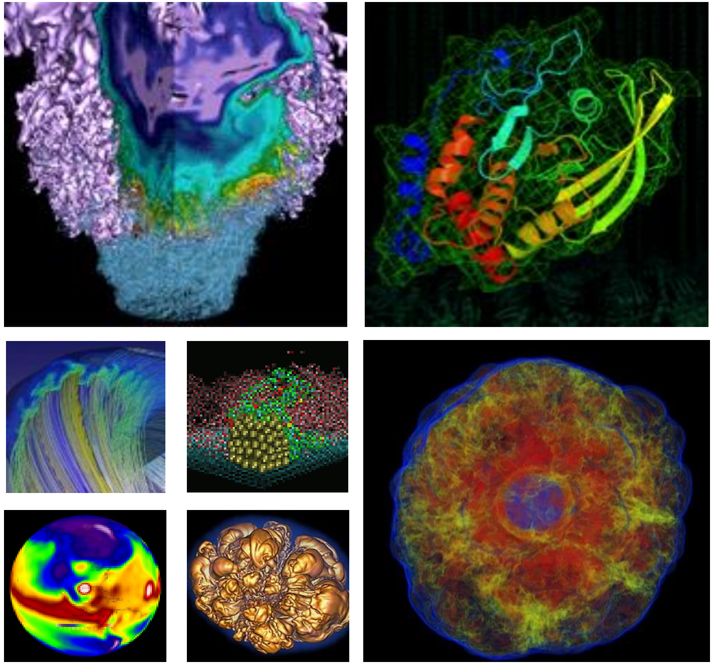
Optimization of the *Model for Prediction Across Scales: Ocean Core* Targeting Production Scale Use of Knights Landing Processor Architecture



NERSC

William Arndt

September 26, 2018
IXPUG Fall Conference
Hillsboro, Oregon

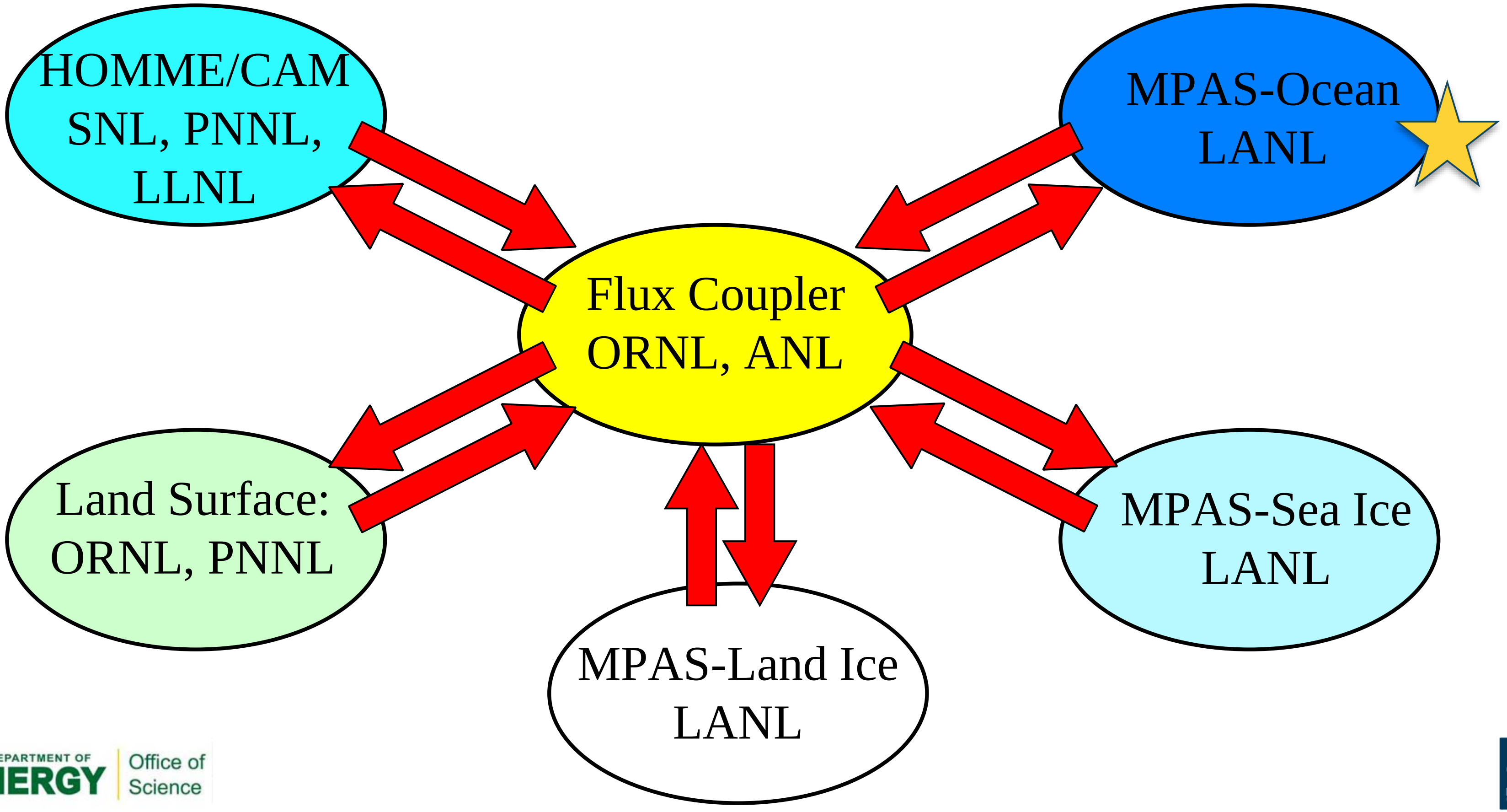
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Project Organization

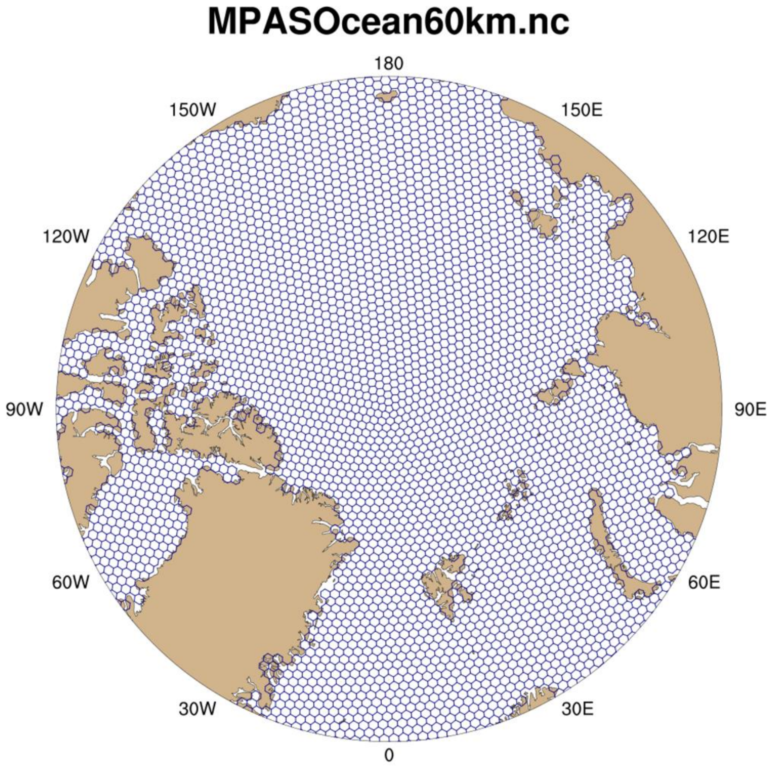
- E^3SM – Energy Exascale Earth System Model
  - Part of ECP initiative at the Department of Energy
  - 300+ contributors over lifetime of project
  - LLNL, PNNL, Sandia, LANL, ORNL, LBNL, Argonne, UCAR
- MPAS framework – Model for Prediction Across Scales
  - 50+ contributors
  - Simulations implemented via unstructured mesh
  - 800,000 lines of Fortran
- NESAP - NERSC Exascale Science Applications Program
  – Fund post docs to adapt high impact applications to KNL

Coupled Climate Model

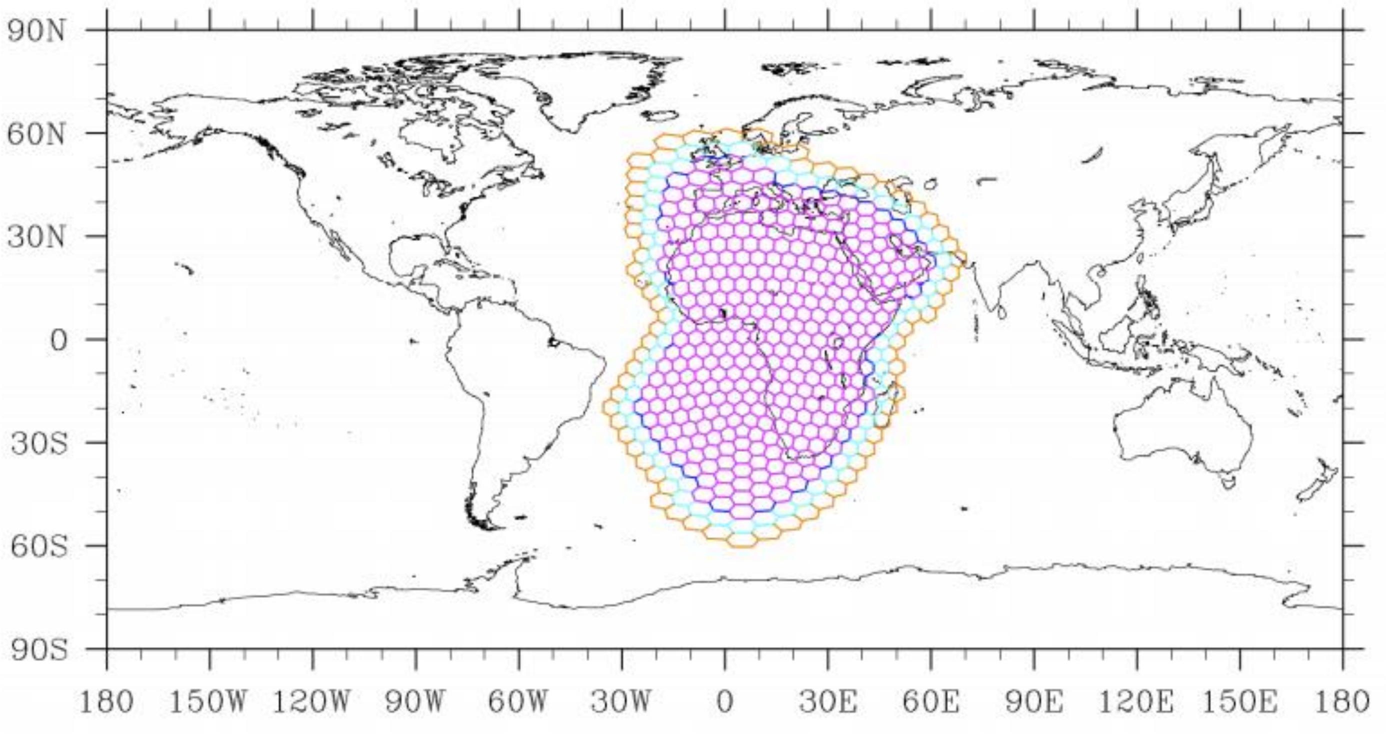


Unstructured Mesh Model of Earth

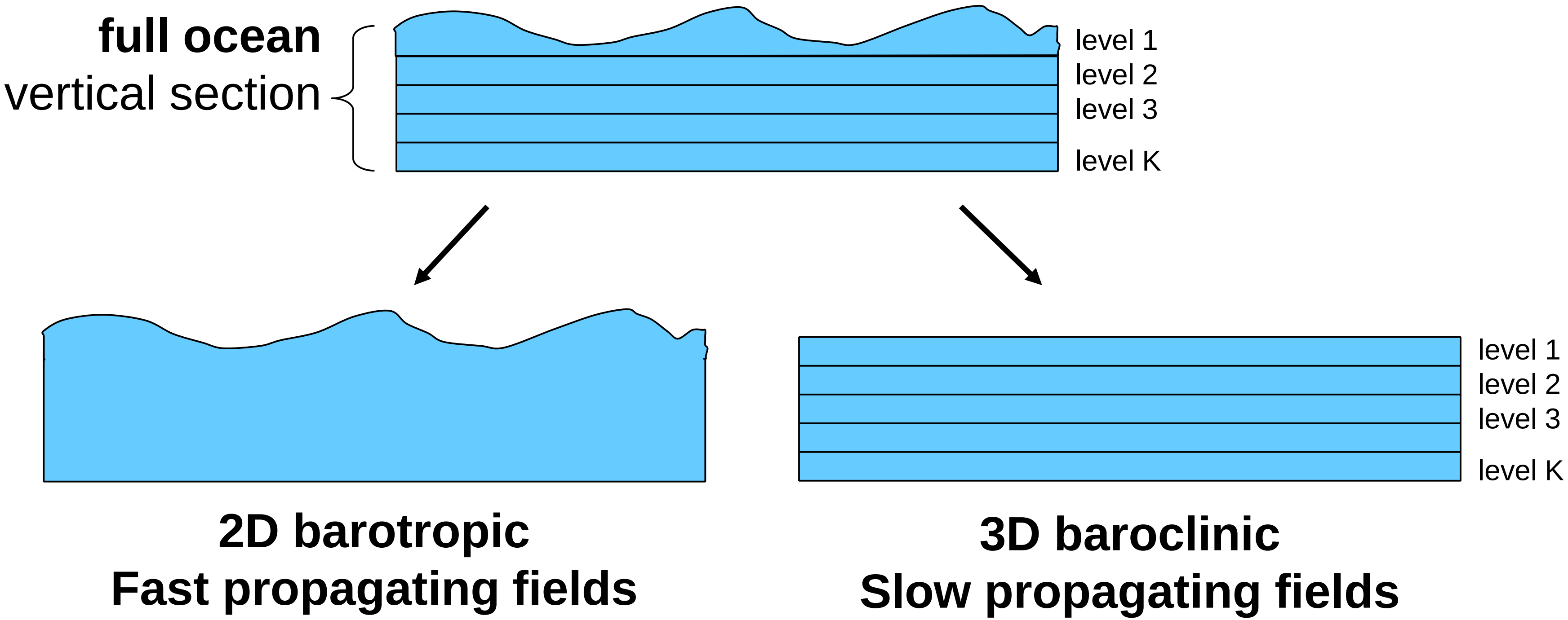
- Low Resolution: 400k Cells
- Med. Resultion: 1.5m Cells
- High Resolution: 6m Cells



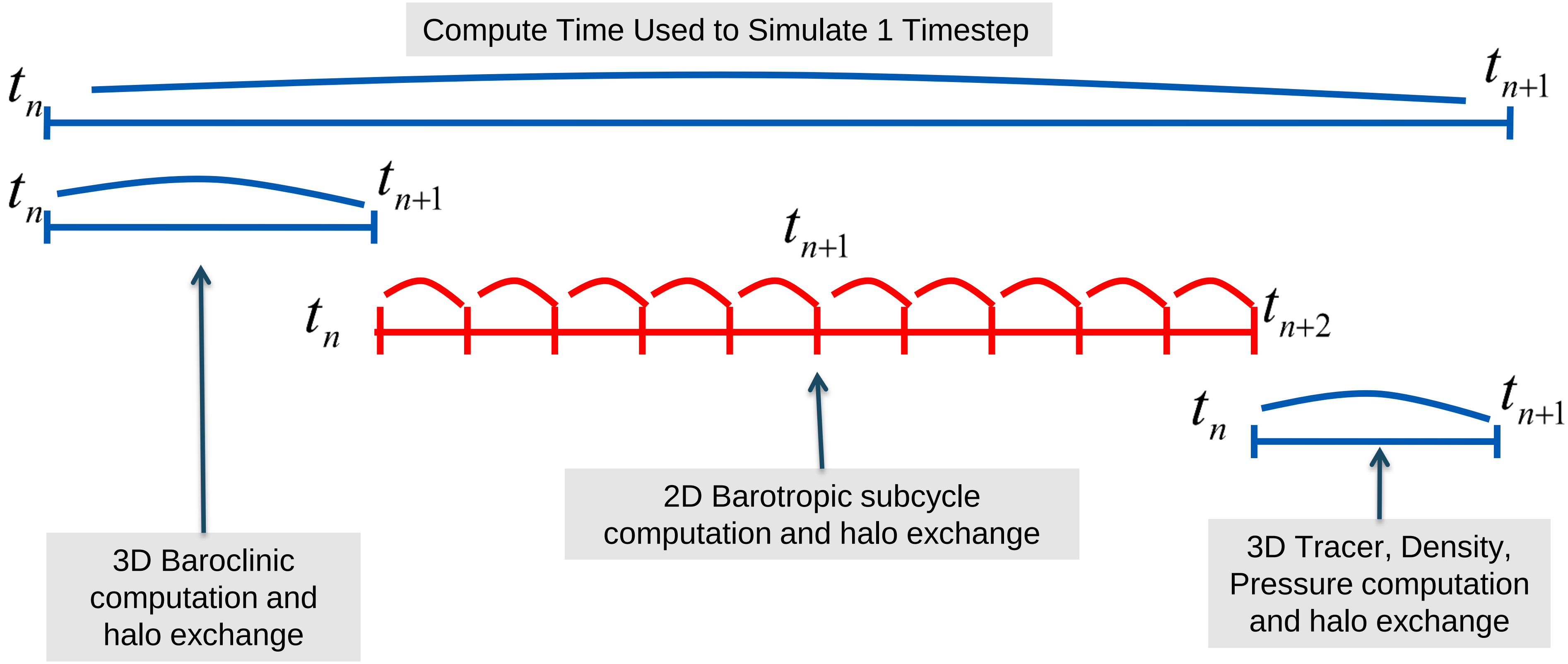
Parallelization by Partition and Halo Exchange



Split-Explicit Integration and Vectorization in Depth



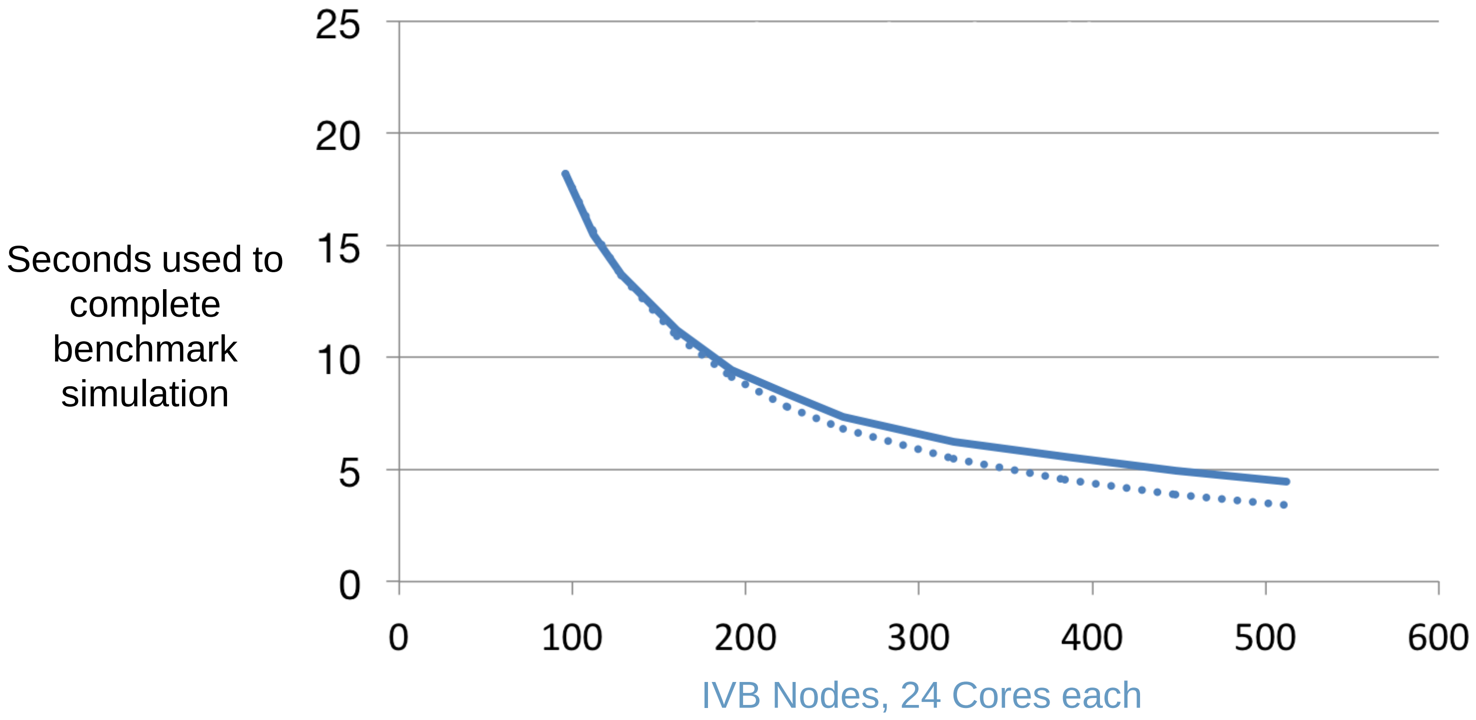
Split-Explicit Integration and Barotropic Subcycling



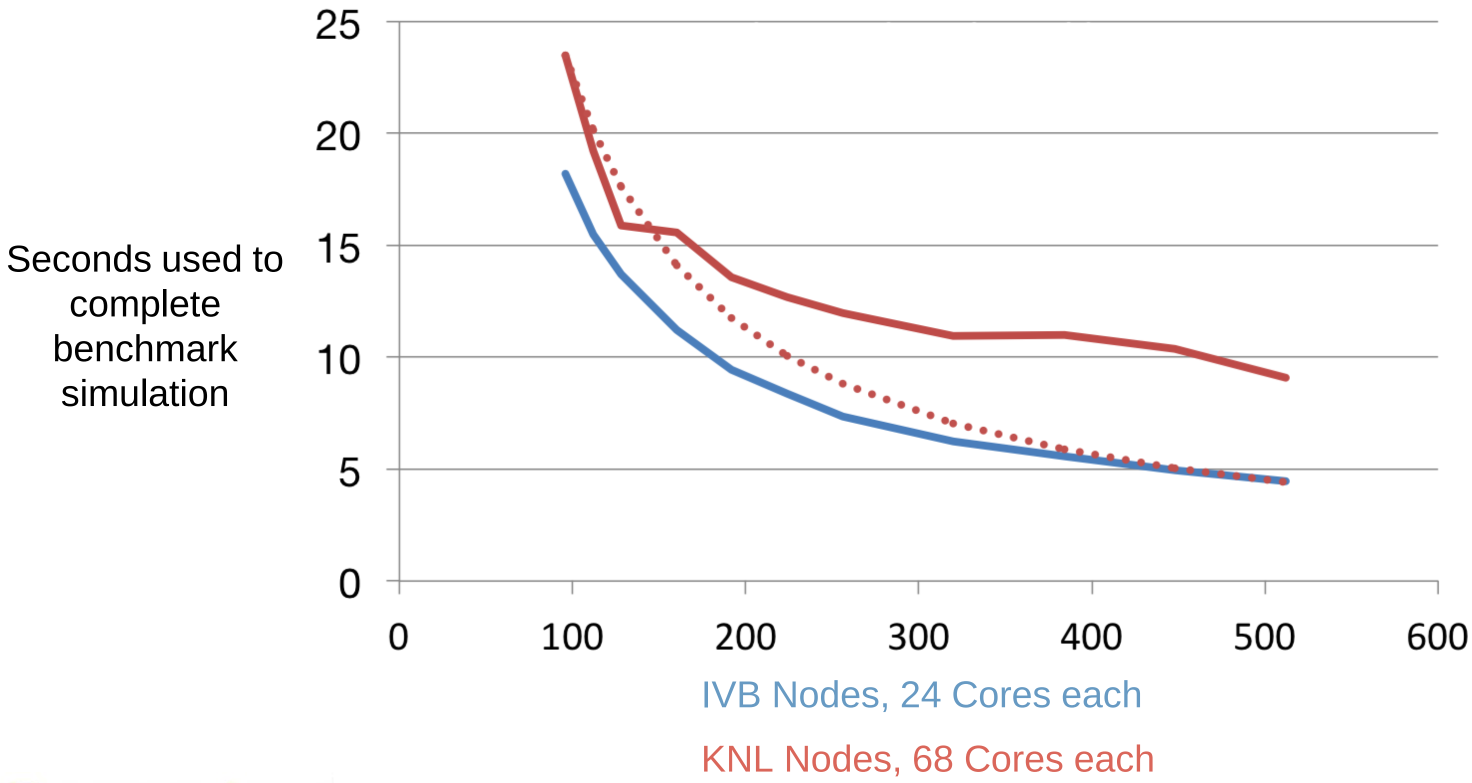
Project Focus

- NERSC systems
  - Edison with Xeon Ivy Bridge (IVB) processors
  - Cori with Xeon Phi Knights Landing (KNL) Processors
- Goals
  - Explain performance disparity between KNL and IVB when MPAS-Ocean runs simulations at production scale (256+ nodes)
  - Find, implement, and push to production optimizations which bring KNL performance on par with IVB

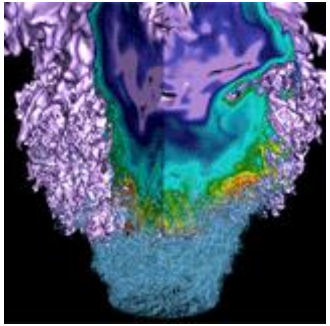
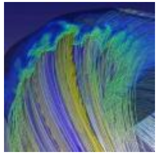
Initial Performance

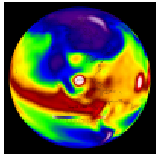
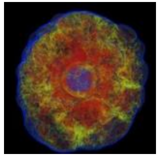
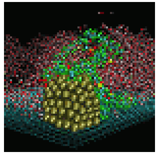


Initial Performance + KNL



Is it load imbalance?

Measuring Load Imbalance

- Can't just measure MPI sync time and be done
  - All Halo exchanges are implemented point to point
  - MPI_Isend, MPI_Irecv
  - Looping through a list to check for expected messages with MPI_Test disguises some imbalance time as compute
- A plan to approximate
  - Wall time is determined by process with most compute. It makes neighbors wait, but does the least waiting itself as messages from it's neighbors are ready sooner
  - The difference between the minimum process wait time and the average wait time is an approximate upper bound on performance gain possible by improving load balance

Measured Load Imbalance

Deviation from average process time spent in communication

12288 IVB processes sorted by communication time

Measured Load Imbalance

11.5% Gain

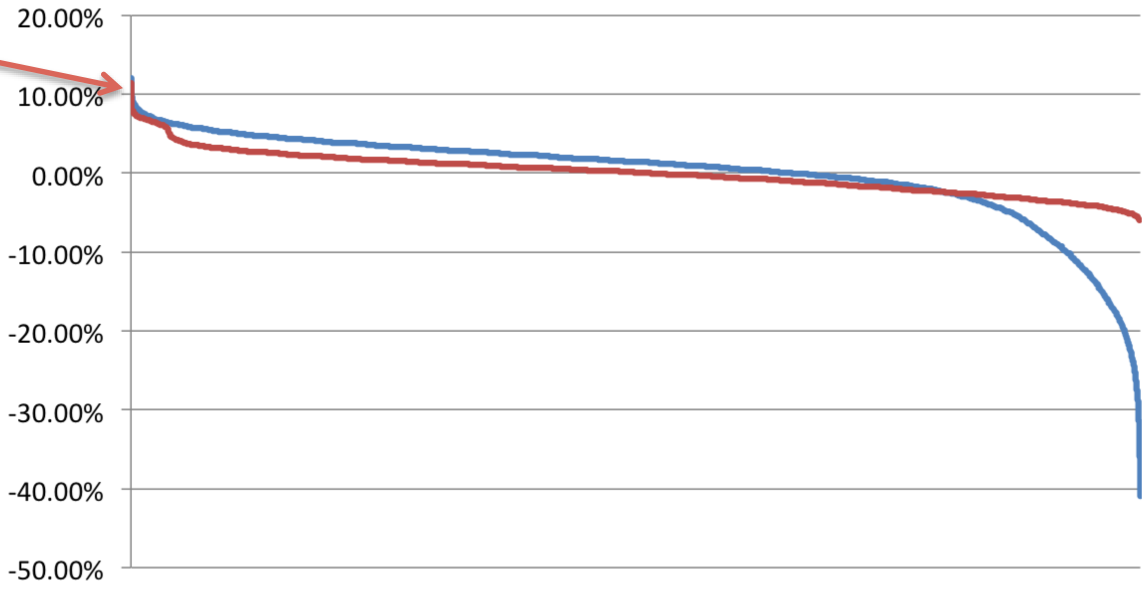
Deviation from average process time spent in communication

12288 IVB processes sorted by communication time

Measured Load Imbalance

11.5% Gain

Deviation from average process time spent in communication



12288 IVB processes sorted by communication time

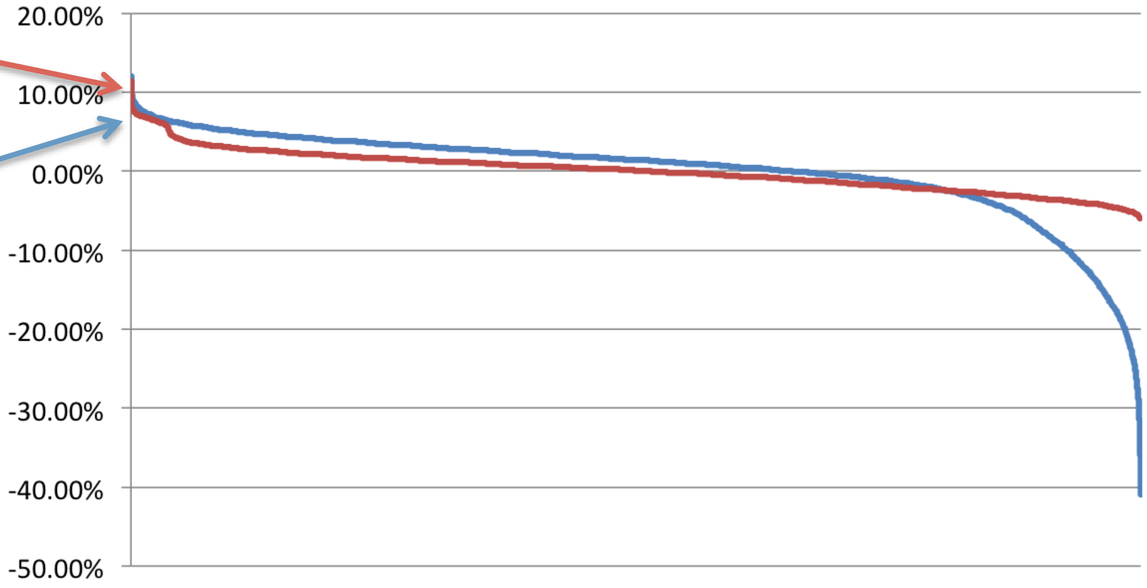
33792 KNL processes sorted by communication time

Measured Load Imbalance

11.5% Gain

12% Gain

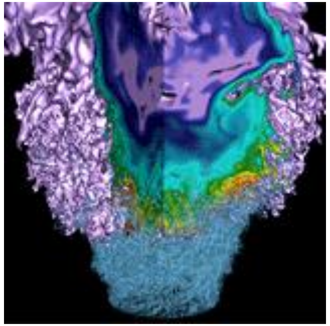
Deviation from average process time spent in communication

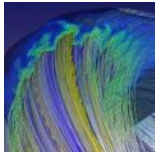


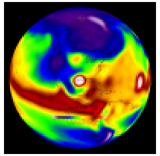
12288 IVB processes sorted by communication time

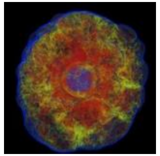
33792 KNL processes sorted by communication time

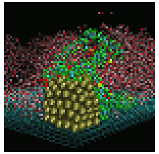
It’s not load imbalance.







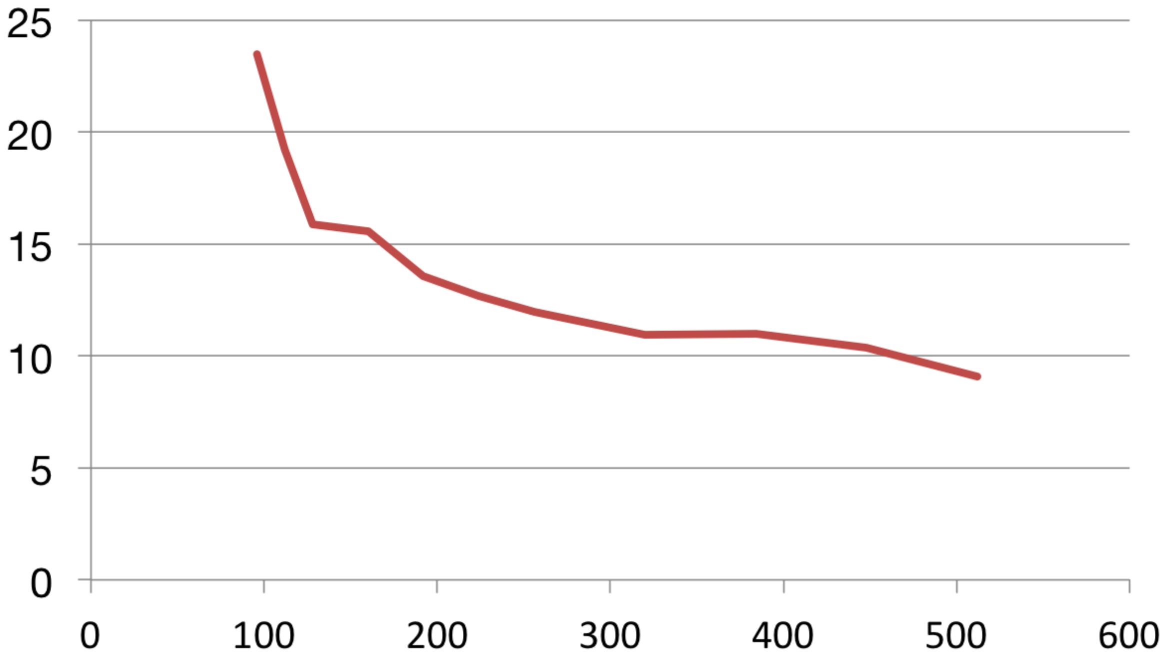




Looking for the bottleneck on KNL

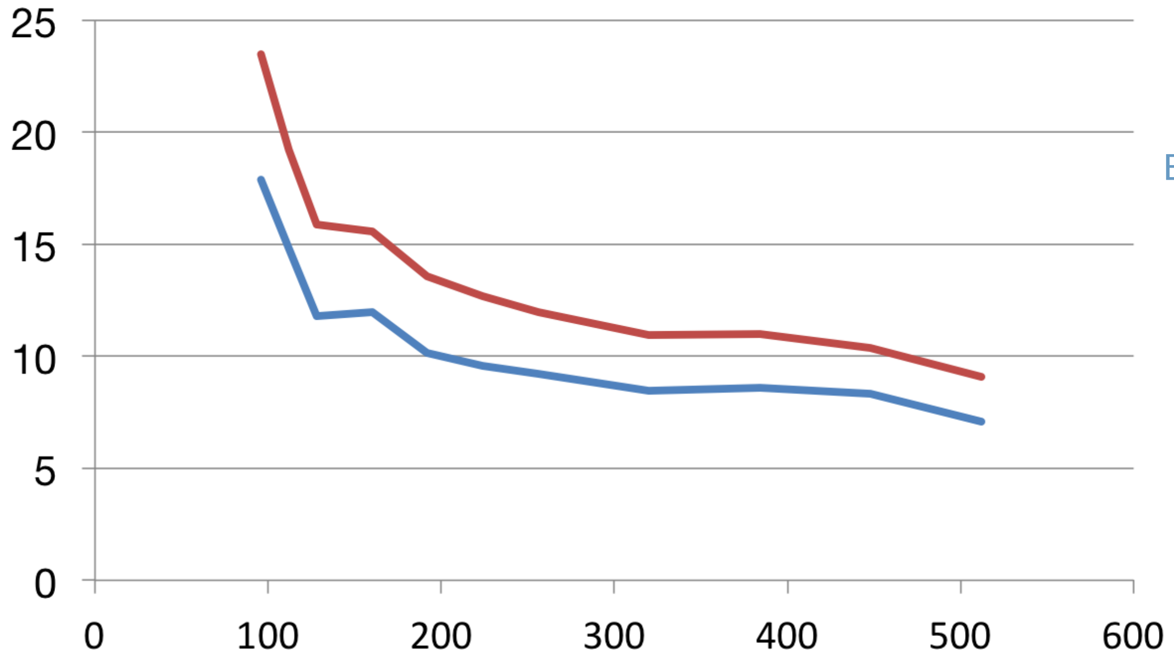
All phases of full time integration

Seconds spent in execution phase during complete benchmark simulation



KNL Nodes, 68 Cores each

Looking for the bottleneck on KNL



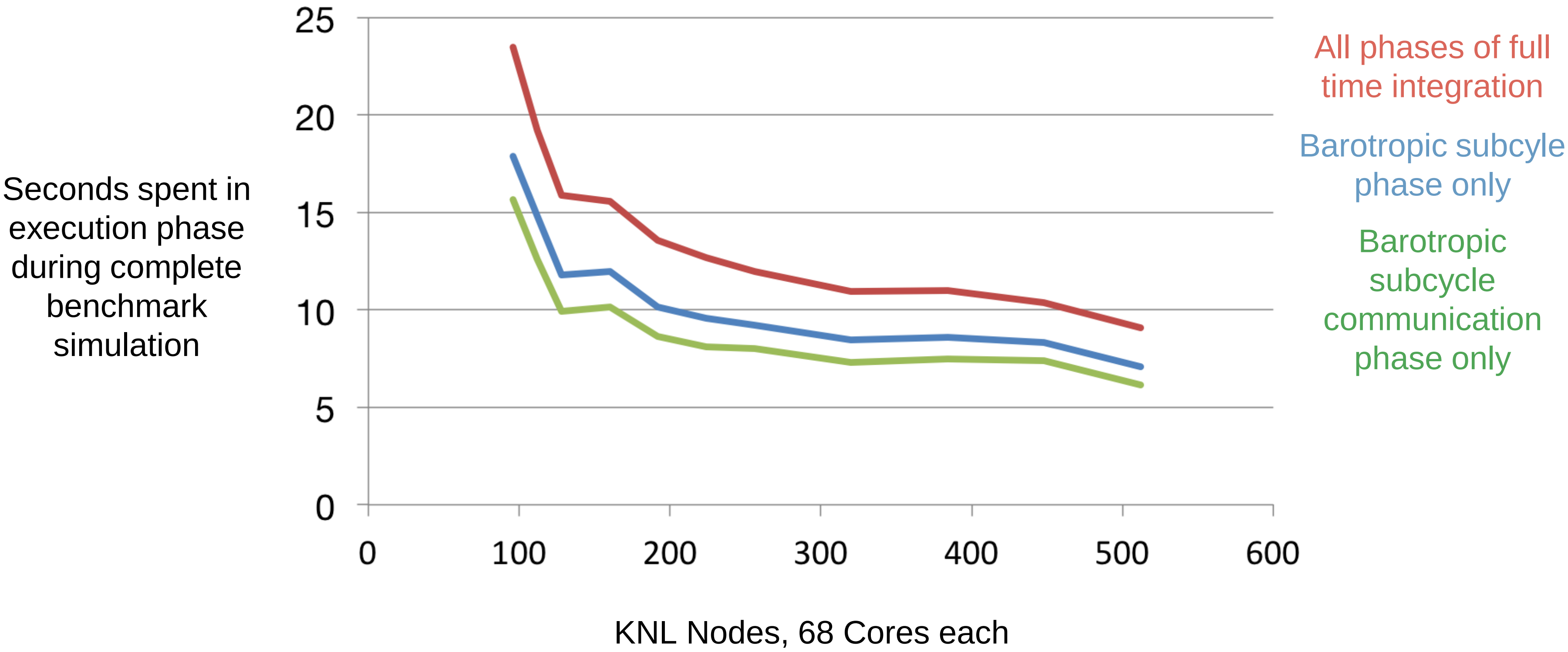
Seconds spent in execution phase during complete benchmark simulation

KNL Nodes, 68 Cores each

All phases of full time integration

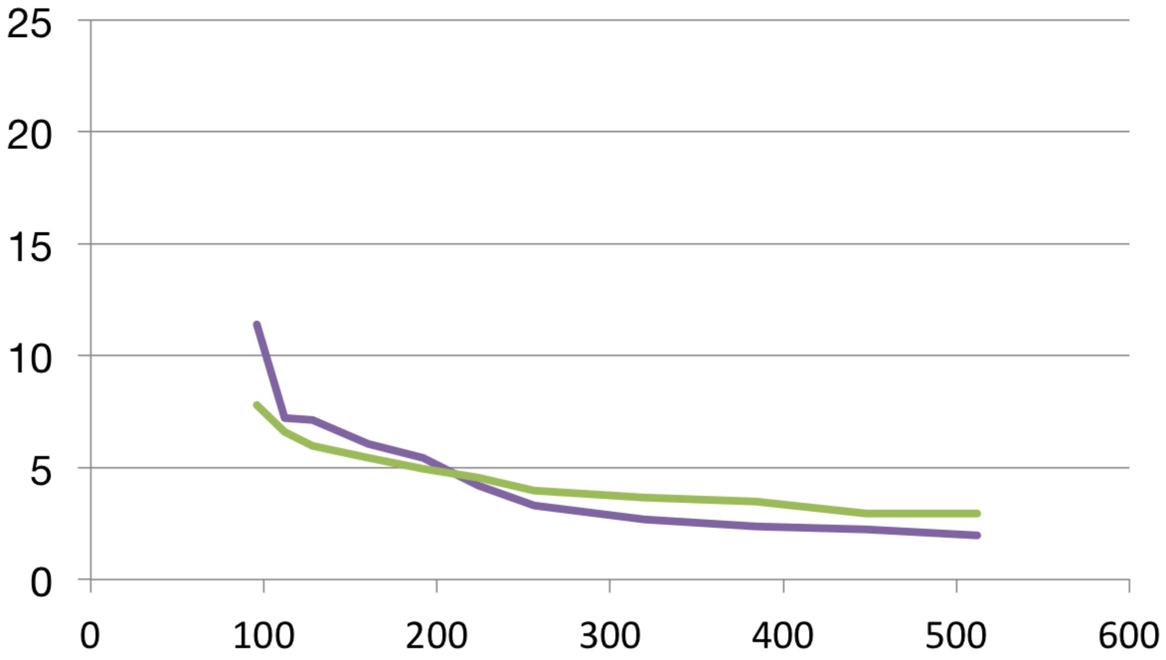
Barotropic subcyle phase only

Looking for the bottleneck on KNL



IVB vs. KNL: Non BTR subcycle halo

Seconds spent not in Barotropic subcycle halo exchange phase during complete benchmark simulation

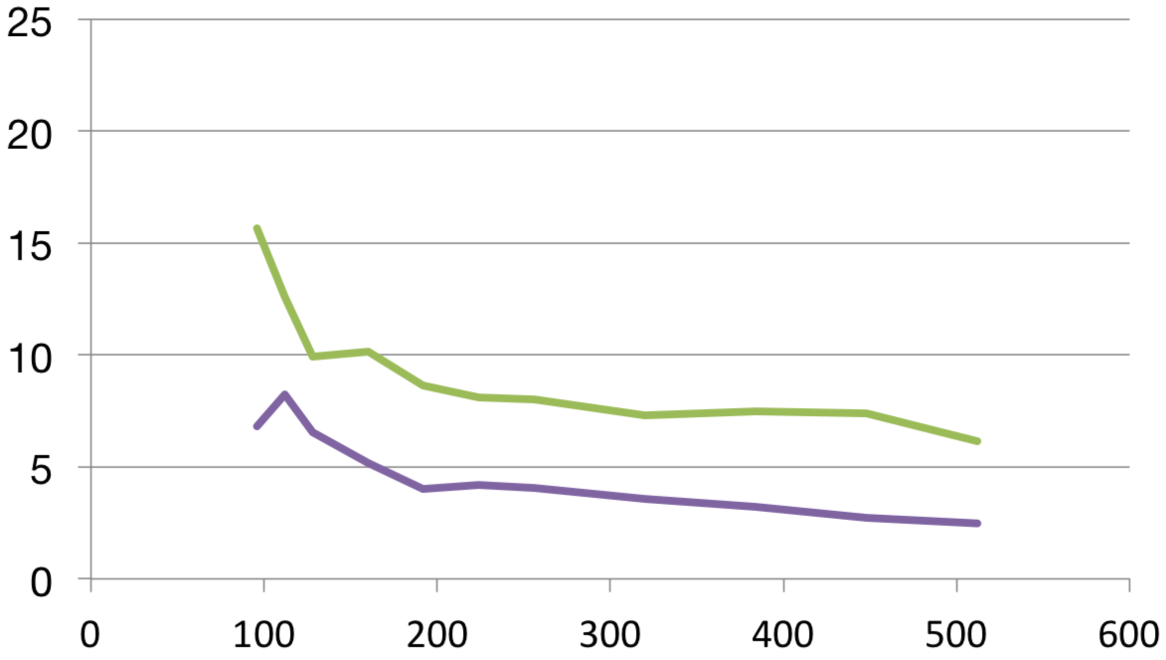


KNL Nodes, 68 Cores each

IVB Nodes, 24 Cores each

IVB vs. KNL: BTR subcycle halo only

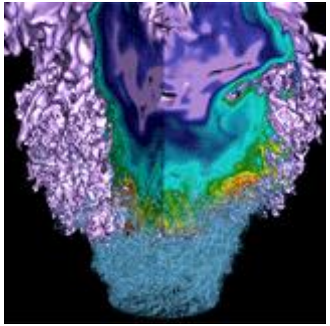
Seconds spent in Barotropic subcycle halo exchange phase during complete benchmark simulation

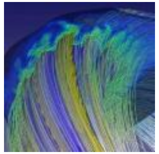


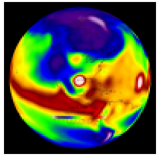
KNL Nodes, 68 Cores each

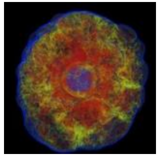
IVB Nodes, 24 Cores each

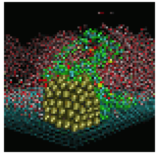
The barotropic subcycle halo exchange is why IVB outperforms KNL.











NERSC

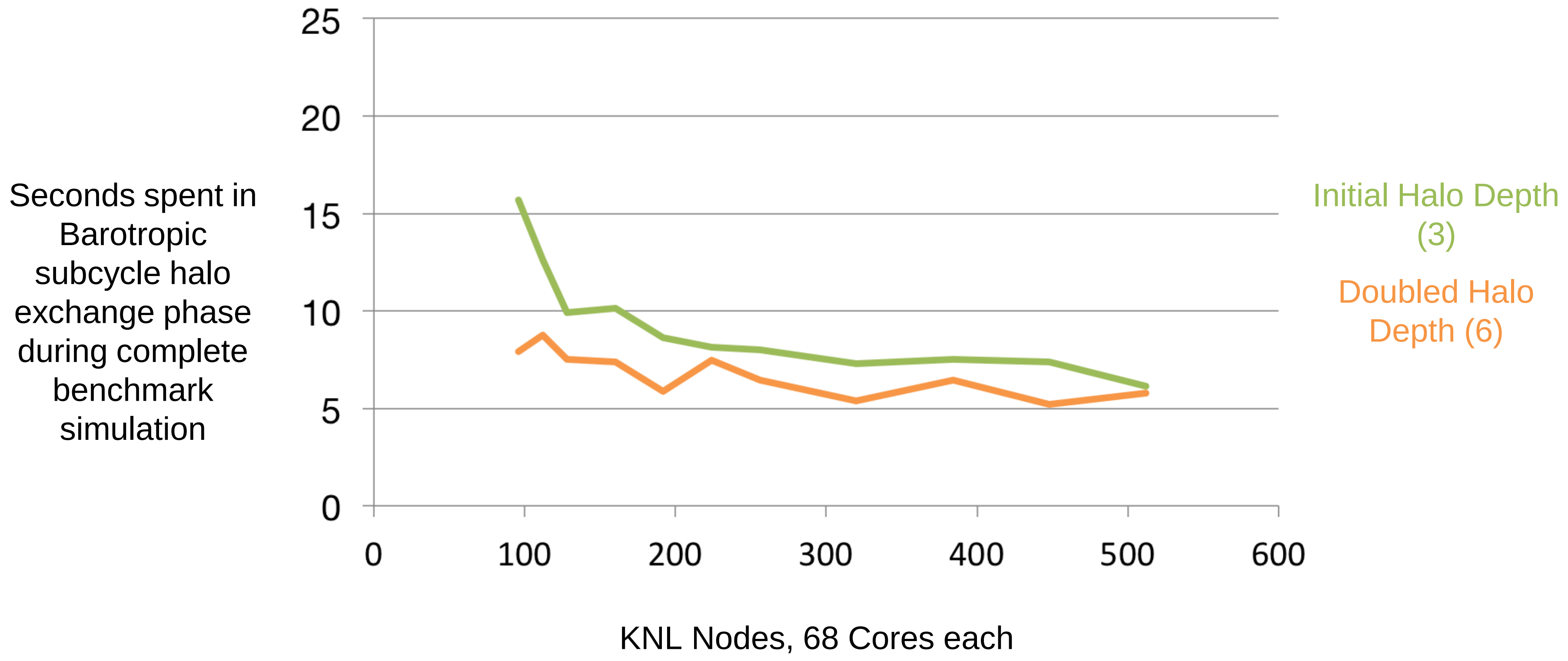
U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

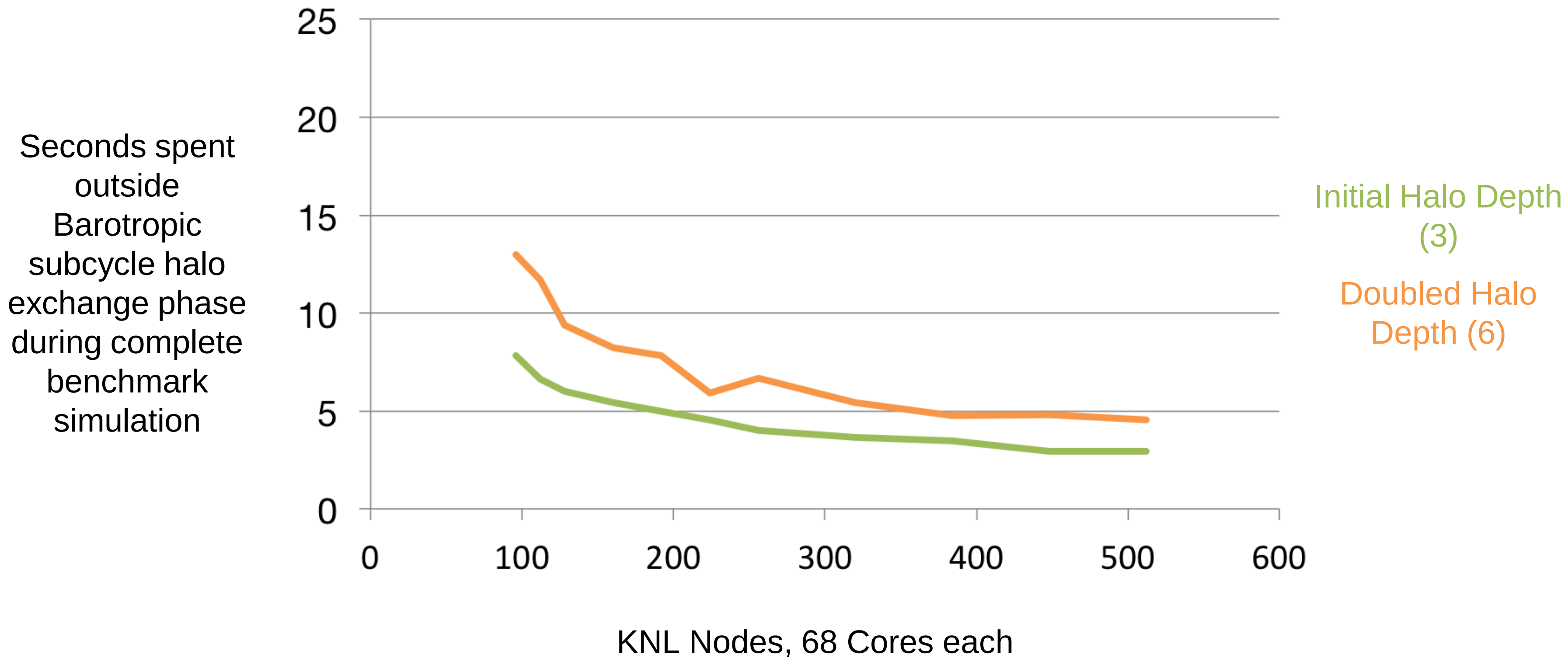
Trying to reduce halo exchanges

- Try deeper halos
  - Doubling the depth of the halo exchange enables two full timesteps per single halo exchange
  - Framework allows easy configuration of halo depth (but globally)
- Drawbacks
  - Some compute at the edges of partitions is duplicated between neighbor processes (but we have plenty of compute to spare)
  - Sending more than twice as much data; more message packing and larger messages over MPI
  - Barotropic subcycle easy to adapt, but a full implementation would require significant labor to convert the rest of the Ocean Core

Normal halo vs. deep: Gains

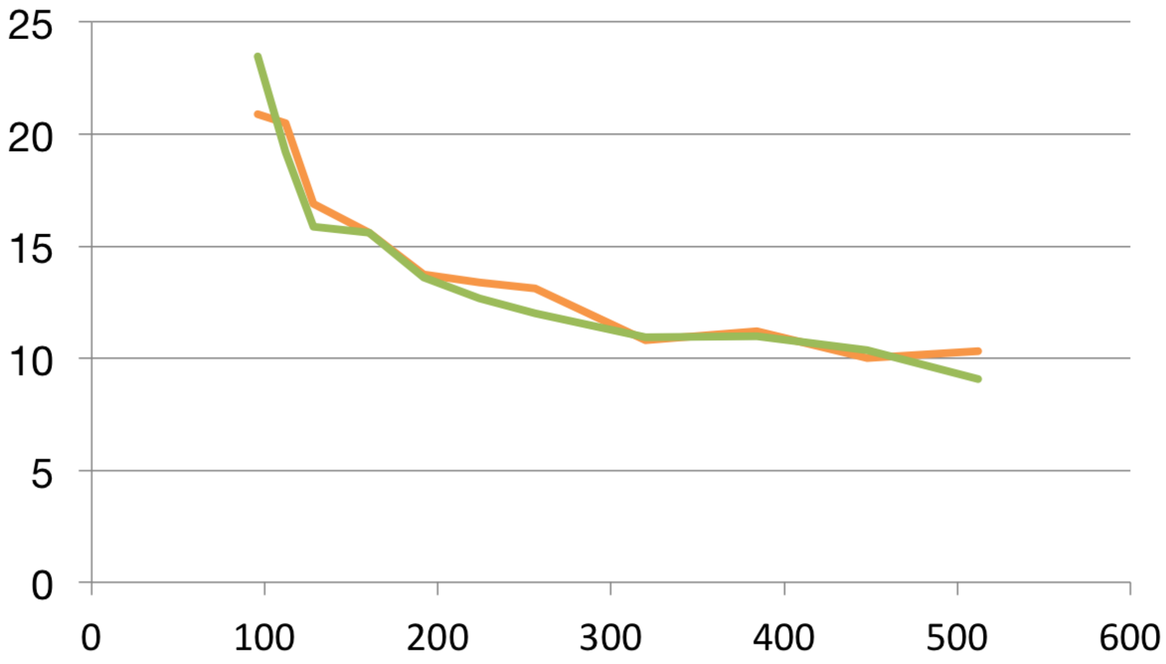


Normal halo vs. deep: Losses



Normal halo vs. deep: It's a wash.

Seconds spent in execution phase during complete benchmark simulation

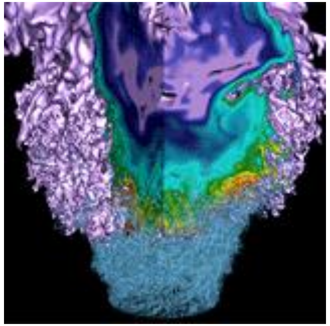


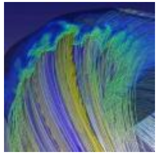
Initial Halo Depth (3)

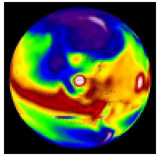
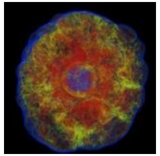
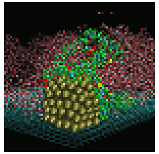
Doubled Halo Depth (6)

KNL Nodes, 68 Cores each

Looking for waste in the framework





NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Threading directives on inner loops

```
commListPtr => exchangeGroup % sendList
do while ( associated(commListPtr) )
   fieldCursor => field
   do while ( associated(fieldCursor) )
      exchListPtr => fieldCursor % sendList % halos(haloLayer) % exchList
      do while ( associated(exchListPtr) )
         if ( exchListPtr % endPointID == commListPtr % procID ) then
            !$omp do schedule(runtime) private(iBuffer)
               do iExch = 1, exchListPtr % nList
                  ! Work to pack communications buffers
               end do
            !$omp end do
         end if
         exchListPtr => exchListPtr % next
      end do
      fieldCursor => fieldCursor % next
   end do
   !$omp master
      ! work to track iBuffer position progress during this iteration
   !$omp end master
   call mpas_threading_barrier()
   commListPtr => commListPtr % next
end do
```

Threading directives on inner loops

```
commListPtr => exchangeGroup % sendList
do while ( associated(commListPtr) )
   fieldCursor => field
   do while ( associated(fieldCursor) )
      exchListPtr => fieldCursor % sendList % halos(haloLayer) % exchList
      do while ( associated(exchListPtr) )
         if ( exchListPtr % endPointID == commListPtr % procID ) then
            !$omp do schedule(runtime) private(iBuffer)
               do iExch = 1, exchListPtr % nList
                  ! Work to pack communications buffers
               end do
            !$omp end do Hundreds of implied OMP barriers here
         end if
         exchListPtr => exchListPtr % next
      end do
      fieldCursor => fieldCursor % next
   end do
   !$omp master
      ! work to track iBuffer position progress during this iteration
   !$omp end master
   call mpas_threading_barrier()
   commListPtr => commListPtr % next
end do
```

Pull OMP directive to outer loop

```
commListPtr => exchangeGroup % sendList
commListSize = commListPtr % commListSize
!$omp do private( commListPtr, ... )
   do listItem = 1, commListSize
      commListPtr => exchangeGroup % sendList
      do listPosition = 2, listItem
         commListPtr => commListPtr % next
      end do
      bufferOffset = commListPtr % bufferOffset
      fieldCursor => field
      ! Same inner loop over fieldCursor
         ! Same inner loop over exchList
            ! Same loop over iExch, but no OMP directive
               ! New Inner loop over memoized size of buffer section to be copied
      ! work to track buffer position progress for this message
      commListPtr => commListPtr % next
   end do
!$omp end do
```

Anatomy of a halo exchange

```
subroutine mpas_dmpar_exch_group_full_halo_exch(domain, groupName, iErr)
   ...
   call mpas_dmpar_exch_group_build_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_start_recv(domain % dminfo, exchGroupPtr)
   call mpas_dmpar_exch_group_pack_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_start_send(domain % dminfo, exchGroupPtr)
   call mpas_dmpar_exch_group_local_exch_fields(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_unpack_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_destroy_buffers(exchGroupPtr)
   ...
```

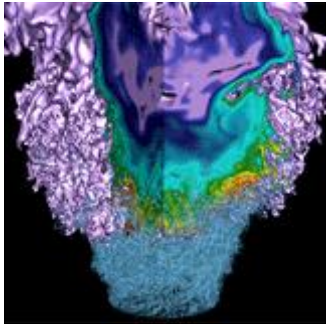
Disection of a halo exchange

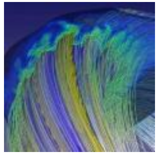
```
subroutine mpas_dmpar_exch_group_full_halo_exch(domain, groupName, iErr)
   ...
   call mpas_dmpar_exch_group_build_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_start_recv(domain % dminfo, exchGroupPtr)
   call mpas_dmpar_exch_group_pack_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_start_send(domain % dminfo, exchGroupPtr)
   call mpas_dmpar_exch_group_local_exch_fields(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_unpack_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_destroy_buffers(exchGroupPtr)
   ...
```

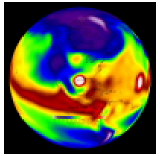
Replacement of a halo exchange

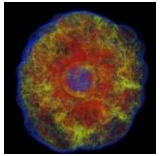
```
function ocn_forward_mode_init(domain, startTimeStamp) result(ierr)
   ...
   call mpas_dmpar_exch_group_create(domain, 'subcycleFields')
   call mpas_dmpar_exch_group_add_field(domain, 'subcycleFields', 'sshSubcycle')
   call mpas_dmpar_exch_group_add_field(domain, 'subcycleFields', 'normalBarotropicVelocitySubcycle')
   call mpas_dmpar_exch_group_build_reusable_buffers(domain, 'subcycleFields')

subroutine mpas_dmpar_exch_group_reuse_halo_exch(domain, groupName, timeLevel, haloLayers, iErr)
   ...
   call mpas_dmpar_exch_group_start_recv(domain % dminfo, exchGroupPtr)
   call mpas_dmpar_exch_group_pack_buffers(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_start_send(domain % dminfo, exchGroupPtr)
   call mpas_dmpar_exch_group_local_exch_fields(domain % allFields, exchGroupPtr)
   call mpas_dmpar_exch_group_unpack_buffers(domain % allFields, exchGroupPtr)
   ...
function ocn_forward_mode_finalize(domain) result(iErr)
   call mpas_dmpar_exch_group_destroy_reusable_buffers(domain, 'subcycleFields')
```

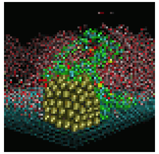
Results





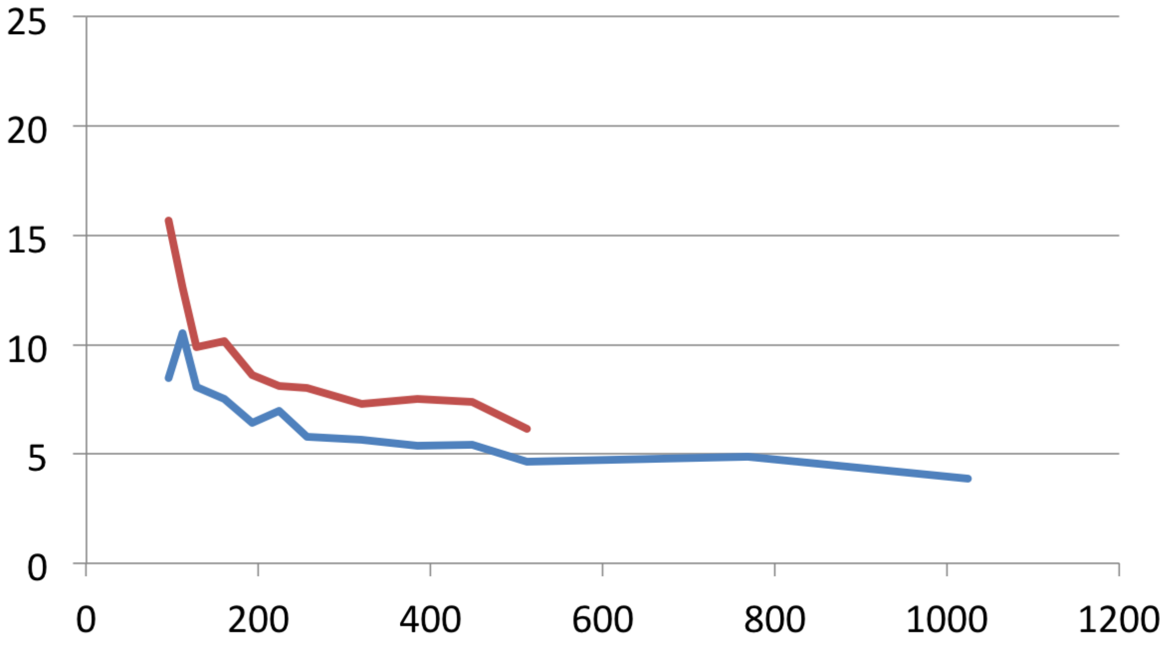






KNL initial vs. modified: sub halo only

Seconds used during Barotropic subcycle halo exchange phase when running benchmark simulation

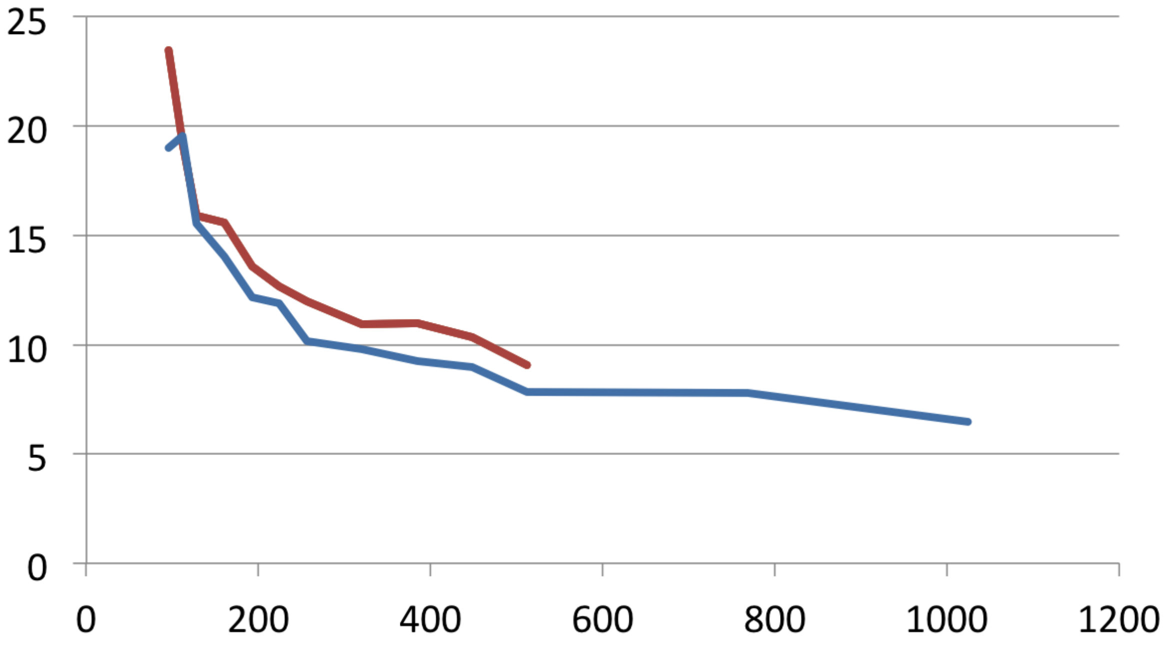


KNL Nodes, 68 Cores each, initial codebase

KNL Nodes, 68 Cores each, modified codebase

KNL modified vs. initial

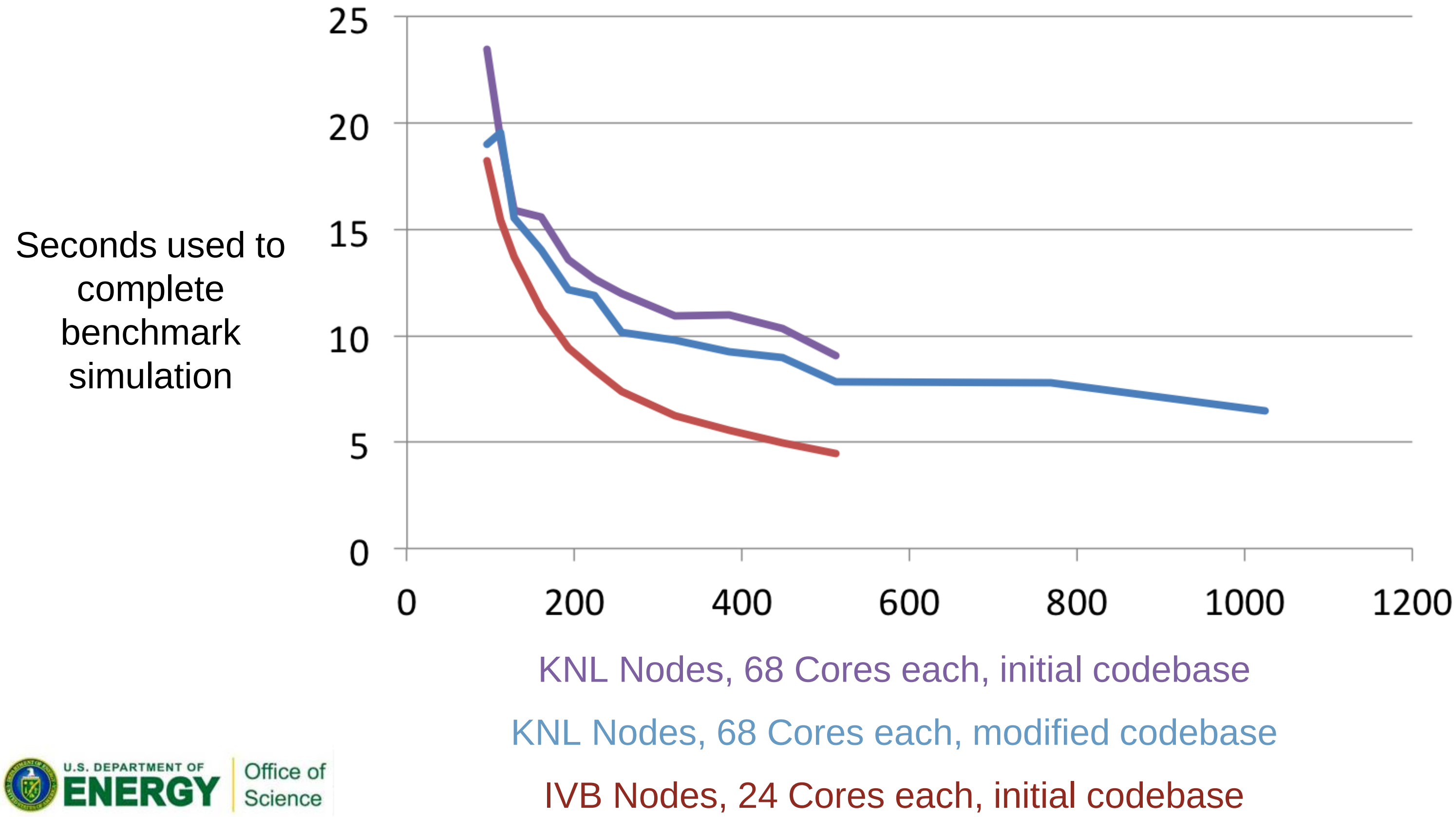
Seconds used to complete benchmark simulation



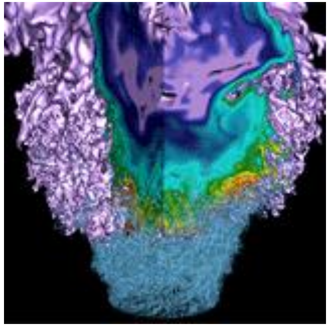
KNL Nodes, 68 Cores each, initial codebase

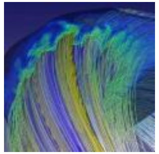
KNL Nodes, 68 Cores each, modified codebase

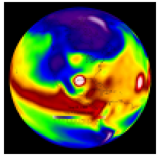
KNL modified vs. IVB

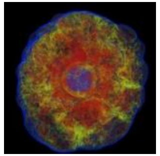


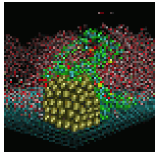
Conclusions











Outcome of goals

- Performance goal: Failed
  - Needed 2x speedup on KNL to match IVB performance
  - Only achieved 1.15x speedup
  - E^3SM only runs 3% faster
  - IVB best configuration doesn’t use threading so no gain there
- Explain why it is slower on KNL.
  - Need much better threading.
    - 210 serial thread sections in framework code
  - KNL needs more processes per node, which inflates total message packing for halo exchanges, which overwhelms memory system

If I had a million FTEs I would…

- Extend framework to address negative halo layers, use to overlap compute and communication
- Build a mapping and add some extra loops to pack halo exchange messages during compute
- Every framework data structure should not be a linked list
- Follow up on potential from deeper halo exchanges:
  - Modify configurations to support different halo sizes for different fields -OR-
  - Adapt entire ocean core (and the others) to support deeper halo exchanges

Credit to:

- Mark Peterson – Los Alamos National Lab
- Philip Jones – Los Alamos National Lab
- Sam Williams – Lawrence Berkeley National Lab
- Leonid Oliker – Lawrence Berkeley National Lab
- Noel Keen – Lawrence Berkeley National Lab
- Brian Friesen – NERSC
- Helen He - NERSC
- The NESAP program at NERSC

Thank You

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB