Experience with mixed-precision within physics parameterizations

John Dennis, Dan Milroy, Allison Baker,
Andrew Gettelman, Dorit Hammerling

September 26, 2019

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

History/Motivation

- Historically:
  - Climate models: 64-bit floating-point
    - On Cray vector systems -> single precision is 64-bit floating-point
    - Small per MPI rank problem size means less cache pressure
    - 4-byte calculations cost same as 8-bytes
    - Needed for conservation
    - Simpler
  - Weather models: 32-bit floating-point
    - Large per MPI rank problem size
    - Lower precision floating reduces cache pressure
- Now:
  - Vector instruction sets means that 32-bit floating-point is potentially 2x that of 64-bit floating-point

Impact of 32-bit floating-point

- Is correctness maintained?
- Does it reduce code execution time?
- Does it negatively impact maintainability?

Approach

- Previous results (WACCM implicit solver)
  - Thrashes L2 cache
  - Code is highly vectorize
  - Virtually no ‘if’ tests in computational kernel
  - 1.97x speedup
- Want something more challenging!
  - Morrison Gettelman Microphysics version 2
  - Relatively expensive: ~5% of total CAM cost
  - Complex code with lots of ‘if’ tests
  - Extensive experience optimizing code base
  - Willing collaborator (KEY)

Optimization approach: vectorize everything

```
real, intent(in)  :: t   ! Temperature in Kelvin
real, intent(out) :: es  ! SVP in Pa

  ! uncertain below -70 C
  es = 10.**(-7.90298*(tboil/t-1.)+ &
       5.02808*log10(tboil/t)- &
       1.3816e-7*(10.**(11.344*(1.-t/tboil))-1.)+ &
       8.1328e-3*(10.**(-3.49149*(tboil/t-1.))-1.)+ &
       log10(1013.246))*100.
```

1 single-precision result
1 double-precision result

```
integer, intent(in) :: vlen
real, intent(in)  :: t(vlen)   ! Temperature in Kelvin
real, intent(out) :: es(vlen)  ! SVP in Pa
integer :: i
  ! uncertain below -70 C
  do i=1,vlen
       es(i) = 10.**(-7.90298*(tboil/t(i)-1.)+ &
          5.02808*log10(tboil/t(i))- &
          1.3816e-7*(10.**(11.344*(1.-t(i)/tboil))-1.)+ &
          8.1328e-3*(10.**(-3.49149*(tboil/t(i)-1.))-1.)+ &
           log10(1013.246))*100.
  enddo
```

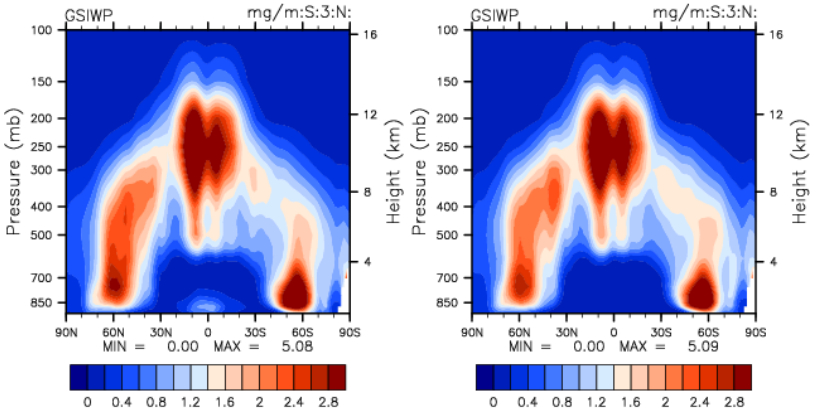
8 single-precision results
4 double-precision results

Is correctness maintained?

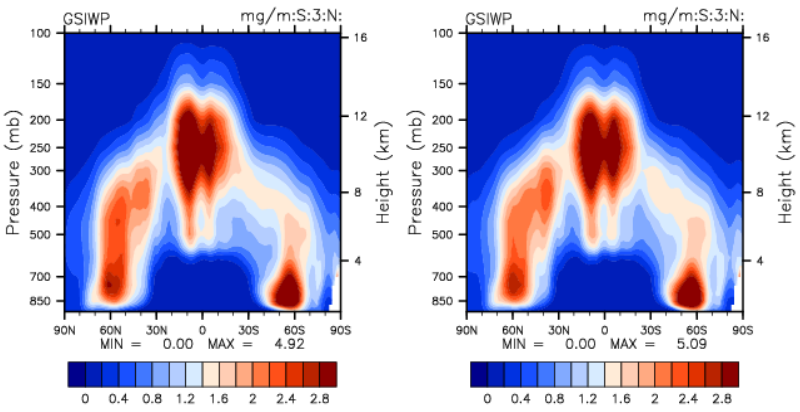
- Did not pass CESM verification test
  - The changes are statistically distinguishable from natural variability
- Systematic differences are apparent in climatological averages (AMWG diagnostic package)
- Look at three different configurations of CAM
  - MG2-CAM-default:
    [64-bit everywhere]
  - MG2-CAM-mixed:
    [32-bit in MG2]
  - MG2-GammaWV-SP:
    [32-bit gamma functions in saturation vapor pressure calculations]

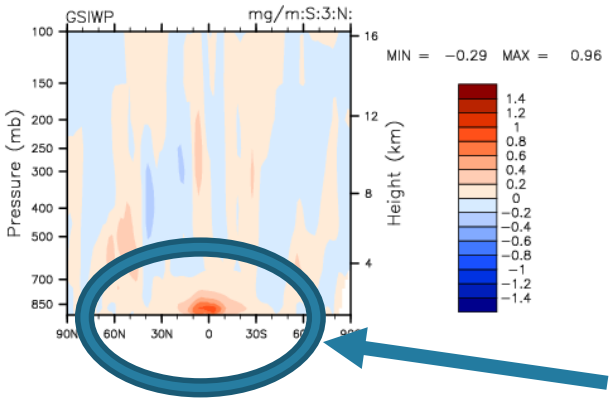
Global annual mean for SIWC (snow plus ice water content)

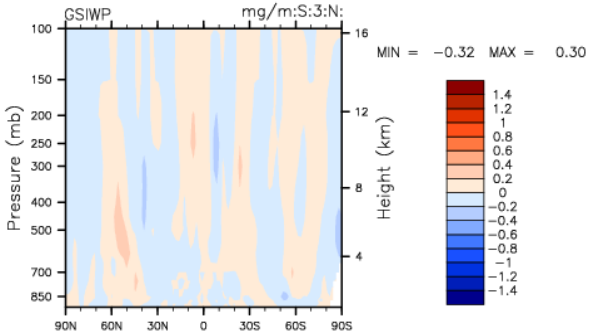
MG2-CAM-mixed



MG-gamaWV-SP







Systematic bias at equator

D. Milroy, A. Baker, J. Dennis, A. Gettelman, D. Hammerling, “Investigating the Impact of Mixed Precision on Correctness for a Large Climate Code”, Correctness 2019 workshop to appear

Impact of 32-bit floating-point

- Is correctness maintained?

  Not really
- Does it reduce code execution time?

Does it reduce execution time?

- MG2 calculation only
  - Cheyenne
    - Kernel: R4 → 1.35x speedup versus R8
    - In CAM: R4 → 1.22x speedup versus R8
- Current R8→R4 speedup is equivalent of Broadwell to Skylake speedup.
- Variation across different MPI ranks:
  - 2x speedup on a few execution paths
  - Could additional execution paths could be optimized?
- Overall impact on CAM: ~ 0.5%
  - Very large overhead in actually calling MG2 from CAM
  - Other parameterizations in CAM are significantly more expensive (CLUBB)

Potential implications of 32-bit floating-point

- Is correctness maintained?

  Not really
- Does it reduce code execution time?

  Somewhat
- Does it negatively impact maintainability?

Negatively impact maintainability?

- Single point to switch from 8-byte to 4-byte calculations ☺
- Multiple entry points into modified code
  - Certain MG2 utility routines are called outside main subroutine
  - Saturation vapor pressure calculations called from multiple locations in CAM
  - Need to include both vector and scalar versions of numerous subroutines ☹
  - 4-byte and 8-byte versions generated by templating capability with CAM
- Constants: Maintain separate 8-byte and 4-byte versions or type conversion of 8-byte constant?

Potential implications of 32-byte floating-point

- Is correctness maintained?

  Not really
- Does it reduce code execution time?

  Somewhat
- Does it negatively impact maintainability?

  Yes

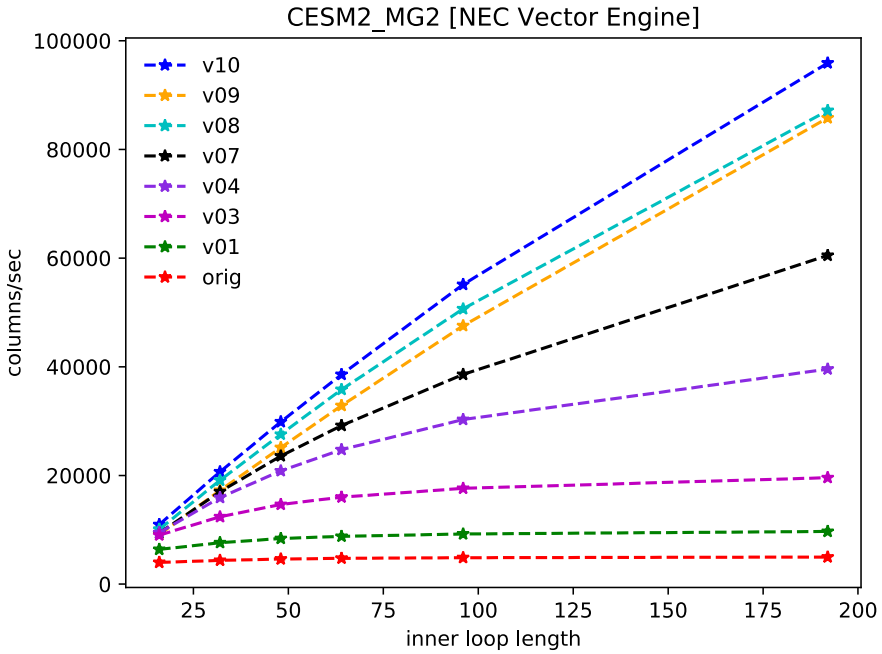
Some reflections

- Parameterization tuned using 8-byte floating-point
  - Can correctness issues in 4-byte version be eliminated by tuning?
- Simplified support for reduced precision will likely be in next version of CESM
- Develop new parameterization that can be switch between single and double precision
  - Focus on 4-byte version
  - Scientific justification for 8-byte

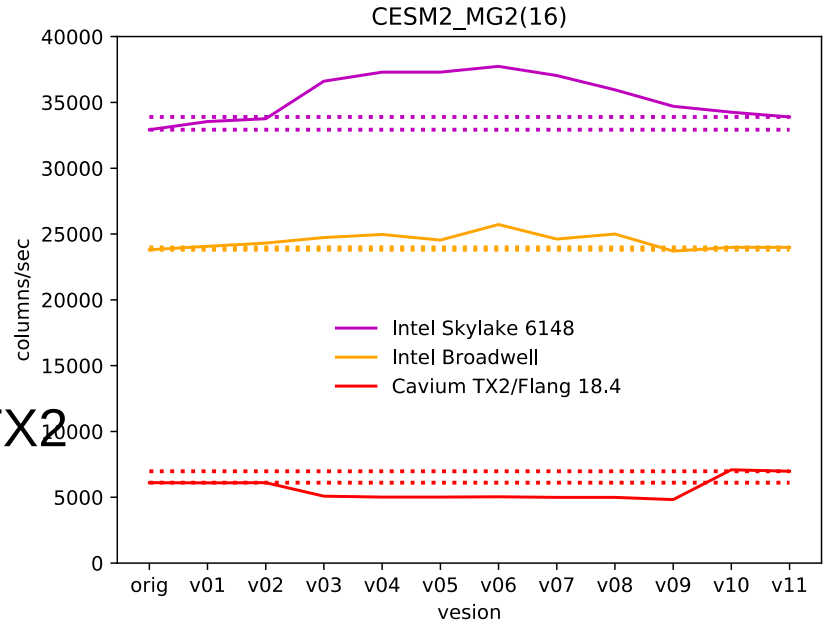
Conclusions/Future work

- Achieves speedup comparable to next generation of processor
- Use of 32-bit floating-point currently breaks correctness
- Does currently impact code maintainability due to call structure in CAM
- Should future parameterizations be 32-bit floating-point?

Extreme vectorization of the CESM2_MG2 kernel



~20x speedup on NEC VE



Performance neutral for Xeon and TX2