Atmospheric Radiation GO GO

Robert Pincus
Walter Hannah
Valentin Clement
Matt Norman (mentor)

We’re working on RRTMGP

- This code computes fluxes of electromagnetic radiation through the atmosphere
- The application is any weather or climate model

Initial Profile

- Experience with similar codes told us which parts of the small problem were expensive
- Algorithms: algebra, transcendentals, integration (loop caries), linear interpolation with lots of indirection
- The real win will come when we can ship and retrieve small amounts to/from GPU. We’re almost there…

Evolution and Strategy

- Our initial hope was to port most of the code to the GPU using OpenACC
- Strategy: kernel by kernel, first loop directives and then data

Results and Final Profile

- What were you able to accomplish
  - Did you achieve speed up? I don’t know*, sorry.
  - Achieved new scientific goals: almost…

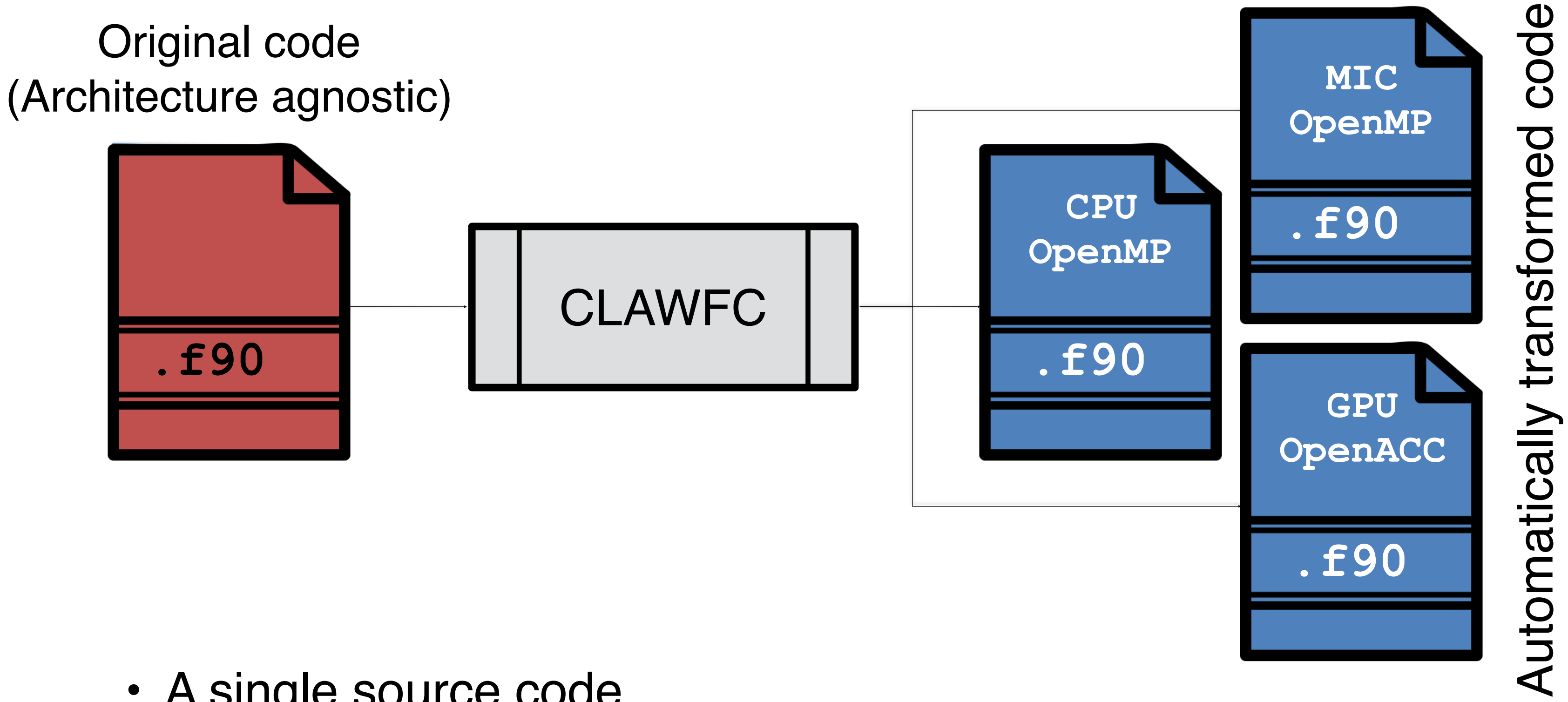
What problems you encountered

- New, clean code make this (relatively) easy
- Careful coding for CPU had to be refactored
- Cray compiler doesn’t work
- Nvprof “gave funky output”

Wishlist

- What do you wish existed to make your life easier?
  - Tools: Less heavy GPU compilers, better debugging tools on GPU
  - Even better: CLAW or similar
  - Language standards: no support for Fortran (:)?
  - Event: no complaint
  - Systems: no complaints

CLAW approach: applying transformations



- A single source code
- Specify a target architecture for the transformation
- Specify a compiler directives language for parallelization

```
clawfc --directive=openacc --target=gpu -o mo_lw_solver.acc.f90 mo_lw_solver.f90
```

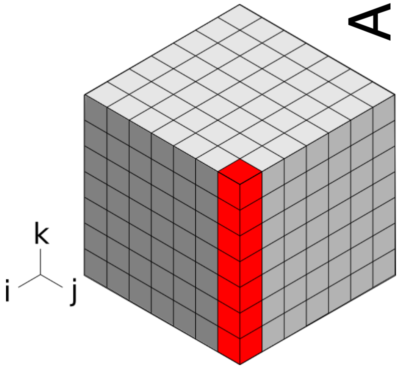
```
clawfc --directive=openmp --target=cpu -o mo_lw_solver.omp.f90 mo_lw_solver.f90
```

```
clawfc --directive=openmp --target=mic -o mo_lw_solver.mic.f90 mo_lw_solver.f90
```

RRTMGP lw_solver - claw code 1 column

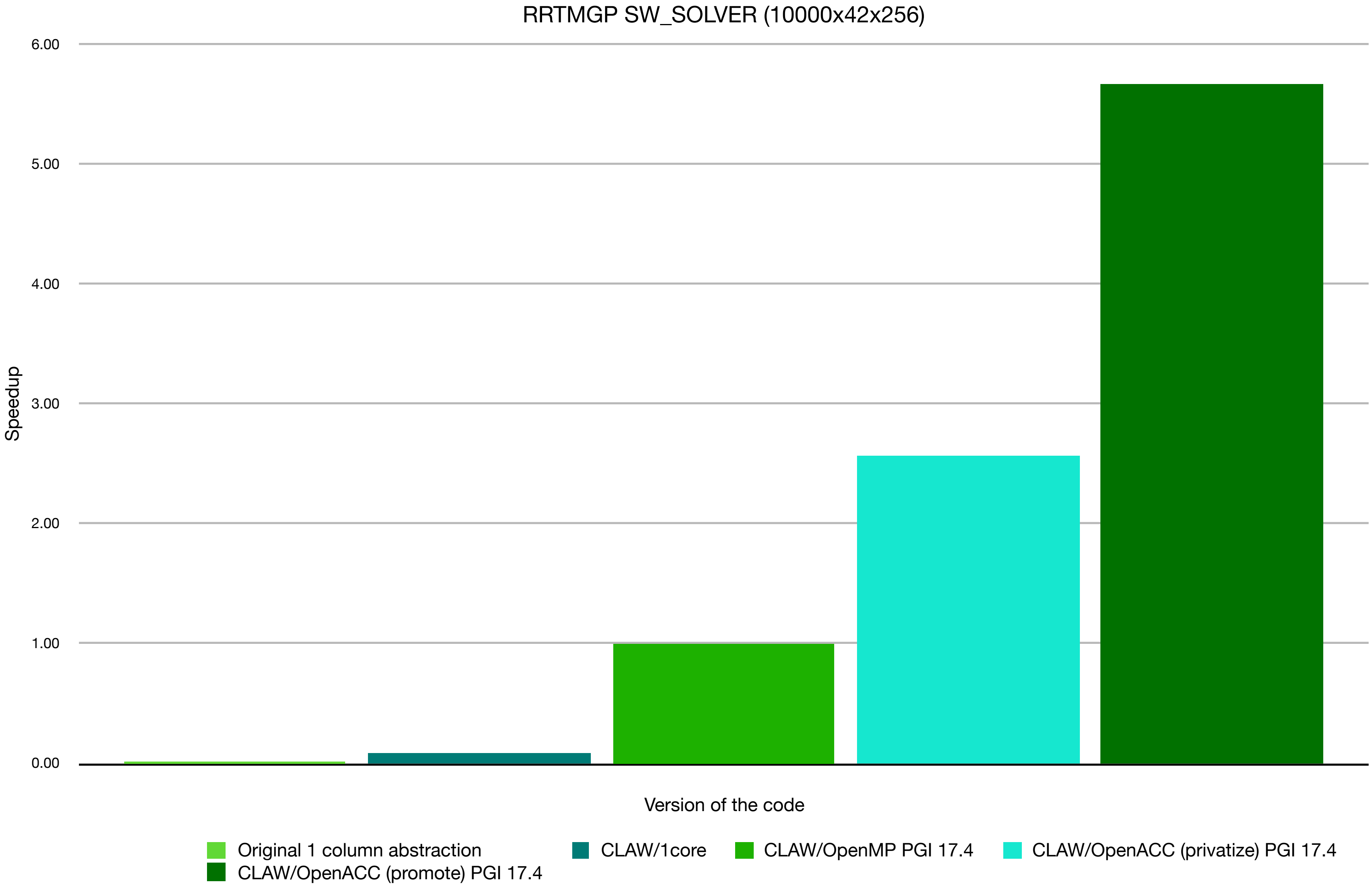
Algorithm for one column only

```
SUBROUTINE lw_solver(ngpt, nlay, tau, …)
  !$claw define dimension icol(1:ncol) &
  !$claw parallelize
  DO igpt = 1, ngpt
    DO ilev = 1, nlay
      tau_loc(ilev) = max(tau(ilev,igpt) …
      trans(ilev) = exp(-tau_loc(ilev))
    END DO
    DO ilev = nlay, 1, -1
      radn_dn(ilev,igpt) = trans(ilev) *
        radn_dn(ilev+1,igpt) …
    END DO
    DO ilev = 2, nlay + 1
      radn_up(ilev,igpt) = trans(ilev-1) *
        radn_up(ilev-1,igpt)
    END DO
  END DO
  radn_up(:,:) = 2._wp * pi * quad_wt * radn_up(:,:)
  radn_dn(:,:) = 2._wp * pi * quad_wt * radn_dn(:,:)
END SUBROUTINE lw_solver
```



Dependency on the vertical dimension only

RRTMGP sw_solver - results



Was it worth it?

- Was this worth it?
  - You - yes
  - Your team – “I learned a lot”
  - Your app – sure
  - Your domain – when this works, very much
  - Your mentors – “This was great for me”
- Will you continue development? Yes!