Towards Performance-Portable Applications through Kokkos:

A Case Study with LAMMPS

Christian Trott
Carter Edwards
Simon Hammond

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.
The challenge – Node parallelism

<table>
<thead>
<tr>
<th></th>
<th>CPU 2001</th>
<th>CPU Now</th>
<th>MIC</th>
<th>APU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU cores</td>
<td>4</td>
<td>256</td>
<td>~2,000</td>
<td>~5,000</td>
<td>~50,000</td>
</tr>
</tbody>
</table>

MPI-Only will not work anymore!
Domains get to small!
We need threading.
The challenge – Memory Access

Memory systems get more complex.

We need to use special hardware capabilities to achieve good performance.
What do we want?

- Single code base
- Support for all current (and future) hardware
- Flexible run configurations
  - MPI-Only
  - MPI + Threads
  - MPI + GPU
  - MPI + GPU + Threads
- Close to optimal performance (i.e. performance of a specialized code)
- Possibility for code specialisation
- Use vendor compilers
- Simple code

*Eierlegende Wollmilchsau (egg-laying wool-milk-sow)*
Kokkos as a solution

A programming model with two major components:

Data access abstraction
- Change data layout at compile time without changing access syntax
  => Optimal access pattern for each device
- Data padding and alignment is transparent
- Access traits for portable support of hardware specific load/store units

Parallel dispatch
- Express algorithms with parallel_for, parallel_reduce etc.
- Using functor concept
- Transparently mapped onto back-end languages (e.g. OpenMP, CUDA)

Goal: Separate science code from hardware details
What is Kokkos?

- C++ template library => almost everything is headers
- Developed as node level parallelism layer for Trilinos
  Trilinos is an Open-Source solver library, development led by Sandia
  www.trilinos.org
- Open-Source
- Standalone (no required dependencies)
- Lead developer: Carter Edwards, SNL
- Will be integrated into Trilinos during 2014

Pre print: Kokkos: Enabling manycore performance portability through polymorphic memory access patterns
H. Carter Edwards, Christian R. Trott; submitted to JPDC
How does it work

Multidimensional Arrays:
View<\texttt{int**}[8][3], \texttt{LayoutRight}, DeviceType> a(“A”,N,M);
- 4D array NxMx8x3
- RowMajor data storage (i.e. 4\textsuperscript{th} index is stride-one access)
- allocated in memory space of DeviceType
- access: double tmp = a(i,j,k,l);

View<\texttt{const int**}[8][3], \texttt{LayoutRight}, Device, RandomRead> b = a;
- b is a const view of the same data as a
- const + RandomRead => use Texture fetches on GPUs

Parallel dispatch:
struct AXPYFunctor {
    typedef Kokkos::Cuda device_type;
    ViewType a,b;
    AXPYFunctor (ViewType A, ViewType B): a(A),b(B) {}\
    void operator() (const int &i) const { a(i) += b(i); }
}

parallel_for(n, AXPYFunctor(a,b));
Performance Portability with Mantevo: MiniFE

Finite element code miniApp in Mantevo (mantevo.org)

*Heat conduction, Matrix assembly, CG solve*

Most variants of any miniApp in Mantevo

*more than 20 implementations in Mantevo repository; 8 in Mantevo 2.0 release*

Models aspects of Sandia’s mechanical engineering codes

---

**MiniFE CG-Solve time**

200x200x200 cells, 200 iterations

Volta

Compton H

XeonPhi B0

XeonPhi C0

Shannon

Teller

Watson

- OpenMP
- TBB
- MPI-Only
- Cilk+(1 Socket)
- Kokkos
- NVIDIA ELL
- OpenCL
- NVIDIA CuSparse

44-57s
Performance Portability with Mantevo: MiniMD

MiniMD Strongscaling
2M atoms; Standard Lennard Jones

Molecular Dynamics application
simplified LAMMPS

Variants:
- Reference (SNL)
- OpenCL (SNL)
- **Kokkos (SNL)**
- Intel Xeon Phi intrinsics (Intel)
- OpenACC (AMD)
- Chapel (Cray)
- Intel intrinsics (Warwick/Intel)
- Qthreads (SNL)
LAMMPS Prototype

Exploration of Kokkos for use in LAMMPS (lammps.sandia.gov)
  replace specialized packages => reduce code redundancy 3x
  enable thread scalability throughout code base

Leverage algorithmic exploration from MiniMD
  transferring thread-scalable algorithms

Get some simple simulations to run well
  Implement framework (data management, device management)
  Get all parts of a simulation run with Kokkos
  First Goal: MiniMD run
LAMMPS Strongscaling

1M atoms; Standard Lennard Jones

- Xeon - Kokkos
- Xeon - OpenMP
- Xeon Phi - Kokkos
- Xeon Phi - OpenMP
- Kepler - Kokkos
- Kepler - Cuda
- Xeon Phi - MPI-only Kokkos

# Devices x Time in sec vs # Devices

Graph showing the strongscaling performance of LAMMPS for different devices and configurations.
A side note: Performance on Xeon Phi

Gather out of cache appears to be inefficient on Xeon Phi.
Features of Kokkos

Backends:
  Pthreads
  OpenMP
  CUDA (UVM support in the plans)

Parallel execution:
  parallel_for
  parallel_reduce (for arbitrary types)
  parallel_scan

2 level threading:
  teams of threads
  primitives (team_scan, team_barrier)
  shared memory

Data abstraction:
  8-dimensional arrays
  View semantics
    (no hidden data transfers)
  compile-time data-layouts
  access traits (random, stream* ...)  
  data padding, alignment

Higher Level Libraries:
  container classes
    “std::vector”, dual-view, map
  sparse linear algebra
    CRS-Matrix, MatVec, ...
Conclusions

Kokkos: Research stable since September (keeping backward compatibility)

**Portable:** *one code for CPUs, MIC, GPUs, ...*
**Performance:** >90% of native implementations
**Extensible:** *use new back-ends without changing code*

Look for:  Mantevo 2.0 release here at SC13 and at [mantevo.org](http://mantevo.org)
              => *get the MiniAPPs*
Kokkos included in Trilinos at [trilinos.org](http://trilinos.org)
LAMMPPS downloads at [lammps.sandia.gov](http://lammps.sandia.gov)