Efforts of Xeon Phi use for CESM

Srinath Vadlamani
Youngsung Kim
and John Dennis
*Mike Levy and Jim Edwards
ASAP-TDD-CISL
National Center for Atmospheric Research

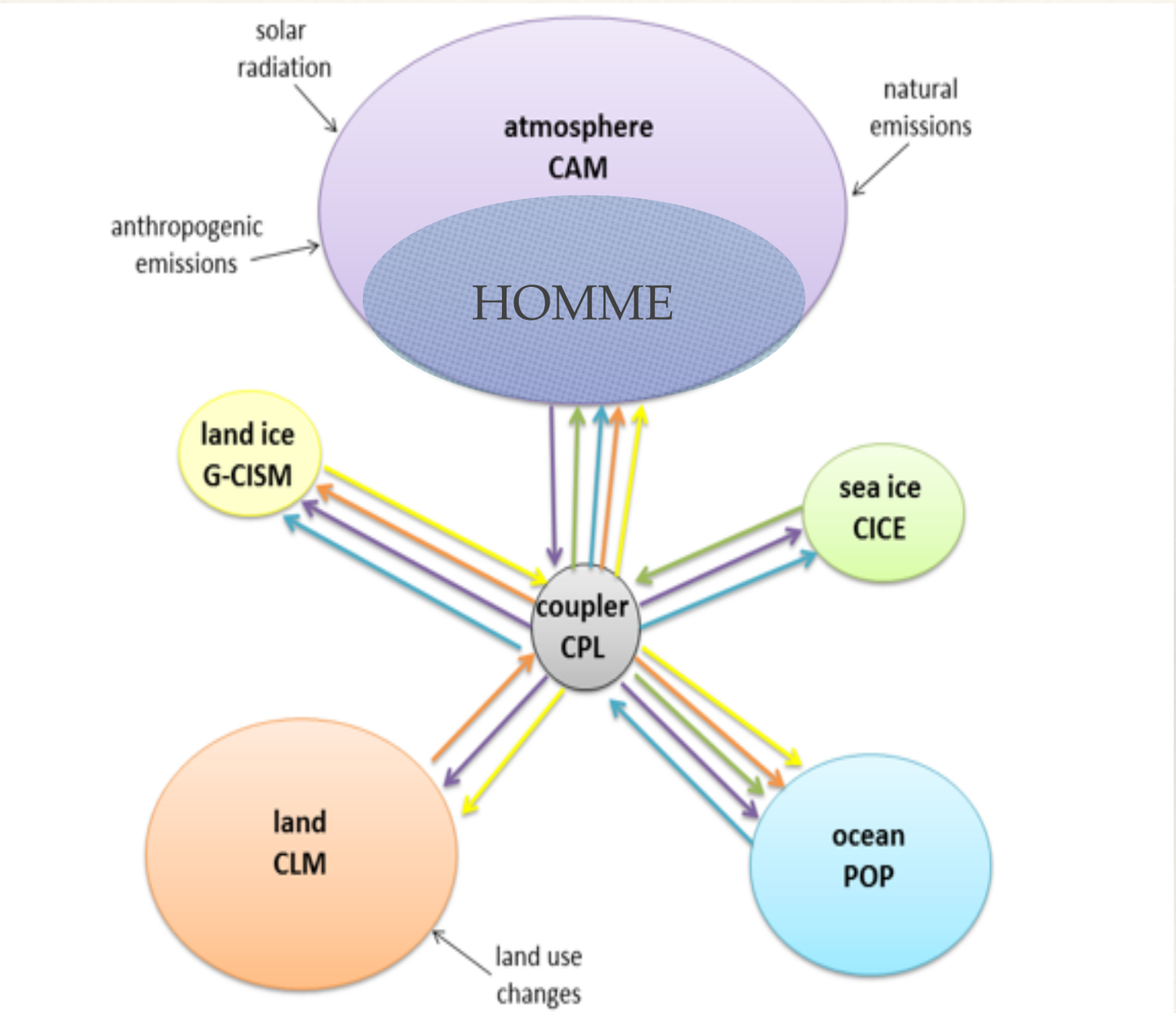
This effort only works with a collaborative effort

- Intel's Early Access Program got us up and going quickly.
  - Weekly teleconferences have led to both compiler improvements, code and even OS improvements.
- Access to Stampede positioned CESM to run on the Xeon Phi in production mode.
  - Kudos for Stampede infrastructure.
    - We've adopted something similar to ibrun.symm and the file mounting paradigm on NCAR's KNC cluster

We need better science throughput for climate codes.

- Climate simulations simulate 100s to 1000s of years of activity.
  - Currently high resolution climate simulations rate is 2 ~ 3 simulated year per day (SYPD) [~40k pes].
- NCAR climate code primarily uses explicit time stepping which means speed up primarily comes from faster calculations.
- We want to use the full range of HPC micro-architectures which adds to the challenge.
- We must use these architectures efficiently for successful SYPD speed up, which requires knowing the hardware!

Community Earth Systems Model (CESM)



- Coupled framework for components consisting of different models.
- >1.5M lines of fortran code
- Many collaborators with acute accuracy requirements.
- Few standalone drivers
  - Kernel extractor (KGEN) is addressing this.
- HOMME (dynamical core) + atmosphere physics and chemistry + active land (CLM) = FC5 ~ CAM-SE
- FIDEAL is only dynamical core with all fluxes from data files [HOMME + coupler].
- Scientist want a "push-button" code.
  - *Tuning* application must eventually be hidden in application configure scripts. This is very very hard.

Porting to Xeon Phi easy to start and difficult to finish.

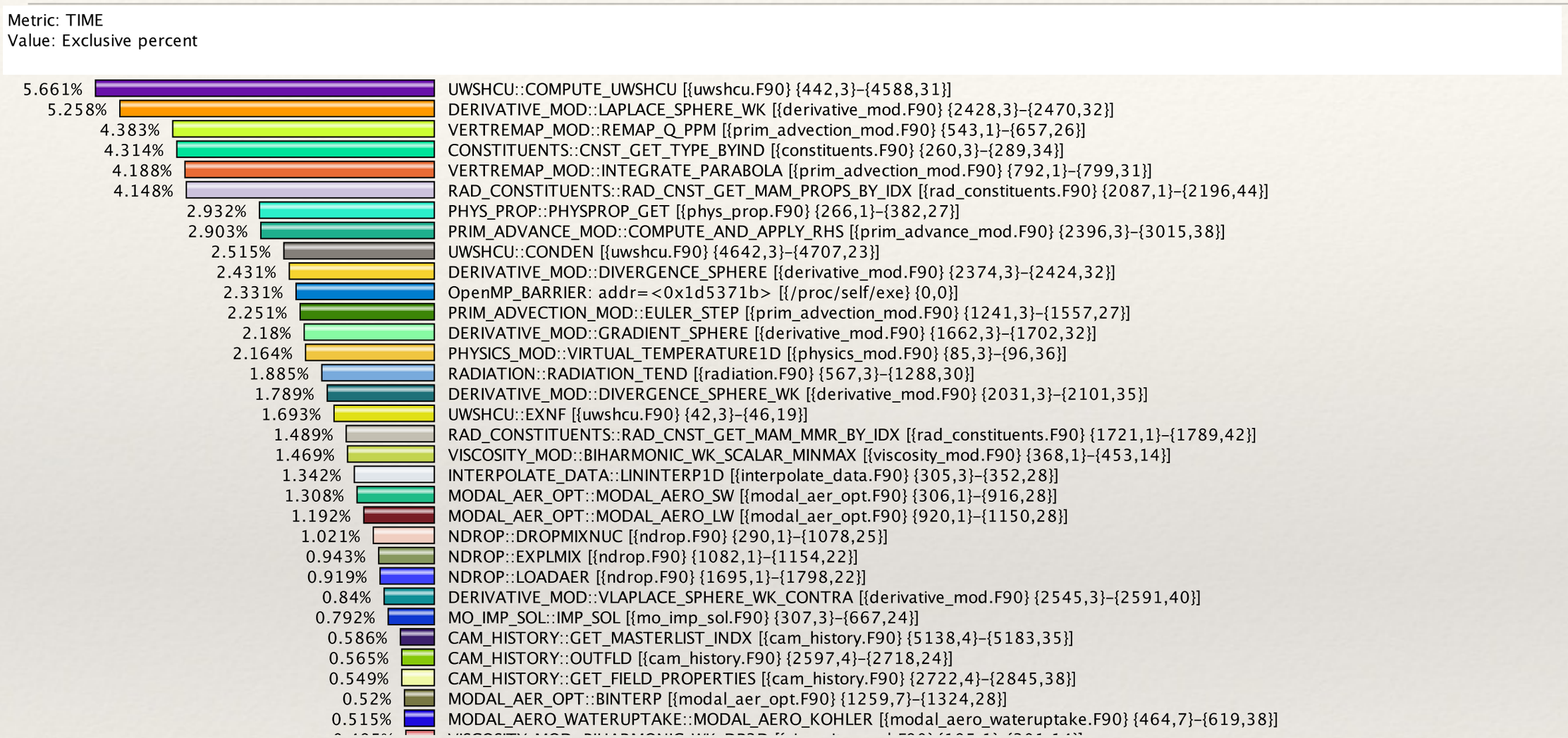
- Programs can run completely on the Xeon Phi (native mode).
  - Compiling with "-mmic" is supposed to do the trick.
  - Somebody needs to come along and build supporting libraries for the Xeon Phi (netcdf, hdf5, …, ?Trilinos?)
- We can run CESM on both the host (Xeon) and coprocessor (Xeon Phi) at same time (symmetric mode) via MPI ranks mapping to specific devices. We are focusing on *native* execution to highlight computational inefficiencies.
  - Working on a configure (processor layout) file generator for mapping ranks so PIO is on host and rest on Xeon Phis.
    - Primarily sequential code should be on the higher frequency chips but this is only a few of the components.

FIDEAL for CESM passes correctness on Stampede's Xeon Phi

- FCIDEAL motivated by efforts collaborative efforts with INTEL w.r.t HOMME
- 25% as fast as Dual Socket Xeon
- 1 yr simulation results ensemble verified… so move on to more complicated CESM system.

Device	Resolution	avg. dt [secs / simulated day]	parallel decomp. [8 nodes]
Dual socket Xeon	ne16_ne16	4.26	16mpi ranks / node, 1 omp thread
Xeon Phi SE10P	ne16_ne16	19.18	8 mpi ranks/ node, 24 omp threads per mpi rank

FC5 CESM is relatively "flat" when looking for expensive subroutines.



- Huge subroutines that are time consuming may require refactoring to discover enhancement possibilities.

We want to use vector instructions efficiently.

- Better vectorization on Xeon will only help with Xeon Phi.
- Contiguous memory access in loops will help.
  - A *$dir vector aligned* to inner loop allows for single stride access shows 1% relative VI increase (divergence_sphere_wk). Not enough to impact overall performance at this small scale.
- An inner loop of 8 iterates works well with the Xeon Phi VPU.

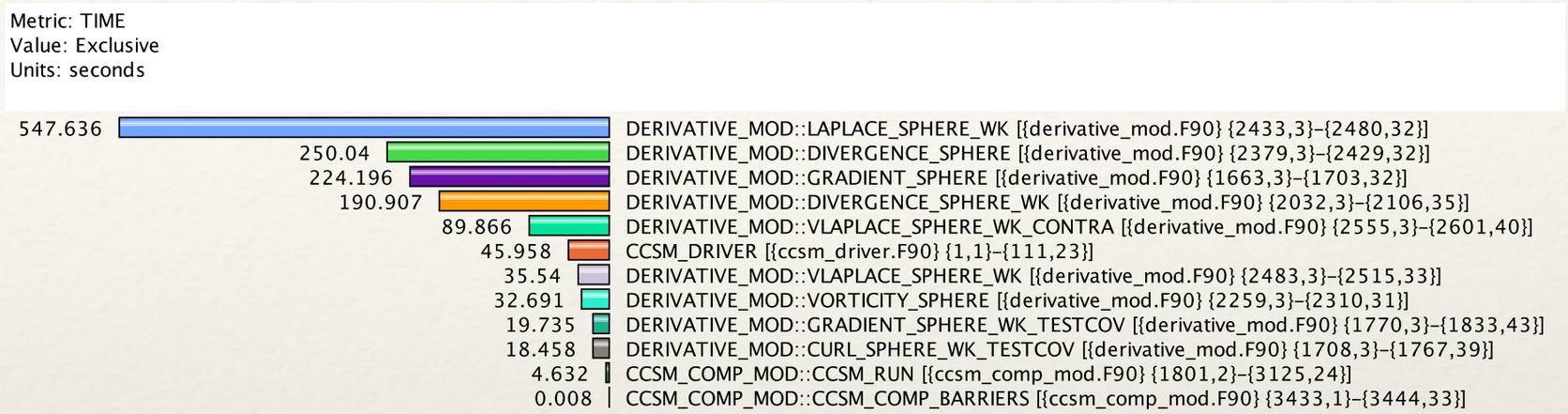


Fig.1: TAU profile, total exclusive execution time on Xeon.

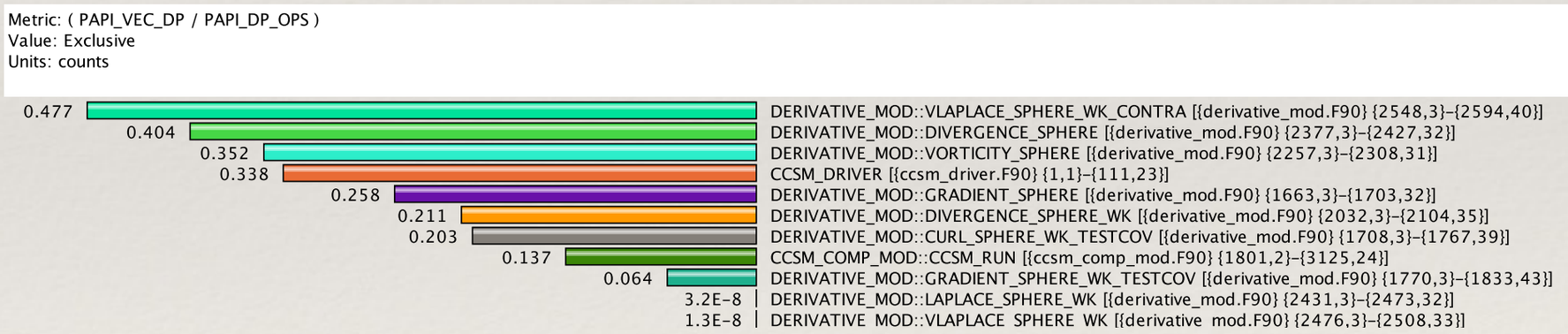
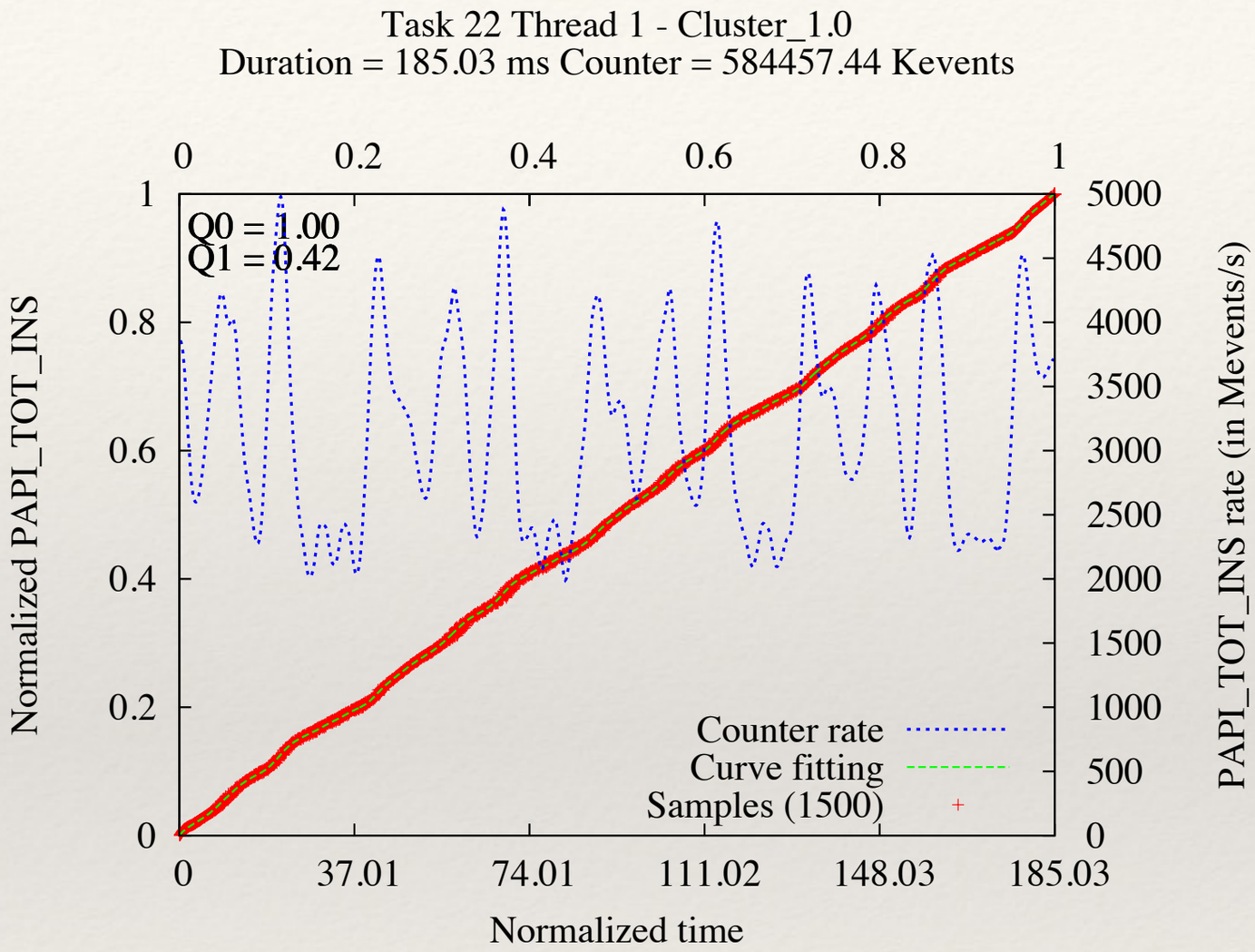


Fig 2.: Vector Intensity with max value of 1 on Xeon

- a=2*(FP_COMP_OPS_EXE:SSE_FP_PACKED_DOUBLE) + 4*(SIMD_FP_256:PACKED_DOUBLE), b=FP_COMP_OPS_EXE:SSE_SCALAR_DOUBLE
- PAPI_VEC_DP/PAPI_DP_OPS= 1/(1+(b/a))

Cycles Per Instruction can highlight issues.



- First derivative minimums correspond to subroutines which are evaluated for acceleration. [BSC: Extrae,Paraver]

Scalars as vectors and reducing loops help. Cutting down number of divides also helps.

```
real(r8) srcs2      ! work variable
real(r8) tc         ! temp in celcius
real(r8) weight     ! fraction of condensate which is ice
real(r8) cldmabs    ! maximum cloud at or above this level
real(r8) cldmabc    ! maximum cloud at or above this level
real(r8) odds       ! limit on removal rate (proportional to prec)
```

```
do k = 1,pver
  do i = 1,ncol
  tc     = t(i,k) - tmelt
  weight = max(0._r8,min(-tc*0.05_r8,1.0_r8)) ! fraction of condensate that is ice
  weight = 0._r8                              ! assume no ice

  pdog = pdel(i,k)/gravit



! ****************** Evaporation ***************************
  ! calculate the fraction of strat precip from above
  !               which evaporates within this layer
  fracev(i) = evaps(i,k)*pdel(i,k)/gravit &
          /max(1.e-12_r8,precabs(i))
```

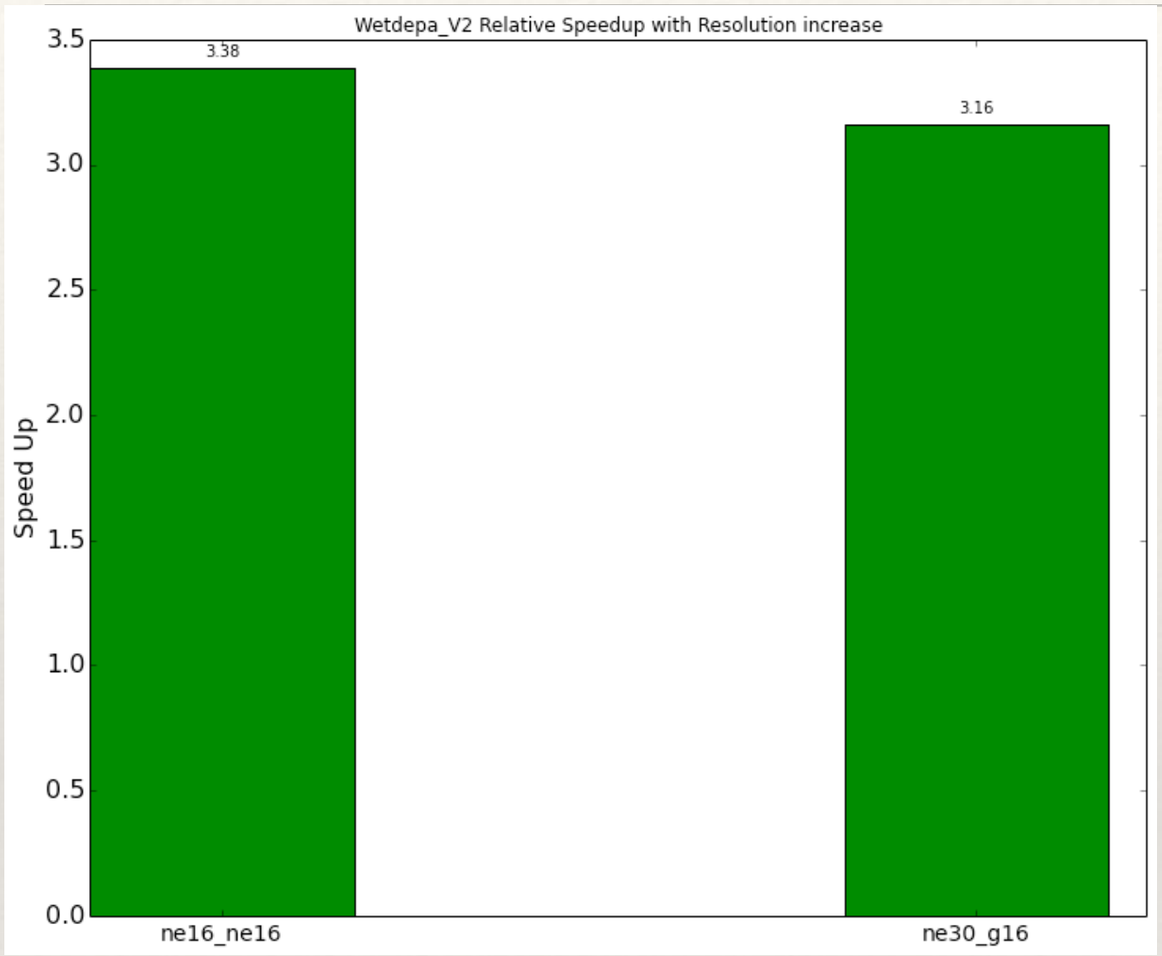
```
real(r8) srcs2(pcols)       ! work variable
real(r8) tc(pcols)          ! temp in celcius
real(r8) weight(pcols)      ! fraction of condensate which is ice
real(r8) cldmabs(pcols)     ! maximum cloud at or above this level
real(r8) cldmabc(pcols)     ! maximum cloud at or above this level
real(r8) odds(pcols)        ! limit on removal rate (proportional to prec)
```

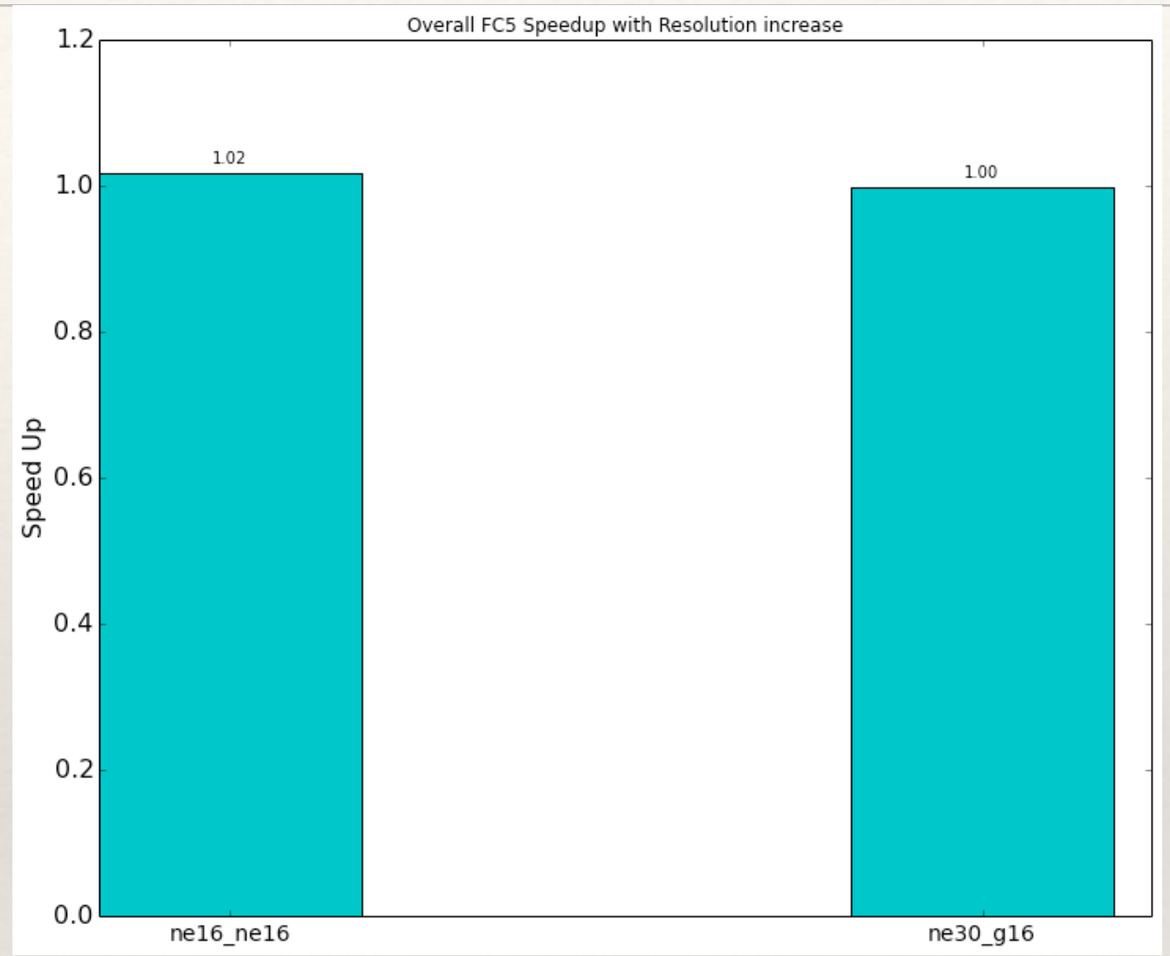
```
do k = 1,pver
    do i = 1,ncol
    tc(i)     = t(i,k) - tmelt
    weight(i) = max(0._r8,min(-tc(i)*0.05_r8,1.0_r8)) ! fraction of condensate
    weight(i) = 0._r8                                 ! assume no ice

    pdog(i)  = pdel(i,k)/gravit
    rpdog(i) = gravit/pdel(i,k)
    rdeltat  = 1.0_r8/deltat

    ! ****************** Evaporation ***************************
    ! calculate the fraction of strat precip from above
    !               which evaporates within this layer
#ifdef SVRECIP
        fracev(i) = evaps(i,k)*pdog(i) &
#else
        fracev(i) = evaps(i,k)*pdel(i,k)/gravit &
#endif
                /max(1.e-12_r8,precabs(i))
```

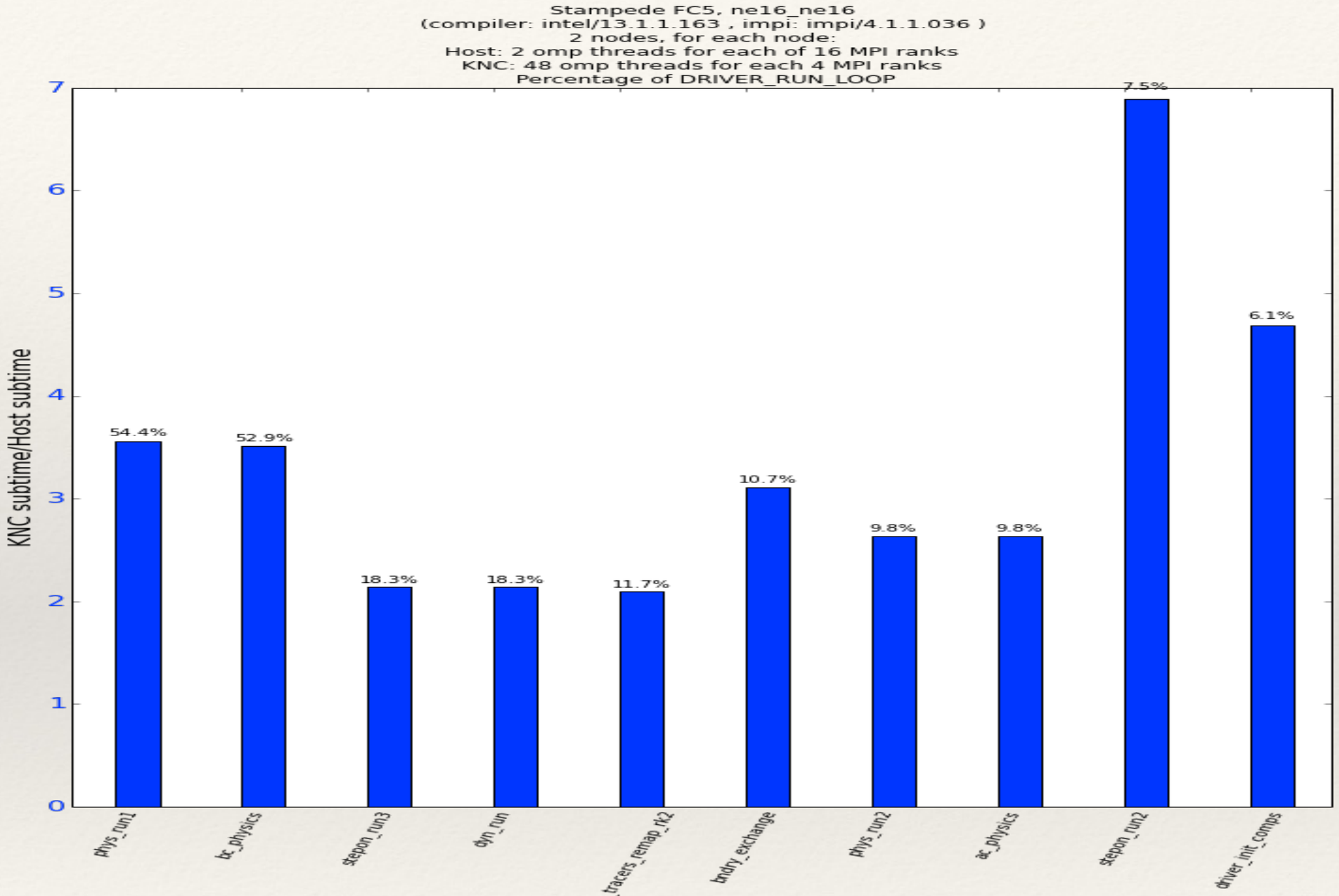
VPU vector length speedup nearly achieved.





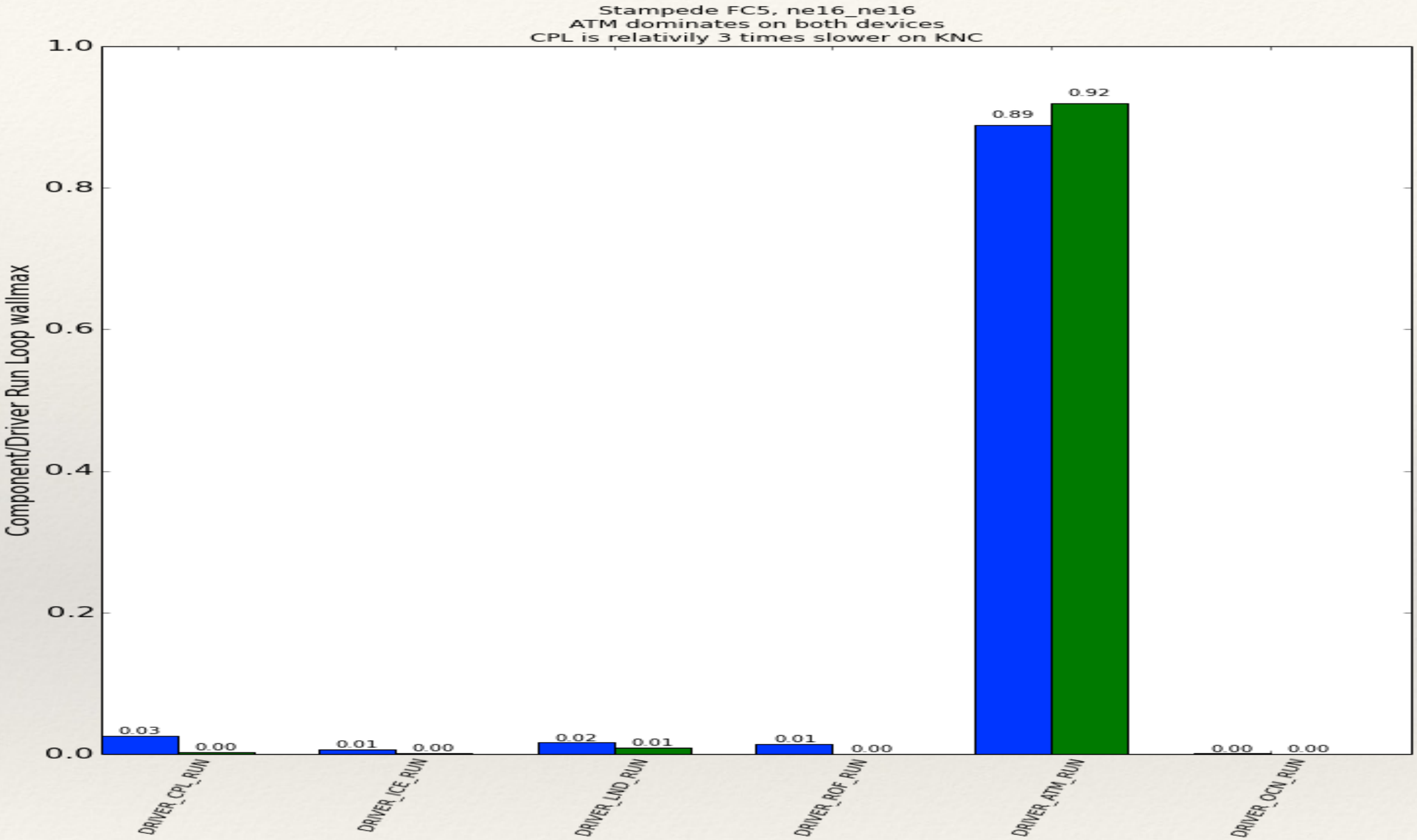
- Overall application speedup is relatively small; we need many of these micro-accelerations.
- This type of acceleration's overall impact is possibly diminished with increased resolution.
- All regression tests passed for the Xeon Phi.

Investigations with HOMME and help from TACC + Intel showed "-mP2OPT_hpo_matrix_opt_framework=0" needed for -O3 on Xeon Phi



- Comparative slowest may not be most expensive.

Components behave nearly the same on both platforms



- Still need to understand why coupler is much slower on Xeon Phi with no restart files.

FC5 (CAM-SE) is .3*Xeon [2 dual socket]= Xeon Phi [2 KNC] and is now verified using ensembles. Is the *bit-for-bit* constraint alleviated?

- Create ensemble of all variables (118) on trusted platform by 101 initial temperature perturbations.
  - 1 year simulation
  - Particular *compset* and resolution
    - 101 member ensemble establishes [min,max] RMSZ score or global mean interval per variable: *baseline intervals*
- Compare 3 similar perturbations on ported platform or new algorithm.
- Success: Less than 10% variables outside of baseline intervals with no repetition among the 3 experiments.

$$Z_{x_i}^m = \frac{x_i^m - \bar{x}_i^{E\setminus m}}{\sigma_{x_i}^{E\setminus m}}$$

- Definition for Z-score for one variable at a global location

$$\mathrm{RMSZ}_X^m = \sqrt{\frac{1}{N_X}\sum_i (Z_{x_i}^m)^2}$$

- Definition of RMSZ
  - X: particular variable,

We are continually working toward Xeon Phi CESM production runs.

- DDT use on Xeon Phi helped expose Intel's OpenMP issue of privatized variables that have allocatable members.
- We want to use KGEN (Y. Kim) to extract more kernels for acceleration on both Xeon and Xeon Phi
- We are working on the fully coupled BC5 Xeon Phi verification.
  - Newest Intel 15 compiler that has fixes applied (compiler flag that optimized FC5) does not run on the Xeon Phi as of yet (but it does for Xeon).
- Ensemble verification process allows for comparisons without bit-for-bit constraint.
- We now have a branch of CESM dedicated to Xeon Phi development.
  - Symmetric run strategies will be explored in this branch. This will be necessary production run status.
- We must be careful that code changes does not hurt performance on other architectures.
  - Regression tests have minimal performance checks (high-level subroutine max times). We would like to integrate more detailed (and still light weight) performance checks in the regression testing.

References

- Computational performance of ultra-high-resolution capability in the Community Earth System Model, Dennis, John M., Vertenstein, Mariana, Worley, Patrick H., Mirin, Arthur A., Craig, Anthony P., Jacob, Robert, Mickelson, Sheri, International Journal of High Performance Computing Applications, Feb 2012; vol. 26: pp. 5-16
- CAM-SE: A scalable spectral element dynamical core for the Community Atmosphere Model, Dennis, John M., Edwards, Jim, Evans, Katherine J., Guba, Oksana, Lauritzen, Peter H., Mirin, Arthur A., St-Cyr, Amik, Taylor, Mark A., Worley, Patrick H. International Journal of High Performance Computing Applications, Feb 2012; vol. 26: pp. 74-89
- TAU: The TAU Parallel Performance System, S. Shende and A. D. Malony. International Journal of High Performance Computing Applications, Volume 20 Number 2 Summer 2006. Pages 287-311.
- Trace Spectral Analysis toward Dynamic Levels of Detail., Llort G, Casas M, Servat H, Huck K, Giménez J, Labarta J17th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2011, Tainan, Taiwan. 2011 :332 - 339.
- A new verification tool for the {Community Earth System Mode, M. Levy and A. H. Baker and J. Anderson and J. M. Dennis and J. Edwards and A. Mai and D. Nychka and J. Tribbia and S. Vadlamani and M. Vertenstein and D. Williamson and H. Xu, in preparation, 2014.