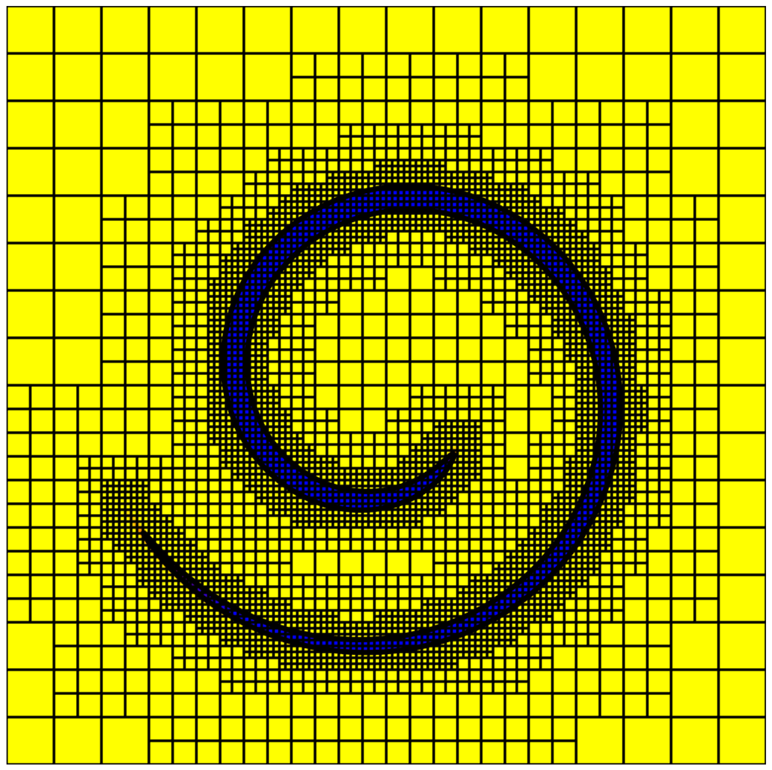
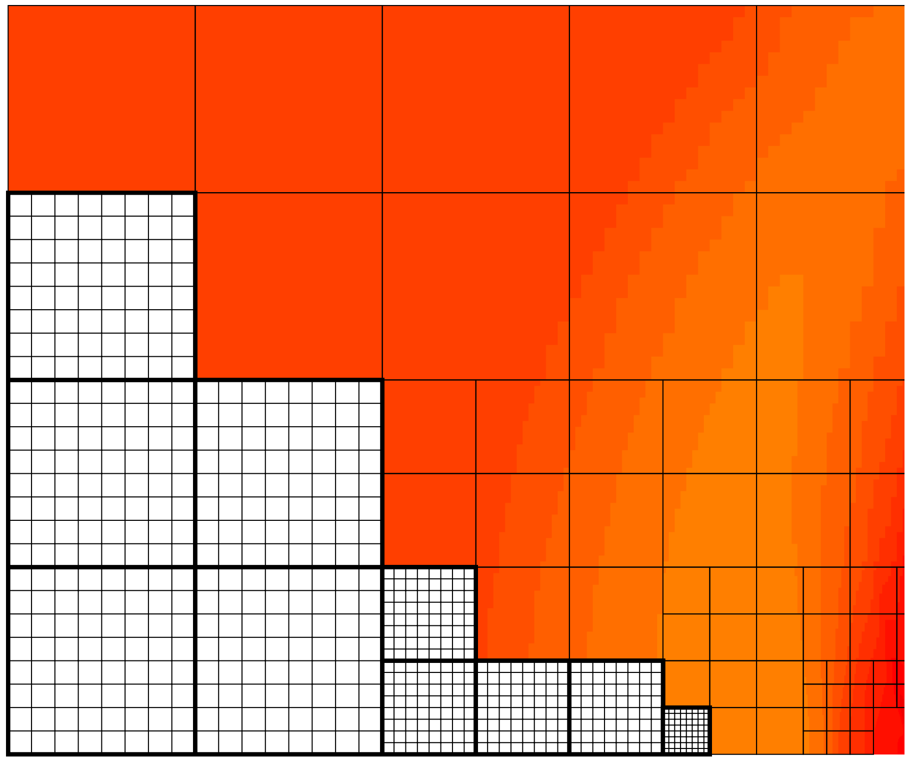
ForestClaw : Parallel, adaptive mesh refinement for Cartesian block structured meshes

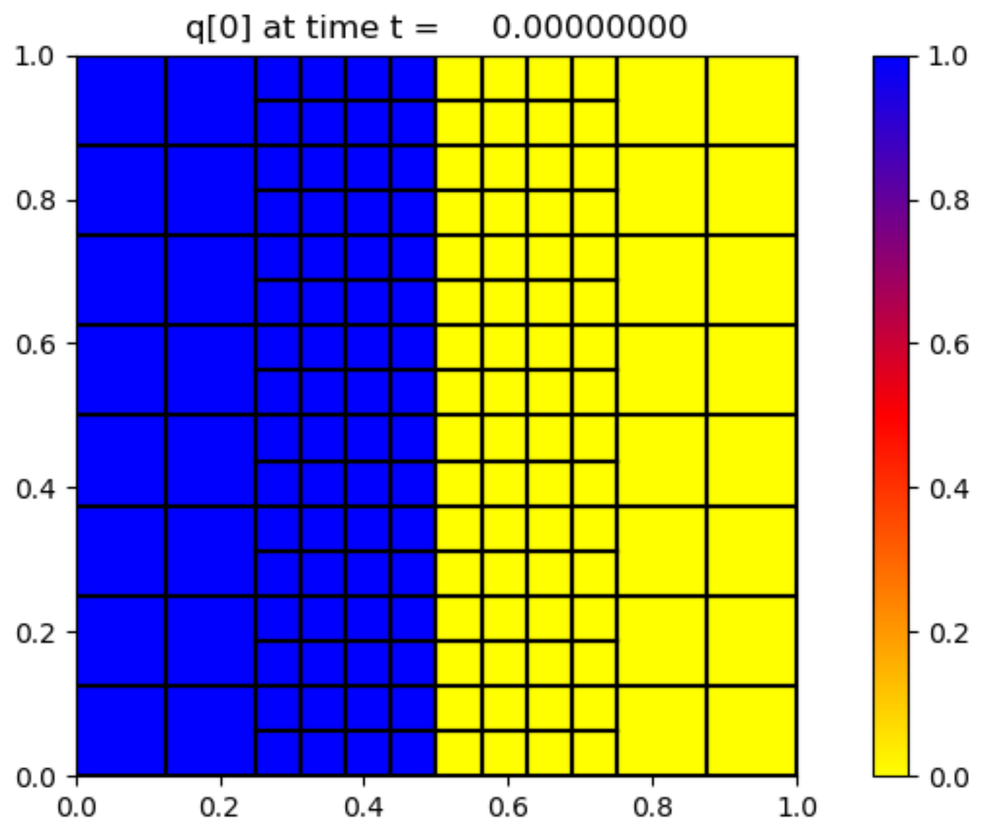
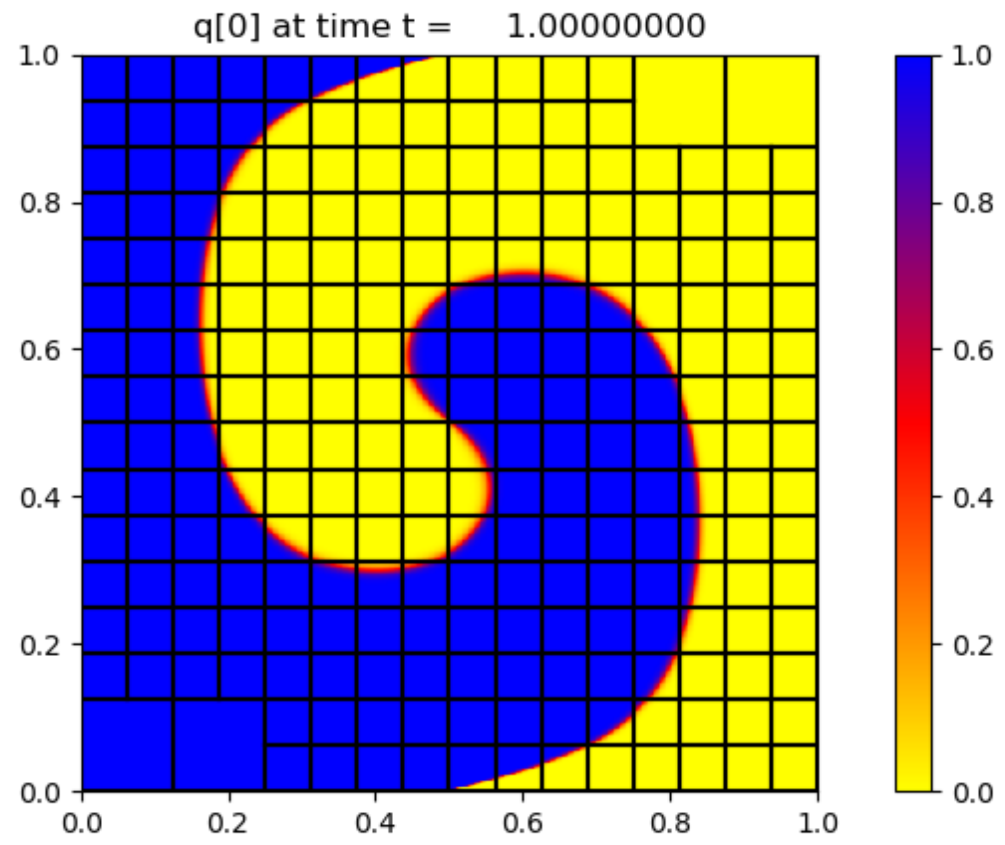
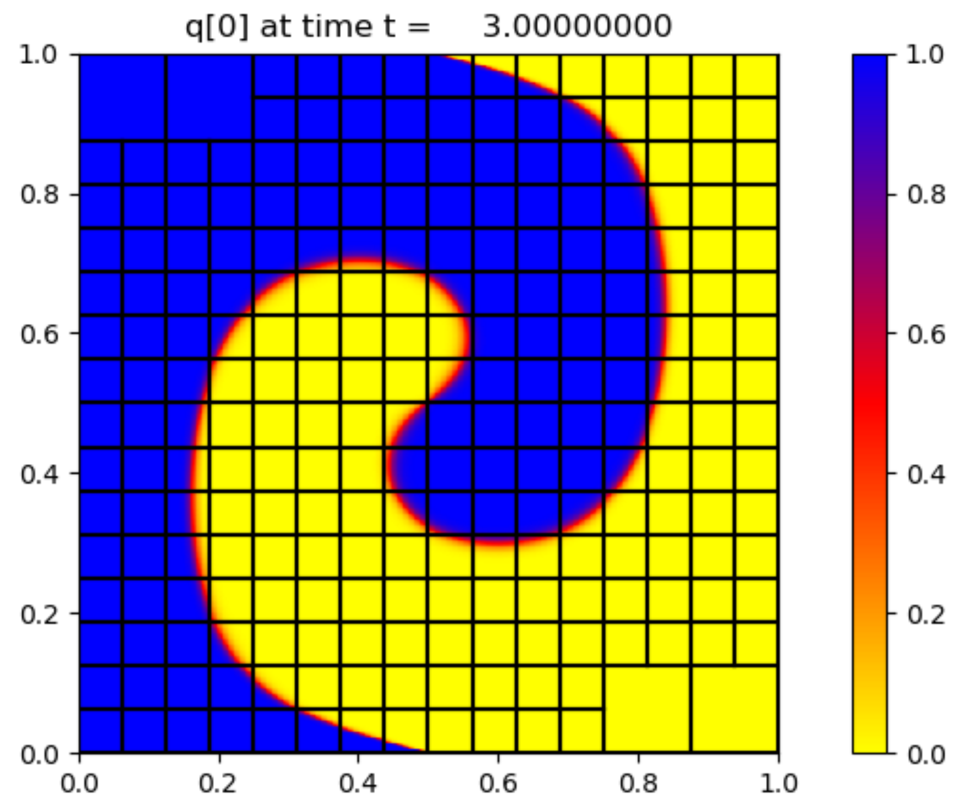
Goal : Improve performance of codes used in natural hazards modeling (tsunamis, storm surges, flooding, debris flows, avalanches, earthquakes and so on).

Solution method : Explicit finite volume methods on structured Cartesian blocks. **Patches are dynamically created and destroyed** to follow the solution features of interest.





GPU approach : Improve performance of kernels used to update the solution on a Cartesian patch.


q[0] at time t = 0.00000000
0.0
0.2
0.4
0.6
0.8
1.0

q[0] at time t = 1.00000000
0.0
0.2
0.4
0.6
0.8
1.0

q[0] at time t = 3.00000000
0.0
0.2
0.4
0.6
0.8
1.0

Progress

Monday :

-- Realized that we needed to coalesce the memory for several patches into one big patch that we could pass to the GPU.

Tuesday :

-- Work on memory coalescence.

-- Add reduction to compute maximum CFL number for variable time stepping. Used the CUB library.

Wednesday :

-- Finished debugging memory coalescence

-- Did extensive profiling.

-- Worked on acoustics problem and shallow water wave equation model.

-- Added timing markers

Thursday :

-- Began adding more complexity the computational kernel (wave limiters; second order terms)

-- Easy acceleration for scalar advection problem : Compute the velocity field on the GPU!

-- Profile with MPI

Friday

-- Debug correction terms to get second order version.

-- Work to get MPI version working with GPU version (didn't quite get this working)

Paradigms used

Solver kernel

- Put much of the solver into a single kernel.   This requires us to re-visit the logic of the original solver (a legacy code written in the early 90s!)
- Cartesian based solver should map very easily to the CUDA thread paradigm
- Lots of independent but lightweight tasks - no brainer for threads?

Approach to developing GPU accelerated code

- Core adaptive mesh refinement (AMR) routines in ForestClaw will remain on the CPU
- Create CUDA extension library of ForestClaw. This library will live alongside all other solver libraries (Clawpack, Ash3d, GeoClaw)
- Convert patch solvers to CUDA from Fortran. Computational patch solvers are small enough and isolated from the rest of the AMR infrastructure that this is fairly straightforward.

Launch the CUDA kernel

- Use a linear pool of threads. Map `threadIdx.x` to `(ix,iy)` and then to patch data
- One block per data patch, even if this means that some threads do more work.
- Allows for more flexibility in the kernel;

Outlook

This project looks very promising!

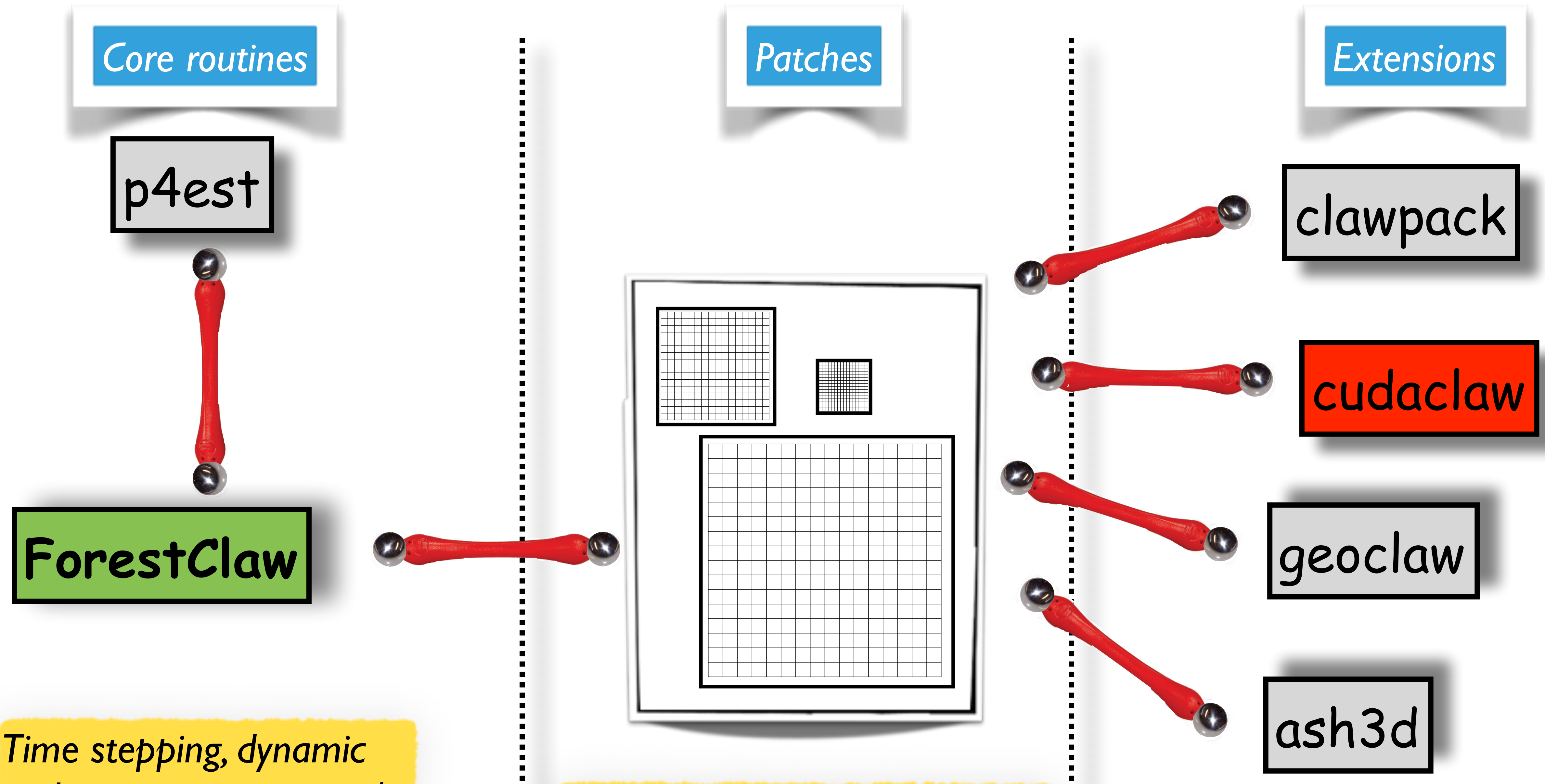
- No major road-blocks (with exception of some solver logic)
- No major branching issues
- We need to try larger problems with more field variables, MPI,
- General questions about best way to include CUDA extension in general library so that users without CUDA can still compile and run our code.
- Other aspects of teh code now look slow - filling ghost cell regions; MPI communication

General question : How does the CUDA programming and parallelization paradigm impact the development of numerical algorithms?

- Should conventional algorithms be revisted?

```
Test          WALLTIME    ADVANCE     GFILL    B4STEP2     STEP2   TOTAL STEPS
------------------------------------------------------------------------------
T1 (gpu)       140.612    137.670     4.691    128.616     8.936        709636
T2 (cpu)       170.595    167.298     4.616    128.629    38.601        709636
T3 (gpu)        15.263      9.941     4.798          0     9.866        707802
------------------------------------------------------------------------------
```

ForestClaw and library extensions



Time stepping, dynamic grid management, input/output.

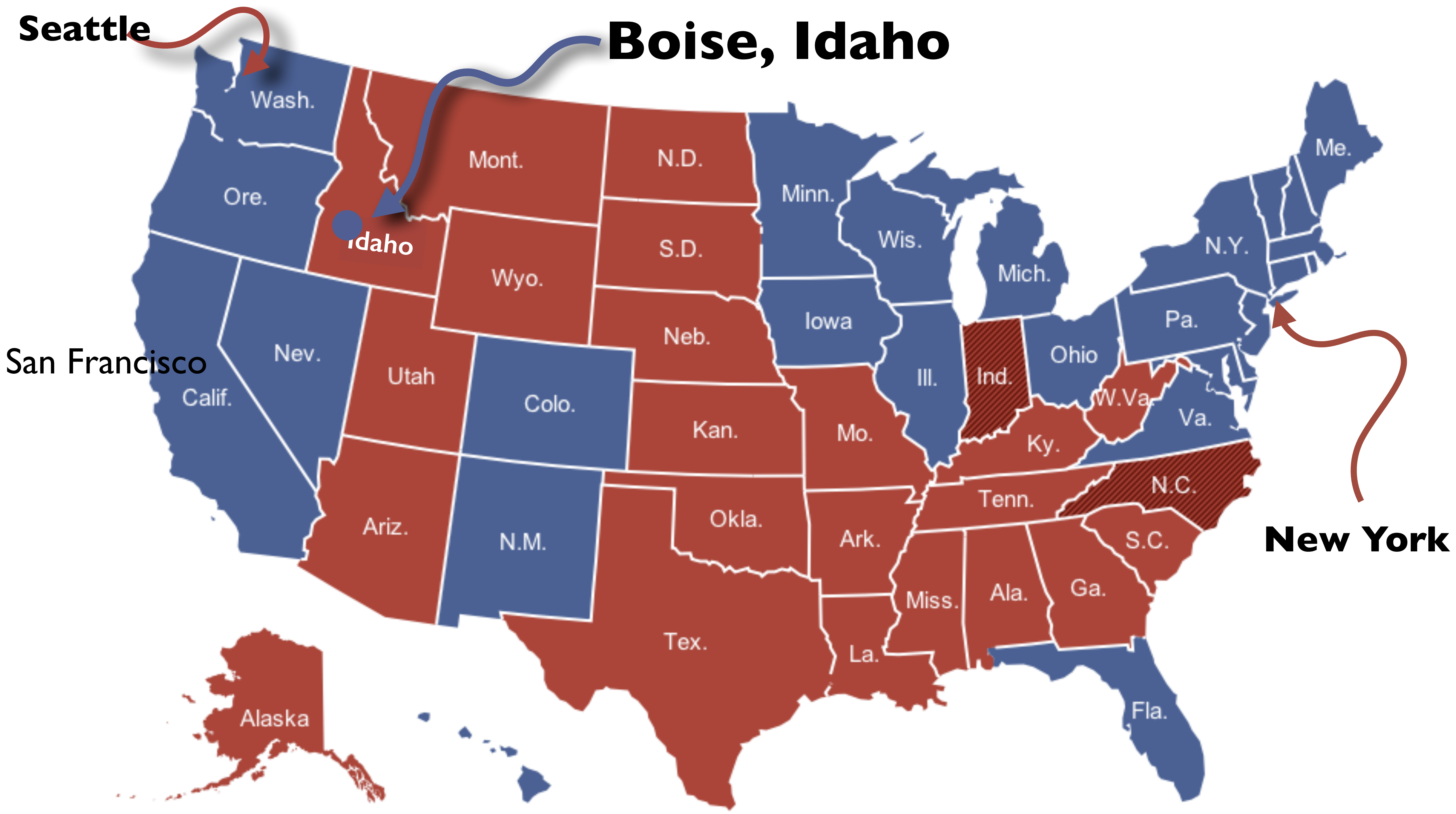
Tasks (tagging, building patches, etc) customized through use of function pointers stored in "virtual tables"

ForestClaw is mostly C; solvers extension libraries left largely untouched (in original Fortran)

Thanks to the Riemann Sweepers!

- **Scott**: Lots of programming experience (less so in CUDA) - still an undergraduate at Boise State University). Expert in Makefile systems, running on Daint. Learning PDE solvers.
- **Melody** : A second year graduate student at NYU (Courant Institute). Has already had significant CUDA experience. Learning wave-propagation algorithm.
- **Shawn** : Advanced graduate student at University of Washington.   Expert in profiling with NVIDA. Knows Wave-propagation algorithm backwards and forwards.
- **Donna :** Computational Mathematician

Thanks to the organizers and Mentors!


Seattle
Boise, Idaho
San Francisco
New York
Wash.
Ore.
Idaho
Mont.
N.D.
Minn.
Wis.
Mich.
Me.
N.Y.
S.D.
Wyo.
Pa.
Iowa
Neb.
Nev.
Calif.
Utah
Colo.
Ill.
Ind.
Ohio
W.Va.
Va.
Kan.
Mo.
Ky.
N.C.
Tenn.
Okla.
Ariz.
N.M.
Ark.
S.C.
Miss.
Ala.
Ga.
Tex.
La.
Alaska
Fla.

BOISE STATE™

Albertsons®

Micron®

Boise Cascade
Engineered Wood Products

SALMON AIR

NATIONAL INTERAGENCY FIRE CENTER
Boise, Idaho