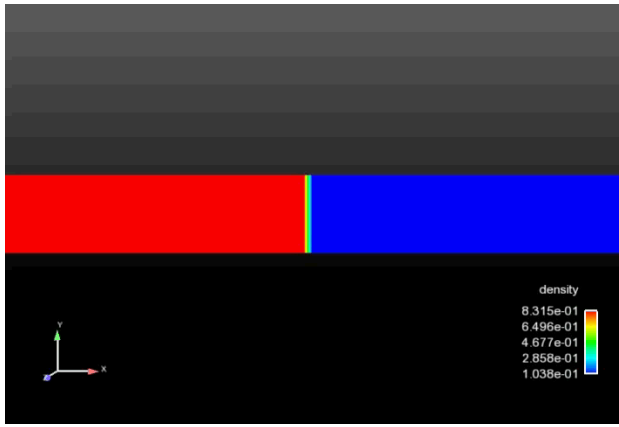
SAND2014-1040P

Exceptional service in the national interest

Sandia National Laboratories

miniAero:MPI+X with Kokkos



Ken Franko

U.S. DEPARTMENT OF ENERGY | NNSA

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Outline

- Motivation/Challenges
- Mathematical Problem/Numerical Method
- Code Design
- Kokkos Example
- Performance
- Conclusions

Relevant Exascale Challenges

Sandia National Laboratories

- CPU power/memory ratio increasing
- Increased cost of communication
- Heterogeneous Processing
- New devices (Intel Phi and GPUs)
- Uncertainty of future computer architectures

See:
The Opportunities and Challenges of Exascale Computing. ASCAC Report

Goals

Sandia National Laboratories

- Generate a miniapp representing some basic physics of Aero problems of interest to Sandia
- Evaluate the use of Kokkos as X in MPI+X
- Measure performance on different architectures including GPU. (MIC in future)

Equations

Sandia National Laboratories

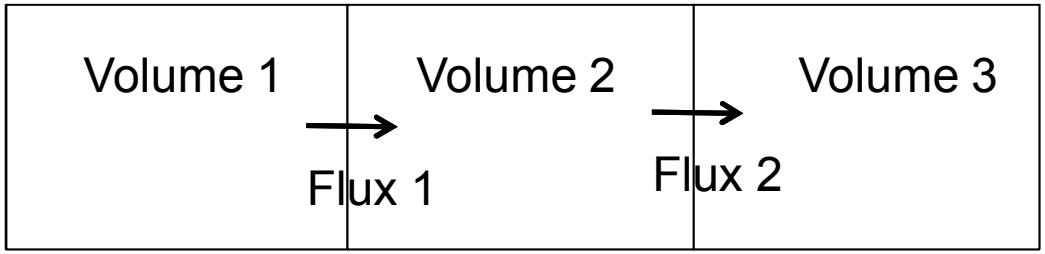
$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0$$ Mass

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_i u_j + P\delta_{ij}\right) = \frac{\partial \tau_{ij}}{\partial x_j}$$ Momentum

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u_j H}{\partial x_j} = -\frac{\partial q_j}{\partial x_j} + \frac{\partial u_i \tau_{ij}}{\partial x_j}$$ Energy

Solution Method

Finite Volume:



- Cell-centered
- 1st order in space
- Flux boundary conditions

Explicit RK4 Time Marching

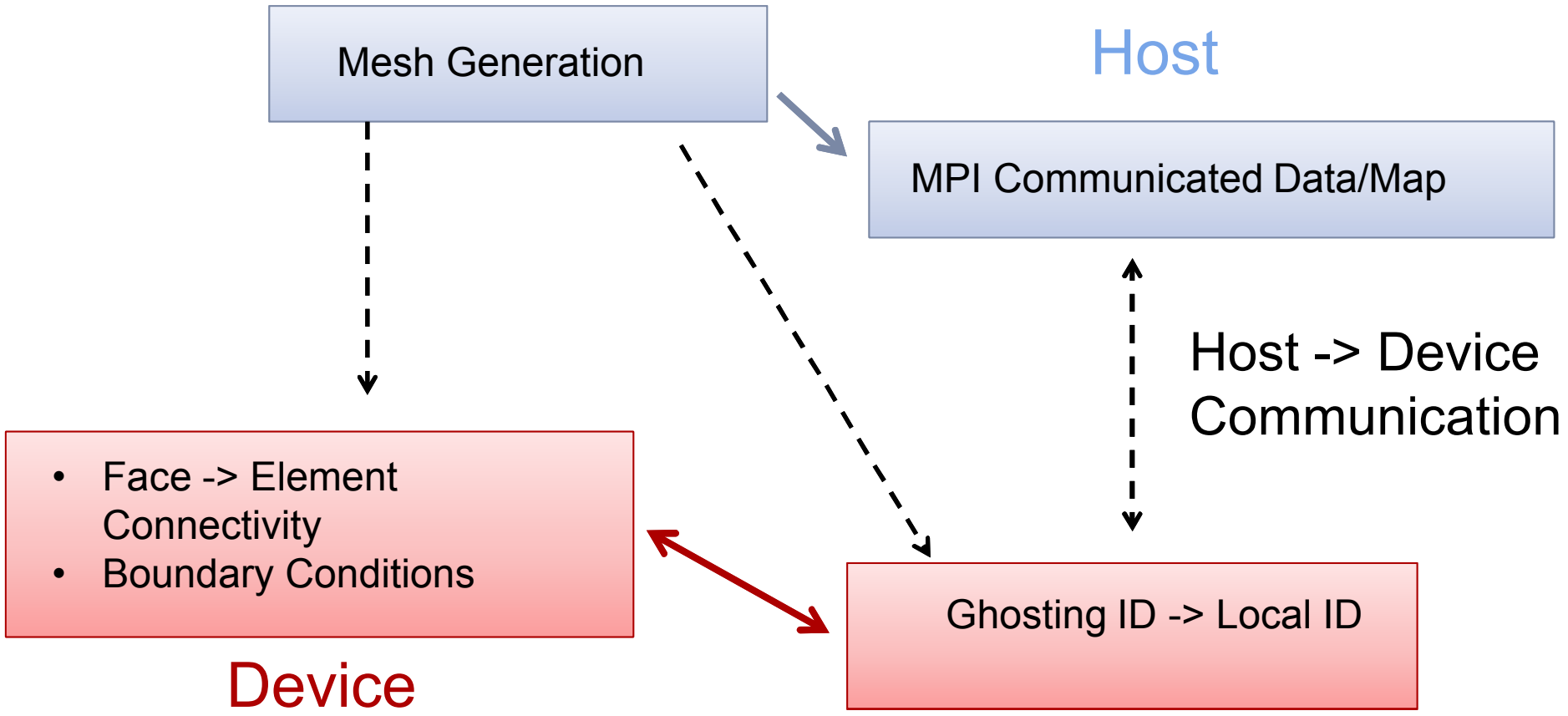
$$\frac{d\boldsymbol{U}}{dt} = \boldsymbol{R}(t, \boldsymbol{U}(t)) \quad \boldsymbol{U}(t_o) = \boldsymbol{U}_o$$

Numerics Summary

Sandia National Laboratories

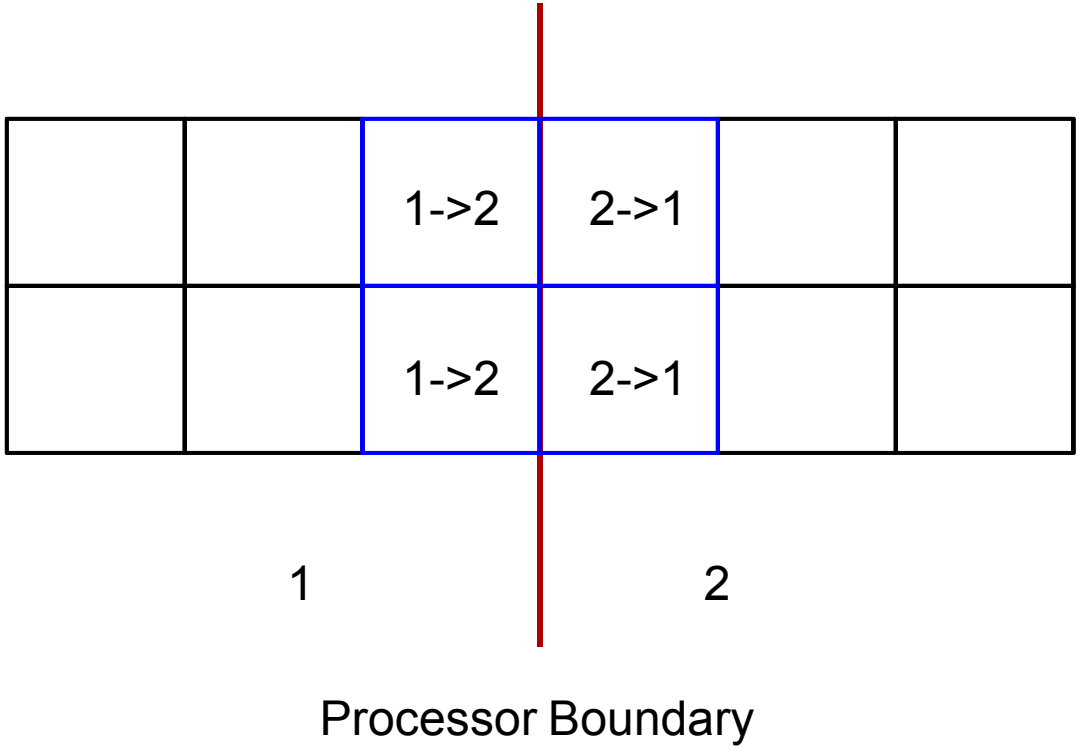
- Fully 3D unstructured finite volume
- Runge-Kutta 4^{th} order time marching
- Inviscid Roe Flux
- BCs: Supersonic inflow, supersonic outflow, and tangent flow

Program Design



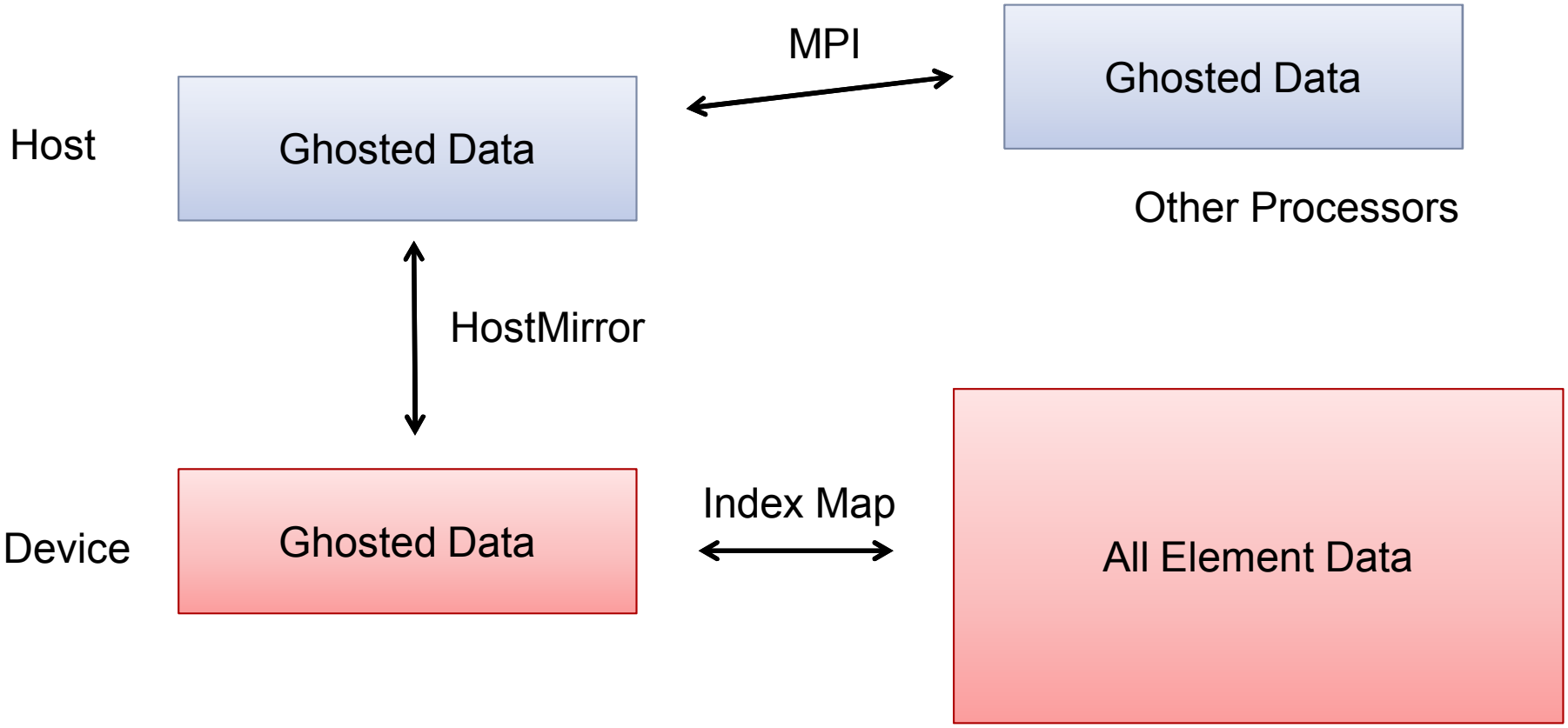
Simple MPI Implementation

- Decompose in a single direction and use ghosting.



Simple MPI Implementation(2)

- Ghosting data needs to be copied to/from device in order to send/receive over MPI.



Kokkos – Data storage

Sandia National Laboratories

Declaration:

*Kokkos::View<double *[5], Device> residual("residual", ncells);*

Access:

residual(15, 0)

miniAero examples:

*face_cell_conn_*

*coordinates_*

*volumes_*

Kokkos - Functor

Sandia National Laboratories

Constructor:

```
template <class Device>
struct initialize_constant{

  typedef Device     device_type;
  typedef Kokkos::View<double *[5], device_type >  flow_var;

  struct Cells<Device> cells_;
  flow_var soln_,solnp1_;
  double flow_state_[5];

  initialize_constant(struct Cells<Device> cells, flow_var soln, flow_var solnp1, double *
flow_state) :
    cells_(cells),
    soln_(soln),
    solnp1_(solnp1)
  {
    for(int i=0; i<5; ++i)
      flow_state_[i]=flow_state[i];
  }
```

Kokkos – Functor(2)

Sandia National Laboratories

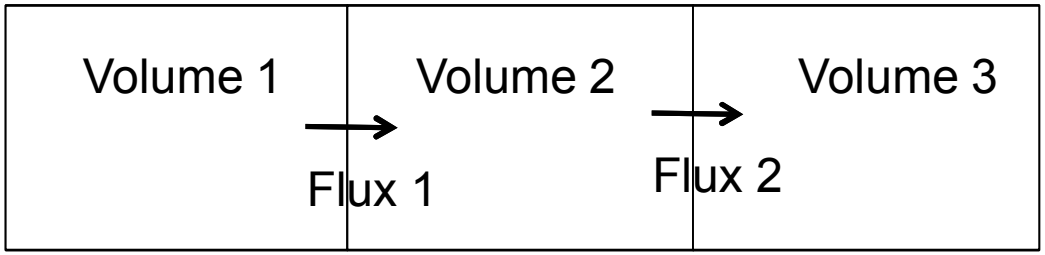
Actual Calculation:

```
KOKKOS_INLINE_FUNCTION
 void operator()( int i )const{
   for(int j=0; j<5; j++)
     soln_(i,j)=flow_state_[j];
   for(int icomp=0; icomp<5; icomp++){
     solnp1_(i,icomp)=soln_(i, icomp);
   }
  }
};
```

Construction and Dispatch:

```
initialize_constant<Device> init_fields(cells, sol_n_vec, sol_np1_vec, &inflow_state[0]);
parallel_for(nowned_cells, init_fields);
```

Assembly – Thread safety



Options

1. Store fluxes at Faces. Two loops – over faces and then over volumes. Downside: Performance and need to store flux direction for each volume.
2. **Stores fluxes for each face on volume. Two loops – over faces and then over volumes.**
   Downside: Increased memory use
3. **Atomic operations. Single loop over faces and no additional memory required.**
   Downside: Could be slow with large number of conflicts.

Summary of Code

Sandia National Laboratories

- C++ - lines of code - 3370
- Minimal dependencies – only on Kokkos.
- Mesh and data structures generated on host then moved to Kokkos Views.
- Physics kernels are functors -> flexible
- Use of templates for device and algorithm choices.

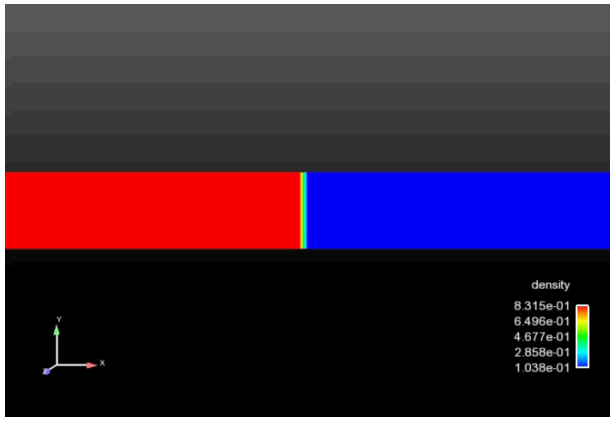
Performance – Node Parallel

Sandia National Laboratories

- Pre-MPI implementation
- Run on Curie
  - 16-core 2.1GHz 64bit AMD Opteron 6200 CPU's
  - NVIDIA Kepler - Tesla K20X

3D Shock Tube Problem

Sandia National Laboratories



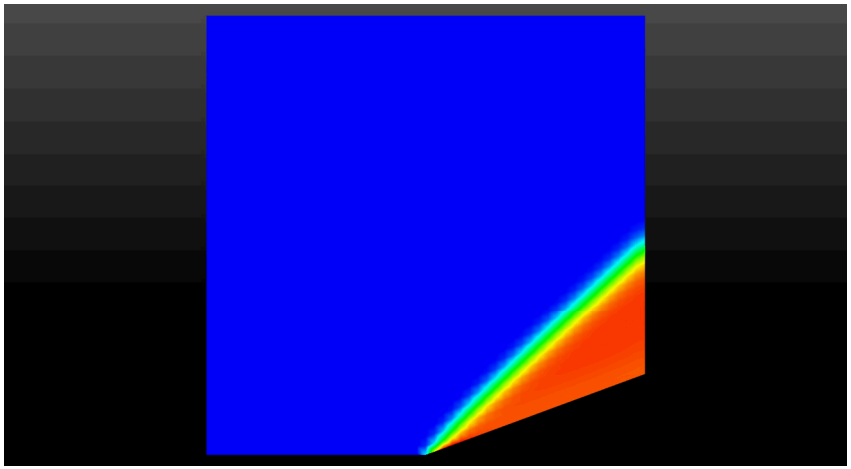
nx = 100, ny = 20, nz = 20, 40000 elements.
500 timesteps

Performance:

	Cell Storage		Atomics	
	Time	Speedup	Time	Speedup
CPU – 1 thread	203s	1x	218s	1x
CPU – 4 threads	91s	2.2x	126s	3.5x
CPU – 8 threads	64s	3.2x	33s	6.6x
CPU – 16 threads	44s	4.6x	16s	13.6x
GPU	7s	29x (6.3x)	6s	36x (2.7x)

Timings on Curie (16 core AMD and NVIDIA Kepler - Tesla K20X)

3D Ramp



nx = 50, ny = 50, nz = 25
62500 elements
500 timesteps
192500 faces
182500 edges

Performance:

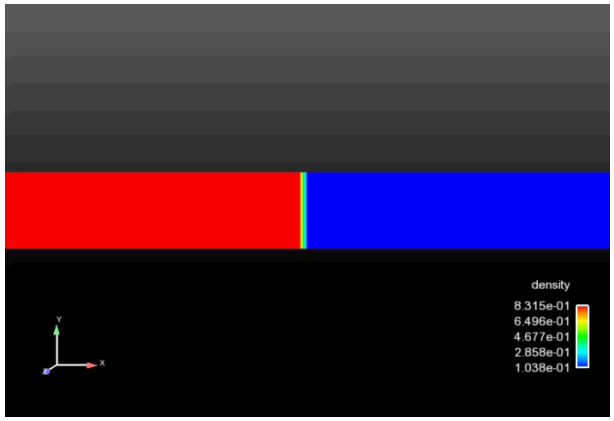
	miniAero	Conchas
Blade 1-CPU	221s	397.0s
Blade 8-CPU	33s ↓ ~6.7x	63.0s ↓ ~6.5x
Curie GPU	9s	

GPU is ~45x faster than current single core Conchas

Performance - MPI+X

- Mainly to test MPI implementation
- Run on Curie
  - 16-core 2.1GHz 64bit AMD Opteron 6200 CPU's
  - NVIDIA Kepler - Tesla K20X
- Run on Chama
  - Dual socket 2.6 GHz Intel Sandy Bridge

MPI+X (GPU)



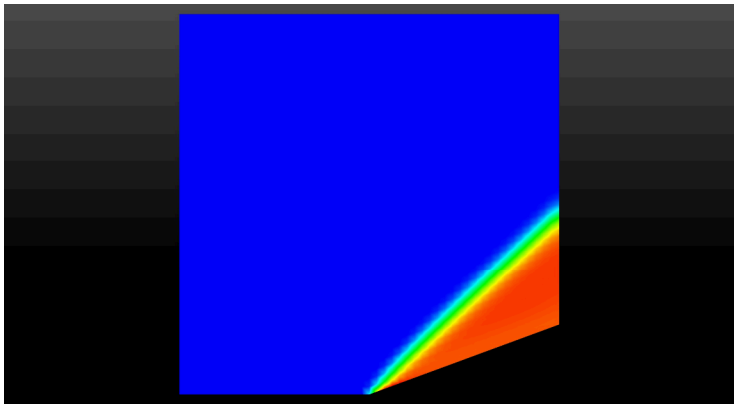
nx = 1000, ny = 64, nz = 64,
~4 million elements.
5000 explicit RK4 timesteps
Decomposed in x direction
Hybrid MPI/GPU

Performance:

	Setup		Run	
Nodes (MPI Ranks and GPU cards)	Time	Speedup	Time	Speedup
1	107s	1x	2723s	1x
2	83s	1.3x	1455s	1.9x
4	50s	2.1x	708s	3.8x
8	35s	3x	369s	7.4x

All timings on Curie (16 core AMD and NVIDIA Kepler - Tesla K20X)

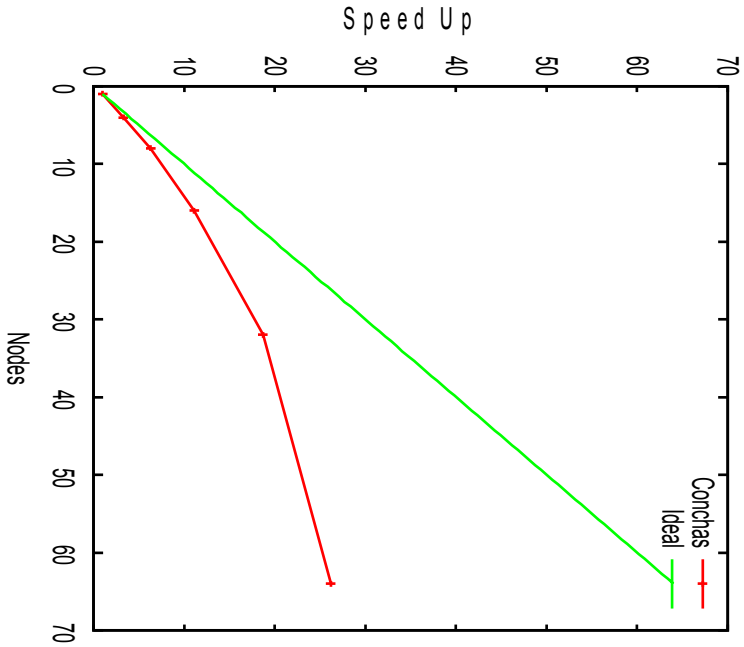
3D Ramp Problem - Conchas



nx = 1024, ny = 64, nz = 64,
~4 million elements.
500 explicit RK4 timesteps
Decomposed in x direction

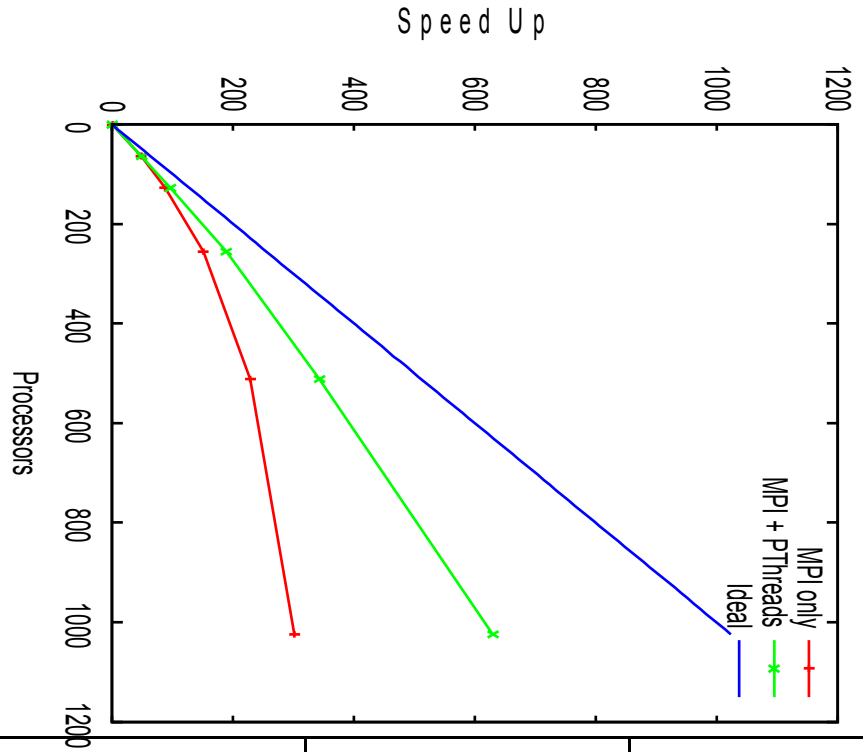
STK-based Conchas:

Nodes	MPI Ranks	Runtime(s)	Speedup
1	16	2042	1x
4	64	619	3.3x
8	128	324	6.3x
16	256	183	11.1x
32	512	109	18.7x
64	1024	78	26.2x



All timings on Chama(16 core dual-socket 2.6 GHz Intel Sandy Bridge)

MiniAero – MPI+X(Pthreads)



Nodes	MPI Ranks	Threads/Rank	Runtime(s)	Speedup
1	1	1	7563	1x
64	1024	1	25	302.0x
64	64	16	12	630.2x

Summary

Sandia National Laboratories

- Kokkos – Promising for programming on heterogenous architectures
- Thread safety required
- Finite volume method (explicit) amenable to threading, both CPU and GPU
- GPU speedups can be huge
- MPI everywhere breaks down

Future Plans

Sandia National Laboratories

- Further performance evaluation (strong/weak scaling, GPU performance, threading performance)
- Additional Hardware Porting and Testing (MIC, Blue Gene Q, Titan)
- Smarter mesh decomposition and ghosting
- Simple Linear Solver – Point Implicit Solver
- Additional Physics (viscous, LES, second-order space)

Acknowledgements

- Micah Howard – SPARC
- Carter Edwards, Dan Sunderland and Christian Trott – Kokkos Team
- Co-design project
- Rob Hoekstra and Ryan Bond
- Pat Notz

Questions

Sandia National Laboratories