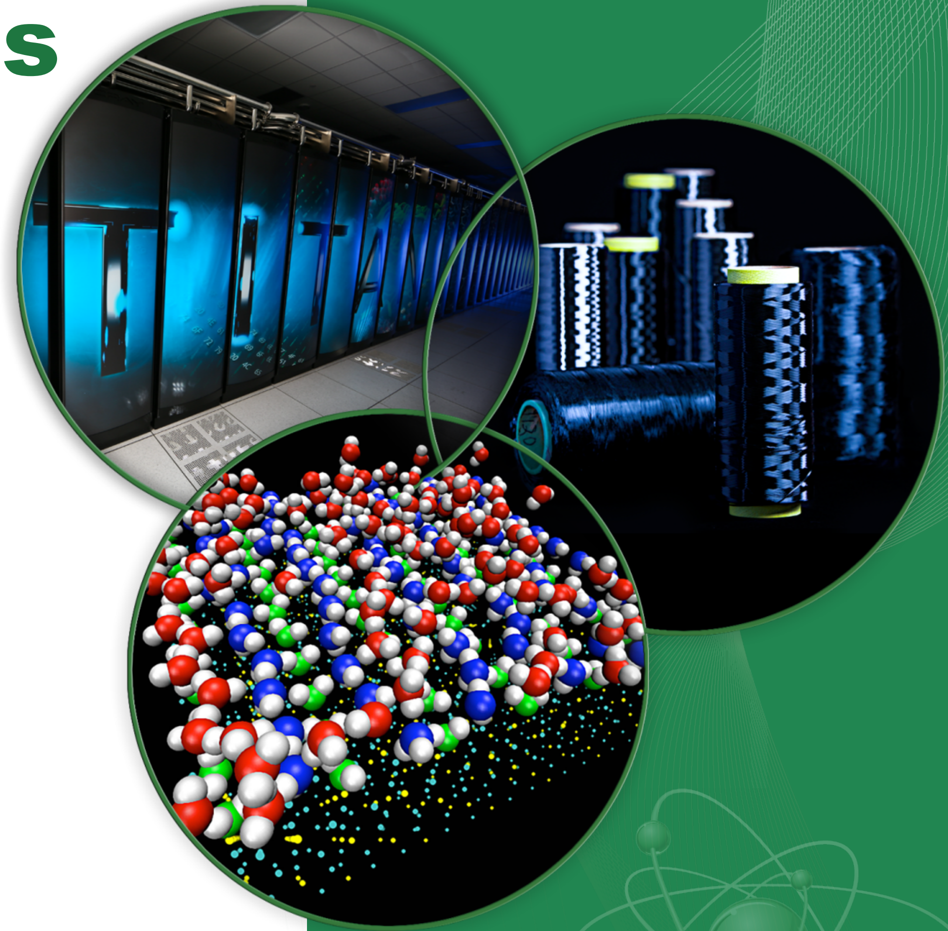
Porting CAM-SE To Use Titan's GPUs



2014 OLCF User's Meeting

Matthew Norman	ORNL
Jeffrey Larkin	Nvidia
Richard Archibald	ORNL
Valentine Anantharaj	ORNL
Ilene Carpenter	NREL
Paulius Micikevicius	Nvidia
Katherine Evans	ORNL

ORNL is managed by UT-Battelle
for the US Department of Energy

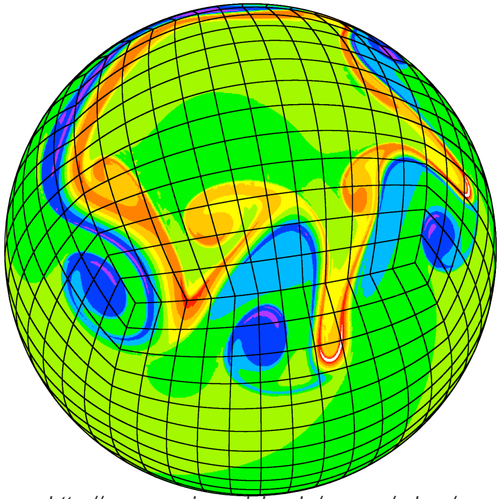
OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

What is CAM-SE

- Climate-scale atmospheric simulation for capability computing
- Comprised of (1) a dynamical core and (2) physics packages

What is CAM-SE

- Climate-scale atmospheric simulation for capability computing
- Comprised of (1) a dynamical core and (2) physics packages



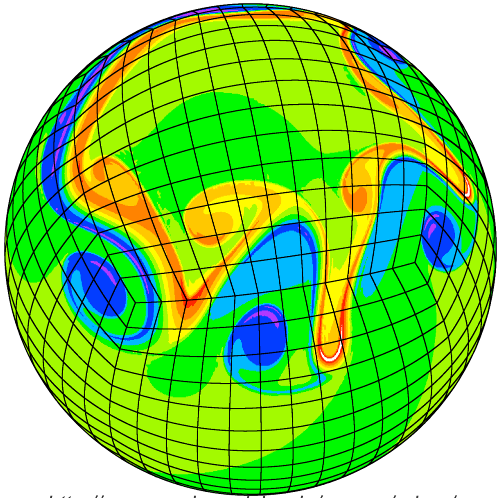
http://esse.engin.umich.edu/groups/admg/ dcmip/jablonowski_cubed_sphere_vorticity.png

Dynamical Core

1. “Dynamics”: wind, energy, & mass
2. “Tracer” Transport: (H_2O, CO_2, O_3, …)
   Transport quantities not advanced by the dynamics

What is CAM-SE

- Climate-scale atmospheric simulation for capability computing
- Comprised of (1) a dynamical core and (2) physics packages



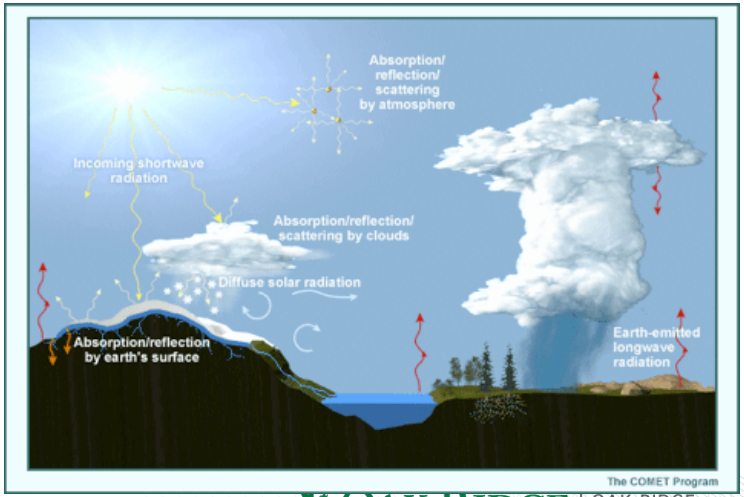
http://esse.engin.umich.edu/groups/admg/dcmip/jablonowski_cubed_sphere_vorticity.png

Dynamical Core

1. "Dynamics": wind, energy, & mass
2. "Tracer" Transport: (H_2O, CO_2, O_3, …)
   Transport quantities not advanced by the dynamics

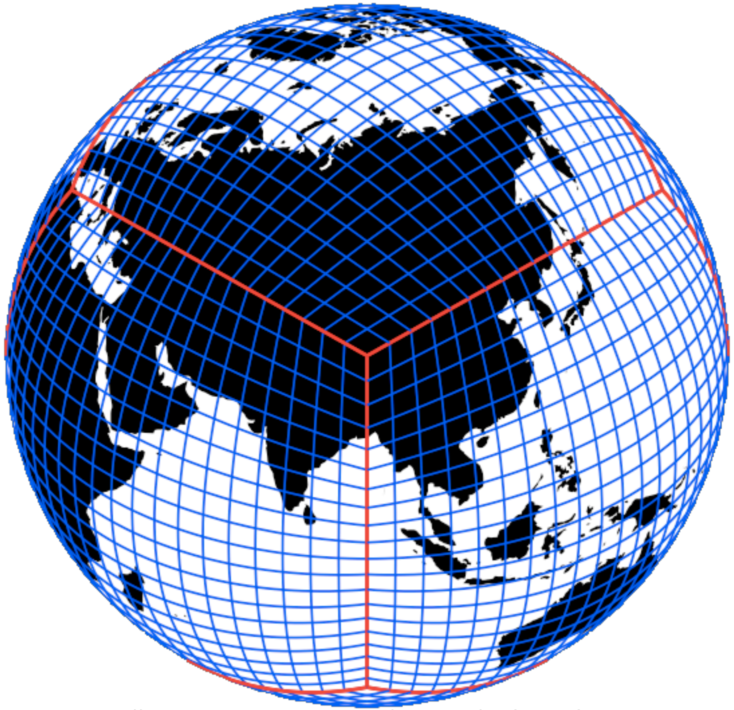
Physics Packages

Resolve anything interesting not included in dynamical core (moist convection, radiation, chemistry, etc)


Absorption/ reflection/ scattering by atmosphere
Incoming shortwave radiation
Absorption/reflection/ scattering by clouds
Diffuse solar radiation
Absorption/reflection by earth's surface
Earth-emitted longwave radiation
The COMET Program

http://web.me.com/macweather/blogger/maweather_files/physprc2.gif

Gridding & Numerics

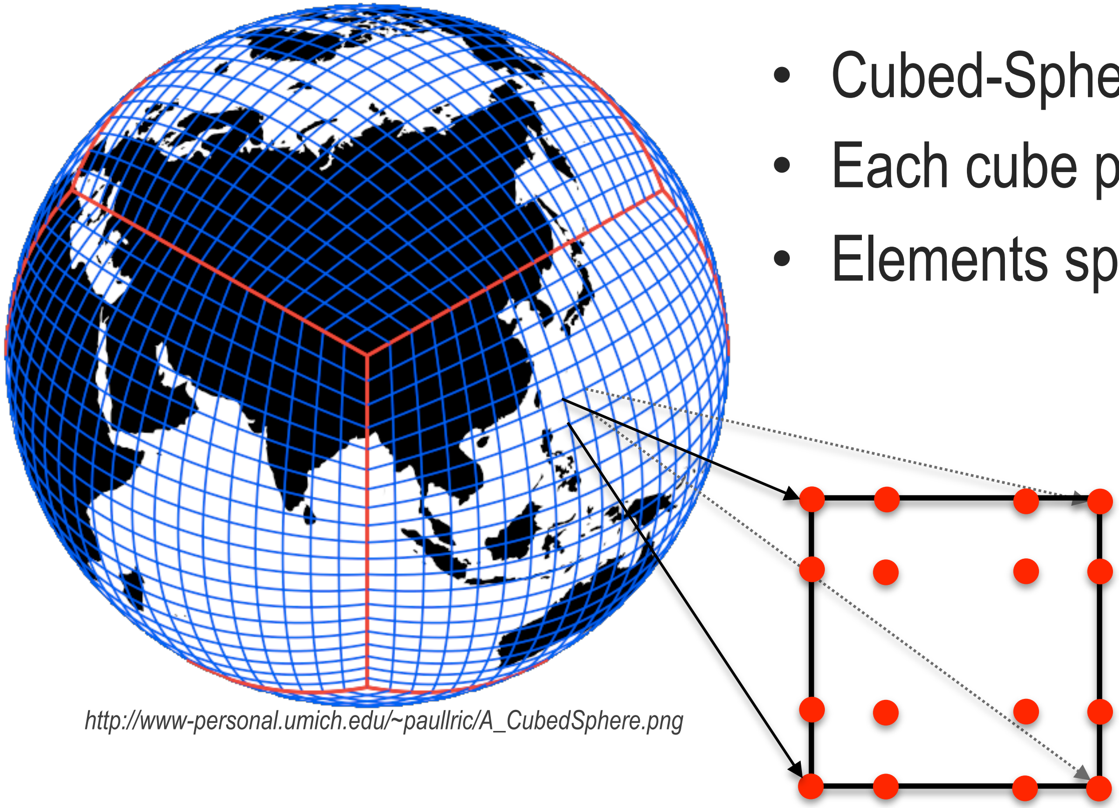


http://www-personal.umich.edu/~paullric/A_CubedSphere.png

- Cubed-Sphere + Spectral Element
- Each cube panel divided into elements

Gridding & Numerics

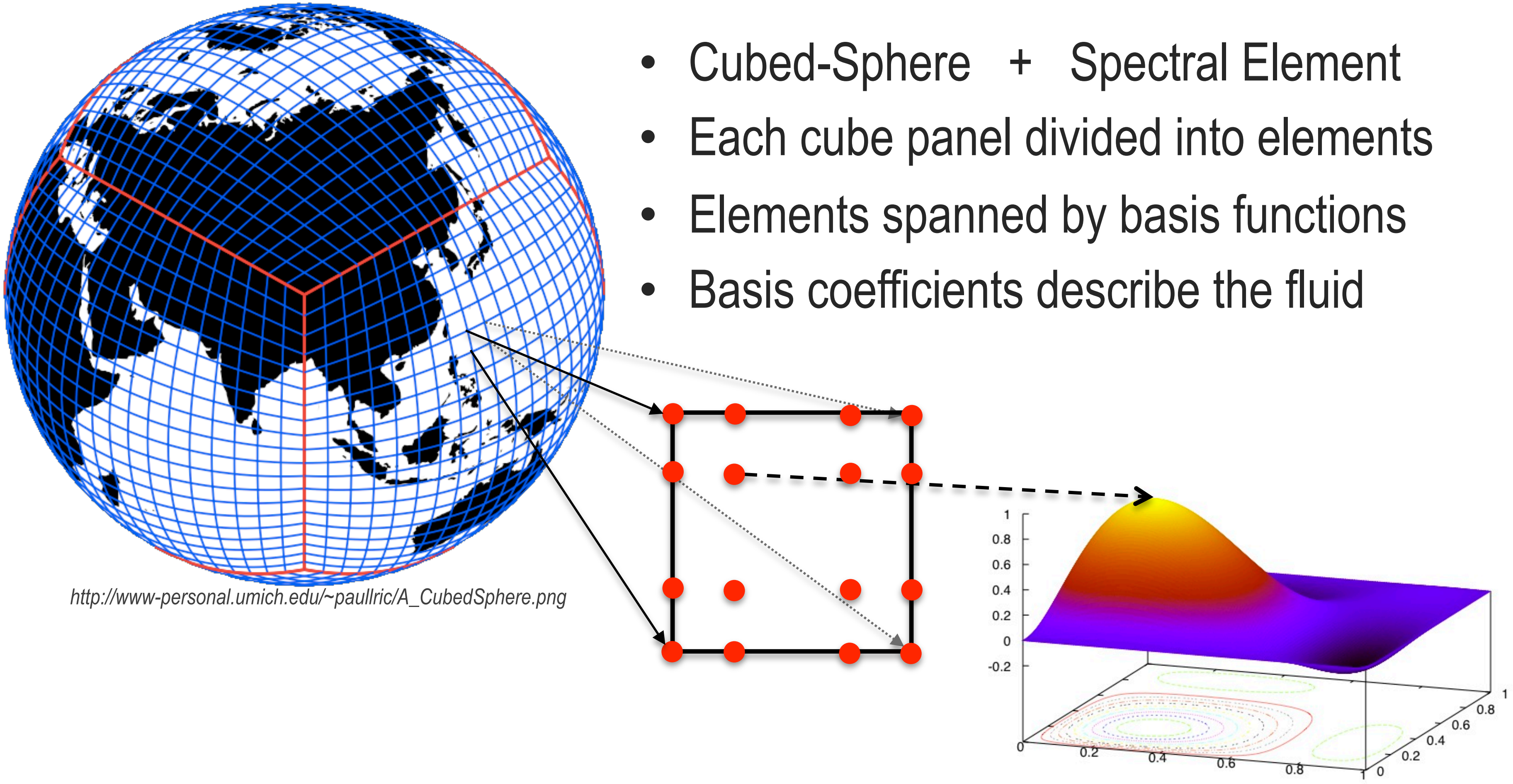
- Cubed-Sphere + Spectral Element
- Each cube panel divided into elements
- Elements spanned by basis functions



http://www-personal.umich.edu/~paullric/A_CubedSphere.png

Gridding & Numerics

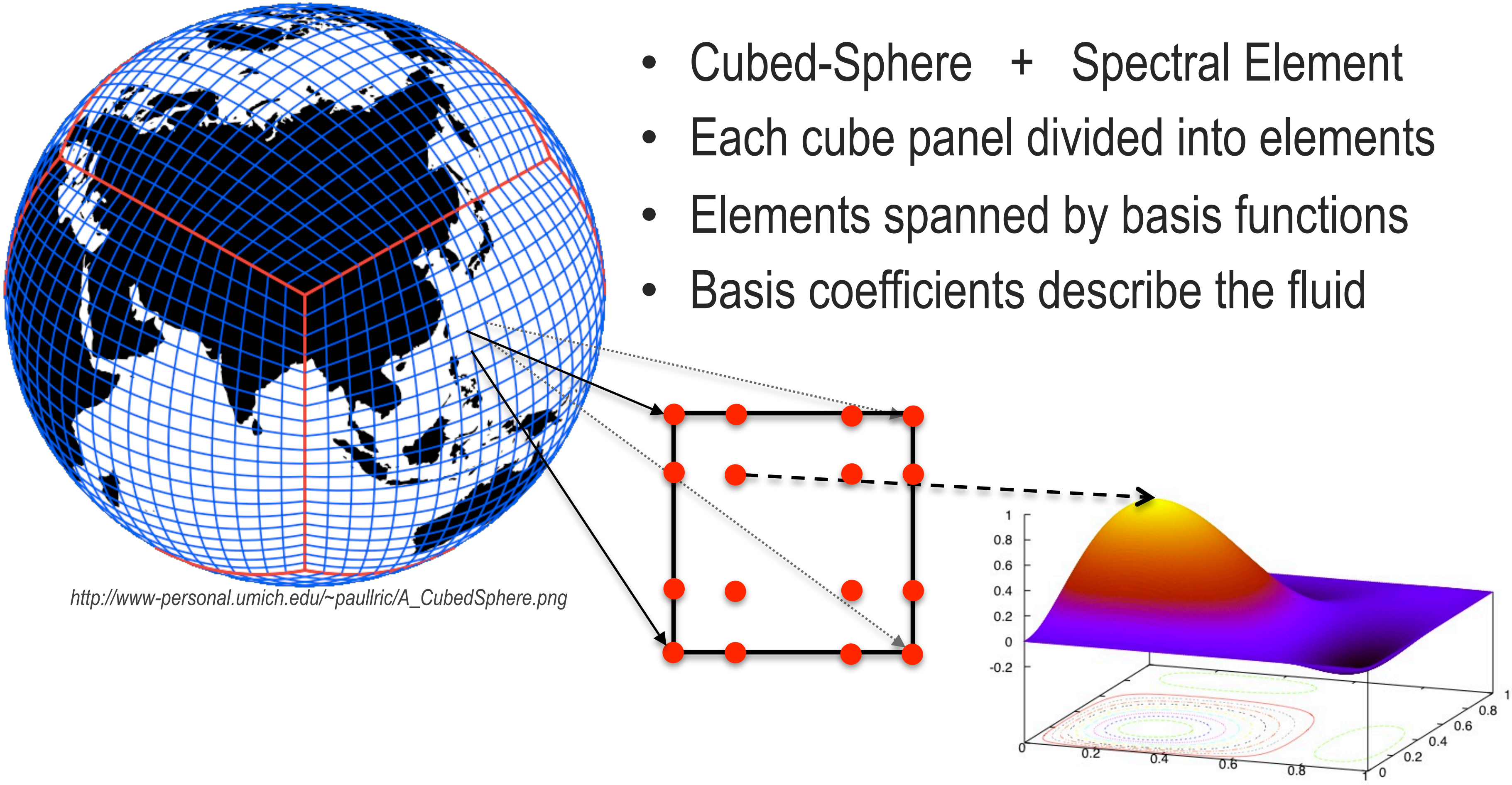
- Cubed-Sphere + Spectral Element
- Each cube panel divided into elements
- Elements spanned by basis functions
- Basis coefficients describe the fluid



http://www-personal.umich.edu/~paullric/A_CubedSphere.png

Gridding & Numerics

- Cubed-Sphere + Spectral Element
- Each cube panel divided into elements
- Elements spanned by basis functions
- Basis coefficients describe the fluid



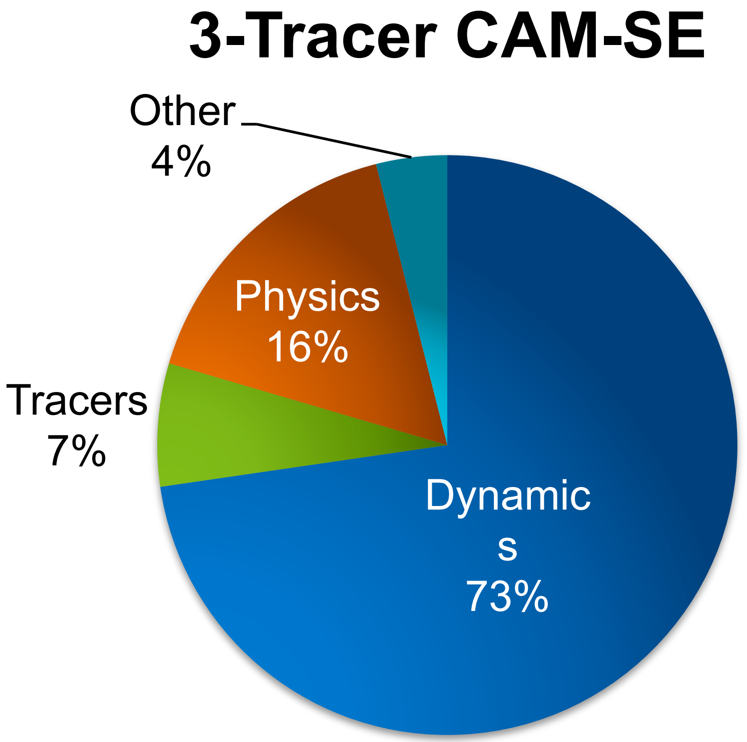
http://www-personal.umich.edu/~paullric/A_CubedSphere.png

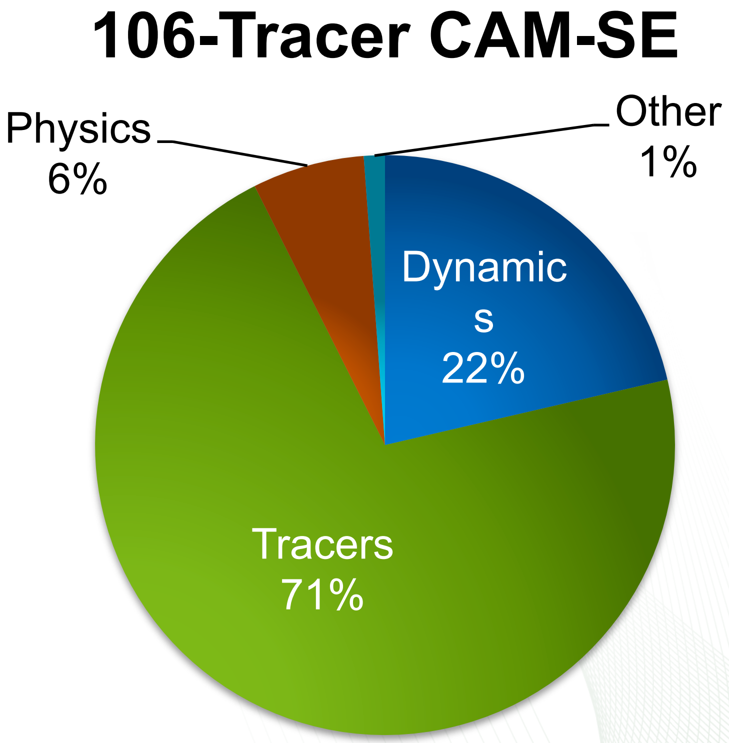
Used CUDA FORTRAN from PGI

OACC Directives: Better software engineering option moving forward

CAM-SE Profile (Cray XT5, 14K nodes)

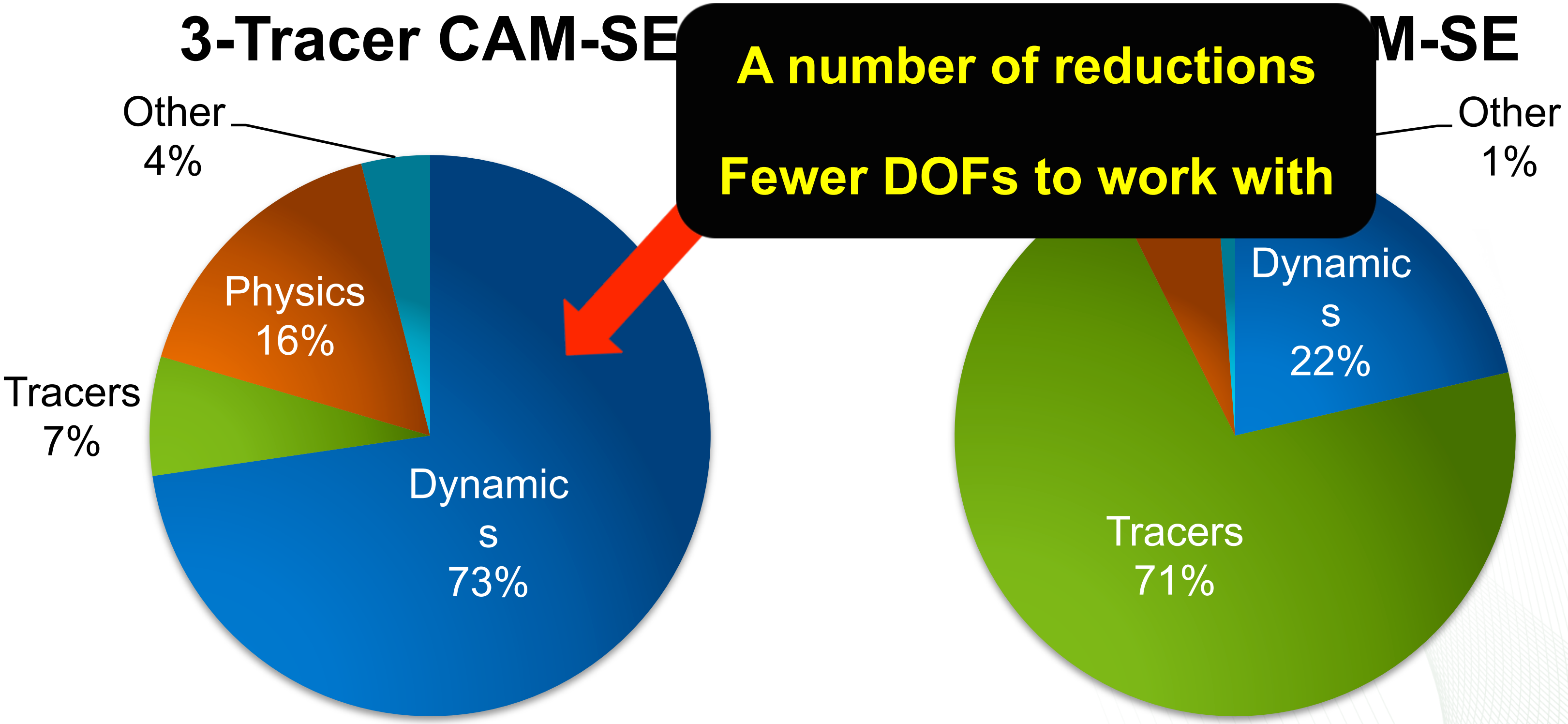
- Original CAM-SE used 3 tracers (20% difficult to port)
- Mozart chemistry provides 106 tracers (7% difficult to port)
  - Centralizes port to tracers with mostly data-parallel routines





CAM-SE Profile (Cray XT5, 14K nodes)

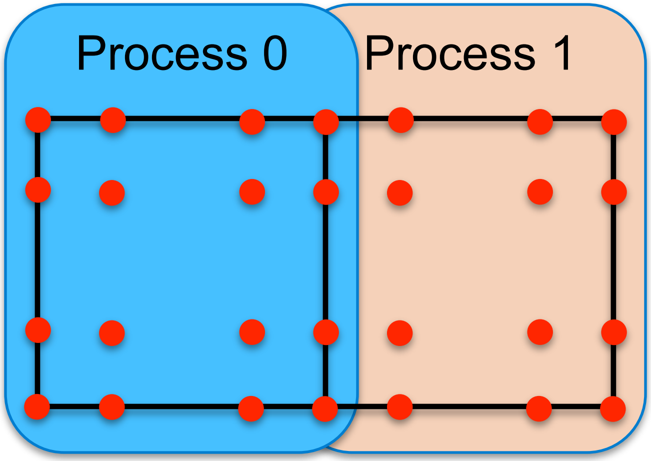
- Original CAM-SE used 3 tracers (20% difficult to port)
- Mozart chemistry provides 106 tracers (7% difficult to port)
  - Centralizes port to tracers with mostly data-parallel routines



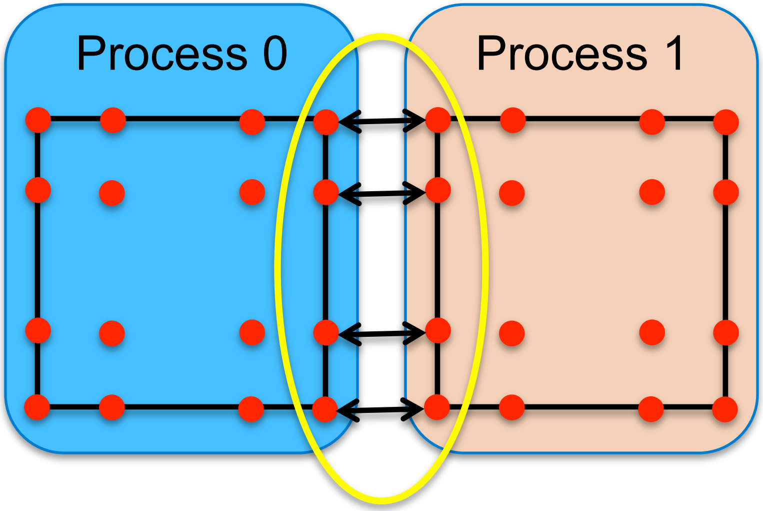
A number of reductions

Fewer DOFs to work with

Communication Between Elements



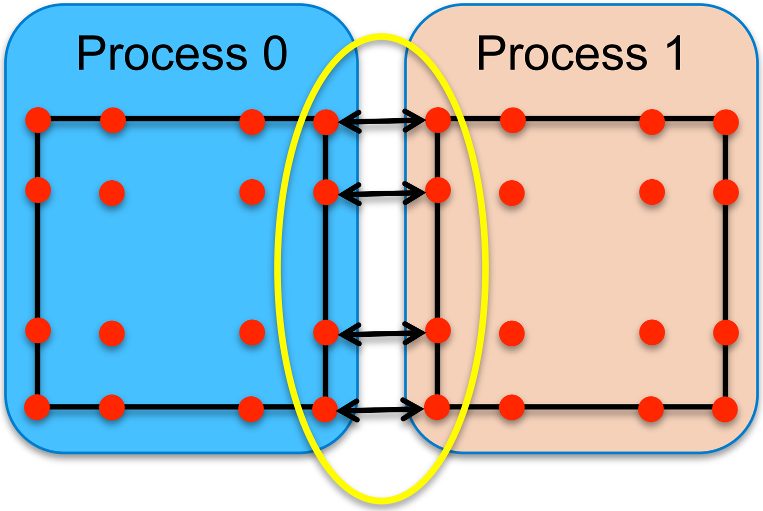
Communication Between Elements



Physically occupy the same location, Spectral Element requires them to be equal

Edges are averaged, and the average replaces both edges

Communication Between Elements



Physically occupy the same location, Spectral Element requires them to be equal

Edges are averaged, and the average replaces both edges

Implementation

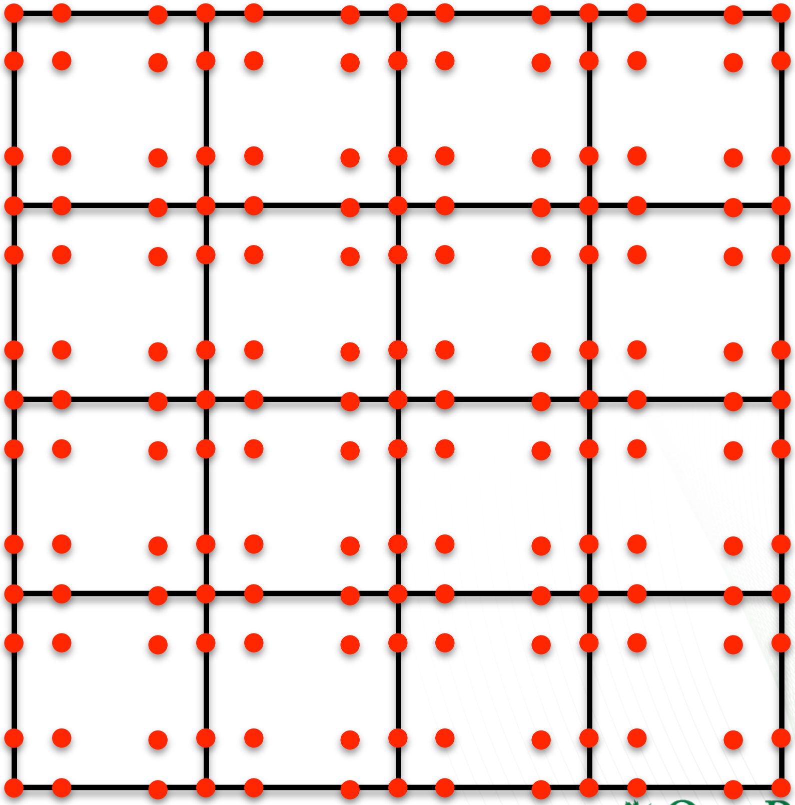
Edge_pack: pack all element edges into process-wide buffer. Data sent over MPI are contiguous in buffer.

Bndry_exchange: Send & receive data at domain decomposition boundaries

Edge_unpack: Perform a weighted sum for data at all element edges.

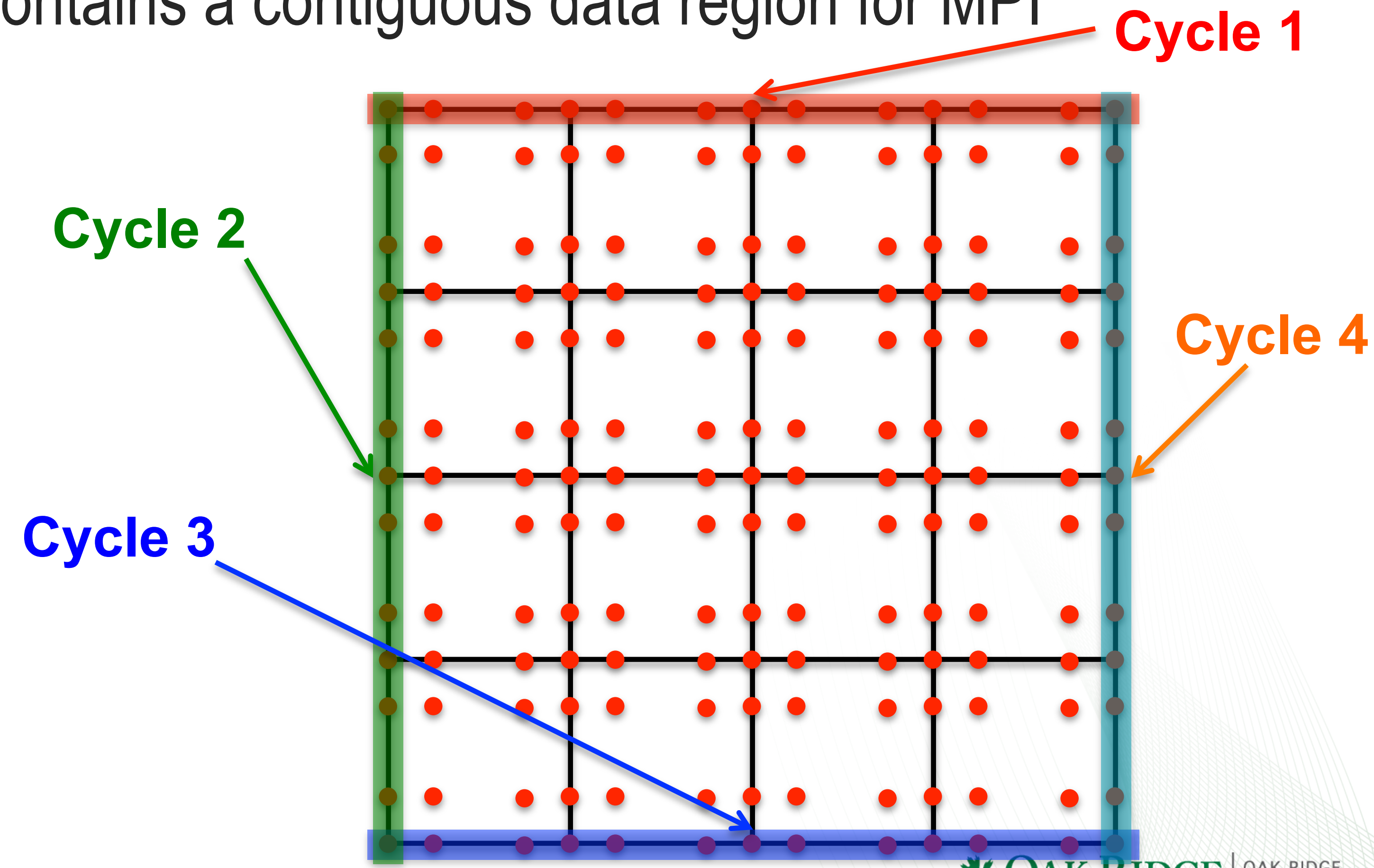
Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in “cycles”



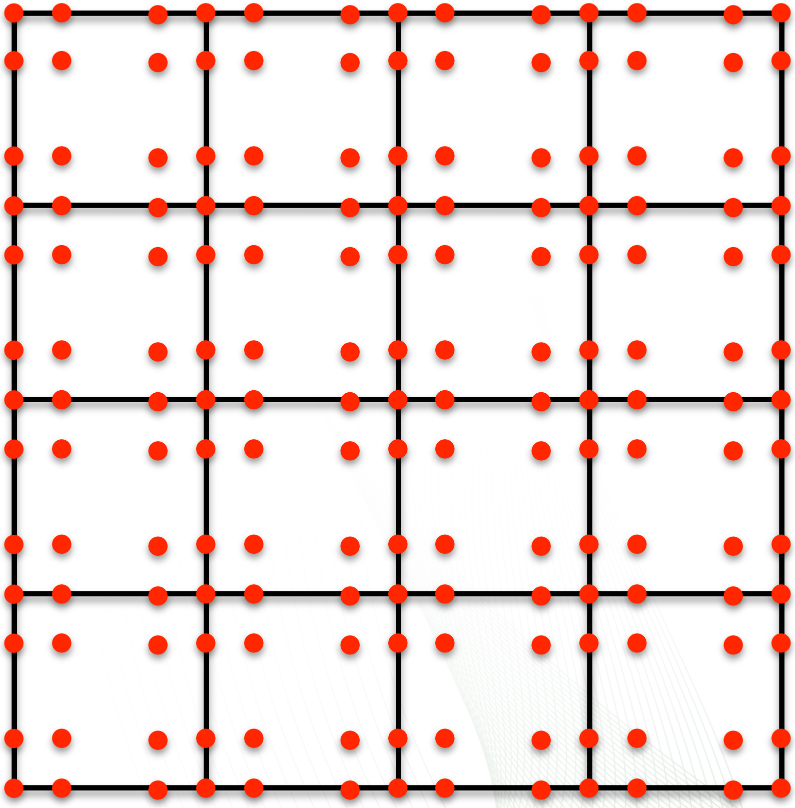
Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in “cycles”
- A cycle contains a contiguous data region for MPI



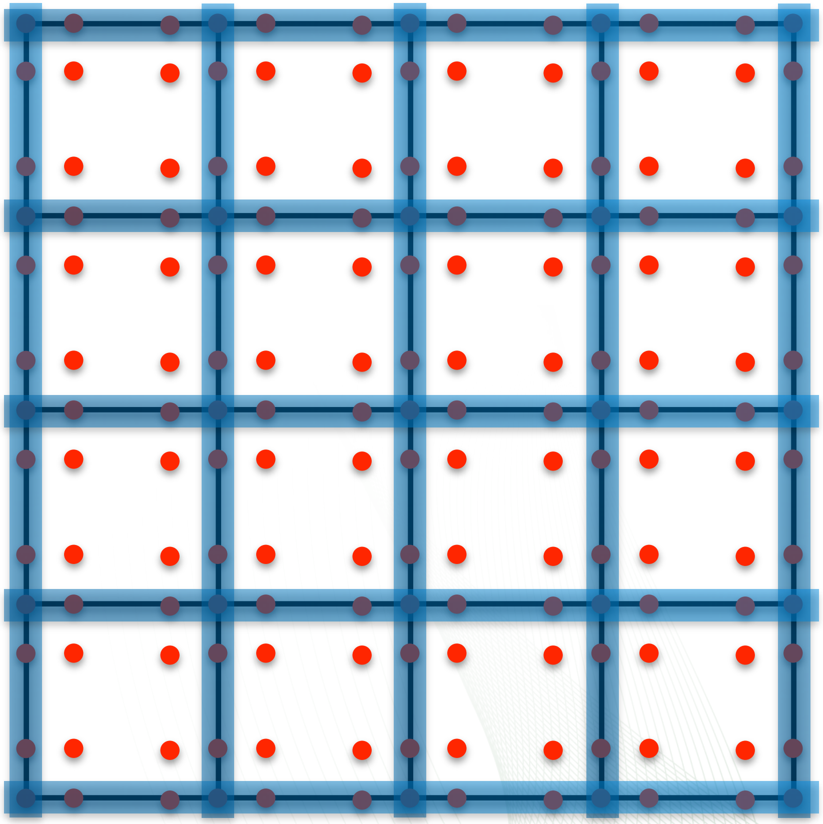
Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in “cycles”
- A cycle contains a contiguous data region for MPI
- Original pack/exchange/unpack



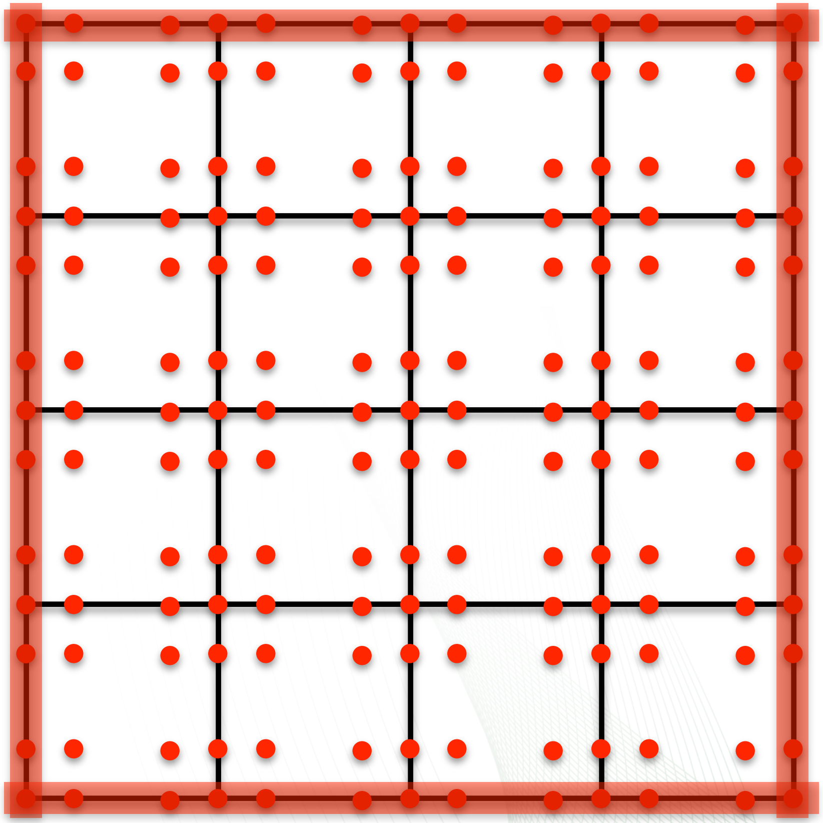
Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in “cycles”
- A cycle contains a contiguous data region for MPI
- Original pack/exchange/unpack
  - Pack all edges in a GPU Kernel



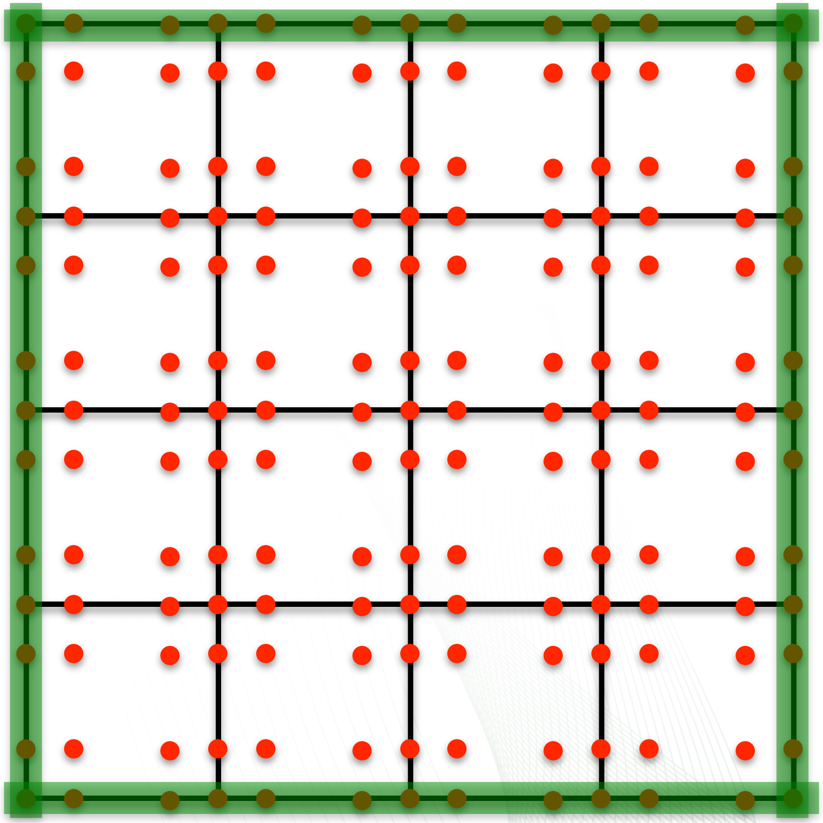
Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in “cycles”
- A cycle contains a contiguous data region for MPI
- Original pack/exchange/unpack
  - Pack all edges in a GPU Kernel
  - For each “send cycle”
    - Send cycle over PCI-e (D2H)
    - MPI_Isend the cycle



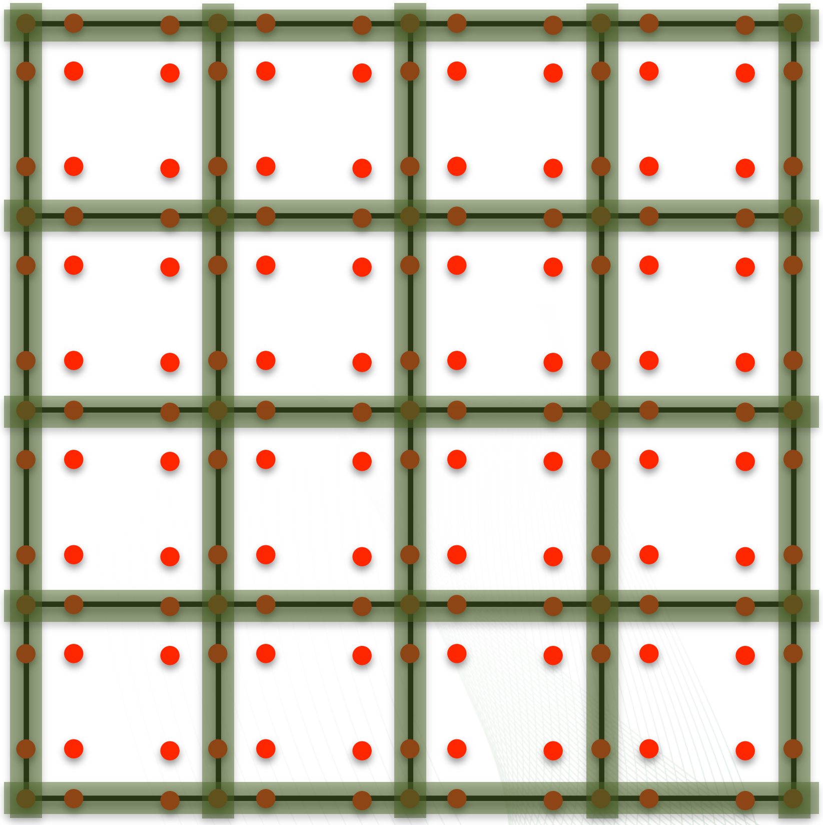
Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in "cycles"
- A cycle contains a contiguous data region for MPI
- Original pack/exchange/unpack
  - Pack all edges in a GPU Kernel
  - For each "send cycle"
    - Send cycle over PCI-e (D2H)
    - MPI_Isend the cycle
  - For each "receive cycle"
    - MPI_Wait for the data
    - Send cycle over PCI-e (H2D)

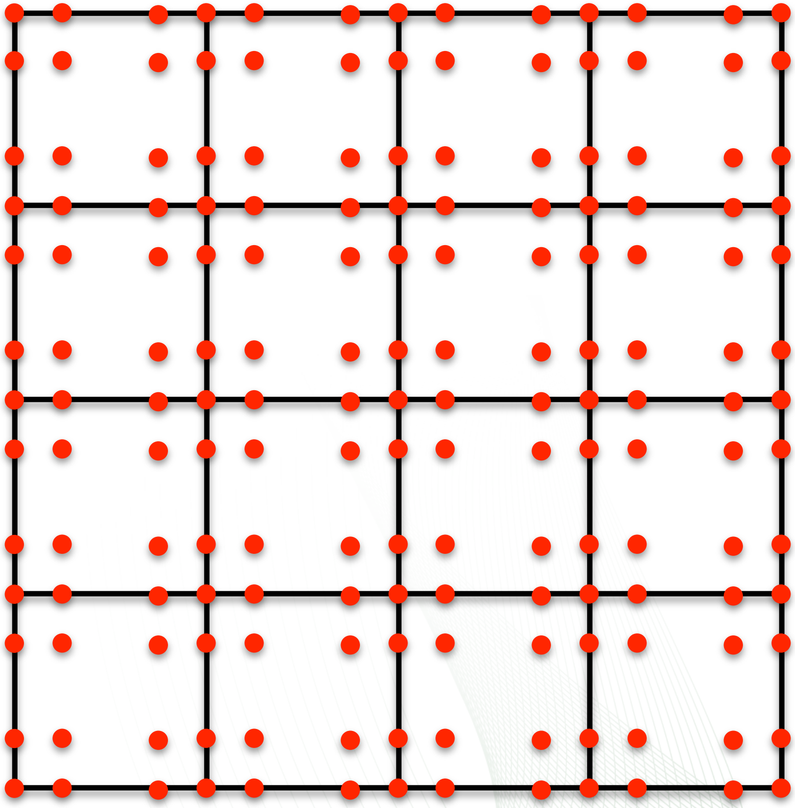


Original Pack/Exchange/Unpack

- Edge_pack ensures data for MPI is contiguous in buffer
- MPI communication occurs in "cycles"
- A cycle contains a contiguous data region for MPI
- Original pack/exchange/unpack
  - Pack all edges in a GPU Kernel
  - For each "send cycle"
    - Send cycle over PCI-e (D2H)
    - MPI_Isend the cycle
  - For each "receive cycle"
    - MPI_Wait for the data
    - Send cycle over PCI-e (H2D)
  - Unpack all edges in a GPU Kernel

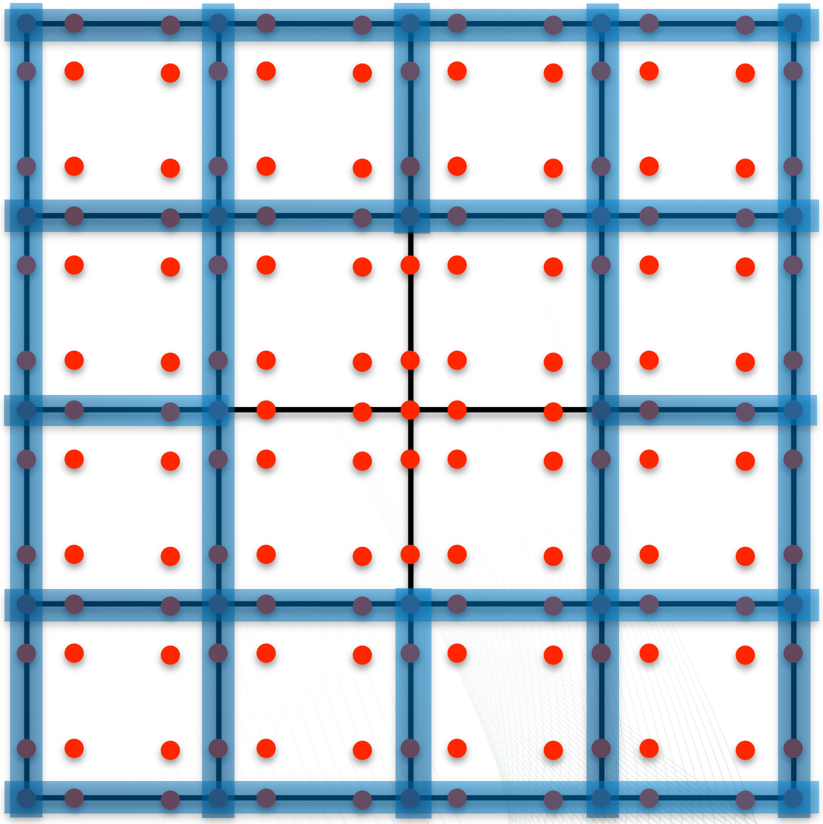


Porting Strategy: Pack/Exchange/Unpack



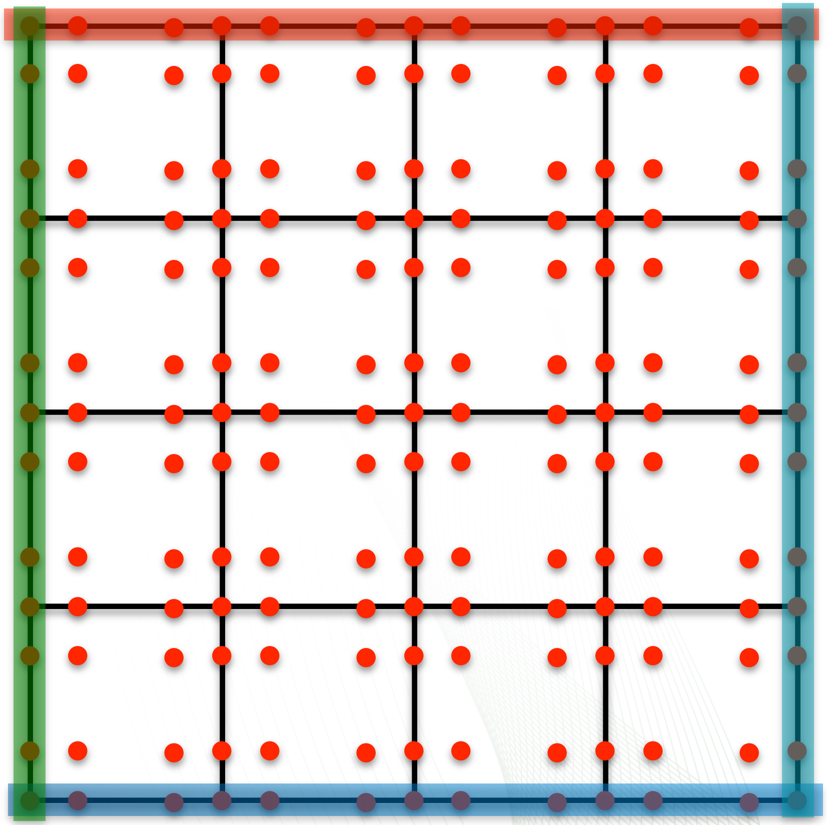
Porting Strategy: Pack/Exchange/Unpack

- Pack external elements that participate with MPI



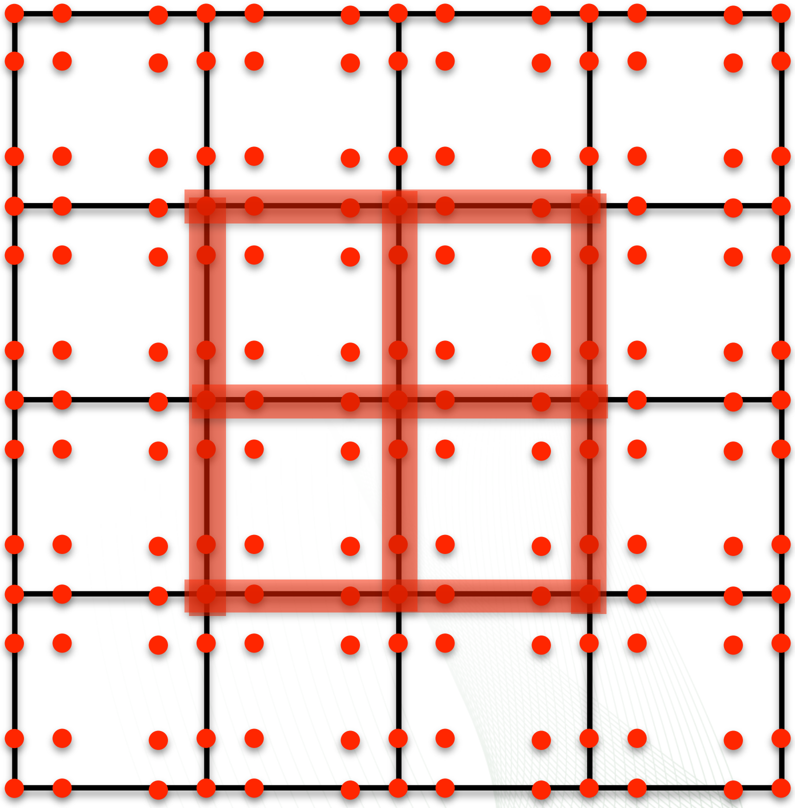
Porting Strategy: Pack/Exchange/Unpack

- Pack external elements that participate with MPI
- Send Cycles over MPI and PCI-e



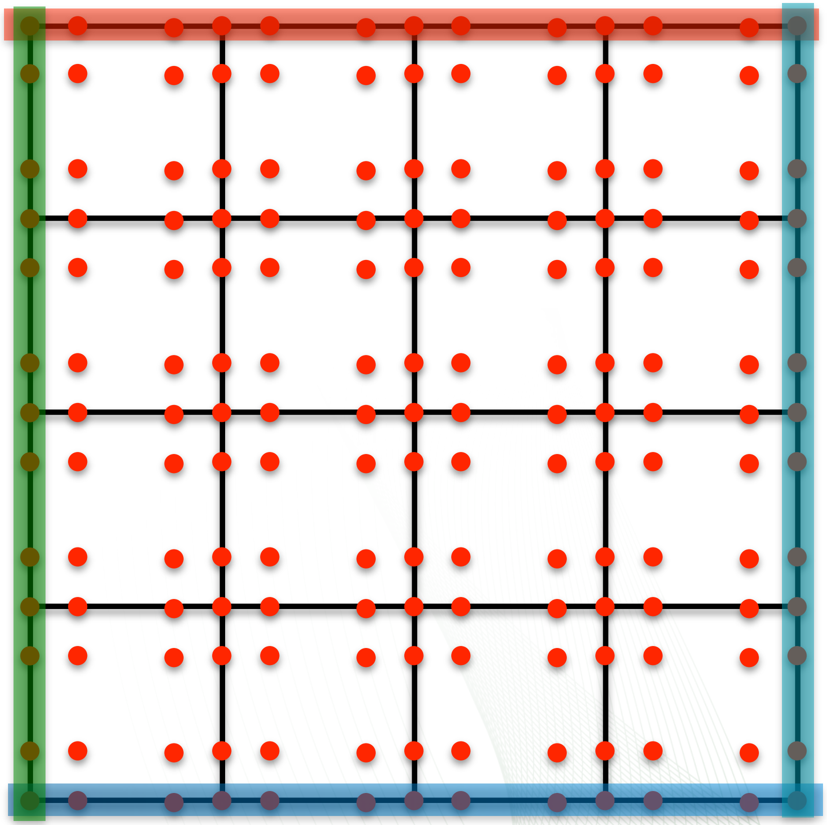
Porting Strategy: Pack/Exchange/Unpack

- Pack external elements that participate with MPI
- Send Cycles over MPI and PCI-e
- Pack and unpack internal elements during MPI / PCI-e



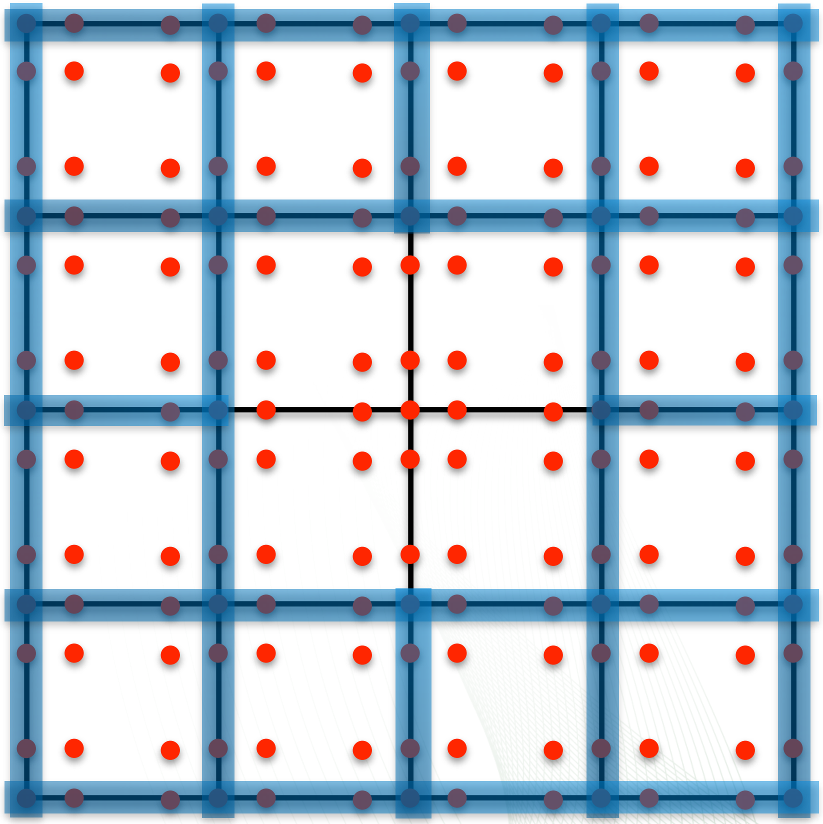
Porting Strategy: Pack/Exchange/Unpack

- Pack external elements that participate with MPI
- Send Cycles over MPI and PCI-e
- Pack and unpack internal elements during MPI / PCI-e
- MPI_irecv and PCI-e to GPU



Porting Strategy: Pack/Exchange/Unpack

- Pack external elements that participate with MPI
- Send Cycles over MPI and PCI-e
- Pack and unpack internal elements during MPI / PCI-e
- MPI_irecv and PCI-e to GPU
- Unpack external elements that participate with MPI



Think Differently About Threading

CPU Code

```
do ie=1,nelemd
 do q=1,qsize
  do k=1,nlev
   do j=1,np
    do i=1,np
     coefs(1,i,j,k,q,ie) = ...
     coefs(2,i,j,k,q,ie) = ...
     coefs(3,i,j,k,q,ie) = ...
```

GPU Code

```
ie = blockidx%y
q  = blockidx%x
k  = threadidx%z
j  = threadidx%y
i  = threadidx%x
coefs(1,i,j,k,q,ie) = ...
coefs(2,i,j,k,q,ie) = ...
coefs(3,i,j,k,q,ie) = ...
```

Think Differently About Threading

CPU Code

```
do ie=1,nelemd
 do q=1,qsize
  do k=1,nlev
   do j=1,np
    do i=1,np
     coefs(1,i,j,k,q,ie) = ...
     coefs(2,i,j,k,q,ie) = ...
     coefs(3,i,j,k,q,ie) = ...
```

Coded to respect cache locality

GPU Code

```
ie = blockidx%y
q  = blockidx%x
k  = threadidx%z
j  = threadidx%y
i  = threadidx%x
coefs(1,i,j,k,q,ie) = ...
coefs(2,i,j,k,q,ie) = ...
coefs(3,i,j,k,q,ie) = ...
```

Think Differently About Threading

CPU Code

```
do ie=1,nelemd
 do q=1,qsize
  do k=1,nlev
   do j=1,np
    do i=1,np
     coefs(1,i,j,k,q,ie) = ...
     coefs(2,i,j,k,q,ie) = ...
     coefs(3,i,j,k,q,ie) = ...
```

Coded to respect cache locality

GPU Code

```
ie = blockidx%y
q  = blockidx%x
k  = threadidx%z
j  = threadidx%y
i  = threadidx%x
coefs(1,i,j,k,q,ie) = ...
coefs(2,i,j,k,q,ie) = ...
coefs(3,i,j,k,q,ie) = ...
```

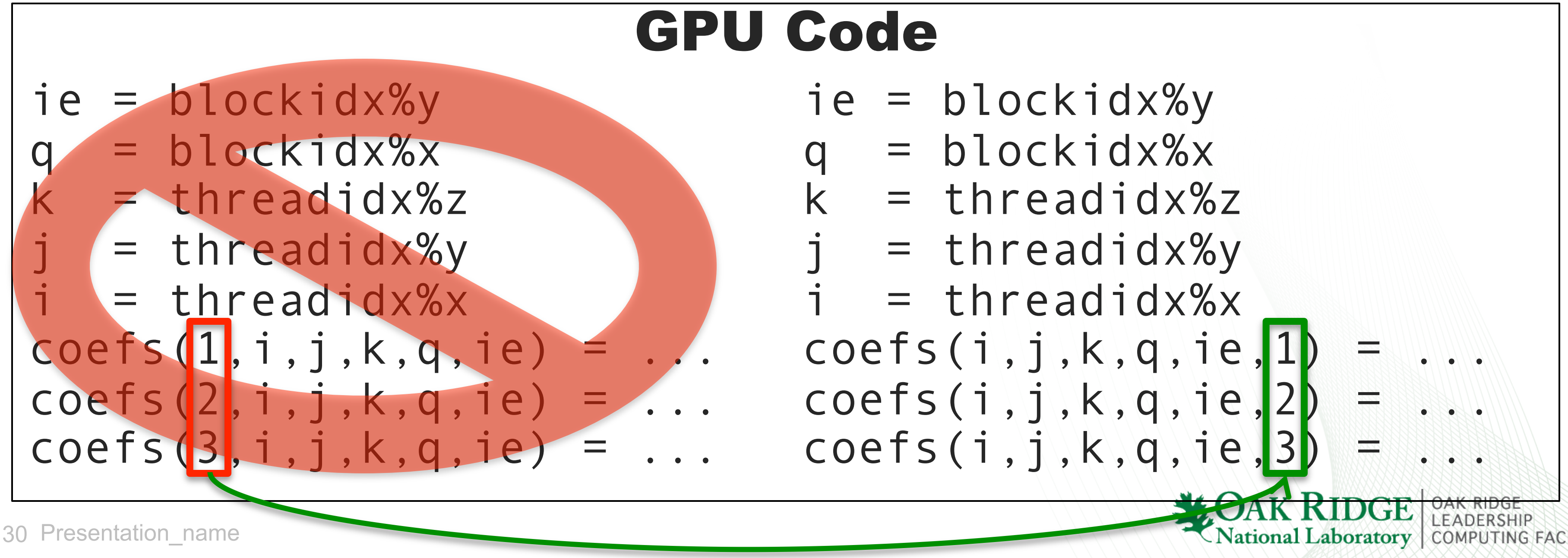
However, this will not coalesce to fill the DRAM bus

Think Differently About Threading

CPU Code

```
do ie=1,nelemd
 do q=1,qsize
  do k=1,nlev
   do j=1,np
    do i=1,np
     coefs(1,i,j,k,q,ie) = ...
     coefs(2,i,j,k,q,ie) = ...
     coefs(3,i,j,k,q,ie) = ...
```

GPU Code



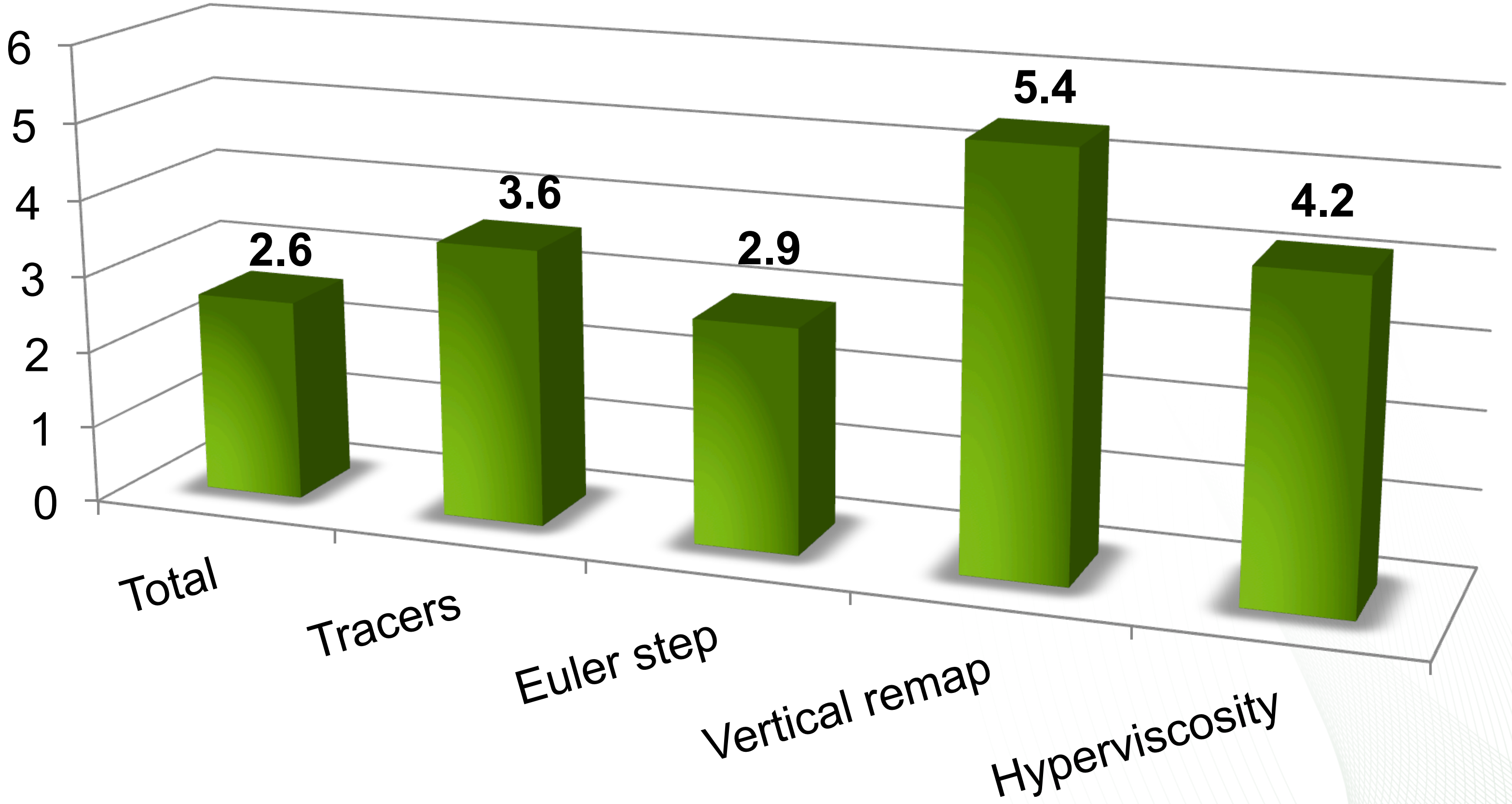
```
ie = blockidx%y
q  = blockidx%x
k  = threadidx%z
j  = threadidx%y
i  = threadidx%x
coefs(1,i,j,k,q,ie) = ...
coefs(2,i,j,k,q,ie) = ...
coefs(3,i,j,k,q,ie) = ...
```

```
ie = blockidx%y
q  = blockidx%x
k  = threadidx%z
j  = threadidx%y
i  = threadidx%x
coefs(i,j,k,q,ie,1) = ...
coefs(i,j,k,q,ie,2) = ...
coefs(i,j,k,q,ie,3) = ...
```

Speed-Up: Fermi GPU vs 1 Interlagos / Node

Older values

- Benchmarks performed on XK6 using end-to-end wall timers
- All PCI-e and MPI communication included



OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Why Was Vertical Remap So Fast?

- Originally used splines for reconstruction
  - Splines require a linear solve → vertical dependence within loops
  - Vertical index could not be threaded, only horizontal
- We replaced reconstruction with Piecewise Parabolic Method
  - Vertically independent → vertical index was threaded → 30x more threads
- Original remapping used a summation to reduce flops
  - Summations are vertically dependent and harder to thread
- We changed it to do two integrations instead
  - This double the work for remapping
  - But it also reduced data requirements and dependence
- As a result, all data in the reconstruction and remap fit into cache
  - Only accesses to DRAM were at the very beginning and end of kernel with a lot of work in between, all done in-cache
  - Thus, >5x speed-up over PPM remap on CPU

Why Was Vertical Remap So Fast?

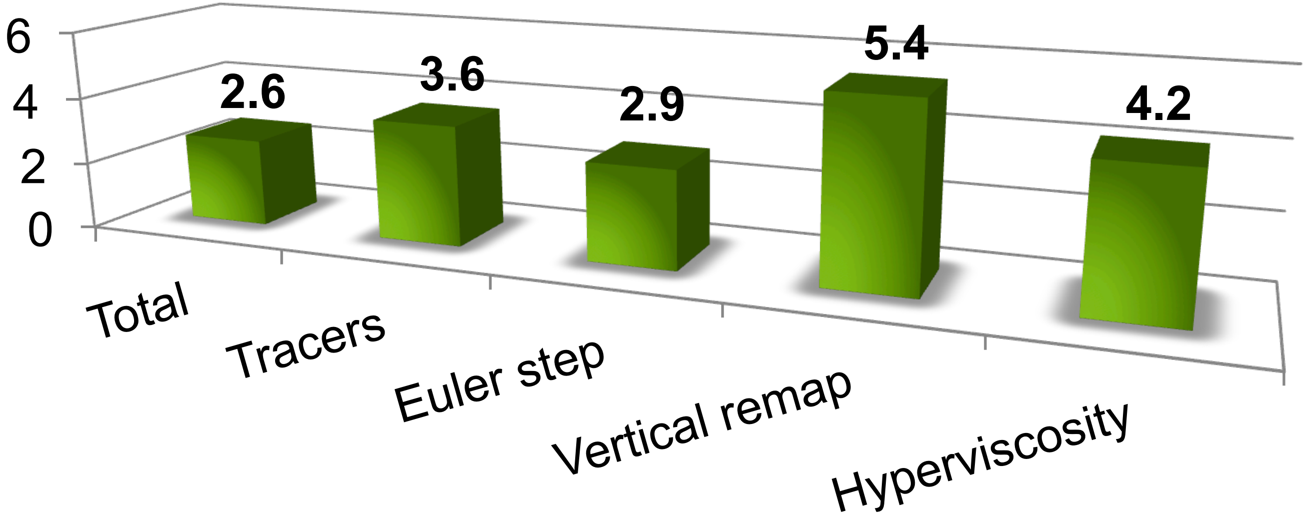
- Originally used splines for reconstruction
  - Splines require a linear solve → vertical dependence within loops
  - Vertical index could not be threaded, only horizontal
- We repl
  - Ver s
- Origina
  - Sur
- We ch
  - This
  - But
- As a res e
  - Only accesses to DRAM were at the very beginning and end of kernel with a lot of work in between, all done in-cache
  - Thus, >5x speed-up over PPM remap on CPU

- **If Increasing The Workload**
  - **Allows More Threading**
  - **Decreases Data Dependence**
  - **Decreases Local Data Requirements**
- **Then It's Worth Investigating**

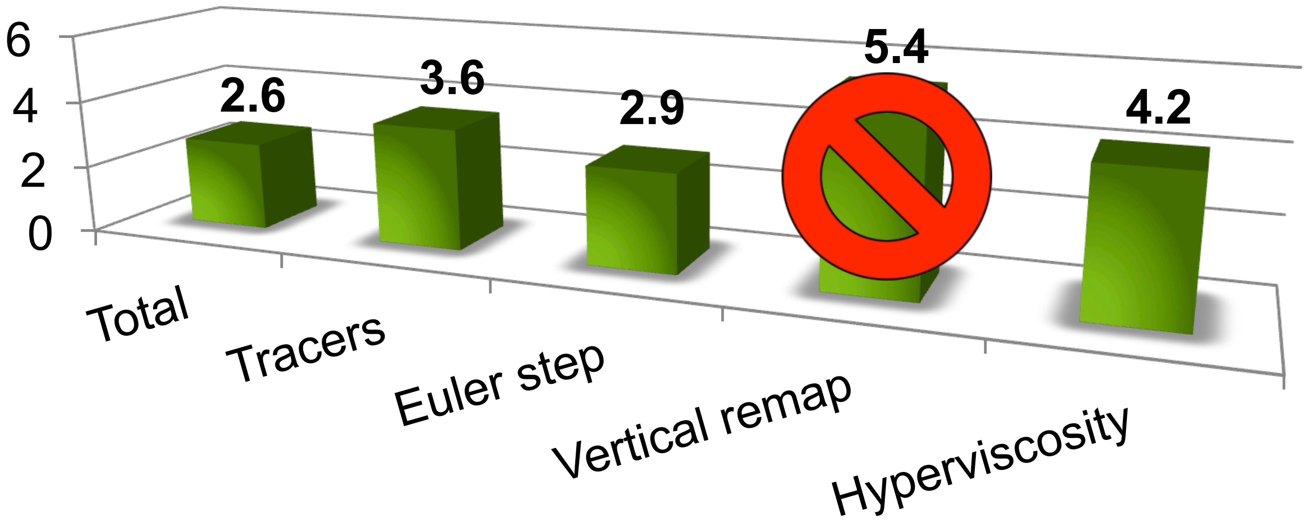
New Algorithms

- High-Order Accuracy
  - Galerkin: Local computation scales as N^{2D+1} (D = “dimensions”)
But watch out for that time step
  - Finite-Volume: Local computation scales as N^{2D-1}
Adjacent stencils nearly entirely re-used
- Time Discretization
  - Mixed-Precision, communication-reducing implicit / iterative methods
  - Communication-avoiding explicit methods
    - ADER: Local computation scales as $N^{2(D+1)}$
Large, high-order time steps w/no stages / comms
    - Multi-step: Save & re-use past steps for high-order w/o stages
- Redundant computations
  - Brake-even point might be further than you think!

Back to the Real World: Codes Change

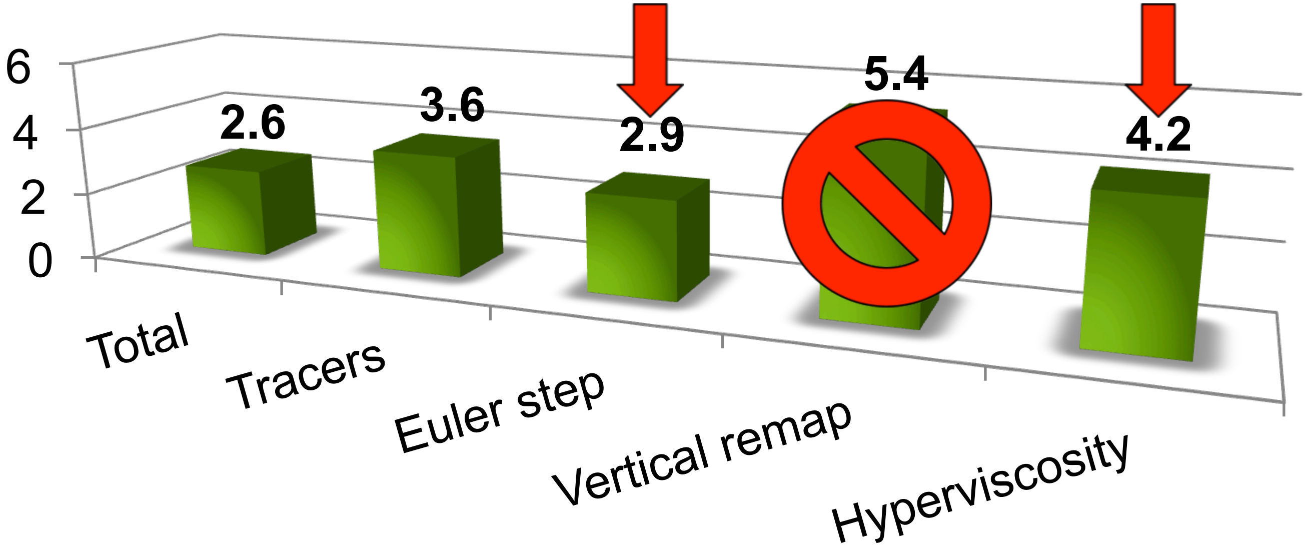


Back to the Real World: Codes Change



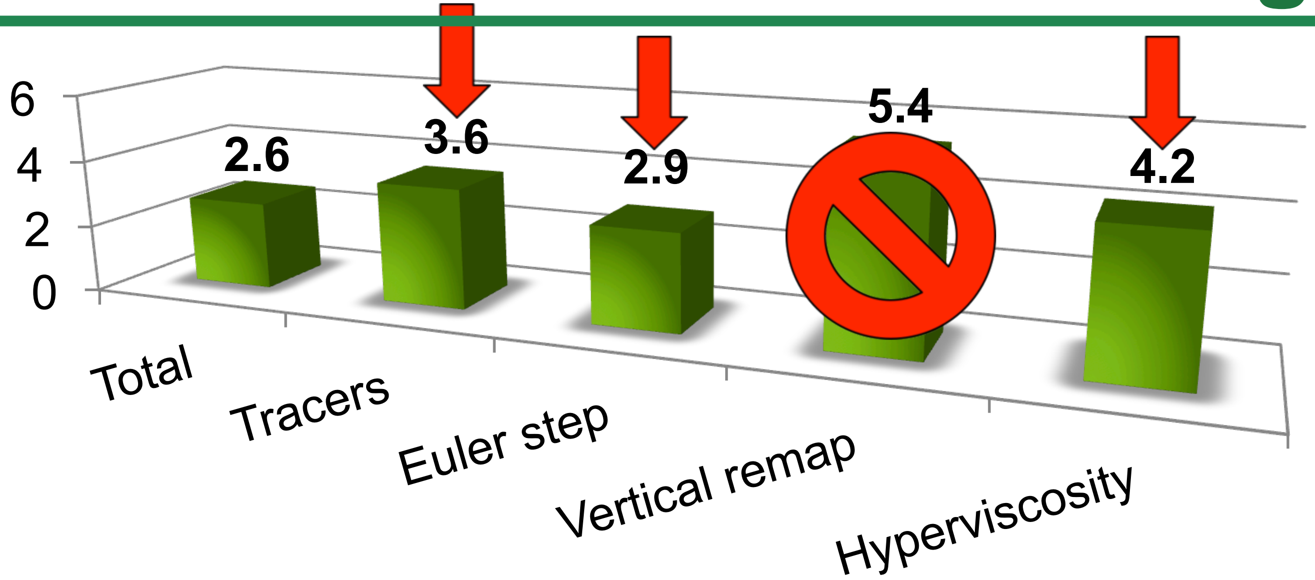
- Vertical Remap basically removed

Back to the Real World: Codes Change



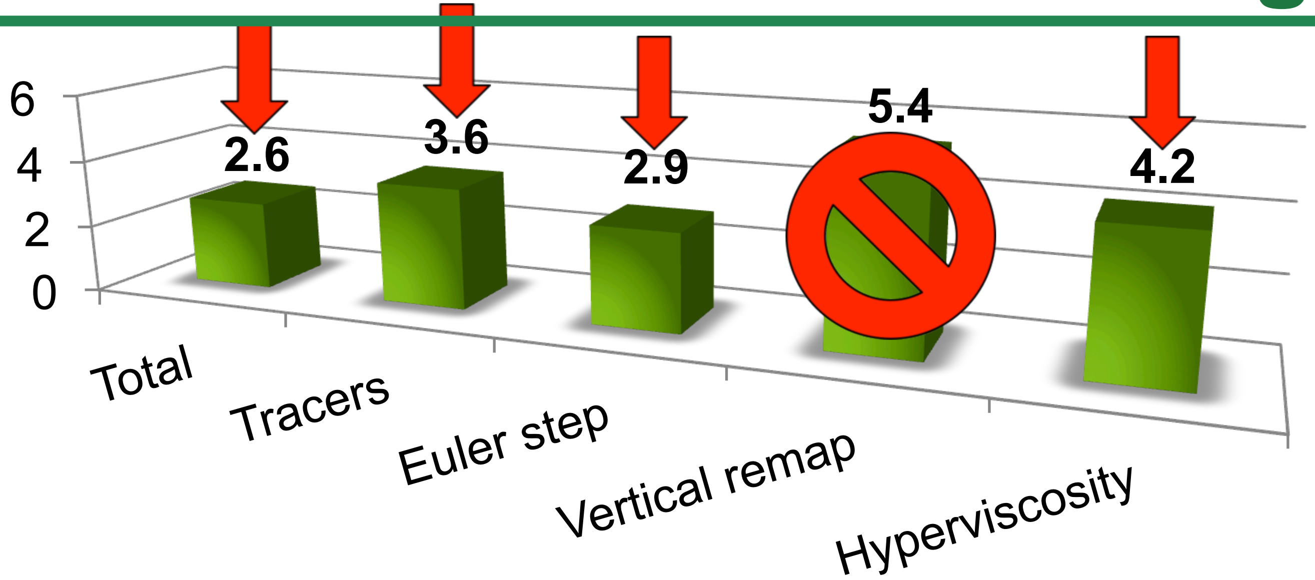
- Vertical Remap basically removed
- New backend for PGI's FORTRAN CUDA

Back to the Real World: Codes Change



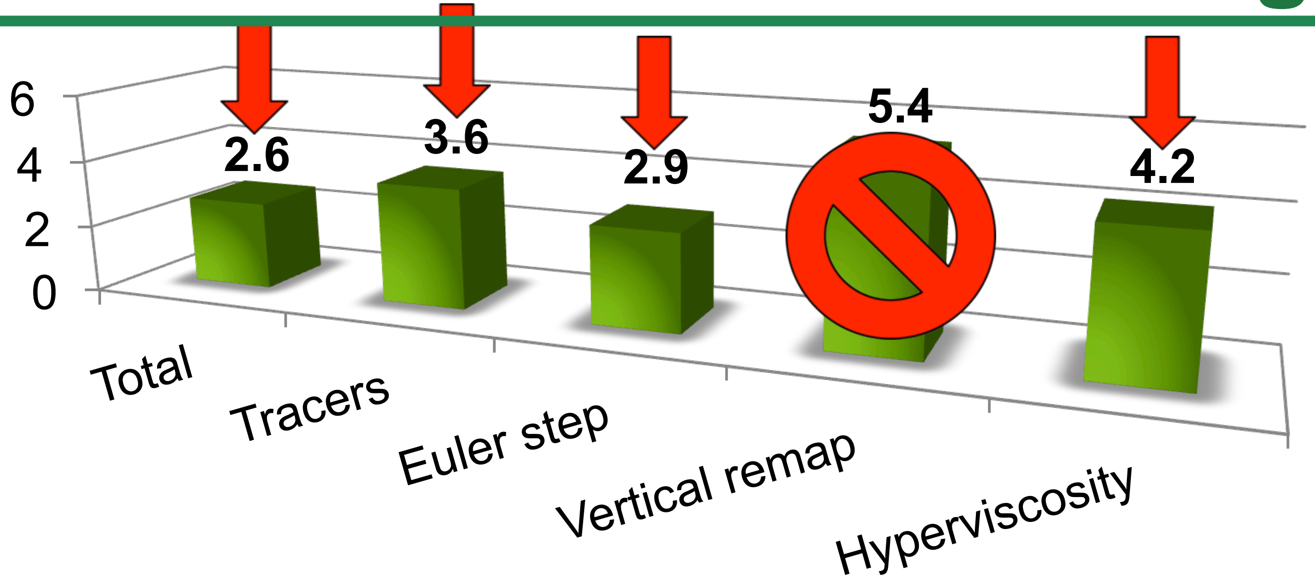
- Vertical Remap basically removed
- New backend for PGI's FORTRAN CUDA
- New sub-cycling methods implemented (More PCI-e traffic)

Back to the Real World: Codes Change



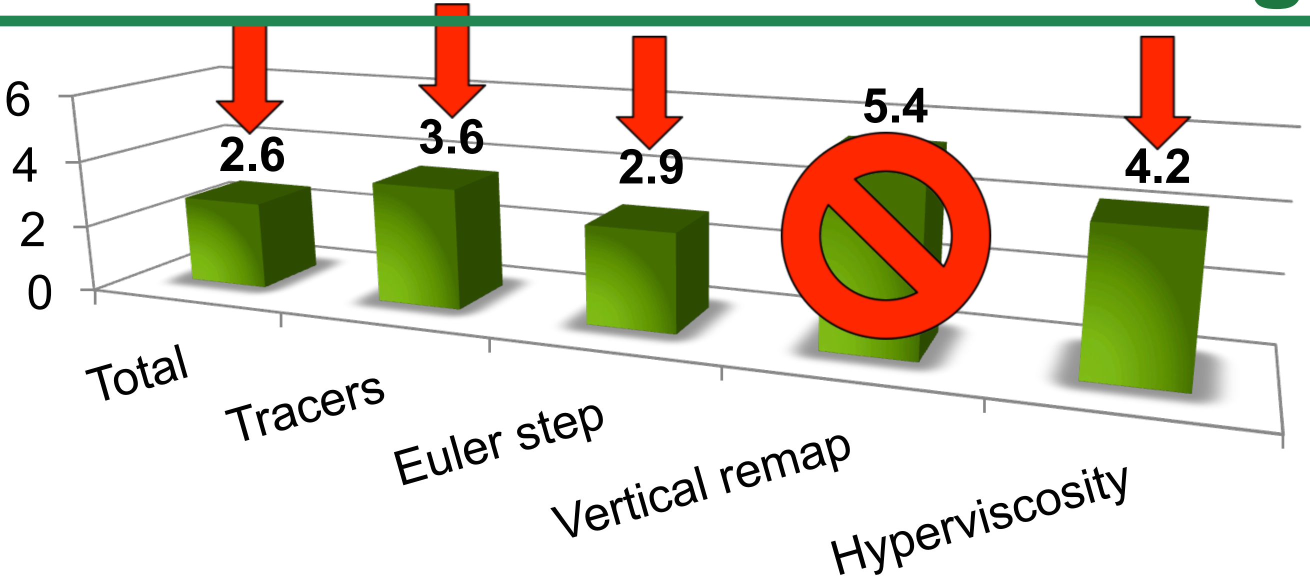
- Vertical Remap basically removed
- New backend for PGI's FORTRAN CUDA
- New sub-cycling methods implemented (More PCI-e traffic)
- New science targets identified

Back to the Real World: Codes Change



- Vertical Remap basically removed
- New backend for PGI's FORTRAN CUDA
- New sub-cycling methods implemented (More PCI-e traffic)
- New science targets identified
- Many communities resistant to code refactoring

Back to the Real World: Codes Change



- Vertical Remap basically removed
- New backend for PGI's FORTRAN CUDA
- New sub-cycling methods implemented (More PCI-e traffic)
- New science targets identified
- Many communities resistant to code refactoring
- Moral of the story: your port must be flexible and maintainable

Next Steps (Joint ACME & OLCF)

- ACME: Accelerated Climate Model for Energy
- Create new better-tuned kernels for newer PGI compiler
- Redo the port using OpenACC
  - OpenACC is very sensitive to code & looping structure
  - Need to discover & disseminate best practices
  - Using best practices, port the “dynamics”
  - Develop new validation suites to maintain GPU port confidence
- Evaluate reproducibility with various Cray & PGI compiler flags
- Efficiently maintain V&V robustness with a changing GPU port

Questions?