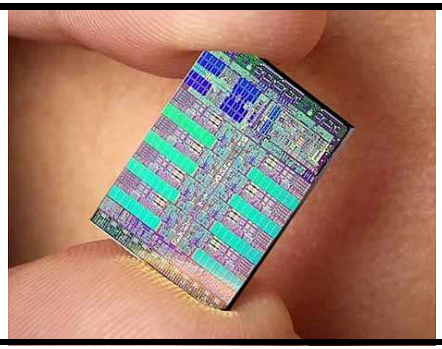
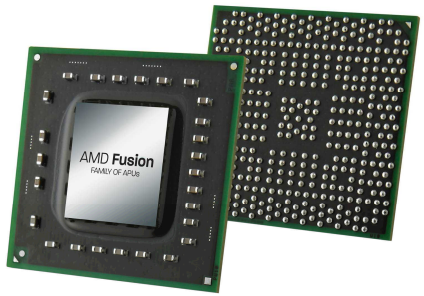
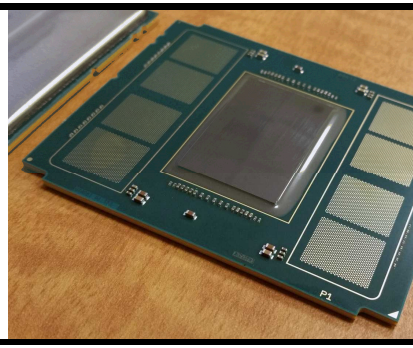
SAND2018-1490C



Kokkos: Hierarchical Parallelism

Christian Trott (crtrott@sandia.gov)

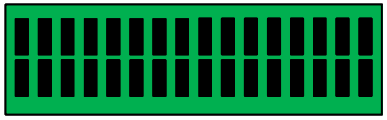
Center for Computing Research, Sandia National Laboratories, NM

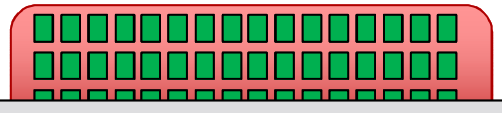
SAND2017-4935 C

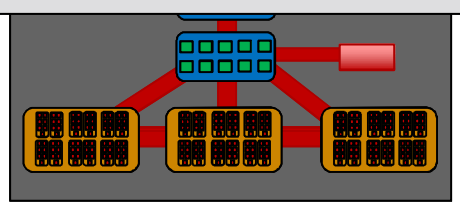
Parallelism on Summit

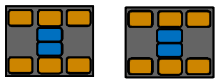
Sandia National Laboratories

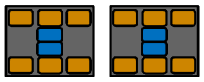
		Concurrently Executing CUDA threads
Warp	32 CUDA threads	32
Streaming Multi	64 Warps	2,048
NVI		
Node	2x (3 GPU + Power 9)	983,040
Machine	~ 4600 Nodes	~4,521,984,000











But logically active parallelism can be much larger:

Concurrent Kernels per GPU: 128
Maximum logical threads per Kernel: 2^31 x 1024
Total GPUs: ~27600

$$7.77 \times 10^{18}$$

Why expose this?

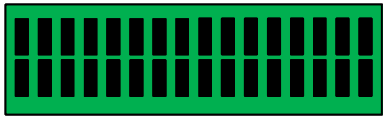
Sandia National Laboratories

- The finer the level the better the collaboration
  - __syncwarp() -> < 10 cycles
  - __syncthreads() -> ~100 cycles
  - Sync GPU -> O(us)
  - Sync Node -> O(us)
  - Sync Machine -> O(100us)??
- Algorithms are often inherently hierarchical
  - Reduction over neighbors to compute a value per cell/particle/element
  - Gather some data into a per-workset temporary buffer before computing
  - Multiple subsequent nested loops
- Outer Loops may not have enough parallelism
  - You may not have billions of particles, but maybe you got billions of interactions

Kokkos Approach

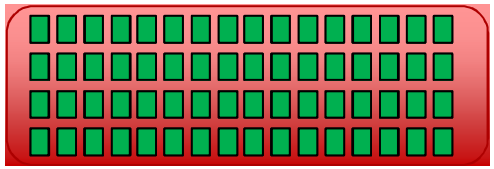
Sandia National Laboratories

Warp



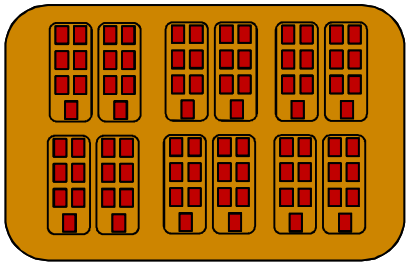
Kokkos::ThreadVectorRange

Streaming Multiprocessor



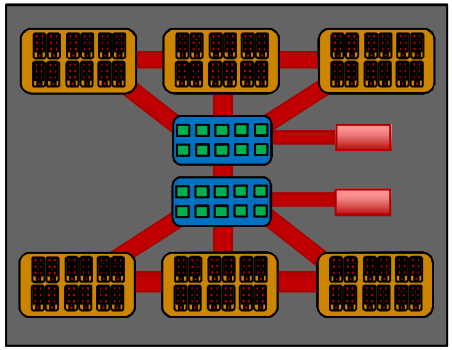
Kokkos::TeamThreadRange
Kokkos::task_spawn

NVIDIA V100



Kokkos::RangePolicy
Kokkos::TeamPolicy
Kokkos::MDRangePolicy
Kokkos::WorkGraphPolicy

Node



Kokkos::partition_master

MPI / Remote Memory Spaces

Machine

MPI / Remote Memory Spaces

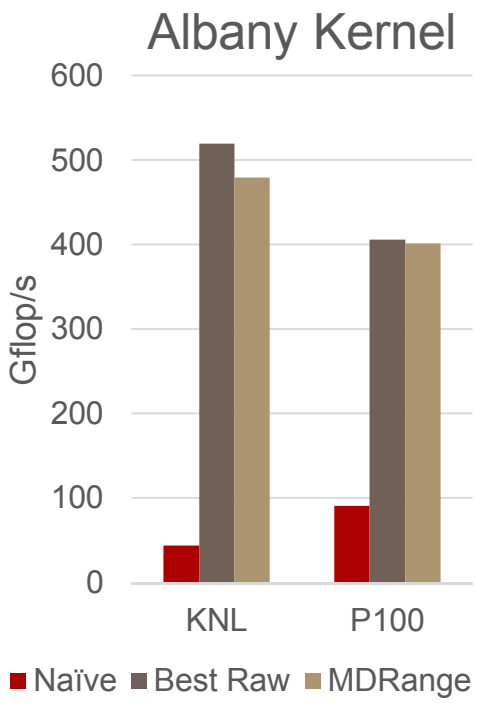
Tightly Nested Loops

- Many people perform structured grid calculations
  - Sandia's codes are predominantly unstructured though
- MDRangePolicy introduced for tightly nested loops
- Iteration Space is divided into tiles
- Tiles are assigned to SMs, inner loops to warps
- Corresponds to OpenMP collapse clause

```
void launch (int N0, int N1, [ARGS]) {
  parallel_for(MDRangePolicy<Rank<3>>({0,0,0},{N0,N1,N2}),
     KOKKOS_LAMBDA (int i0, int i1, int i2)
  {/*...*/});
}
```

- Optionally set iteration order and tiling:

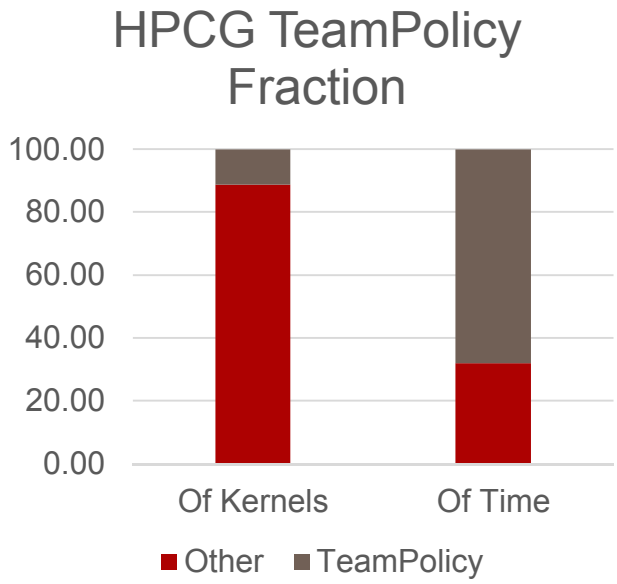
```
MDRangePolicy<Rank<3,Iterate::Right,Iterate::Left>>
  ({0,0,0},{N0,N1,N2},{T0,T1,T2})
```



Non-Tightly Nested Loops

- Often Loops are not tightly nested
- Exploit that parallelism through nested parallel_for/reduce/scan
- Exposed through Kokkos::TeamPolicy and Kokkos::host_spawn
  - Provides “team” handle to functor, which allows nested parallel patterns
- Kokkos::TeamPolicy now very commonly used, and often responsible for the dominant kernels

```
void launch (int N, int M, [ARGS]) {
  parallel_for(TeamPolicy<>(N,AUTO,8),
      KOKKOS_LAMBDA (const team_t& team) {
      ...
      parallel_reduce(TeamThreadRange(team,M),
        [&] (const j) {
      },val);
      ...
      parallel_for(TeamThreadRange(team,M),
        [&] (const j) {
      });
  });
}
```



Experimental: Process Partitioning

Sandia National Laboratories

- Subdividing a process into multiple workers common pattern
  - Uintah Task Framework runs multiple schedulers per process
  - QMCPack runs multiple workers per process, which share a big table
- Kokkos allows symetric resource splits via Kokkos::partition_master
  - Currently only implemented for OpenMP
  - Plan to support CUDA by summer 2018
- With each master thread: normal use of Kokkos parallel_for etc, which will use subset of resources.
- One possible approach to support multiple GPUs per process and exploit NVLink coherency between GPUs.

Under Development: Remote Memory Spaces

Sandia National Laboratories

- Node level interconnects are becoming more powerful
- System level interconnects become more memory bus like
  - Support higher message rate
  - Remote Atomics
- Expose this through Remote Memory Spaces
  - Goal: easy to use one sided communication through Kokkos Views
- Use Cases:
  - Implicit Halo exchange (think MPI one-sided)
  - Switch between MPI-only phase and MPI+threads (current LANL production runs)
  - PGAS programming model
- Now: Initial Development Phase, big R&D component

http://www.github.com/kokkos