SAND2015-10209C

Exceptional service in the national interest

Photos placed in horizontal position with even amount of white space between photos and header

Maintainability and Performance for LAMMPS

Christian Trott, Tzu-Ray Shan, Stan Moore, Aidan Thompson and Steve Plimpton

Center for Computing Research; Sandia National Laboratories

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

LAMMPS a general purpose MD code

- C++, MPI based open source code: lammps.sandia.gov
- Modular design for easy extensibility by expert users
- Wide variety of supported particle physics:
  - Bio simulations, semi conductors, metals, granular materials
  - E.g. blood transport, strain simulations, grain flow, glass forming, self assembly of nano materials, neutron star matter
- Large flexibility in system constrains
  - Regions, walls, geometric shapes, external forces, particle injection
- Scalable: simulations with up to 6 Million MPI ranks demonstrated

Estimate: 500 Performance Critical Kernels

LAMMPS on next generation platforms

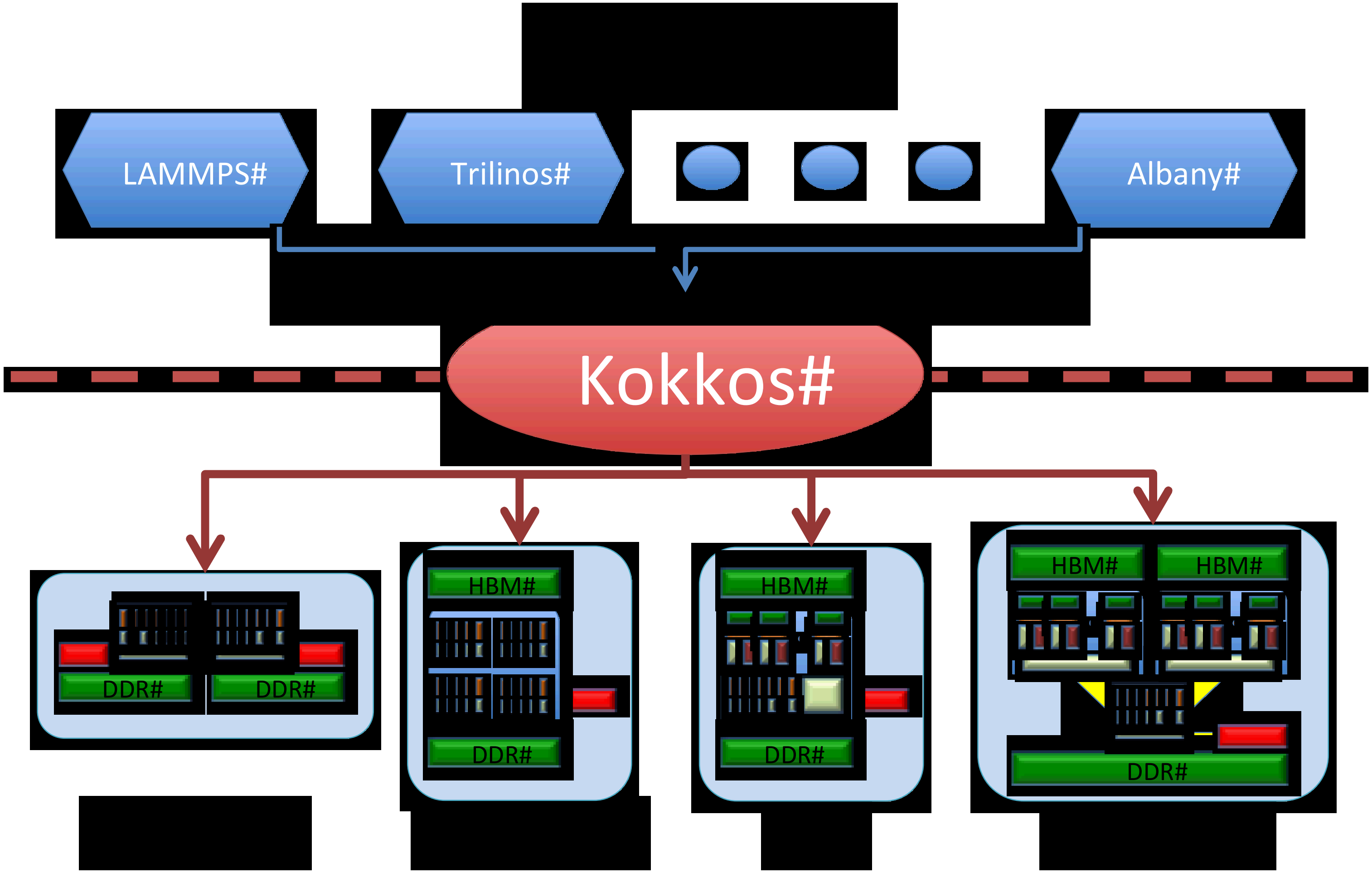
- Next generation platform support through packages
- GPU

Packages replicate existing physics modules:

Hard to maintain.
Prone to inconsistencies.
Much more code.

- INTEL
  - Intel Offload pragmas for Xeon Phi
  - Offloads force calculations (non-bonded, long range coulomb)

Kokkos: *Performance, Portability and Productivity*

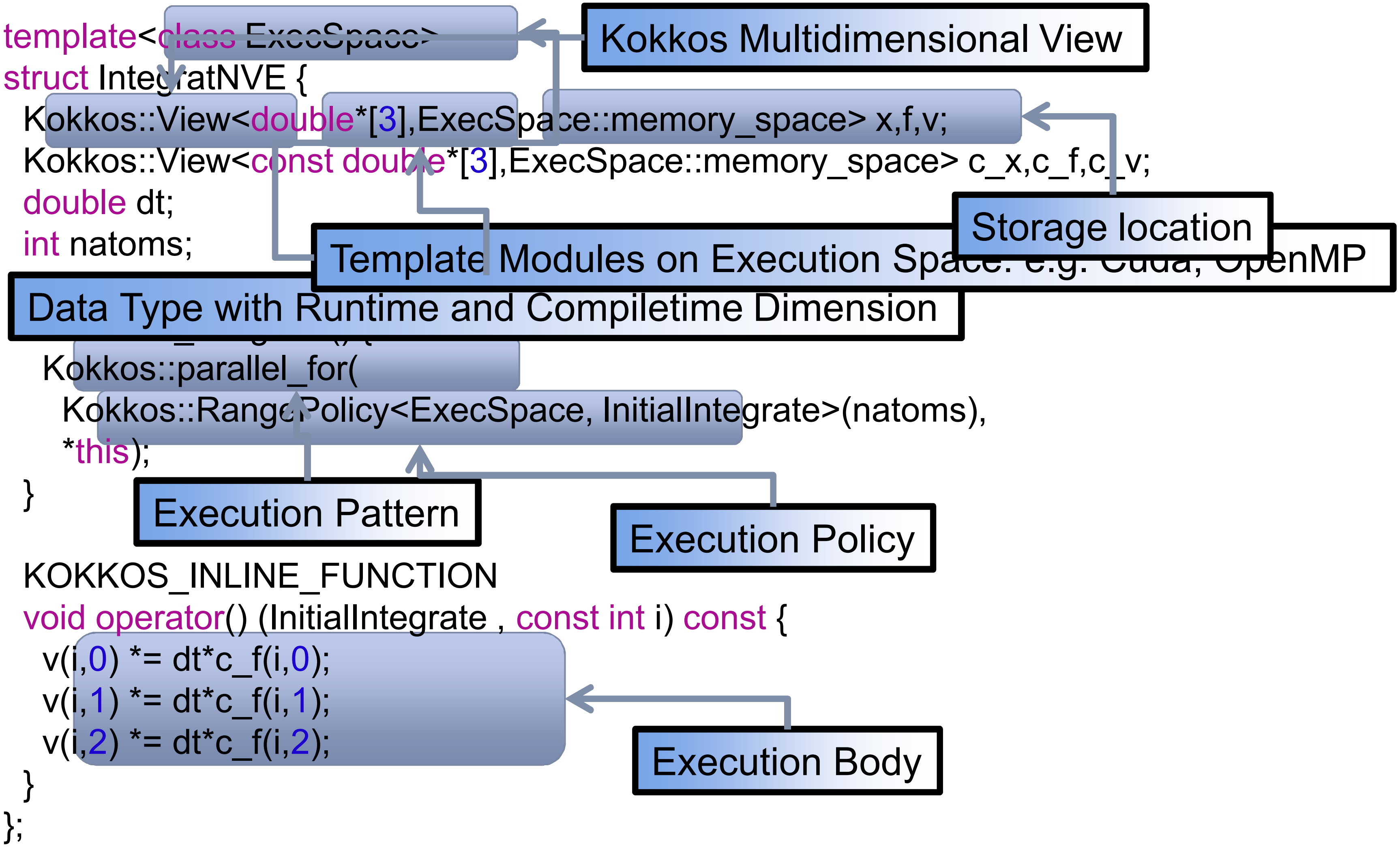


[redacted]

Kokkos: *Performance, Portability and Productivity*

- A programming model implemented as a C++ library
- Open Source (BSD): https://github.com/kokkos
- Abstractions for Parallel Execution and Data Management
  - Execution Pattern: What kind of operation (for-each, reduction, scan, task)
  - Execution Policy: How to execute (Range Policy, Team Policy, DAG)
  - Execution Space: Where to execute (GPU, Host Threads, PIM)
  - Memory Layout:
  - Memory Traits: How to access the data (Random, Stream, Atomic)
  - Memory Space: Where does the data live (High Bandwidth, DDR, NV)
- Supports multiple backends: OpenMP, Pthreads, Cuda, Qthreads, Kalmar (experimental)
- Sandia application teams committed to Kokkos as its path for transitioning legacy codes, and as part of its new codes
  - Trilinos, LAMMPS, Albany, Sierra Mechanics, …

Kokkos in LAMMPS: Examples I



```
template<class ExecSpace>
struct IntegratNVE {
  Kokkos::View<double*[3],ExecSpace::memory_space> x,f,v;
  Kokkos::View<const double*[3],ExecSpace::memory_space> c_x,c_f,c_v;
  double dt;
  int natoms;

   Kokkos::parallel_for(
     Kokkos::RangePolicy<ExecSpace, InitialIntegrate>(natoms),
     *this);
  }

  KOKKOS_INLINE_FUNCTION
  void operator() (InitialIntegrate , const int i) const {
   v(i,0) *= dt*c_f(i,0);
   v(i,1) *= dt*c_f(i,1);
   v(i,2) *= dt*c_f(i,2);
  }
};
```

Kokkos Multidimensional View

Storage location

Template Modules on Execution Space: e.g. Cuda, OpenMP

Data Type with Runtime and Compiletime Dimension

Execution Pattern

Execution Policy

Execution Body

Kokkos in LAMMPS: Examples II

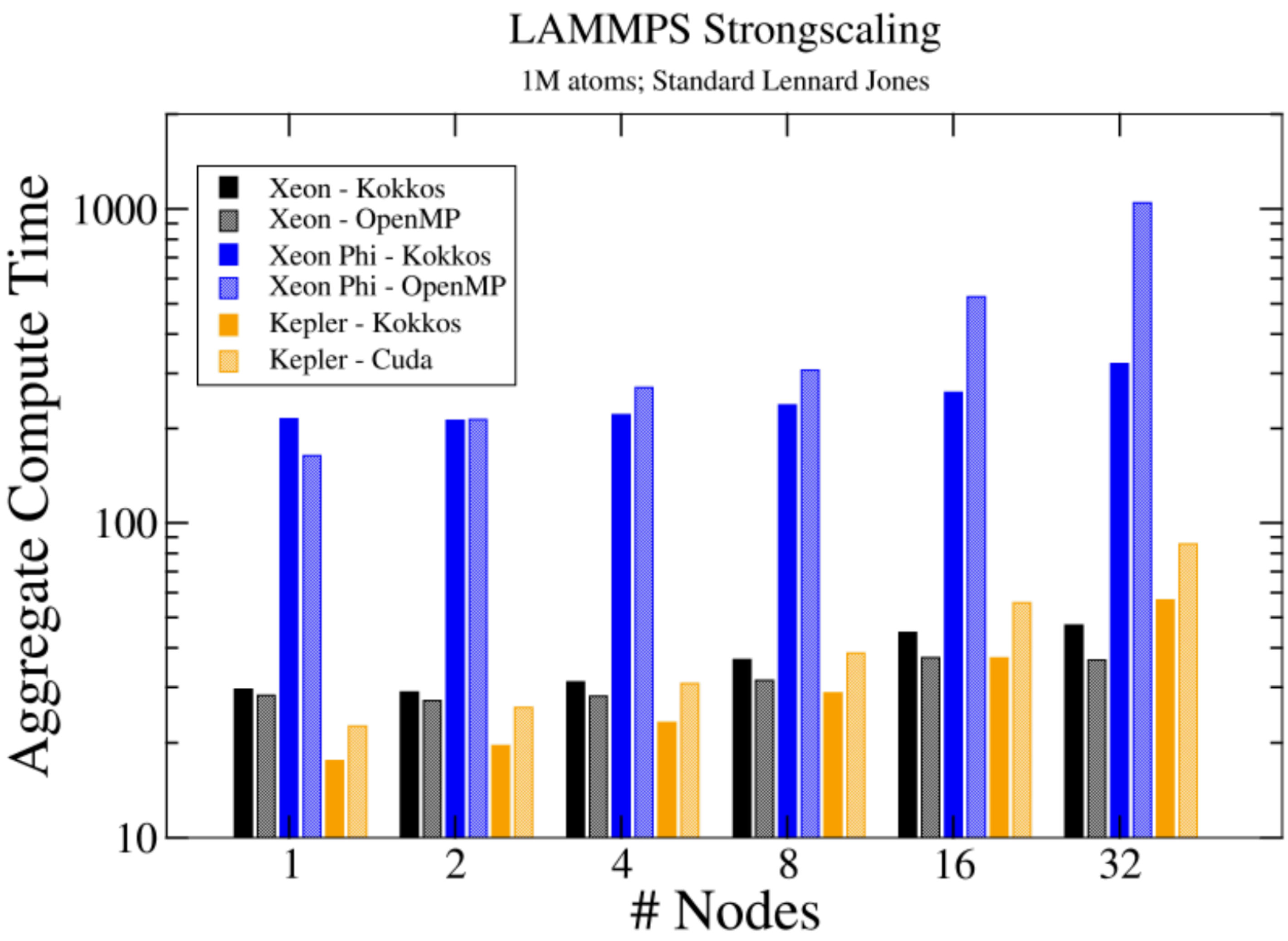
```
template<class ForceType, bool half_neigh>
struct PairForce {
  typedef Kokkos::MemoryTraits<
    typename std::if_c<half_neigh,Kokkos::Atomic,0>::type> atomic_trait;
  Kokkos::View<double*[3], ExecSpace::memory_space, atomic_trait> f;
  Kokkos::View<double*[3], ExecSpace::memory_space,
             Kokkos::MemoryTraits<Kokkos::RandomAccess> > x;
  Kokkos::View<int**, NeighLayout,ExecSpace::memory_space> neighbors;
  ForceType force;

  KOKKOS_INLINE_FUNCTION
  void operator() (const int& i) const {
    const double x_i = x(i,0);
    for(int jj=0; jj<numneigh(i); jj++) {
      const int j = neighbors(i,jj);
      const double dx = x(j,0) - x_i;
      const double rsq = dx*dx+dy*dy+dz*dz;
      if(rsq < cutoff) {
        f_ij = force.eval(rsq);
        fx_i += dx * f_ij;
        if(half_neigh) f(j,0) -= dx*f_ij;
      }
    }
  }
};
```

Choose Atomic Access for Half Neighbor List

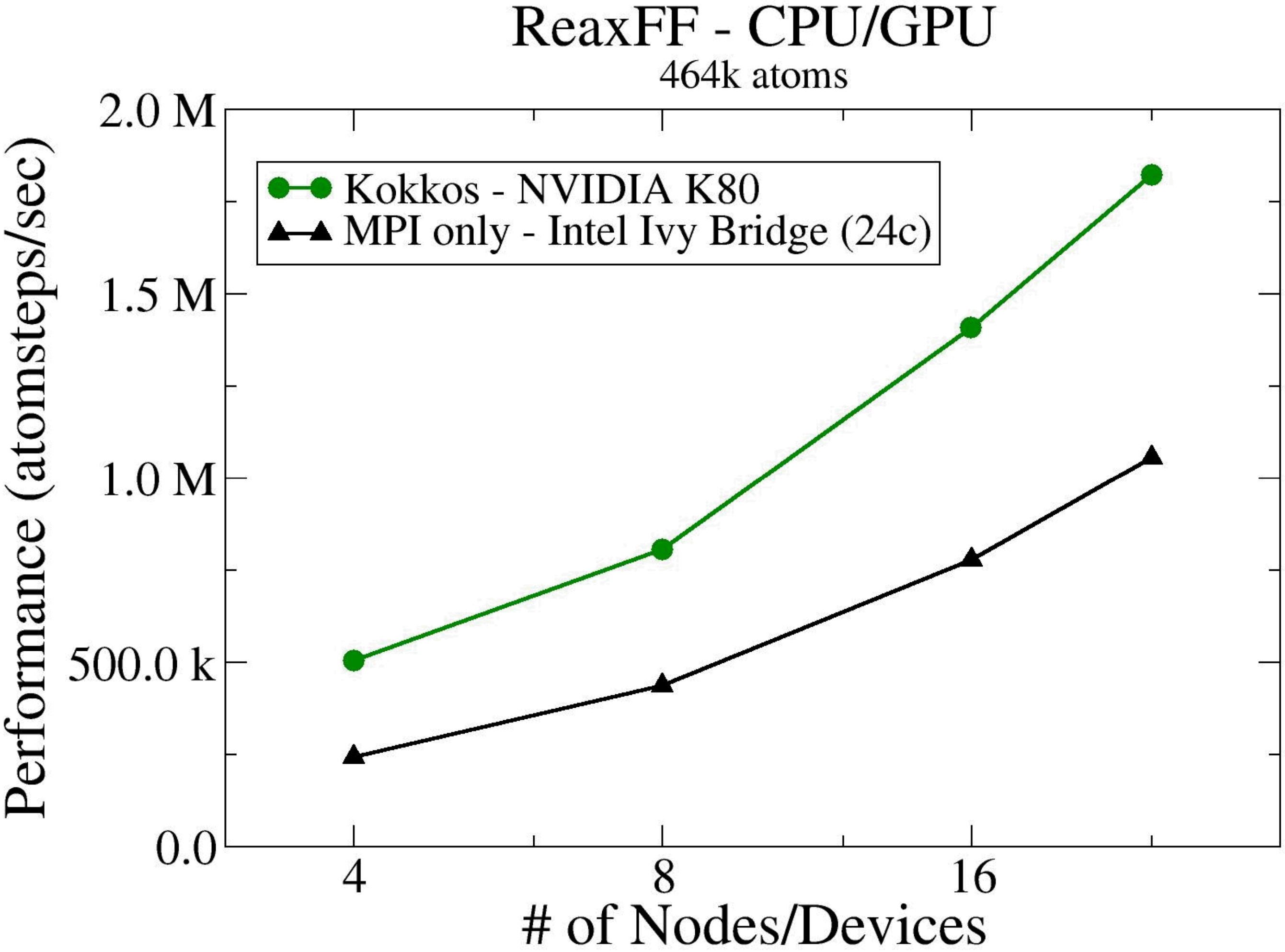
Use Random Access Hint to get Texture Fetches

Use Optimal Memory Layout for each Architecture

Performance Evaluation (I)



Performance Evaluation (II)



The Way Forward

- Kokkos in LAMMPS appears to deliver on performance, portability and productivity
- We believe it is a practical solution to the needs of codes with large loop counts
  - If the number of performance critical regions is small, specialisation might be less intrusive
- Institutional support for Kokkos through Sandia ensures longevity (and for us: in-house expertise)
- We started on long process of providing Kokkos versions for all modules, most new capabilities developed at Sandia will be Kokkos from the get-go
- Expect to be ready for Summit/Sierra platforms in 2018