ARGONNE TRAINING PROGRAM ON EXTREME-SCALE COMPUTING

Produced by Argonne National Laboratory, a U.S. Department of Energy Laboratory managed by UChicagoArgonne, LLC under contract DE-AC02-06CH11357.

Special thanks to the National Energy Research Scientific Computing Center (NERSC) and Oak Ridge Leadership Computing Facility (OLCF) for the use of their resources during the training event.