PROGRESS TOWARDS ACCELERATING CAM-SE ON HYBRID MULTI-CORE SYSTEMS

Presented by: Rick Archibald

Climate and Earth System Modeling PI Meeting
Model Interoperability

September 19-22, 2011
Grand Hyatt, Washington, DC

Team Members and Resources

Core Readiness Team	Location
Rick Archibald	ORNL
Jeff Larkin	Cray
Ilene Carpenter	NREL
Paulius Micikevicius	NVIDIA
Matthew Norman	ORNL
Valentine Anantharaj	ORNL

Climate Change SCIENCE INSTITUTE

Center for Accelerated Application Research(CAAR)

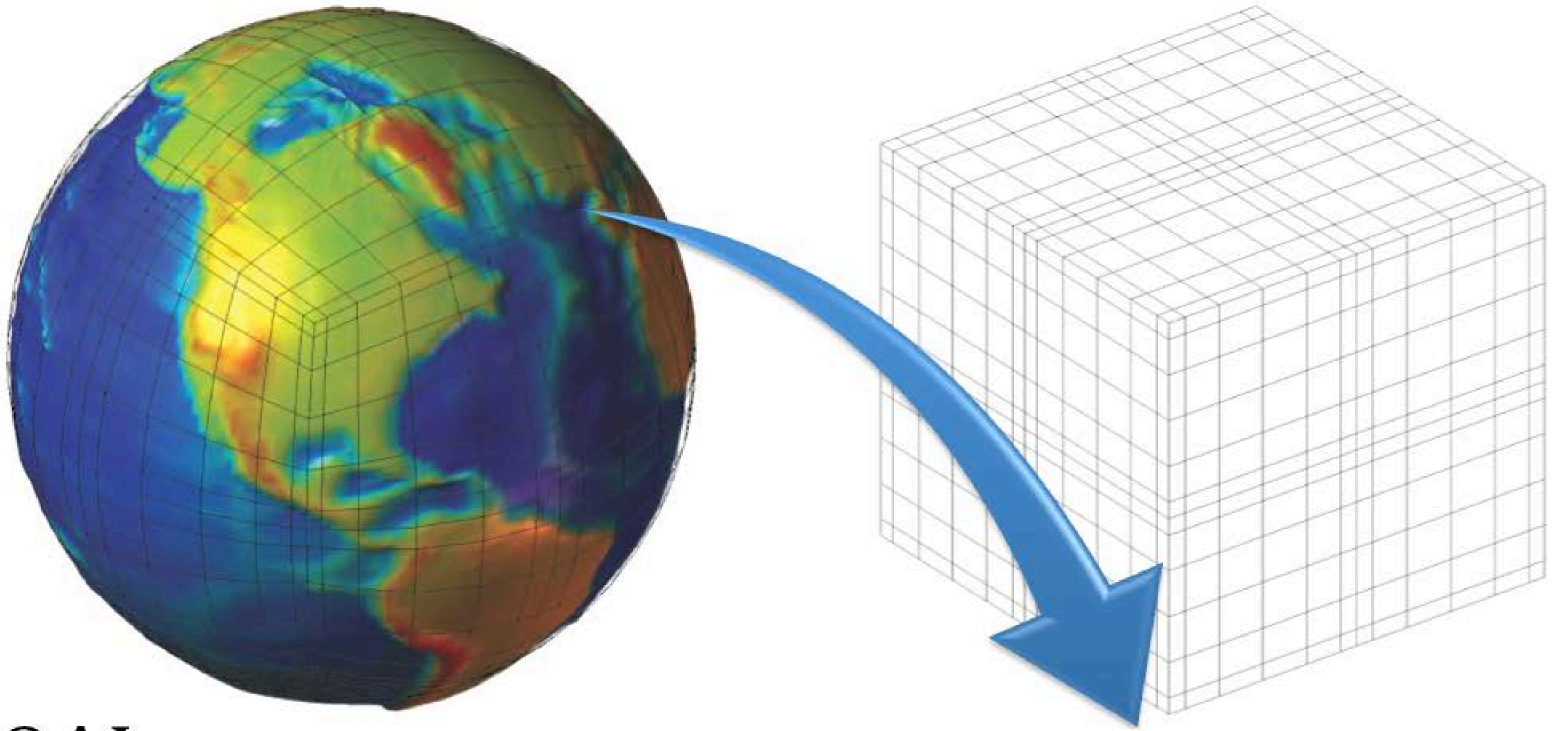
OLCF
OAK RIDGE LEADERSHIP COMPUTING FACILITY

CRAY
THE SUPERCOMPUTER COMPANY

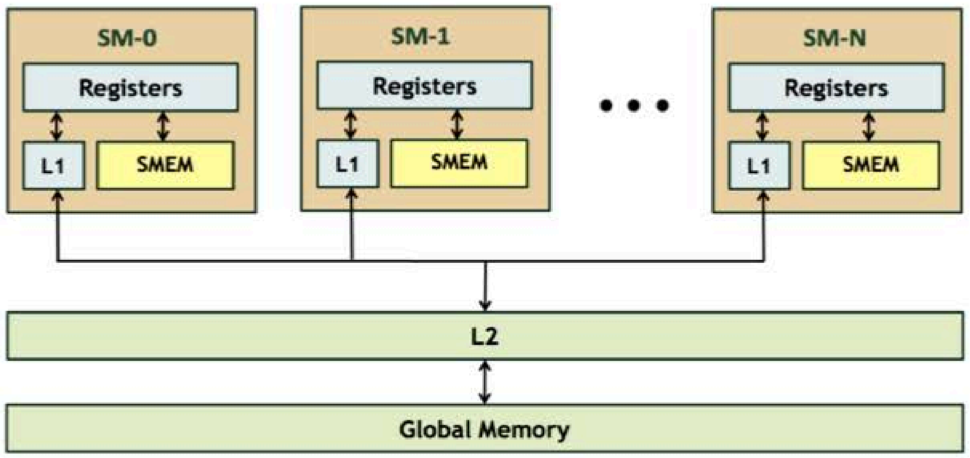
NVIDIA

Why Hybrid Systems

- Multi-threaded architecture is the current standard in computing from multi-core CPUs to the relatively massive threading of GPUs.
- GPUs have evolved to augment CPU capacity. Hybrid systems take advantage of this synergy by having duel socket nodes with both a multi-core CPU and GPU.
- Beneficial GPU properties:
  - Leveraged Commodity
  - Reliability at scale
  - High flop/watt
- Challenging GPU properties:
  - Complex memory hierarchy
  - Massive parallelism



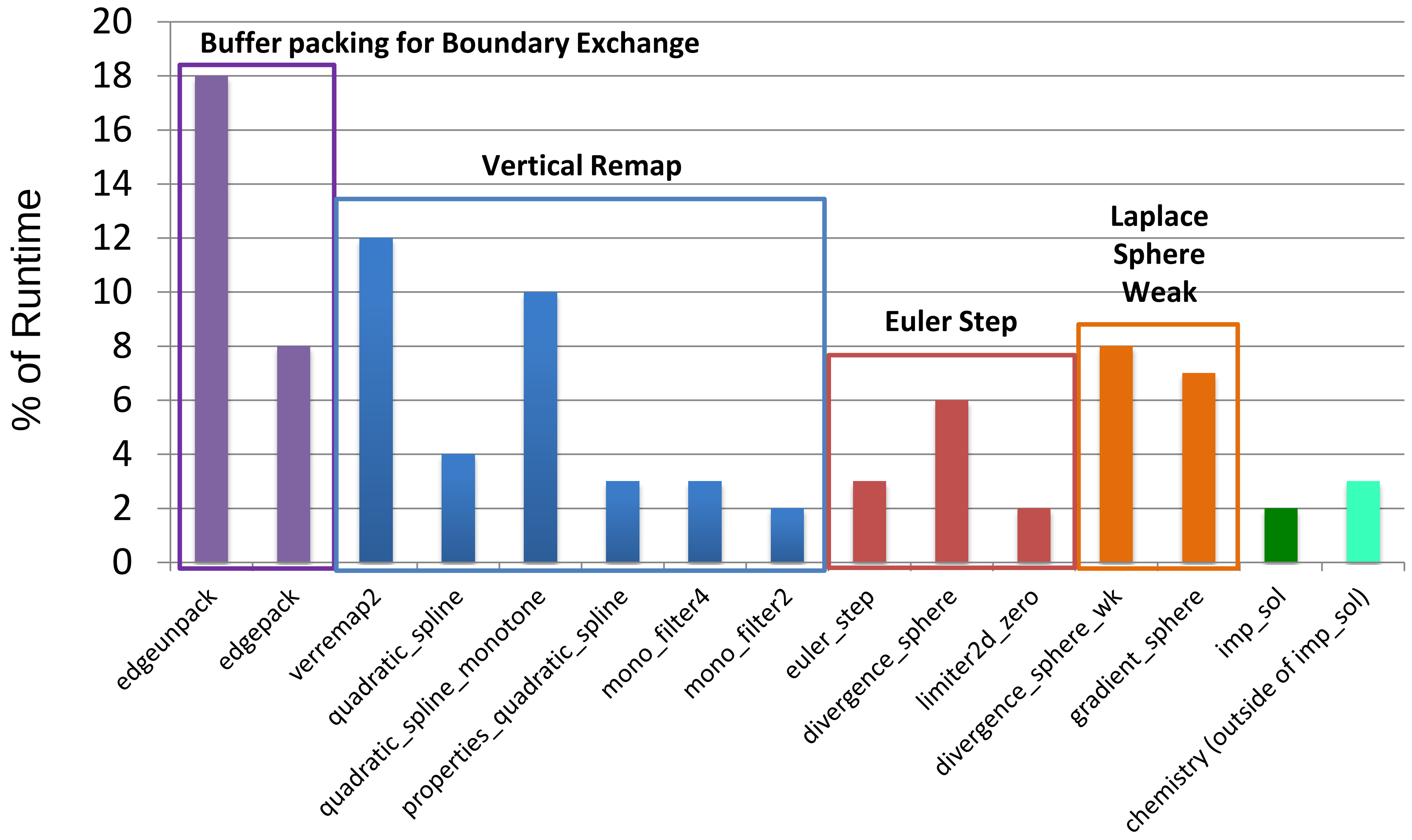
GOAL:
Accelerating
Scalable
Sustainable
Climate Dynamics



Target Problem

- 1/8 degree CAM, using CAM-SE dynamical core and Mozart tropospheric chemistry. Land model will be run on CPUs.
- Why is acceleration needed to “do” the problem?
  - When including all the tracers associated with Mozart atmospheric chemistry, the simulation is too expensive to run at high resolution on today’s systems.
- What unrealized parallelism needs to be exposed?
  - In many parts of the dynamics, parallelism needs to include levels and chemical constituents.

Projected Profile of Runtime



Kernels extracted

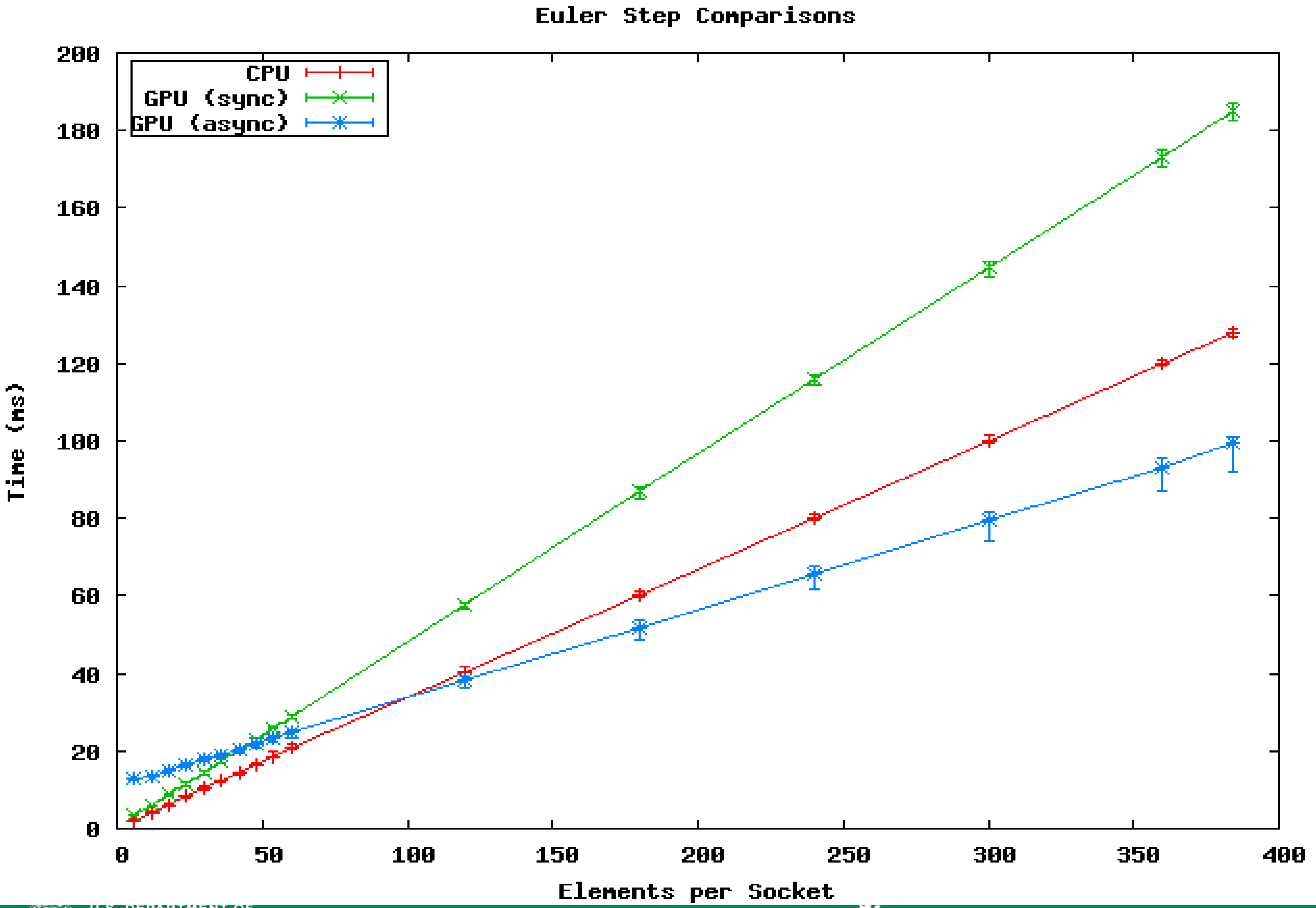
- Vertical remapping
- Euler_step + limiter2d_zero
- Laplace_sphere_wk
- Small kernels from dynamics (gradient, divergence, vorticity etc.), used by euler_step and laplace_sphere_wk
- Tendency physics before coupling (for compiler analysis, not run-able).
- Chemistry implicit solver (smaller than expected % of final runtime, further development put on hold)

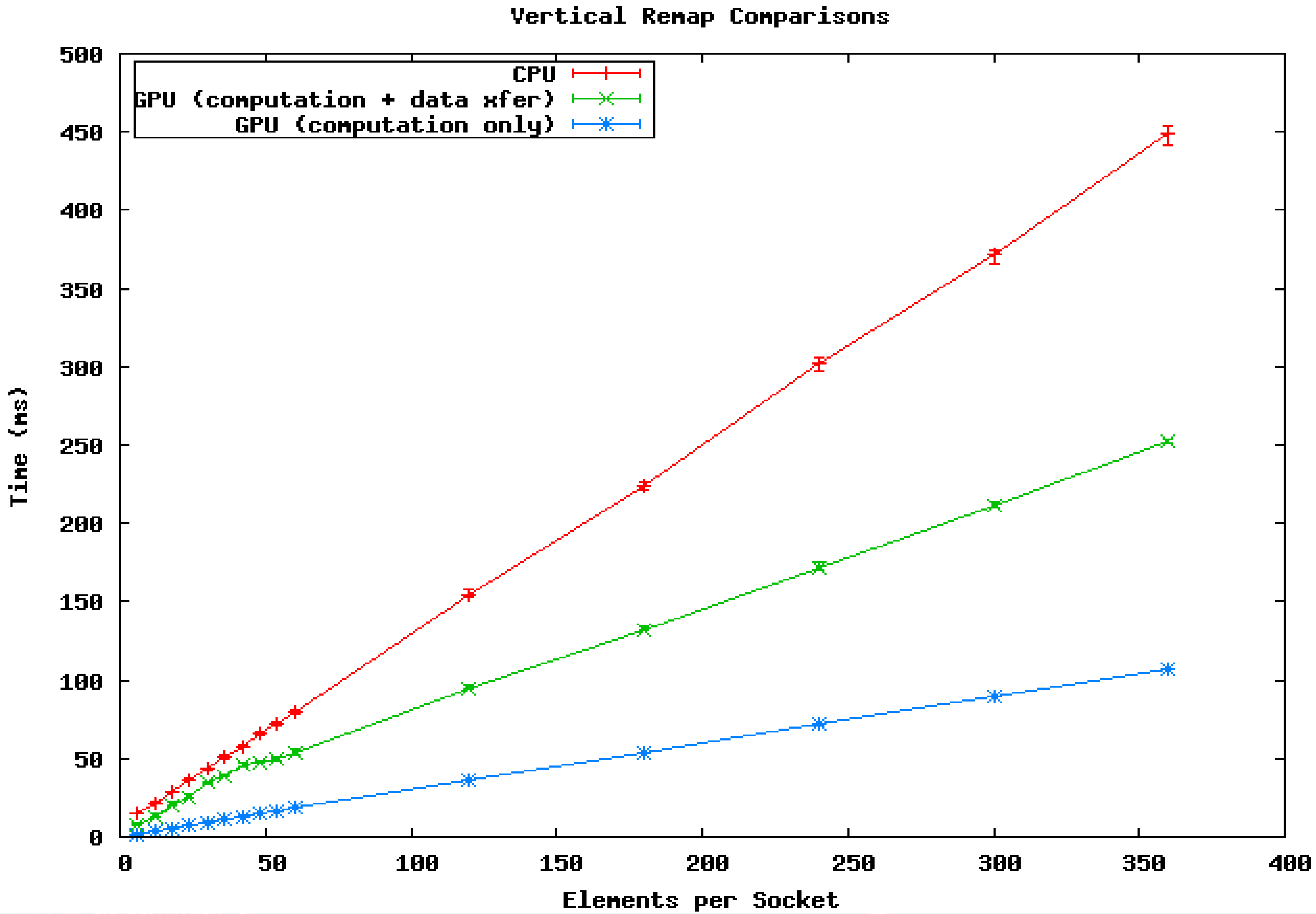
Method of acceleration

- CUDA
  - We primarily used PGI CUDA Fortran for HOMME kernels.
    - Coding intensive, but yields best results with most flexibility
- Directives
  - Open-MP pragma style implementation.
    - Relative ease of implementation, by requires specific forms
- Libraries
  - Application leverage device optimization of libraries
    - In specific application, no significant math library used

Method of acceleration: Why?

- **CAM-SE** dynamical core has the same loop and index patterns appear repeatedly, which makes CUDA work tractable.
- **Mozart chemistry** code developers are likely to accommodate code changes. Both CUDA and directives will be explored. Implicit solver code is generated by a pre-processor, so GPU-specific code can be generated without affecting CPU version.
- **Physics** code will need to accelerated using directives if this is to be run on GPUs
  - to prevent version bifurcation, involves many routines written by many different people who have no knowledge accelerators
  - profile for the physics is extremely flat
  - physics is less than 1% of target job so low priority now


Euler Step Comparisons
CPU
GPU (sync)
GPU (async)
Time (ms)
Elements per Socket
0
20
40
60
80
100
120
140
160
180
200
0
50
100
150
200
250
300
350
400


Vertical Remap Comparisons
CPU
GPU (computation + data xfer)
GPU (computation only)
Time (ms)
Elements per Socket
0
50
100
150
200
250
300
350
400
450
500

Plan of work – what remains to be done?

- Continue to refine profiles as inefficient code is rewritten and work on routines that take a significant percentage of the runtime.
- Optimize on-GPU boundary exchange (pack and unpack).
- Optimize buffers containing boundary data that need to go to CPU to generate MPI messages.
- Move up a level with GPU kernels to include data that can be left on GPU using new boundary exchange methods.
- Integrate GPU kernels into HOMME trunk (and via that, into CESM)

Summary

- The community is moving towards high-resolution modeling with atmospheric chemistry.
- We have demonstrated that such a run is dominated by tracer advection.
- We have demonstrated that enough parallelism exists to run tracer advection on the GPU.

Additional help from

Jim Rosinski, Mark Taylor, John Dennis, Kate Evans, Oscar Hernandez, Jim Schwarzmeier, Tom Court, Abdulla Bataineh

CPU code:

```
Loop over elements
  Loop over advected constituents (q)
    Loop over j
       Loop over i
          Loops over levels
           all computations from quadratic_spline and
           quadratic_spline_monotone manually inlined
```

GPU code:

```
 tblock=dim3(nv,nv,nb_elm) – columns associated with 6 elements
 tgrid=dim3(qsize,1,1)  - each block has a different value of q

 Copy data from element structure to local arrays
 Transfer data to GPU
 Call remap_Q_kernel<<<tgrid,tblock>>>(…)
 Transfer results back to CPU
 Copy data back to elem structure

attributes(global) subroutine &
      remap_Q_kernel(Qdp,hyai,hybi,ps_v,dpdn,ps0,dt)

  i  = threadIdx%x
  j  = threadIdx%y
  ie = threadIdx%z
  q  = blockIdx%x

 ! each thread has one column of points for one value of q

   Loops over levels

     all computations from quadratic_splime and
     quadratic_spline_monotone manually inlined

end subroutine remap_Q_kernel
```