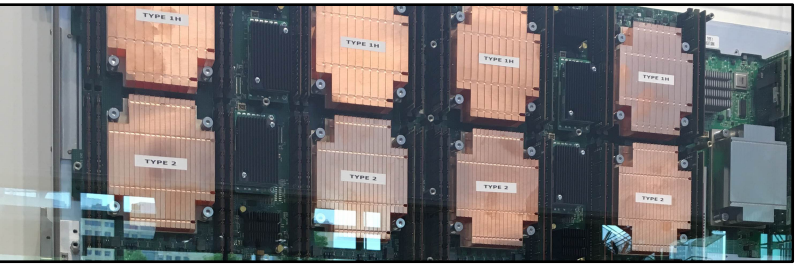
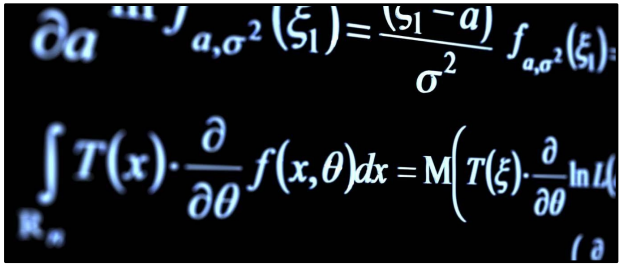
SAND2020-12413PE

Exceptional service in the national interest

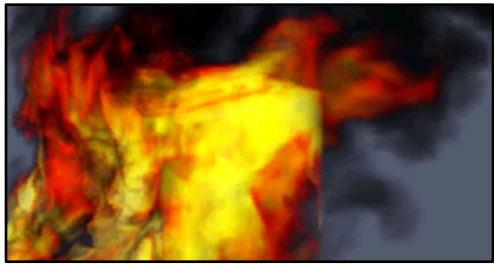
Sandia National Laboratories





$$\frac{\partial}{\partial a} f_{a,\sigma^2}(\xi_1) = \frac{(\xi_1 - a)}{\sigma^2} f_{a,\sigma^2}(\xi_1)$$

$$\int T(x) \cdot \frac{\partial}{\partial \theta} f(x,\theta) dx = \mathrm{M}\left(T(\xi) \cdot \frac{\partial}{\partial \theta} \ln L \right)$$



Kokkos COE Status Update

Unclassified Unlimited Release

Christian R. Trott, - Center for Computing Research

Sandia National Laboratories/NM

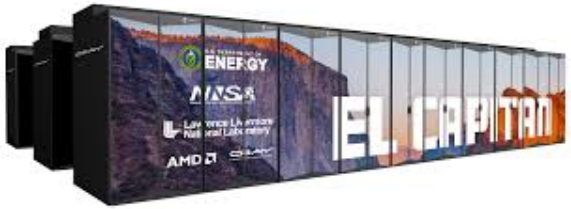
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Exascale Readiness

Sandia National Laboratories

Frontier/El Capitan: HIP and OpenMP 5

- Primary development at ORNL
- Most Capabilities ready
- PR testing for Kokkos on AMD GPUs in place
- ArborX, Cabana, LAMMPS working



Aurora: DPC++ and OpenMP 5

- DPC++ was long blocked by compiler issues
  - Now actively merging capabilities in
- OpenMP 5 a little bit behind HIP but most capabilities now working (Intel GPUs have parity with NVIDIA builds)**

Kokkos 3.3 (Nov 2020): OpenMP 5 and HIP expected to be largely feature complete
***Note: primary testing of OpenMP 5 right now with main-line clang on NVIDIA*

HIP Status

Sandia National Laboratories

- **All primary features are there**
- ROCM 3.8 required (showstopper bugs were fixed)
- Kokkos Kernels works except for Complex
  - Trying to track this down
- Missing:
  - Kokkos Task System (i.e. dynamic task graphs on the device, but VERY few users)
- Identified a few bugs we are working around in our tests but which apps would likely hit
- Will be part of Kokkos 3.3 (coming in the next two weeks)
  - But already available via develop branches on our main repos.

HIP Issues

Sandia National Laboratories

- This is as of ROCM 3.8
  - We had a lot of critical issues before 3.8 which prevented progress
- Just fresh: some regressions with ROCM 3.9 vs ROCM 3.8
  - Tracking this now.
- Race condition in shuffle operations
  - We had something similar with Clang as CUDA compiler for NVIDIA
  - But the workaround for that doesn't work in HIP
  - Workaround in HIP is more heavy handed and could potentially break some code (like special data types such as our simd library)
- A bug associated with register spilling
  - Reducing launch bounds can often resolve that, but no guarantees
- Some weird bug which makes Kokkos::finalize crash

OpenMPTarget Backend

Sandia National Laboratories

- Started work on this more than 2 years ago
  - Hindered by compiler bugs: 15 min work on backend, 6 hours work on compiler bug reproducer, 6 months wait for fix, repeat
  - With Clang 9 first time this isn't the case
- Got many capabilities now ready:
  - RangePolicy: parallel_for, parallel_reduce
  - MDRangePolicy: parallel_for, parallel_reduce
  - TeamPolicy: parallel_for, parallel_reduce (also nested)
  - Views
  - Atomic operations for up to 64 bit types
- Significant performance questions
- We are using patterns which many codes may not have exercised before
  - Will discuss in a bit

OpenMPTarget General Issues (i)

- Virtual Functions
  - Technically not supported: need to discuss with compiler vendors
  - Cuda style support would be sufficient (get the vtable from wherever you construct the object)
- TeamPolicy:
  - Scratch memory: How do we do this properly?
  - **Nested reductions use array reductions of length 1! Needs Optimization!**
  - Handling of SIMD? – In OpenMP right now inconsistent across vendors:
    - Does it do anything? Are vector lanes redundantly executing?

Kokkos

```
parallel_for("K",TeamPolicy<>(N,AUTO,8),
  [=] (team_t& team) {
    int i = team.league_rank();
    int t = team.team_rank();
    parallel_for(TeamThreadRange(team,M),
      [&] (int j) {
      parallel_for(ThreadVectorRange(team,K),
        [&] (int k) { … });
    });
});
```

OpenMP

```
#pragma omp teams distribute target
for(int i=0; i<N; i++) {
  #pragma omp parallel
  {
    int t = omp_thread_num();
    #pragma omp for
    for(int j=0; j<M; j++) {
      #pragma omp simd
      for(int k=0; k<K; k++) { ... }
}}}
```

OpenMPTarget General Issues (ii)

Sandia National Laboratories

- Arbitrary reductions
  - Stateful reducers are cumbersome (need to replace the value type with some wrapper, which contains the stateful reducer)
- Can we get the equivalent of CudaSpace, CudaUVMSpace, and CudaHostPinnedSpace?
  - Not clear that we can currently do that in OpenMP, problem for Trilinos which currently relies on page migratable memory …
- Equivalent of Stream support, or at least asynchronous dispatch?
  - omp_queue_depend_object + omp target depend?
  - Required to make really use of the Kokkos Graph (equi. to CUDA graphs)
- Arbitrary Atomic Operations
  - Compare and swap: we are using GCC intrinsics right now
  - Need to implement our own most likely
  - Not clear how to get our lock tables working which rely on a global variable …

Planned Work

Sandia National Laboratories

- Focus on LAMMPS, Trilinos and Uintah for the next couple months
  - Trilinos prerequisite for our ASC apps
  - LAMMPS generally easy to deal with code and pretty performance sensitive to a lot of capabilities in Kokkos
    - Including launch latencies, bandwidth, compute, reductions, atomics, register usage, caching, nested loops/reductions etc.
  - Uintah: early science code for Aurora which requires thread safe Kokkos
    - they split the process thread into multiple main threads
    - Each main thread dispatches Kokkos kernels independently
- Get it to run and make performance analysis
- Developing of performance test suite
  - Will include contributions of Kokkos variants to the RAJA Perf Suite
- Start working with Cray software stack


Sandia
National
Laboratories