MAY 4, 2017

PERFORMANCE PORTABILITY

TIM WILLIAMS
Deputy Director of Science
ALCF

Argonne NATIONAL LABORATORY

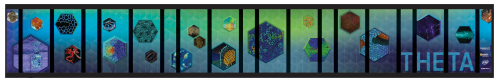
TWO ARCHITECTURAL FOR PRE-EXASCALE SYSTEMS

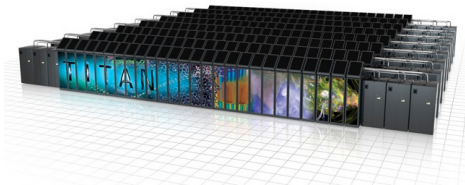
MANY CORE

- Tens of thousands of nodes
- Millions of homogeneous cores
- Multiple memory levels:
  - HBM (on-package), DDR, NVM

HYBRID MULTI-CORE

- Few thousand nodes
- CPU + multiple GPUs
- Coherent shared memory on node
- Multiple memory levels
  - On-package, DDR, NVM





TWO ARCHITECTURAL FOR PRE-EXASCALE SYSTEMS

MANY CORE

- Tens of thousands of nodes
- [illegible]neous cores
- [illegible]vels:
  [illegible]ge), DDR, NVM

DOE:

Maintain architectural diversity

HYBRID MULTI-CORE

- Few thousand nodes
- CPU + multiple GPUs
- Coherent shared memory on node
- Multiple memory levels
  - On-package, DDR, NVM

TWO ARCHITECTURAL FOR PRE-EXASCALE SYSTEMS

MANY CORE

- Tens of thousands of nodes
- [illegible] neous cores
- [illegible] vels:
  [illegible]

HYBRID MULTI-CORE

- Few thousand nodes
- CPU + multiple GPUs
- Coherent shared memory on node
- [illegible] tiple memory levels
  On-package, DDR, NVM

DOE:

Maintain architectural diversity

…and make applications portable

TWO ARCHITECTURAL FOR PRE-EXASCALE SYSTEMS

MANY CORE

- Tens of thousands of nodes
- [illegible] neous cores
- [illegible] vels:

HYBRID MULTI-CORE

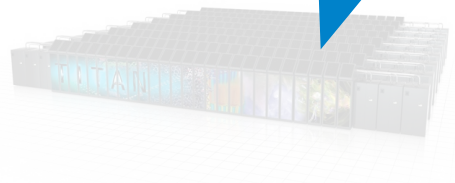
- Few thousand node
- CPU + multi[illegible]
- Coherent [illegible]
- [illegible] tiple [illegible]
- On-[illegible]

DOE:

Maintain architectural diversity

…and make applications portable

…with high performance!



ALCF, OLCF, NERSC, LLNL, SNL, LANL, LBN, JLAB, IBM, INTEL, CRAY, NVIDIA

- ALCF, OLCF, NERSC

DOE Centers of Excellence Performance Portability Meeting

April 19–21, 2016
Glendale, Arizona

- https://asc.llnl.gov/DOE-COE-Mtg-2016/
- Application Experiences
- Performance Portable Abstractions
- Managing Memory Hierarchy
- Experience with OpenMP
- Tools
- DSLs
- …

PERFORMANCE PORTABILITY: WHAT IS IT?

PERFORMANCE PORTABILITY: WHAT IS IT?

- 2014 FASTMath meeting:
  - *Same piece of code (from the user perspective) runs on different architectures with 'good' performance*
  - *A relatively small amount of effort is needed to make a change to get good performance within advertised (algorithmic or performance) tolerances across both current and future architectures*

- Kokkos
  - *The amount of user code which can be compiled for diverse manycore architectures and obtain the same, or nearly the same, performance as an architecture specialized version of that code.*

- A Metric for Performance Portability (Pennycook et al.)
  - *A measurement of an application's performance efficiency for a given problem that can be executed correctly on all platforms in a given set*

https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/Edwards-2013-XSCALE13-Kokkos.pdf https://arxiv.org/pdf/1611.07409.pdf

PERFORMANCE PORTABILITY STRATEGIES

1. Conditional compilation
2. Directives
3. Libraries
4. Frameworks
5. General-purpose high-level programming languages
6. DSLs (Domain specific languages)
7. Common HPC development environment
8. Co-design hardware and software using scientific apps

CONDITIONAL COMPILATION: XGC

Gyrokinetic particle-in-cell tokamak plasma simulation

- Particles (ions & electrons) move continuously through toroidal spatial grid
- Accumulate charge and current densities from particles to grid points
- Solve Maxwell's equations on the grid
- Gather electromagnetic field values from grid to particle positions
- Move particles via Lorentz force of EM fields on charged particle

CONDITIONAL COMPILATION: XGC

Gyrokinetic particle-in-cell tokamak plasma simulation

- Particles (ions & electrons) move continuously through toroidal spatial grid
- Accumulate charge and current densities from particles to grid points
- Solve Maxwell's equations on the grid
- Gather electromagnetic field values from grid to particle positions
- Move particles via Lorentz force of EM fields on charged particle

CONDITIONAL COMPILATION: XGC

Gyrokinetic particle-in-cell tokamak plasma simulation

- Particles (ions & electrons) move continuously through toroidal spatial grid
- Accumulate charge and current densities from particles to grid points
- Solve Maxwell's equations on the grid
- Gather electromagnetic field values from grid to particle positions
- Move particles via Lorentz force of EM fields on charged particle

push

CONDITIONAL COMPILATION: XGC

Gyrokinetic particle-in-cell tokamak plasma simulation

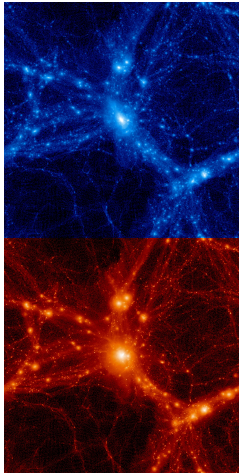
- Particles (ions & electrons) move continuously through toroidal spatial grid
- Accumulate charge and current densities from particles to grid points
- Solve Maxwell's equations on the grid
- Gather electromagnetic field values from grid to particle positions
- Move particles via Lorentz force of EM fields on charged particle

push

MIRA, THETA (MANY CORE)	TITAN (CPU + GPU)
▪ MPI + OpenMP	▪ Accumulation + field solve: MPI + OpenMP on CPU ▪ Push: CUDA Fortran on GPU

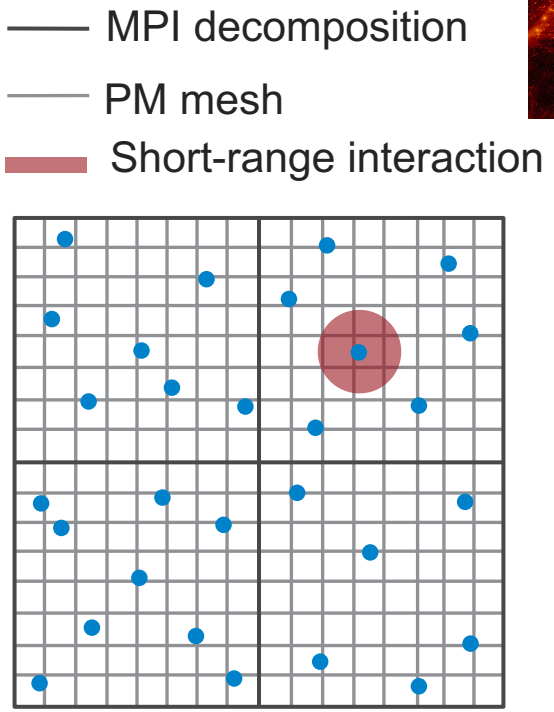
CONDITIONAL COMPILATION: HACC

Hardware Accelerated Cosmology Code



- Underlying calculation is the gravitational force.
- Brute force method $\mathcal{O}(N_p^2)$. Number of particles $N_p \sim 10^{12}$.

1) Long-range component.
   - Particle mesh (PM) method
     - Global FFT across all MPI ranks.
2) Short-range component.
   - Specifics depend on architecture:
     - GPU: Direct pairwise force
     - CPU: Tree computation
   - Bulk of computational time spent here.
   - Requires only rank-local shared memory.



DIRECTIVES

- Can MPI + OpenMP 4+ give us performance portability?

```
double x[128], y[128];
#pragma omp target data map(to:x[0:64])
    map(tofrom:y[0;64])
{
#pragma omp target
  {
    // y computed on device
  }
}
```

```
double x[128], y[128];
#pragma omp for simd aligned(x, y: 32)
for (int i=0; i<128; i++ ) {
  // thread's iterates → SIMD lanes
}
```

```
#pragma omp target data map(to:x) {
  #pragma omp target map(tofrom:y)
  #pragma omp teams
  #pragma omp distribute
  #pragma omp parallel for
  for (int i = 0; i < n; ++i) {
    y[i] = a*x[i] + y[i];
  }
}
```

DIRECTIVES

- OpenMP 5.x – proposed memory management support (TR 5)
  - Memory space: value of memory traits define characteristics of a space:
    - `distance = near, far`
    - `bandwidth = highest, lowest`
    - `latency = highest, lowest`
    - `location = core, socket, device`
    - `optimized = bandwidth, latency, capacity, none`

Lots of ongoing work trying MPI+OpenMP 4+ on multiple architectures. Comparisons with OpenACC 2+, other non-directives approaches.

http://www.openmp.org/wp-content/uploads/openmp-TR5-final.pdf

LIBRARIES / FRAMEWORKS

SOLVERS/MATHEMATICAL

- PETSc
- Trilinos
- ESSL
- ScaLAPACK
- MKL
- MAGMA
- FFTW
- HYPRE
- BoxLib

PARALLEL ALGORITHMS/STRUCTURES

- Thrust
  - Containers, iterators
  - Algorithms
    - Transforms w/functors
    - Parallel prefix-sums
- ADLB
- Global Arrays
- Legion - data centric
- HPX
- Kokkos
- RAJA

WHY ALL THIS C++?

- Templates
  - `template <typename Policy> class A;`
    - `Policy` specifes *how* class `A` implements functions
    - Write tuned specializations for different Policy types
    - Functions are non-virtual and can be inlined for speed
  - Operator overloading (+, -, …)
  - Parenthesis “()” overloading
  - Compile-time template metaprogramming
  - lambda expressions
  - Traits classes – useful properties of a type wrapped in a class
    - C++11: `include <type_traits>`
      - `is_array, is_class, is_pointer, alignment_of, …`

RAJA

- Traversals & execution policies (loop scheduling, execution)
- IndexSets (iteration space partitioning, ordering, dependencies, placement, etc.)
- Reduction types (programming model portability)

How loop iterations are scheduled to hardware: OpenMP, sequential, CUDA

```
forall<exec_policy>(iset, [&] (Index_type i) {
    y[ i ] += a * x[ i ] ;
});
```

https://asc.llnl.gov/DOE-COE-Mtg-2016/talks/2-09_Beckingsale.pdf

https://asc.llnl.gov/DOE-COE-Mtg-2016/talks/1-23_Hornung.pdf

RAJA

```
double* x ; double* y ;
double a, tsum = 0, tmin = MYMAX;
…
for ( int i = begin; i < end; ++i ) {
  y[i] += a * x[i] ;
  tsum += y[i] ;
  if ( y[i] < tmin ) tmin = y[i];
}
```

```
double* x ; double* y ; double a;
RAJA::SumReduction<reduce_policy, double> tsum(0);
RAJA::MinReduction<reduce_policy, double> tmin(MYMAX);
…
RAJA::forall< exec_policy > ( IndexSet , [=] (int i) {
    y[i] += a * x[i] ;
    tsum += y[i];
    tmin.min( y[i] );
} );
```

https://asc.llnl.gov/DOE-COE-Mtg-2016/talks/2-09_Beckingsale.pdf

https://asc.llnl.gov/DOE-COE-Mtg-2016/talks/1-23_Hornung.pdf

KOKKOS

- Data Structures
  - View – Multidimensional Array
    - View<double**, MemoryTraits<Atomic> > a_atomic = a;
  - Kokkos Containers: DualView<type,device>, Vector<t,d>, UnorderedMap<Key,Value,Device>
- Parallel Execution
  - parallel_for, parallel_reduce, parallel_scan
- KokkosKernels (under dev COEPP)
  - BLAS, Sparse, Graph, Tensor kernels

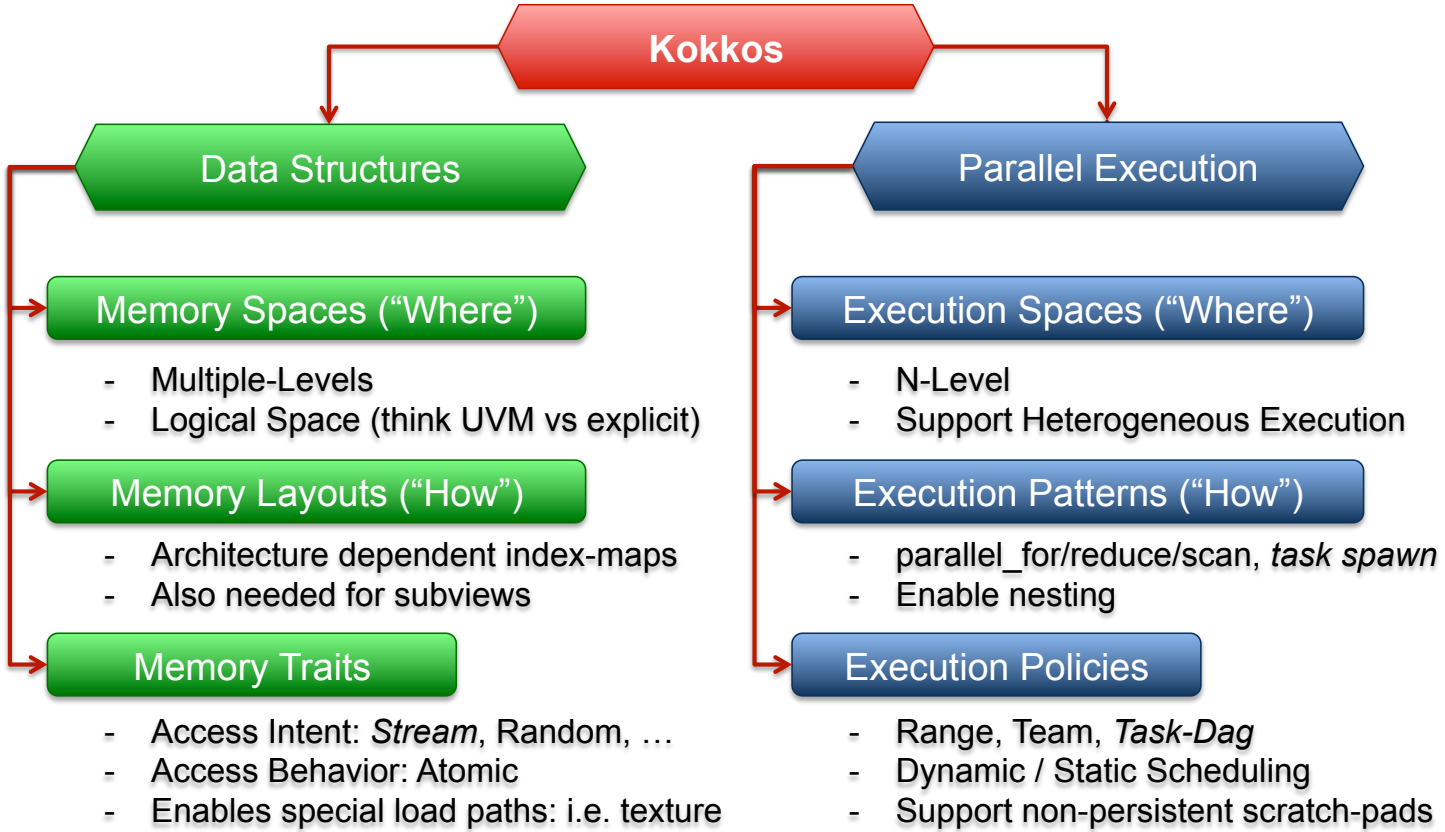
https://github.com/kokkos

https://asc.llnl.gov/DOE-COE-Mtg-2016/talks/1-22_Trott.pdf

Performance Portability through Abstraction

Sandia National Laboratories

Separating of Concerns for Future Systems…



GENERAL-PURPOSE HIGH-LEVEL LANGUAGES

- Charm++ 🔗
- UPC 🔗
- X10 🔗
- Coarray Fortran (Fortran 2008) 🔗
- HPF 🔗
- Chapel 🔗

DSLs (DOMAIN SPECIFIC LANGUAGES)

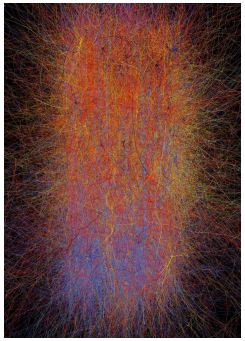
- "Natural" language of the scientific/mathematical domain
- Compact, unambiguous
- Example: NMODL DSL
  - Domain: computational neuroscience

```
NEURON {
  SUFFIX leak
  NONSPECIFIC_CURRENT I
  RANGE i, e, g
}

PARAMETER {
  g = 0.001  (siemens/cm2)  < 0, 1e9 >
  e = -65    (millivolt)
}

ASSIGNED {
  i  (milliamp/cm2)
  v  (millivolt)
}
```

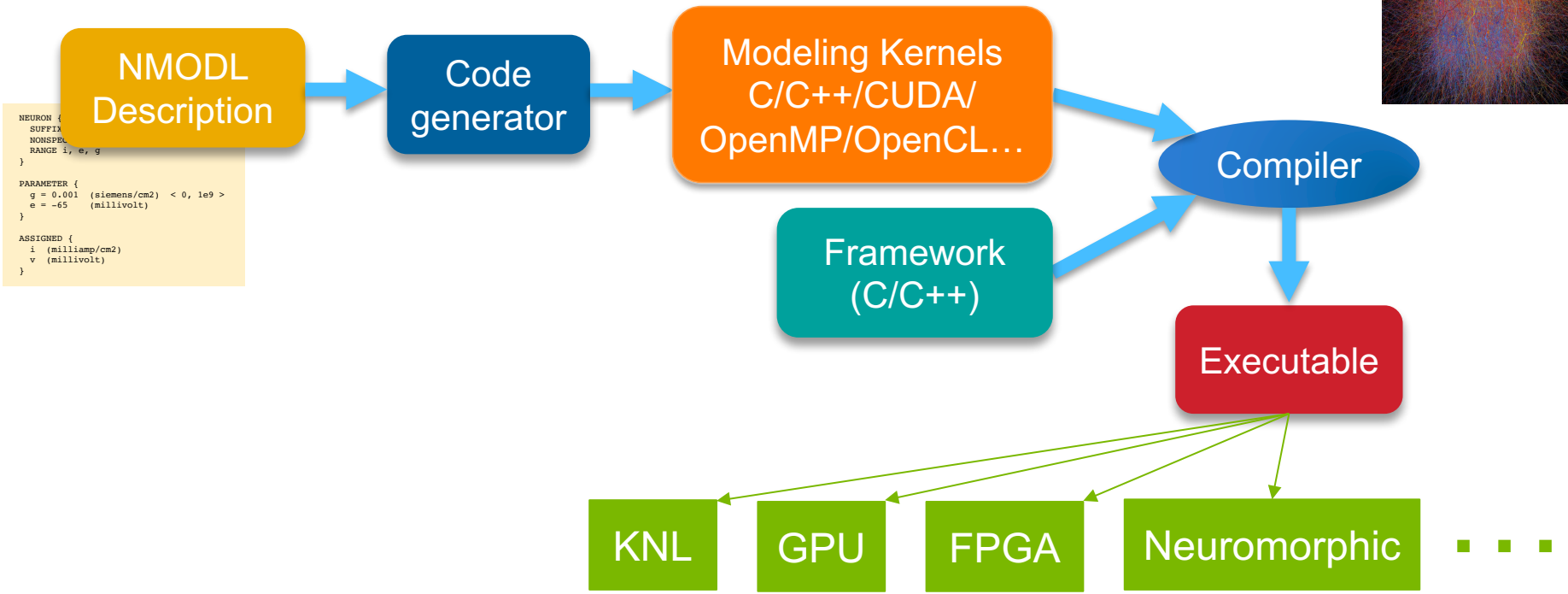

http://mcraveiro.blogspot.com/2015/11/nerd-food-tooling-in-computational.html

DSL FOR PERFORMANCE PORTABILITY

- CoreNEURON brain tissue simulation



```
NEURON {
  SUFFIX
  NONSPE
  RANGE i, e, g
}

PARAMETER {
  g = 0.001  (siemens/cm2)  < 0, 1e9 >
  e = -65    (millivolt)
}

ASSIGNED {
  i  (milliamp/cm2)
  v  (millivolt)
}
```



http://www.alcf.anl.gov/projects/large-scale-simulation-brain-tissue-blue-brain-project-epfl

FREE ADVICE

- Establish performance targets/bounds
- Detailed characterization of performance
- Extract kernels/mini-apps
- Encapsulate portability challenges
  - Modularity
  - High-level abstractions
  - Libraries
  - …good software engineering practices
- Avoid architecture specific models:
  - Intel Thread Building Blocks
  - NVIDIA CUDA
  - If necessary, encapsulate
- Good coding practices
  - Parameters for thread counts a d thread placements
  - Data structures flexibly allocatable to different memory spaces
  - Task level flexibility so work can be allocated on different compute elements (GPU & CPU)

NOT DONE YET

- Another one in 2017

- Concrete portability studies:

ALCF	NERSC	OLCF
• NekBone	• BoxLib MG solvers	• DSL-based library for MD

- Developing best practices guide