DIRAC/NVIDIA HACKATHON

Swansea, September 2018

TEAMS

Presenting on results and lessons learned

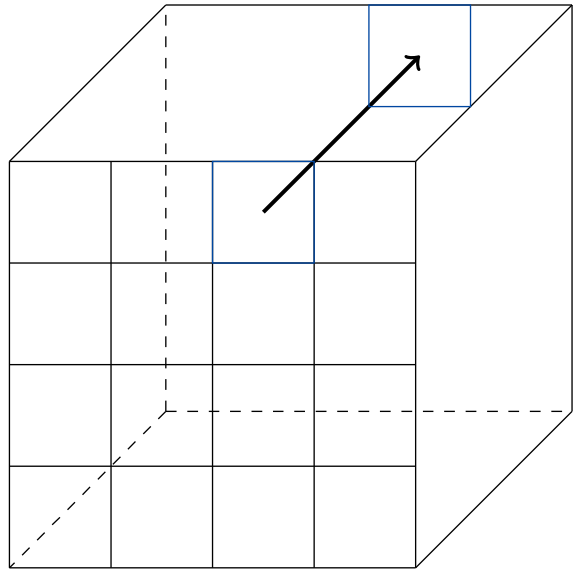
- AREGPU
- GRID
- CURSE
- TROVE
- FARGO
- The Mighty Atom

TEAM 1: AREGPU

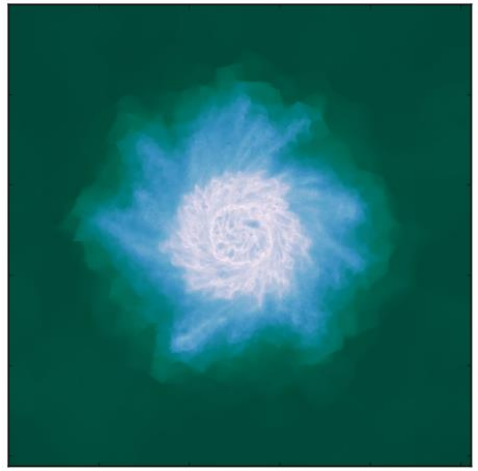
TEAM 1: AREGPU

Visualising AREPO simulations

- AREPO: moving-mesh cosmological hydro solver used in astrophysical simulations, e.g. galaxy formation
- We modify the visualisation module, which ray-traces through the simulation volume to produce projections of physical quantities.
- We parallelise this ray-tracing procedure, so that each GPU thread carries a ray.



Simulation volume



Projection

TEAM 1: AREGPU

```
int voronoi_proj_run(struct projection_data *pdata)
{
    int i=0, rays_left;

    /* rays all advance serially through simulation until all
        rays are finished */
    do
    {
        rays_left = advance_rays_one_cell(pdata);
        ++i;
    }
    while(rays_left);

    return 0
}
```

Original CPU pseudo-code

```
int voronoi_proj_run(struct projection_data *pdata)
{
#pragma acc update device(DC[0:MaxNvc])
#pragma acc kernels copy(pdata) copyin(SphP[0:All.MaxPartSph])
{
    int i;

    /* Loop over each ray; parallelised on GPU */
    #pragma acc loop independent
    for (i = 0; i < pdata->MaxNray; i++)
    {
        /* Loop over cells that ray intersects */
        while (pdata->Ray[i].len < pdata->Ray[i].target_len)
        {
            /* inline code to advance through each cell and
                update integral */
            ...
        }
    }
}
  return 0;
}
```

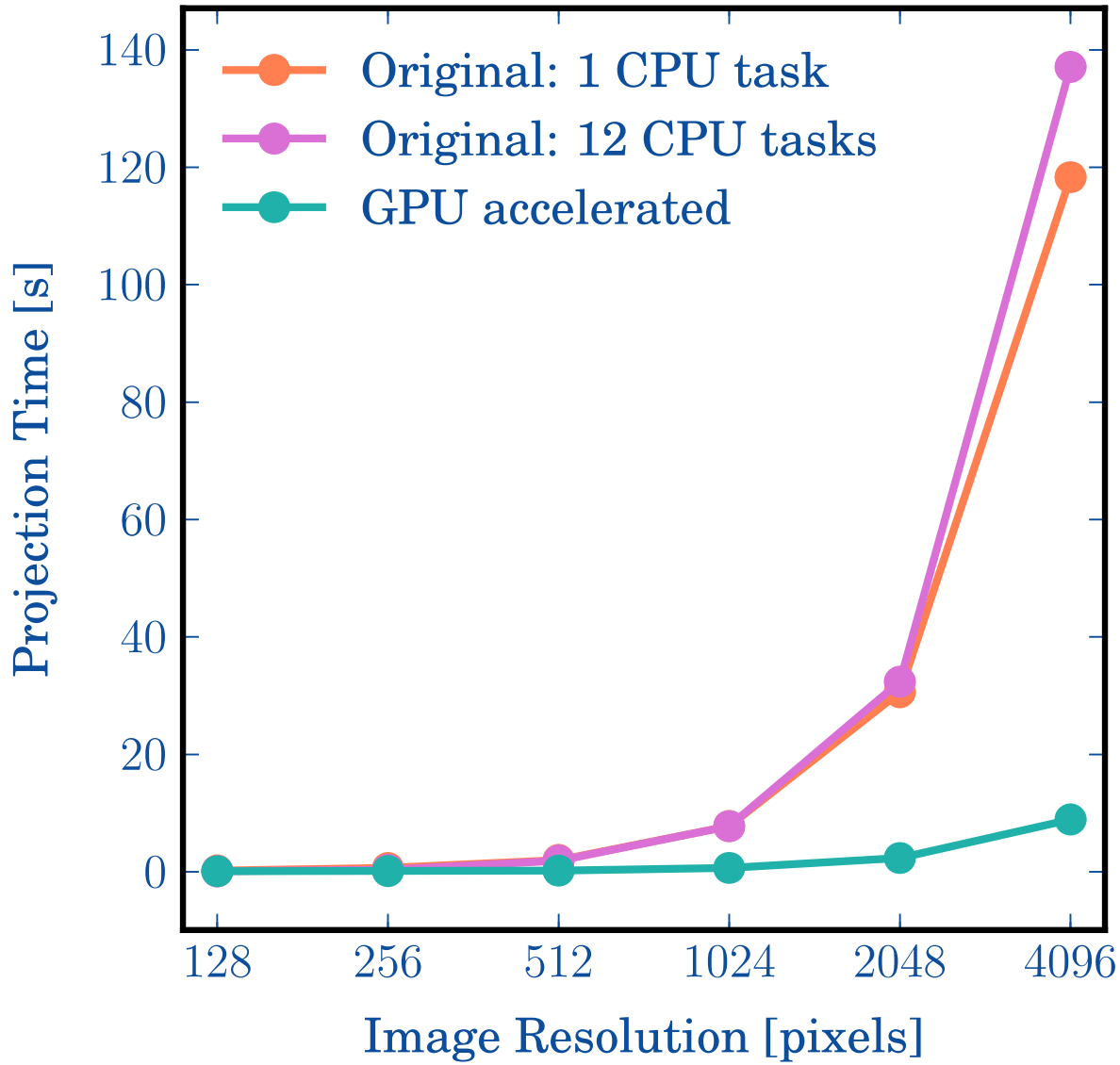
Accelerated GPU pseudo-code

DiRAC OpenACC More Science, Less Programming NVIDIA DEEP LEARNING INSTITUTE

TEAM 1: AREGPU

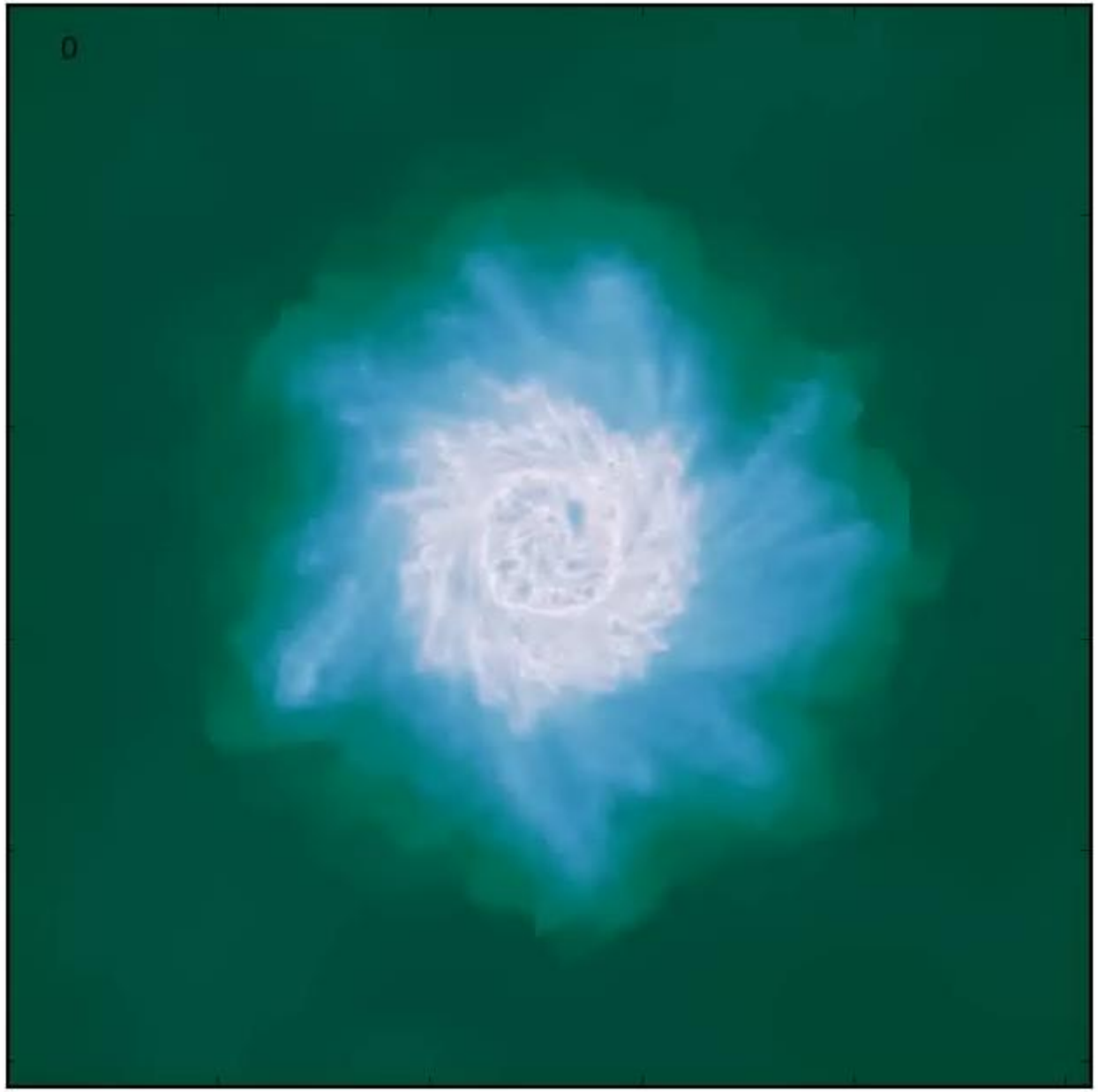
Results

- The GPU-accelerated projection routine achieves a ~10x speedup compared with the original CPU code run on one core.
- Note: the multi-core CPU run is longer than the single-core because the simulation domain decomposition is optimised for hydrodynamics and not for ray-tracing.
- The GPU code is currently incompatible with the domain decomposition, and so can only be used in post-processing. An on-the-fly implementation is left for future work.



DiRAC OpenACC More Science, Less Programming NVIDIA DEEP LEARNING INSTITUTE

The accelerated code can produce animated visualisations by rapidly ray-tracing through different angles.



TEAM 2: GRID

Presenting on results and lessons learned

- Starting point
- Multi-GPU peer-to-peer
- Deterministic GPU reductions

STARTING POINT

STARTING POINT

Results

- Grid Lattice QCD code
- https://www.github.com/paboyle/Grid
- UKQCD and USQCD plan of record for Exascale performance portability
- Runs on
    - SSE4, AVX, AVX2, AVX512, ARM Neon, IBM QPX
    - Intel Xeon, Intel Knights, AMD Zen, ARM ThunderX2, BlueGene/Q
- Work in progress (with DOE USQCD Exascale Computing Project)
    - *feature/gpu-port*
- Targets ORNL Summit (fastest computer in world; 180 PF/s; Power 9 + 6x Nvidia Volta V100)

SUBTEAM 1: STARTING POINT

Starting point (after 6 months prior work: single GPU functioning)

- Compare to QUDA library on **single** V100
-

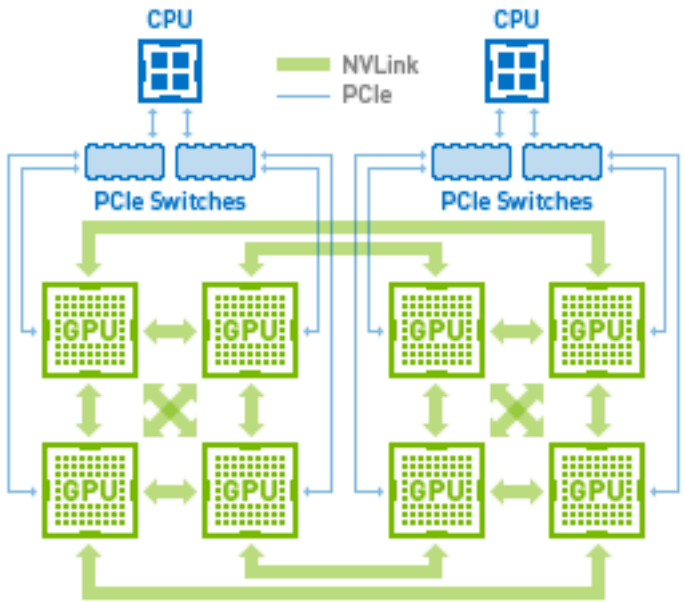
Volume	Grid	QUDA
12^4	1.4TF/s	1.7TF/s
24^4	1.9TF/s	2.3TF/s

GOAL: MULTI GPU PER NODE USING NVLINK

SUBTEAM 1: PEER 2 PEER

Results

- Use one MPI rank per GPU but use GPU-GPU direct peer to peer access for intra-node communication.
- Implemented faces exchange to neighbour GPU
  - Similar to use of Unix shared memory on two socket nodes
- No new functions in code; implementation change in existing func
  - GPU direct RDMA should already work too for free
  - i.e. never bounce through host memory



- **Runs across PCIe on Cambridge 4 GPU nodes (no Nvlink, not perf representative)**
- Runs on DGX-1 (8 Tesla V100) in Nvidia HQ
  - **6.4 TF/s on DGX/1** ; still need to coalesce accesses to accelerate halo - exchg

DiRAC OpenACC NVIDIA DEEP LEARNING INSTITUTE

SUBTEAM 1: PEER 2 PEER

Lessons learned

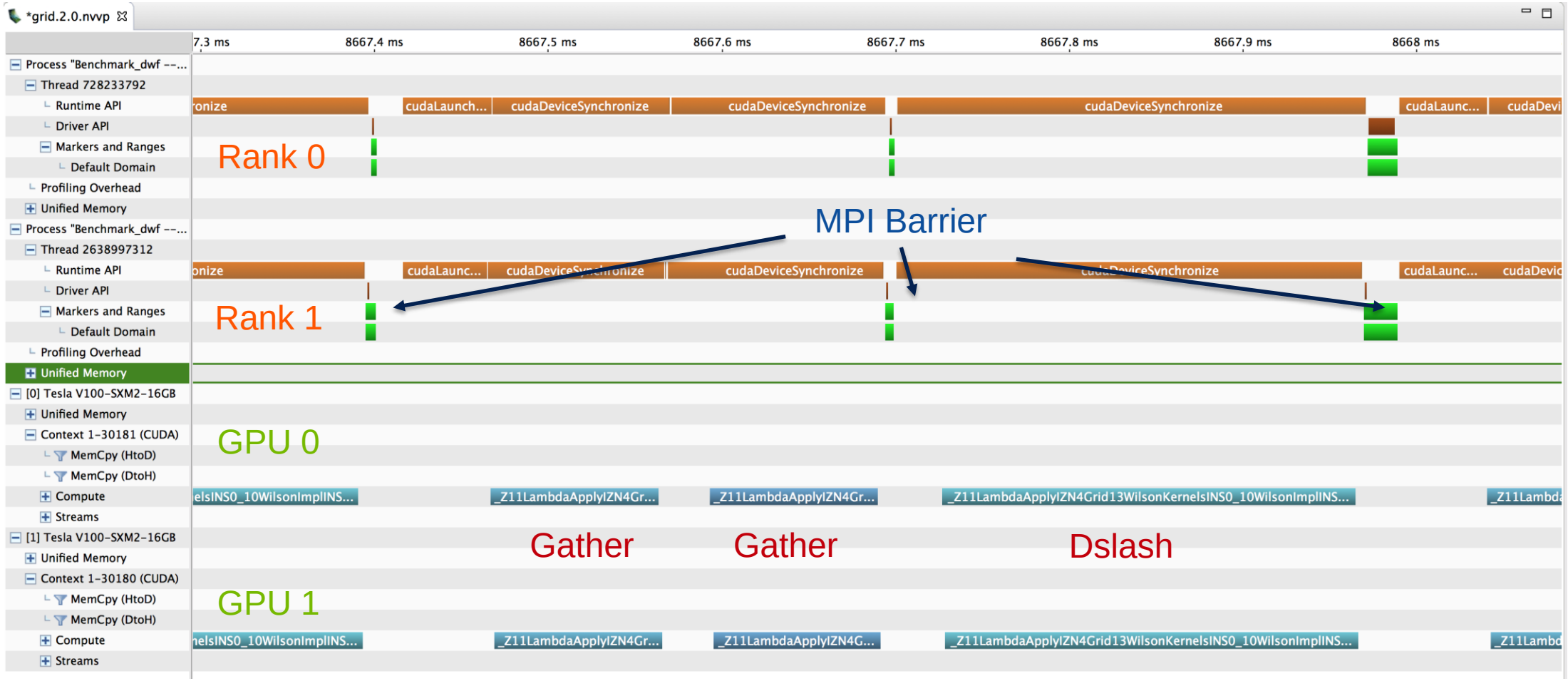
- Met expectations.
  - 5 hours getting modules & compiler versions on Cambridge working
  - **2h modifying the code and debug**
  - 1 day chasing an unrelated memory issue

```
// Each MPI rank should allocate our own device buffer
cudaMalloc(&ShmCommBuf, bytes);


// Loop over mpi ranks/gpu's on our node — 1:1 mapping assumed
for(int r=0;r<WorldShmSize;r++){


cudaIpcMemHandle_t handle;  void * thisBuf;


// If “rank” is me, pass around the IPC access key
if ( r==WorldShmRank ) cudaIpcGetMemHandle(&handle,ShmCommBuf);


// Share this IPC handle across the Shm Comm
MPI_Bcast(&handle,sizeof(handle),MPI_BYTE,r,WorldShmComm);


// If I am not the rank, overwrite thisBuf with remote buffer
if ( r!=WorldShmRank ) cudaIpcOpenMemHandle(&thisBuf,handle,cudaIpcMemLazyEnablePeerAccess);
else thisBuf = ShmCommBuf;


// Save a copy of the peer device buffer for this rank; every GPU can access these pointers
WorldShmCommBufs[r] = thisBuf;


}
```

DiRAC OpenACC More Science, Less Programming NVIDIA DEEP LEARNING INSTITUTE

SUBTEAM 1: PEER 2 PEER

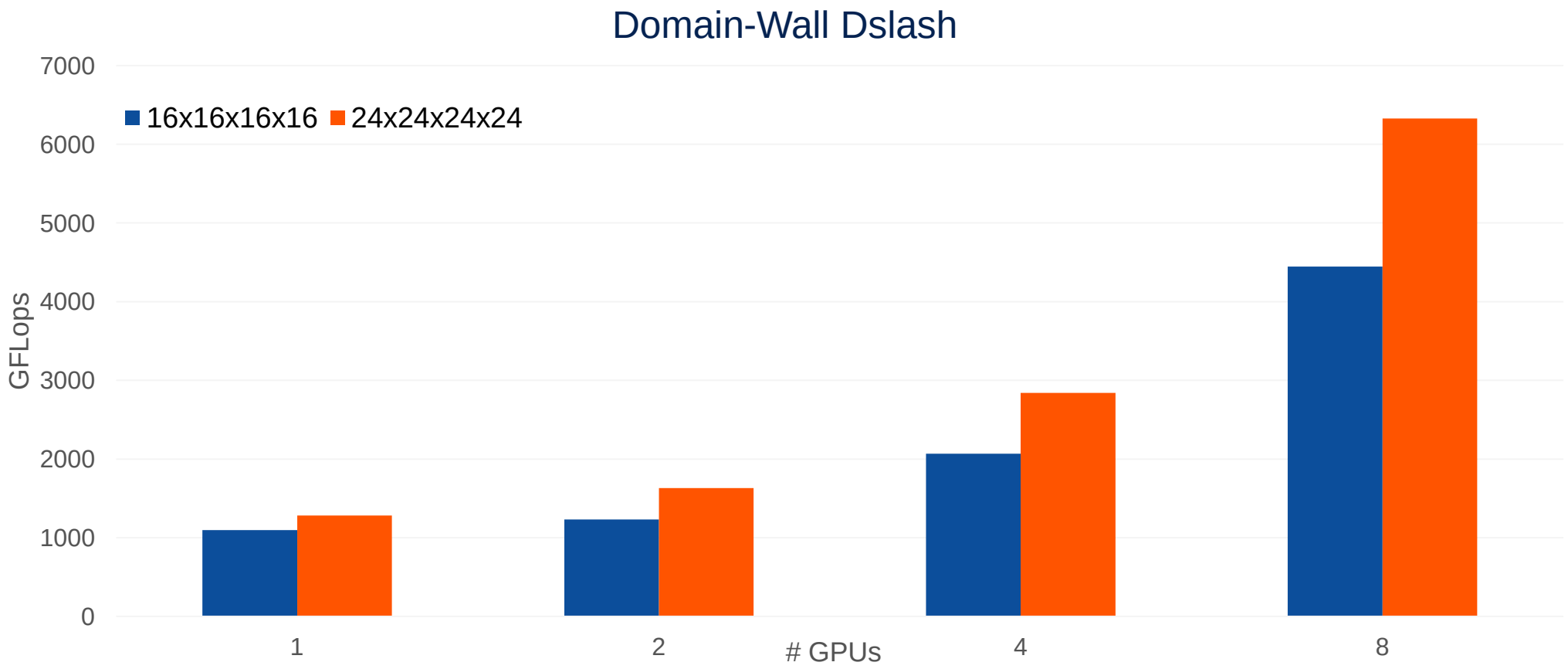
Timeline



DiRAC OpenACC More Science, Less Programming NVIDIA DEEP LEARNING INSTITUTE

SUBTEAM 1: PEER 2 PEER

Initial weak scaling on DGX-1



DiRAC OpenACC More Science, Less Programming NVIDIA DEEP LEARNING INSTITUTE

GPU REDUCTIONS

SUBTEAM 1: REDUCTION

Results

- Grid QCD code
- Implement summation across GPU threads (formerly host only)
- Looked at Nvidia thrust reductions
  - Assessed whether these were reproducible
  - Implement first cut in lib/lattice/Lattice_reduction.h
- Prerequisite for entire Dirac krylov solver running on GPU's

TEAM 4: TROVE

TEAM 4: TROVE

Results

- Problem requires over 50% of eigenvectors of a dense matrix with N>300k (ideally N up to 1M)
- Many eigenvalue solvers struggle with dense matrices larger than N>200,000. In addition. A combination of both MPI and GPUs may satisfy large memory requirements whilst accelerating performance.
- Distributed routines (ScaLAPACK) identified and profiled. Calls to the underlying serial BLAS routines ported cuBLAS to test resulting GPU acceleration.
- Incorrect results due to unresolved bugs in some calls to cuBLAS.

TEAM 4: TROVE

Lessons learned

- Varying previous GPU experience in the team – sharing experience is important!
- Problem intended primarily as learning experience.
- Increased confidence working with ScaLAPACK, BLAS, cuBLAS and CUDA source
- Pair-programming allowed unfamiliar libraries to be understood quickly.
- Reducing iteration time of build and testing process was crucial. Build setup and workflows will be useful for future code development.
- Use reservations to avoid queues!

TEAM 5: FARGO

TEAM 5: FARGO

Results

- Dusty fargo: A code to simulate the growth and dynamics of dust grains in protoplanetary discs.
- Aims: Port the grain growth module to the GPU (the rest of the code already runs on the GPU)
- Got the code running successfully on the GPU, giving correct results.
- 3× speedup, still room to optimise memory management

TEAM 5: FARGO

Lessons learned

- Managing memory with OpenACC surprisingly straightforward
- A lot of time spent interfacing with existing libraries
- Should have started by writing tests, defining correctness
- Isolating the problem very useful

TEAM 6: THE MIGHTY ATOM (UKAEA)

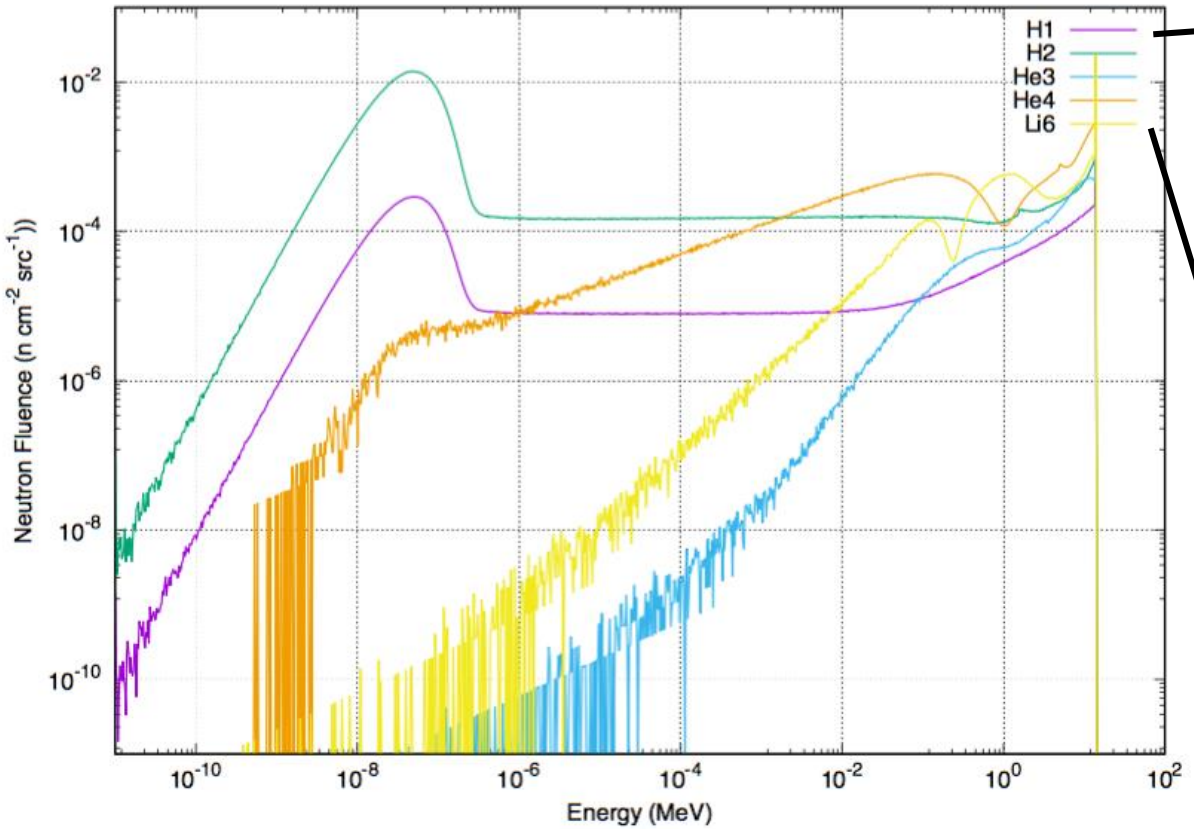
TEAM 6: THE MIGHTY ATOM

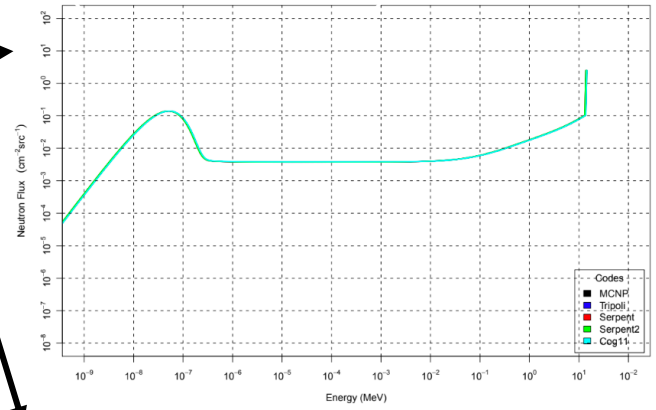
Results

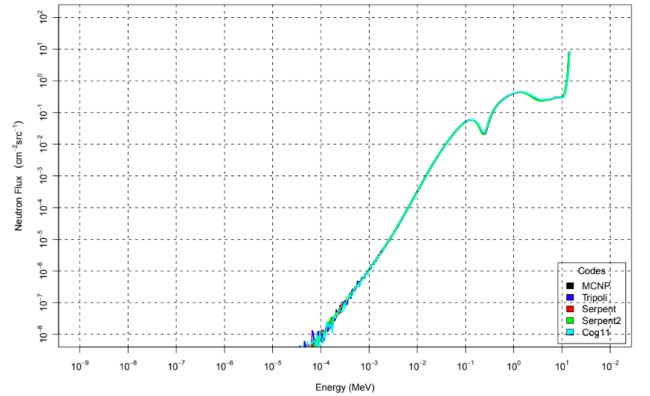
- Warp - GPU Based Neutron Transport code from UC Berkley
- Code was produced by PhD student around 2-3 years ago and has suffered significant bit rot since
- Code compiled and run - with dependencies CUDA 8.0, CUDPP 2.3, OptiX 5.0.1
- Can successfully (and robustly) run fission problems with modern nuclear data, code changes made for fusion problems (particles terminated correctly)
- Code can run up to 1,000,000,000 particles but interaction cross sections take memory

TEAM 6: THE MIGHTY ATOM

Results







DiRAC OpenACC More Science, Less Programming NVIDIA DEEP LEARNING INSTITUTE

TEAM 6: THE MIGHTY ATOM

Lessons learned

- Progress was slow
- Not atypical with bug finding in existing large codebase
- Next stages to integrate mesh based geometry into it to see how OptiX works with mesh geometry and rationalise the handling of materials

THANK YOU

DEEP
LEARNING
INSTITUTE