Kokkos Update for Trilinos Developers

Trilinos Developers; June 5, 2014

SAND2014-4545P (Unlimited Release)
Increasingly Complex Heterogeneous Future
¿ Performance Portable and Future Proof Codes?

**Memory Spaces**
- Bulk non-volatile (Flash?)
- Standard DDR (DDR4)
- Fast memory (HBM/HMC)
- (Segmented) scratch-pad on die

**Execution Spaces**
- Throughput cores (GPU)
- Latency optimized cores (CPU)
- Processing in memory

**Special Hardware**
- Non caching loads
- Read only cache
- Atomics

**Programming models**
- GPU: CUDA-ish
- CPU: OpenMP
- PIM: ???
Vision for Managing Heterogeneous Future

- **“MPI + X” Programming Model, separate concerns**
  - Inter-node: MPI and domain specific libraries layered on MPI
  - Intra-node: Kokkos and domain specific libraries layered on Kokkos

- **Intra-node parallelism, heterogeneity & diversity concerns**
  - Execution spaces’ (CPU, GPU, PIM, ...) diverse performance requirements
  - Memory spaces’ diverse capabilities and performance characteristics
  - Vendors’ diverse programming models for optimal utilization of hardware

- **Desire standardized performance portable programming model**
  - Via vendors’ (slow) negotiations: OpenMP, OpenACC, OpenCL, C++17
  - Vendors’ (biased) solutions: C++AMP, Thrust, CilkPlus, TBB, ArrayFire, ...
  - Researchers’ solutions: HPX, StarPU, Bolt, Charm++, ...

- **Necessary condition: address execution & memory space diversity**
  - SNL Computing Research Center’s Kokkos (C++ library) solution
  - Engagement with ISO C++ Standard committee to influence C++17
Programmatics

- **ASC/CSSE (FY11 start):** Heterogeneous Computing project
  - Tight integration with co-design, mini-application, and testbed projects
  - Manycore (CPU, GPU, Xeon Phi, ...) performance portable “X” for MPI+X
    - Kokkos library is the “X” for fine grain data parallelism
  - 1.0-1.4 FTE split among ~2 staff + interns (FY14 @ 1.0 FTE)

- **ASCR/EASI:** Sparse Linear Algebra Kernels on Manycore
  - Some portion of this project also working on Kokkos core

- **LDRD (FY14 start):** Unified Task+Data Manycore Parallelism
  - For solver-preconditioners, finite elements, informatics, transport sweeps, ...
  - 0.9 FTE split among ~4 staff

- Internal/external interests, and resource challenge ahead
  - Trilinos, LAMMPS, SIERRA, other ASC codes (SNL, LANL, LLNL), AWE, ...
  - ISO C++ standards addressing fine grain parallelism (am a voting member)

  - Currently under-resourced for production-growth support
Kokkos: A Layered Collection of Libraries

- Standard C++, Not a language extension
  - In *spirit* of TBB, Thrust & CUSP, C++AMP, LLNL’s RAJA, ...
  - *Not* a language extension like OpenMP, OpenACC, OpenCL, CUDA, ...
- Uses C++ template meta-programming
  - Rely on C++1998 standard (supported everywhere except IBM’s xlC)
  - Moving to C++2011 for concise lambda syntax (required by LLNL’s RAJA)
    - Vendors slowly catching up to C++2011 language compliance

Application and Domain Specific Library Layer

- Kokkos Sparse Linear Algebra
- Kokkos Containers
- Kokkos Core

Back-ends: OpenMP, pthreads, Cuda, vendor libraries ...
Evaluate Performance Overhead of Abstraction
Kokkos competitive with native programming models

- MiniFE: finite element linear system iterative solver mini-app
- Compare to versions specialized for programming models
- Running on hardware testbeds

![MiniFE CG-Solve time for 200 iterations on 200^3 mesh chart]

- K20X
- IvyBridge
- SandyBridge
- XeonPhi B0
- XeonPhi C0
- IBM Power7+

- NVIDIA ELL
- NVIDIA CuSparse
- Kokkos
- OpenMP
- MPI-Only
- OpenCL
- TBB
- Cilk+(1 Socket)
Thread-Scalable Fill of Sparse Linear System

- MiniFENL: Newton iteration of FEM: \( x_{n+1} = x_n - J^{-1}(x_n) r(x_n) \)
- Thread-scalable pattern: Scatter-Atomic-Add or Gather-Sum?
  - Scatter-Atomic-Add
    + Simpler
    + Less memory
    - Slower HW atomic
  - Gather-Sum
    + Bit-wise reproducibility
- Performance win?
  - Scatter-atomic-add
  - \(~equal~Xeon~PHI\)
  - \(40\%~faster~Kepler~GPU\)
- Pattern chosen
  - Feedback to HW vendors: performant atomics

![Diagram showing comparison between Scatter-Atomic-Add and Gather-Sum methods in terms of performance and characteristics.]

![Graph showing matrix fill time for different finite element node numbers for various hardware configurations.]

- Phi-60 GatherSum
- Phi-60 ScatterAtomic
- Phi-240 GatherSum
- Phi-240 ScatterAtomic
- K40X GatherSum
- K40X ScatterAtomic

Number of finite element nodes
Matrix Fill: microsec/node
Thread-Scalable Sparse Matrix Construction

- MiniFENL: Construct sparse matrix graph from FEM connectivity
- Thread scalable algorithm for constructing a data structure
  1. Parallel-for: fill Kokkos lock-free unordered map with FEM node-node pairs
  2. Parallel-scan: sparse matrix rows’ column counts into row offsets
  3. Parallel-for: query unordered map to fill sparse matrix column-index array
  4. Parallel-for: sort rows’ column-index subarray

- Pattern and tools generally applicable to construction and dynamic modification of data structures
Tpetra: Domain Specific Library Layer for Sparse Linear Algebra Solvers

- Funded by ASC/Algorithms and ASCR/EASI
- Tpetra: Sandia’s templated C++ library for sparse linear algebra
  - Templated on “scalar” type: float, double, automatic derivatives, UQ, ...
  - Incremental refactoring from pure-MPI to MPI+Kokkos
- CUDA UVM (unified virtual memory) codesign success
  - Sandia’s early access to CUDA 6.0 via Sandia/NVIDIA collaboration
  - Hidden in Kokkos, can neglect memory spaces and maintain correctness
  - Enables incremental refactoring and testing
- Early access to UVM a win-win
  - Expedited refactoring + early evaluation
  - Identified performance issue in driver
  - NVIDIA fixed before their release

![Graph showing comparison between different configurations of MiniFE and Tpetra](image)
LAMMPS (molecular dynamics application)
Porting to Kokkos has begun

- Funded by LAMMPS’ projects
- Enable thread scalability throughout code
  - Replace redundant hardware-specialized manycore parallel packages
- Current release (last week) has optional use of Kokkos
  - Data and device management
  - Some simple simulations can now run entirely on device
- Performs as well or better than original hardware-specialized packages
Improvements toward Production-Growth

- Address build (configure, compile, link) complexity
  - Especially for Cuda cross-compiling

- Address runtime initialization complexity
  - HWLOC not as portable as hoped (IBM, Mac)
  - Single Kokkos::initialize( argc , argv ) for all configured devices

- Redesign Device abstraction (classes)
  - Split into Execution Space and Memory Space abstractions (classes)
  - Configure-time default allows this template argument to be omitted

- Enable C++11 Lambda to be used in addition to Functor
  - Allow simple kernels to be defined inline for parallel_{for,reduce,scan}
  - Prototype available – limited to availability of C++11

- Better compile error messages for user’s template errors

- Documentation and project website

- Path-forward technical review by alpha-users on July 16
Task-Data Parallelism: Kokkos/Qthreads LDRD

- Task-DAG where tasks can be data parallel {for, reduce, scan}
  - Add *course grain* task API to Kokkos where a task runs on a team of threads
  - Add thread team task scheduling to Qthreads
  - Qthreads becomes a new Execution Space for Kokkos

- Support informatics graph data structures and algorithms
  - Prototype small subset of Multithreaded Graph Library on Kokkos/Qthreads

- Current API thoughts, in a nutshell
  - Self-describing task functor with simple spawn interface:
    ```
    Future<double> r = spawn( task_functor, dependences );
    ```

- Key design challenge: tasks cannot *explicitly* wait on other tasks
  - Some tasks spawn other tasks and need to wait (e.g., task recursion)
  - GPU kernels once dispatched must complete, they cannot block / wait
  - Solution: executing task respawns (reschedules) itself with new dependences