Application-Driven Requirements for Node Resource Management in Next-Generation Systems

Edgar A León
Balazs Gerofi
Julien Jaeger
Guillaume Mercier
Rolf Riesen
Masamichi Takagi
Brice Goglin

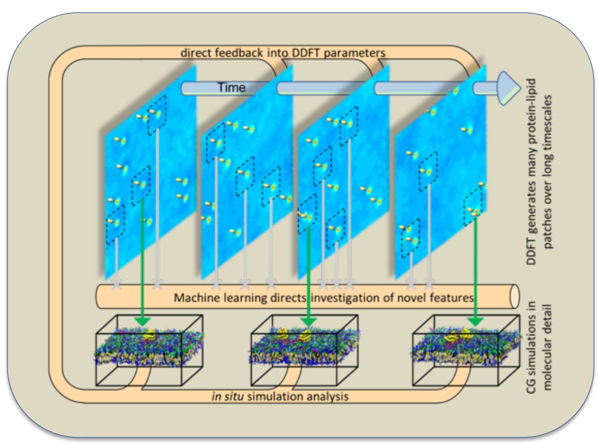
13 November 2020

LLNL-PRES-816236
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Increasingly complex workflows are pushing the limits of HPC software environments

- Multi-physics simulations
  — National security applications
- Data science workloads
  — TensorFlow
  — PyTorch
  — LBANN
- Multi-kernel applications
  — Weather prediction models
- Cognitive simulations
  — Conventional HPC + Deep-Learning



A Massively Parallel Infrastructure for Adaptive Multiscale Simulations.
Di Natale et al., SC 2019

Heterogeneous supercomputing systems pose challenges to application and library developers

- Users must utilize a combination of programming models and runtimes
- Components assume dedicated resources
  - Coordination is left to application and library developers

Heterogeneous Systems
Accelerators
Latency opt processors
Heterogeneous memory

Growing Gap

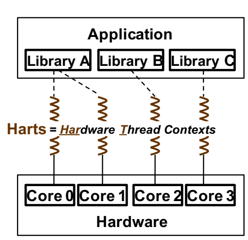
MPI
OpenMP, Pthreads
US DOE RAJA, Kokkos
CUDA, HIP, OpenMP 4+

Application Needs
Portability
Productivity
Performance

Poor or no coordination of resources among node-local components

While not new, this problem is exacerbated by workflow complexity and system complexity

- Promising research in this area
  - Lithe, QUO
  - Argo, Hobbes, multi-kernels
  - Resource and workflow managers



H. Pan's PhD Dissertation, MIT 2010

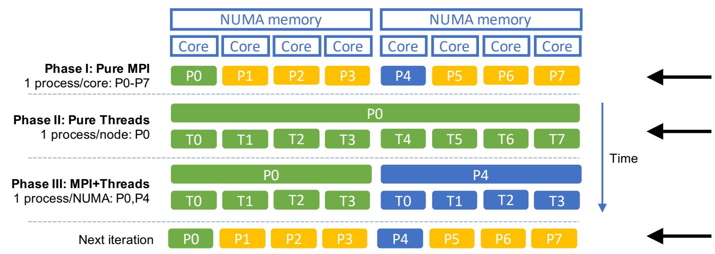
- Yet, problem is getting worse

Our work:

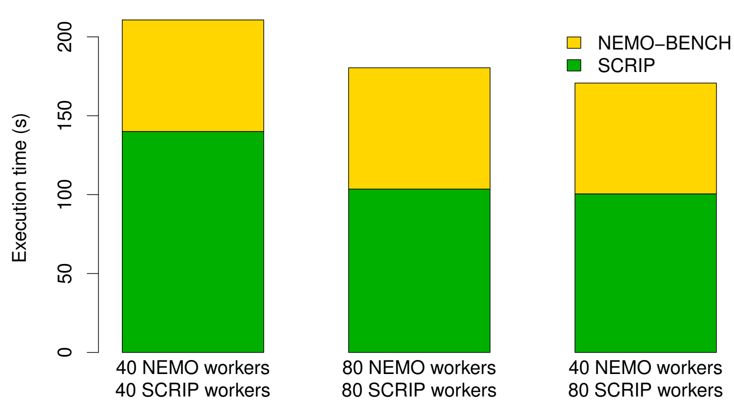
- Understand requirements of current and emerging applications
- Group challenges into a handful of themes we can study
- Propose a strawman solution for application composition

Climate modeling applications integrate multiple kernels with different runtime configurations

- Multiple domains and models
  - Ocean physics
  - Atmospheric physics
  - Biogeochemistry
- Model-specific kernels are developed independently
  - Need to exchange data
  - May use different runtimes
  - May have different optimal points



Thread heterogeneity from different models in one application is difficult to program and impacts performance



\# workers per node

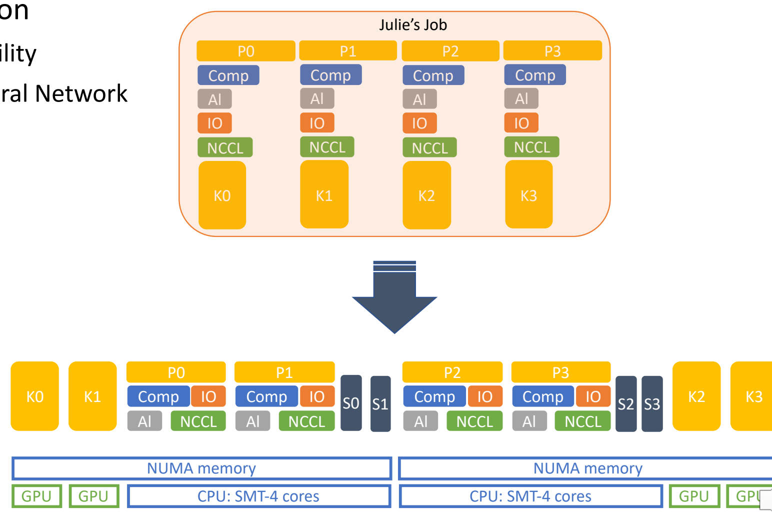
Best configuration
NEMO: 1 MPI task per core
SCRIP: 2 OpenMP threads per core

Each compute node: 40 cores, 2 HT/core

- OASIS from CERFACS
- NEMO-BENCH
  - Oceanic circulation model benchmark
  - MPI parallelism
- SCRIP
  - Interpolation library
  - OpenMP parallelism

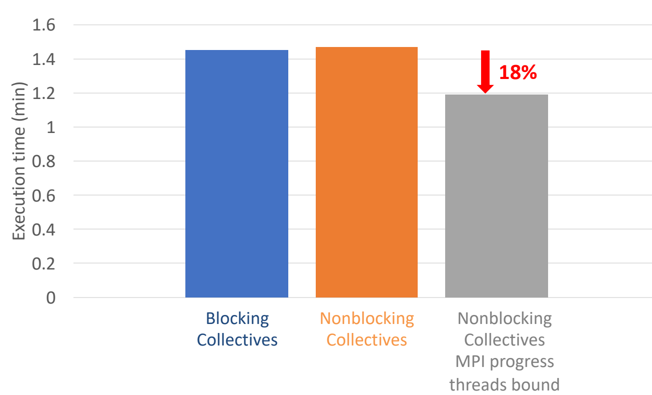
Coordinating multiple components, each with a different type and number of workers, is challenging and error-prone

- Inertial Confinement Fusion
  - LLNL's National Ignition Facility
  - Livermore Big Artificial Neural Network
- I/O
  - C++ threads on CPU
- Compute
  - OpenMP threads on CPU
  - GPU kernels
- Communication
  - Aluminum Pthreads
  - NCCL Pthreads



Placement of compute and utility threads is an important factor to enable computation and communication overlap

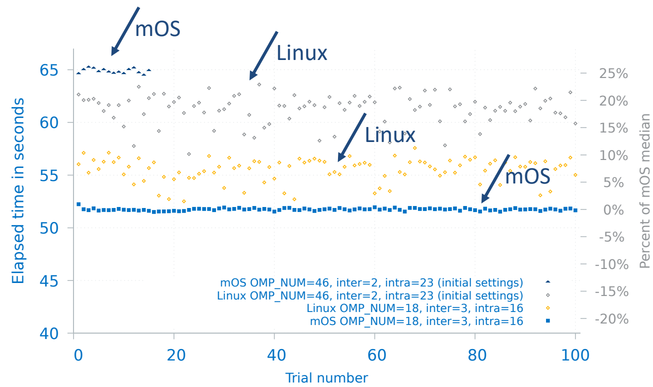
- Water flow simulations
  - GeoFEM from University of Tokyo
- Overlap comm & computation
  - Nonblocking MPI collectives
  - MPI's progress threads
- Considerations
  - Placement of utility threads matters
  - MPI library has no visibility into workers from other libraries



8 MPI tasks/node, 8 OpenMP threads/task
68 cores per compute node

Conflicting directives from different parts of the software stack may hinder performance and the ability to optimize

- Deep learning in Cancer problems
  - CANDLE benchmark
    Pilot 3 data set from ECP
  - TensorFlow + Intel MKL-DNN
- TensorFlow
  - Two thread pools to stage work
  - User controls size but not placement
- Intel MKL-DNN
  - OpenMP threads
  - User controls size and placement



Compute node with 48 cores

Our work:

- Understand requirements of current and emerging applications
- Group challenges into a handful of themes we can study
- Propose a strawman solution for application composition

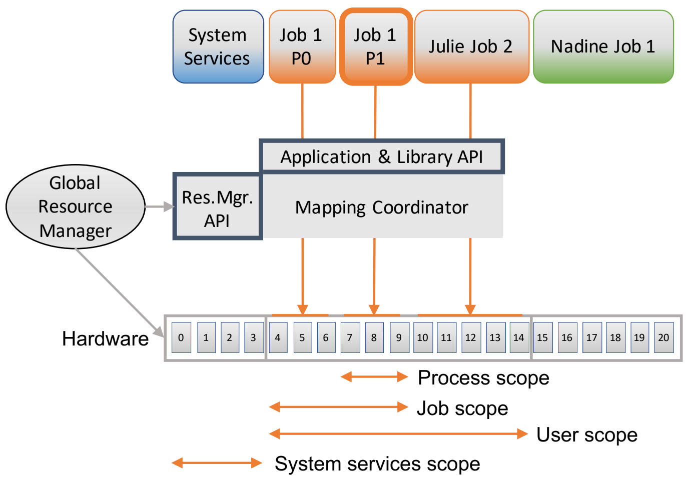
We created 4 themes to capture application requirements and simplify their study

	CANDLE	GeoFEM	NWChem	NEMO BENCH	Hydro	ICF LBANN	SCALE-LETKF
Multiple types of workers							
Dynamic compute workers							
Dynamic utility workers							
Remapping of tasks/threads							
Multiple applications							
MPI							
OpenMP							
POSIX threads							
NVIDIA CUDA							
C++ threads							

Our work:

- Understand requirements of current and emerging applications
- Group challenges into a handful of themes we can study
- Propose a strawman solution for application composition

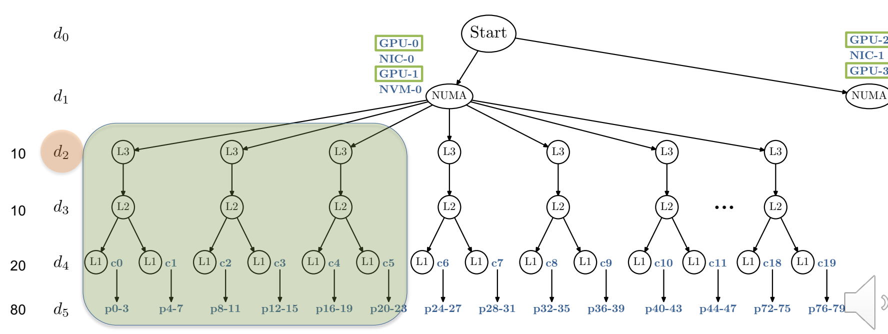
Our *Mapping Coordinator* can provide the functionality needed to meet application demands



- Provides two-prong interface
  - Low-level / high-level functions
- Reconfigures worker types dynamically
- Manages node-local HW
  - Affinity and binding
  - Query availability
  - Request & release
  - Arbitrate access
- Coordinates with the global resource manager

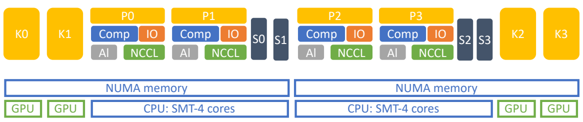
Scopes and mapping policies are high-level abstractions to help application developers

- Mapping policies
  - High-level interface
    No HW topology specific info
    Portable and applied dynamically
  - Low-level interface
- `mpibind`
  - Simple interface: # workers
  - Memory driven
  - Heterogeneous architectures
  - Hybrid applications

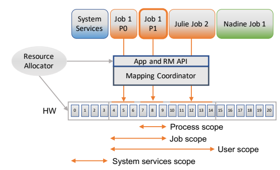


Enabling application composition will be paramount for emerging science applications on tomorrow's systems

- Components assume dedicated resources
  - Coordination left to app/library developers
- Problem not new, but is getting worse
  - Application workflows are more complex
  - Systems are more heterogeneous



- Identified challenges based on real applications
  - Multiple, uncoordinated types of threads
  - Dynamic work by auxiliary libraries
  - Rebalance and remap of workers
  - Multiple applications working together
- Proposed a Mapping Coordinator strawman to mitigate challenges



Lawrence Livermore National Laboratory