The Vision

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

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

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

3 Programming models
- GPU: CUDAish
- CPU: OpenMP
- PIM: ??

PIM

DDR

NIC

NVRAM
Kokkos

C++ template library, C++98, part of Trilinos but no internal dependencies

Data abstraction layer
- Multi-dimensional arrays
- Compile time data layouts
- Memory placement attribute
- Memory access traits
- NUMA aware
- View Semantics

Parallel dispatch
- Functor based
  - parallel_for, parallel_reduce, parallel_scan
- Different backend models
  - Pthreads, OpenMP, Cuda, more to come
- 2D Threading parallelism (thread-teams)
- Tasking layer under development
Memory Spaces

A 4D Kokkos View, 2 runtime+2 compile time dimensions on a NVIDIA GPU:

```cpp
typedef View<
    double**[4][8],
    Cuda>
    view_type;

view_type a("A", n, m);
```

Create a view on host with same data layout, padding etc.:

```cpp
view_type::HostMirror b = create_host_mirror_view(a);
```

Exchange data:

```cpp
depth_copy(a, b);
```

Control data layout:

```cpp
View<
    double **[8],
    LayoutRight,
    Cuda>
    c;

View<
    double **,
    LayoutTileLeft<4, 4>,
    OpenMP>
    c;
```
Execution Spaces

Device concept: *Execution Space + Memory Space*

Execution Space is tied to a Backend Programming Model
   e.g. Cuda + CudaSpace, Cuda + CudaUVM, OpenMP + HostSpace

A functor specifies the execution space

```cpp
#include < Kokkos_Core.hpp>

template<class view_type>
struct Dot {
    // Dispatch work to closest execution space
    typedef view_type::device_type device_type;
    view_type a, b;

    void operator (const view_type& a, const view_type& b) { a(a_in), b(b_in) {};

    KOKKOS_INLINE_FUNCTION
    void operator (int i, volatile double& sum) { sum += a(i)*b(i); }

    KOKKOS_INLINE_FUNCTION
    void join(volatile double& sum, double& add) { sum += add; } }
```
Special Hardware

Access Traits in 4th template parameter
How is it used: e.g. Random Access, Stream, Non Caching

Shallow copy conversion: think pointer casting.
\[
\text{view<const double**}[8], \text{LayoutRight, Cuda, Random}> \ d = c;
\]

Implementation via Template specialization of access operator:

```cpp
template<typename iTypex>

KOKKOS_INLINE_FUNCTION

double operator[] (const iType &i) const
{
    #if defined(__CUDA_ARCH__) && (300 <= __CUDA_ARCH__)
        AliasType v = tex1Dfetch<AliasType>(obj, i);
        return *(reinterpret_cast<ValueType*>(&v));
    
#else
        return ptr[i];
    #endif
}
```

Actual: Multiple layers with enable-if syntax to filter out incompatible types and deal with MemorySpace and Layout index calculation.
Impact of Layout

MiniMD Lennard Jones Force Kernel, 864k atoms, 2.5σ

Performance in GFlop/s

- Optimal settings
- Texture fetch disabled (Kepler)
- Wrong data layout

Xeon   | Xeon Phi | Kepler
---|---|---
50 | 50 | 200
Libraries

(Sparse) LinAlg

Vector
- Dot, Norm, Add, Scale

MultiVector capabilities
- Dot, Norm, Add, Scale

CrsMatrix
- MatVec

TPL support
- can call MKL, CuSparse if Layout and data types allow

Containers

vector
- std::vector drop-in replacement
- supports most common functions

DualView
- Holds views to two memory spaces
- Tracking/Synchronization

Unordered Map
- thread-scalable insert and retrieve

Some Guidelines:
- use general entry functions
- template expansion to get desired attributes (e.g. access traits)
- silent fallback for TPLs
MiniFE – CG-Solve

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

44-57s
Tpetra
- Templated distributed memory layer in Trilinos
- Prior node-level parallelism by ClassicKokkos:
  limited use(-fulness) outside Tpetra
  no data-layouts or access traits

- *Incremental* Porting to Kokkos
  Change Data structures
  Port Kernels
  *Keep Backward compatibility*

- MultiVector, Vector, CrsMatrix can use new Kokkos Devices
- Basic linear algebra kernels working
- OpenMP, Pthreads and CUDA backend working

- CUDA: using UVM in CUDA 6.0 for data management
  => reduced development time
  => no need to identify every possible execution path with Host access
Tpetra/MiniFE – CG-Solve

200^3 weak scaled, MiniFE – problem, Dual Intel SB/K20x

Time [s]

# of Nodes

Tpetra Cuda
MiniFE-Cuda
MiniFE-MKL

Tpetra Pthread
MiniFE-CuSparse

Tpetra TPI
MiniFE-Pthreads
Kokkos Projects / More Info

www.trilinos.org
- Trilinos with Tpetra, Kokkos etc.
- Documentation, Tutorials, Downloads

www.mantevo.org
- Mantevo Suite of miniApps (R&D 100 in 2013)
- Kokkos variants of miniMD and miniFE, more to come

lammmps.sandia.gov
- LAMMPS molecular dynamics code
- Kokkos integration started

Siam PP: MS27 Portable Manycore Sparse Linear System Assembly Algorithms
MS70 Unified Task-Data-Vector Parallelism on Manycore Architectures
MS73 Layered DSLs for Portable Manycore Performance

crtrott@sandia.gov, hcedwar@sandia.gov