

# Porting the RTE+RRTMGP radiative transfer package for next-generation supercomputers

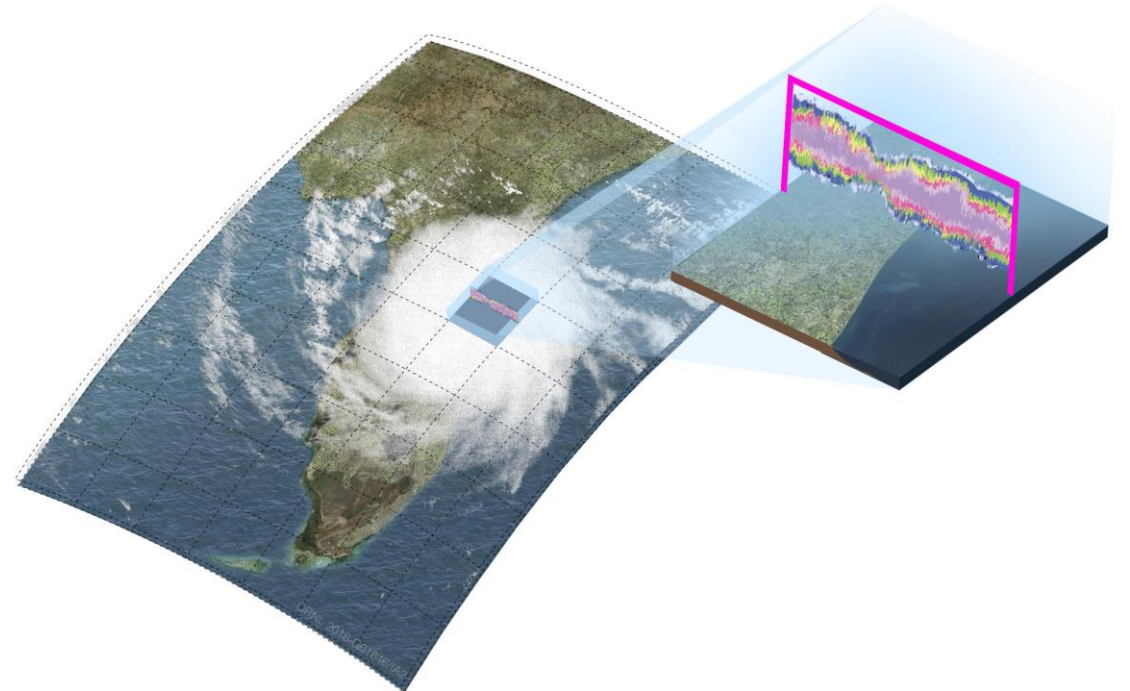
Approved for public release



Benjamin R. Hillman (SNL), Matthew Norman (ORNL), Robert Pincus (CU)

# Two paths toward a DOE global cloud-permitting model

- Simple Cloud-Resolving E3SM Atmosphere Model (SCREAM)
  - Rewrite our existing atmosphere in C++/kokkos for *performance portable* GPU support with simplified physics
  - Scale up to 3km resolution
  - Target simulations in 2021
- E3SM using the Multi-scale Modeling Framework (E3SM-MMF)
  - Multiscale modeling approach, “superparameterization”
  - Cloud resolving convection
  - Very high computational intensity – ideal for GPUs
  - Fortran with OpenACC for GPU support

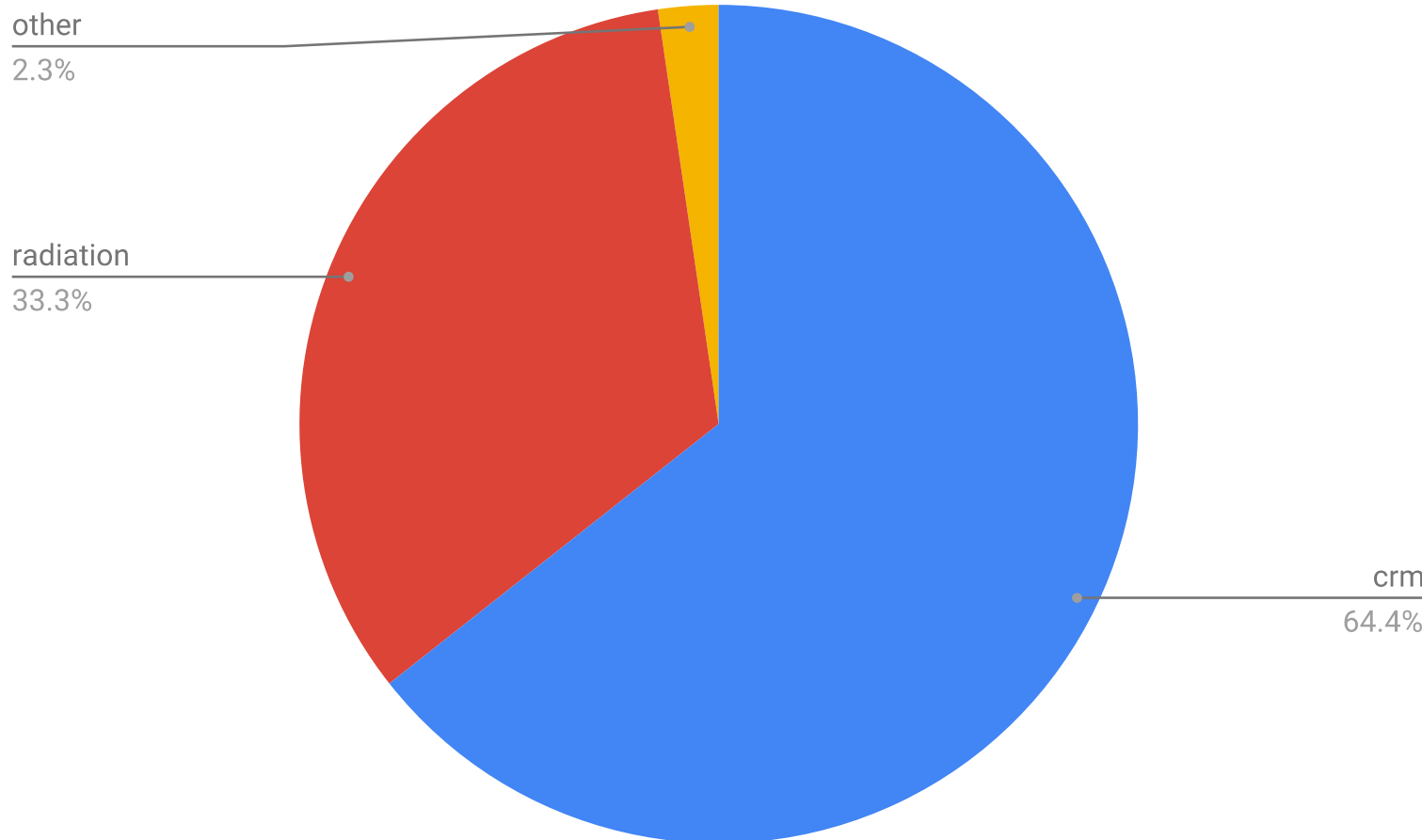


# E3SM-MMF Highlights

- Complete port of the CRM superparameterization to GPUs
  - refactored 30K lines of code to enable openACC acceleration
  - represents about 50% of the cost of the model
  - Port of remaining 40% (RRTMGP package) recently completed
- Summit Early Science Simulation
  - 1024 Summit nodes, running at 0.62 SYPD
  - 6 year simulation, 300K node-hours
  - Running a weather resolving global model (25km) with a cloud resolving 2D CRM (1km superparameterization)
- Gordon Bell Submission SC2019
  - 4600 Summit nodes, ~5.4PF
  - 1.8 SYPD with 2km resolution
  - 0.22 SYPD at 500m resolution

# Radiative transfer cost

- Radiative transfer is expensive: ~1/3 the cost of the atmospheric physics
- CRM has already been ported to GPU on Summit: ~15x speed-up
- This talk: efforts to port the radiative transfer package to GPU



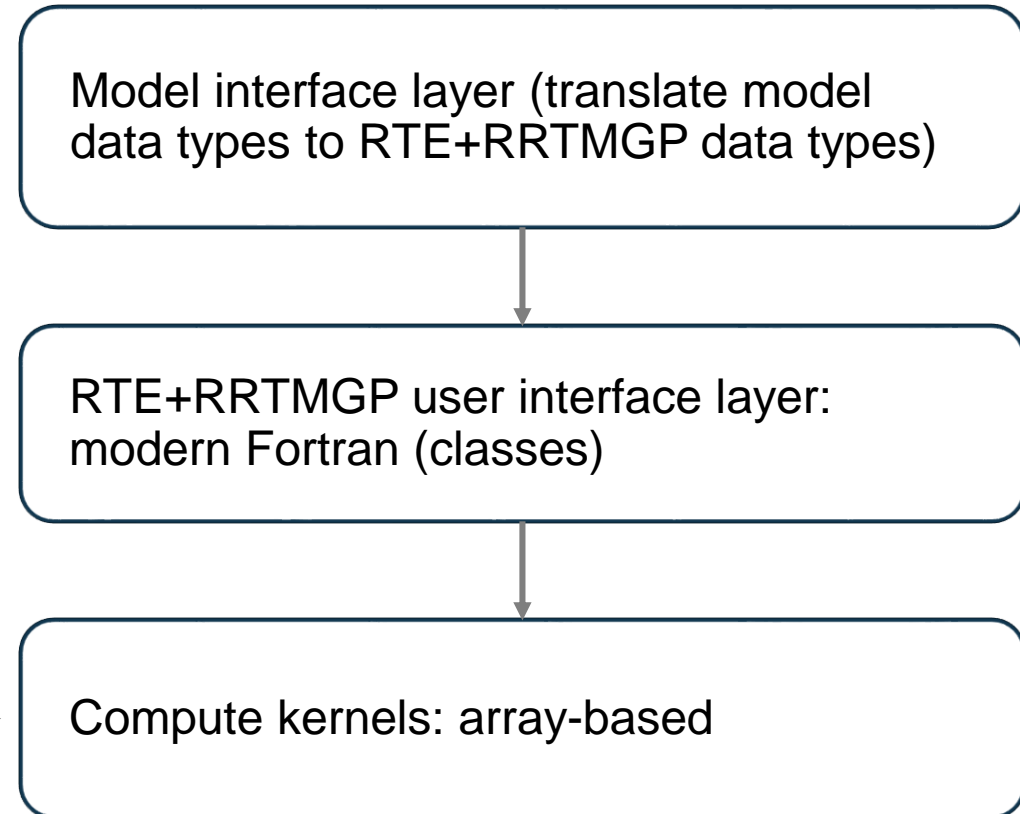
Relative cost of physics packages on Intel Sandy Bridge

# Radiative transfer package: RTE+RRTMGP

- Rewrite of popular RRTMG radiation package
- Expose parallelism
- Modern software practices

Goal: port *kernels* for performance portability, leaving *interface* largely untouched

Implementation: levels of abstraction



# Porting RTE+RRTMGP using OpenACC

- Goal: RTE+RRTMGP fully running on Summit GPU
- Steps:
  - Expose parallelism
  - Wrap with OpenACC directives *without* explicit data management
  - Compile with ptxinfo flag to highlight generation of implicit data copying code
  - Add explicit data management to directives



# Testing

- How do we know we have the right answer (and didn't screw anything up)?
- Need to test after each code addition!
  - Rapid, easy to launch regression tests
- Testing framework based on RTE+RRTMGP RFMIP example code (provided in RTE+RRTMGP Git repo)
  - End-to-end, stand-alone test
  - Code: reads in example atmosphere data, computes radiative fluxes due to gaseous absorption
  - Test: compare outputs from a test run with outputs from a baseline (before the code modification)
  - Challenge: answers are *not* bit-for-bit due to floating point differences arising from atomic updates on the GPU (cannot guarantee order of updates)



# Testing: example

## Diffs between CPU and reference:

Variable rlu: No diffs

Variable rld differs (max abs difference: 3.814697e-06; max frac. difference: 1.178709e-05%)

Variable rsu differs (max abs difference: 3.051758e-05; max frac. difference: 1.185221e-05%)

Variable rsd differs (max abs difference: 6.103516e-05; max frac. difference: 1.087066e-05%)

## Diffs between GPU and reference:

Variable rlu: No diffs

Variable rld differs (max abs difference: 1.490116e-08; max frac. difference: 1.173428e-05%)

Variable rsu differs (max abs difference: 3.051758e-05; max frac. difference: 1.184619e-05%)

Variable rsd differs (max abs difference: 6.103516e-05; max frac. difference: 1.087066e-05%)

## Diffs between CPU and GPU:

Variable rlu: No diffs

Variable rld differs (max abs difference: 3.814697e-06; max frac. difference: 1.178709e-05%)

Variable rsu differs (max abs difference: 3.051758e-05; max frac. difference: 1.185221e-05%)

Variable rsd differs (max abs difference: 3.051758e-05; max frac. difference: 9.782132e-06%)

Subjectively, differences order 1e-5 are “tolerable”

# When things go bad...

```
call zero_array(block_size, def_tsi)
!$acc parallel loop collapse(2) copy(def_tsi) copyin(toa_flux)
do igpt = 1, ngpt
  do icol = 1, block_size
    !$acc atomic update
    def_tsi(icol) = def_tsi(icol) + toa_flux(icol, igpt)
  end do
end do
```

```
call zero_array(block_size, def_tsi)
!$acc parallel loop collapse(2) copy(def_tsi) copyin(toa_flux)
do igpt = 1, ngpt
  do icol = 1, block_size
    !$acc atomic update
    def_tsi(icol) = def_tsi(icol) + toa_flux(icol, igpt)
  end do
end do
```

Missing atomic update in reduction operation leads to wrong answers!

```
Variable rlu: No diffs
Variable rld differs (max abs difference: 1.490116e-08; max frac. difference: 1.173428e-05%)
Variable rsu differs (max abs difference: 4.540662e+06; max frac. difference: 1.999758e+02%)
Variable rsd differs (max abs difference: 2.117698e+07; max frac. difference: 1.999758e+02%)
```

# Debugging tools

- Cuda-memcheck
- Valgrind (on CPU)
- Bounds checking (on CPU)
- Simplifying data movement

# Profiling tools

- PGI\_ACC\_TIME=1: quick timing info for compute vs data movement

```
/autofs/nccs-svm1_home1/brhillman/codes/rte-rrtmgp/branches/master/build/./rte/kernels-openacc/mo_rte_solver_kernels.F90
lw_source_noscat NVIDIA devicenum=0
time(us): 10,078
495: compute region reached 1 time
    495: kernel launched 1 time
        grid: [65535] block: [128]
            device time(us): total=10,078 max=10,078 min=10,078 avg=10,078
            elapsed time(us): total=10,113 max=10,113 min=10,113 avg=10,113
495: data region reached 2 times
```

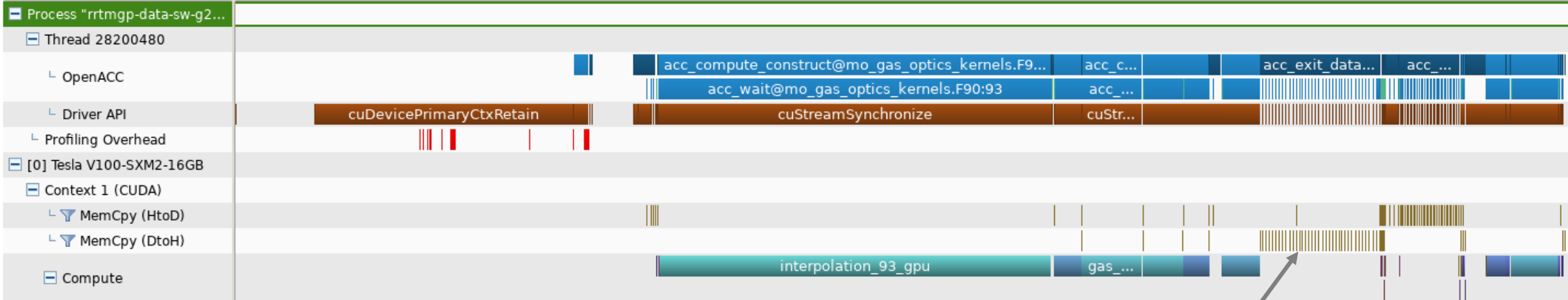
- NVPROF: visual representation of profiling data
  - Run code on compute node, save nvprof output
  - View using nvvp
  - Useful for identifying bottlenecks and excessive data movement

# PGI\_ACC\_TIME=1 example

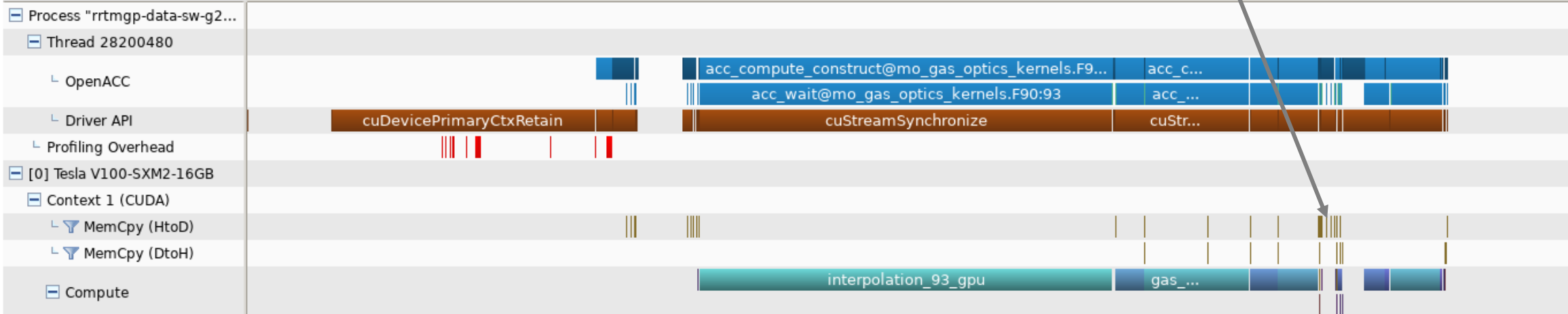
```
/autofs/nccs-svm1_home1/brhillman/codes/rte-rrtmgp/branches/master/examples/rfmip-clear-sky/rrtmgp_rfmip_lw.F90
rrtmgp_rfmip_lw NVIDIA devicenum=0
time(us): 131
228: data region reached 1 time
    228: data copyin transfers: 1
        device time(us): total=20 max=20 min=20 avg=20
229: data region reached 1 time
    229: data copyin transfers: 1
        device time(us): total=13 max=13 min=13 avg=13
230: data region reached 1 time
    230: data copyin transfers: 4
        device time(us): total=32 max=8 min=8 avg=8
253: compute region reached 1 time
    253: kernel launched 1 time
        grid: [225] block: [128]
        device time(us): total=14 max=14 min=14 avg=14
        elapsed time(us): total=159 max=159 min=159 avg=159
253: data region reached 4 times
    253: data copyin transfers: 1
        device time(us): total=9 max=9 min=9 avg=9
301: data region reached 1 time
302: data region reached 1 time
    302: data copyin transfers: 1
        device time(us): total=11 max=11 min=11 avg=11
303: data region reached 1 time
    303: data copyin transfers: 4
        device time(us): total=32 max=8 min=8 avg=8
304: data region reached 1 time
```

← This is a high-level routine doing a lot of data movement

# NVPROF example



After explicit data movement: much less device to host transfers



# Future directions: transition to OpenMP Offload, and managed memory

- For enhanced portability, we are creating an OpenMP 4.5+ version of the code
  - OpenMP 4.5+ includes a kernel offload for accelerators
  - OpenMP4.5 and OpenACC have a nearly 1:1 correspondence
    - !\$acc copyin() --> !\$omp map(to:)
    - !\$acc update host() --> !\$omp target update(from:)
    - !\$acc parallel loop --> !\$omp target teams distribute parallel for
  - Deep copy issues get a little more hairy, but we plan to sidestep that
- We plan to use managed memory
  - Automatically pages data to/from GPU (no more data statements!)
  - -ta=nvidia,managed for PGI for now (currently there are bugs, though)
  - We will replace “allocate()” with custom cudaMallocManaged() routine using the LLNL [Umpire pool allocator](#)

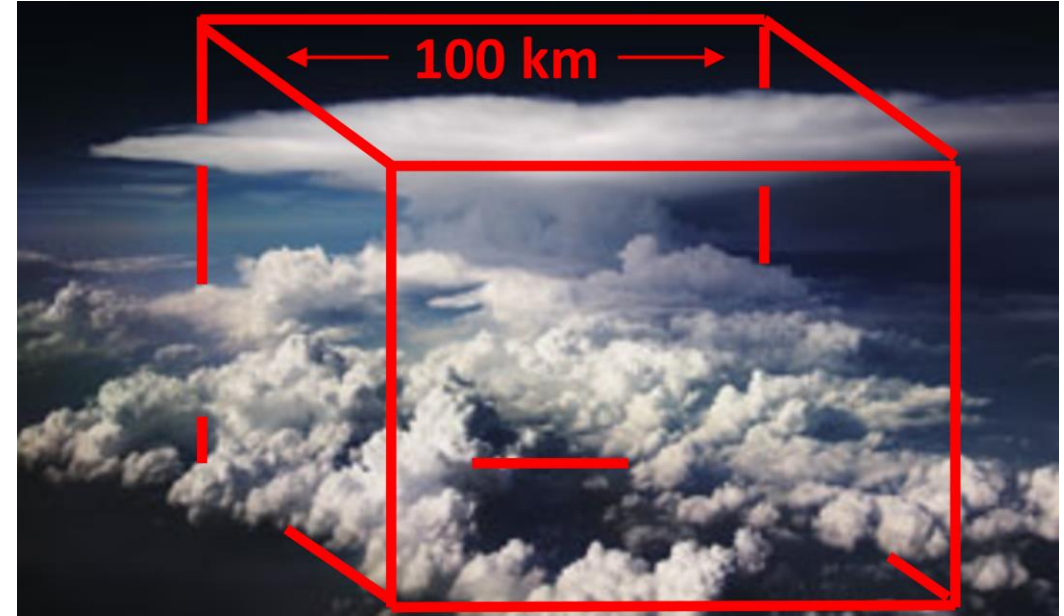
# Summary and challenges

- RTE+RRTMGP radiative transfer code ported to GPU using OpenACC directives
- The need to minimize data movement between device and host requires adding directives pretty high up in the code – impossible to confine to kernels
- A number of compiler bug work-arounds needed
- Next step: evaluating performance in the full model



# Extra slides

# Context: Developing a cloud-permitting climate model for DOE exascale architectures



*How do we parameterize this sub-grid variability?*

# Radiative transfer package: RTE+RRTMGP

- Separation of concerns

## RRTMGP

- Optical properties
- Source functions
- Spectral discretization:  
correlated k-distribution

## RTE: solvers

- One-dimensional plane-parallel RT equations
- Absorption/emission or two-stream
- Adding for transport
- Extensible to multi-stream methods