Team

Pawel Biernacki, Andreas Bleuler, Sebastien Deldon, Claudio Gheller, Andreas Jocksch, Douglas Potter, Romain Teyssier

Problem trying to solve

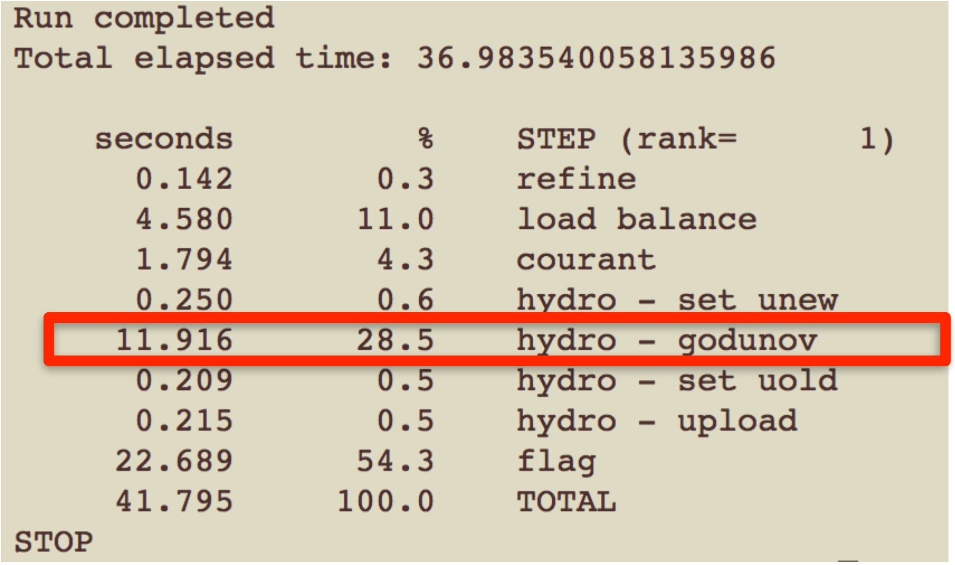
- Astrophysics hydrodynamics for star formation and galaxy evolution based on Adaptive Mesh Refinement approach

- Speeding up hydro kernel computations - GPU as an accelerator approach

Prior Profile

- Focusing on hiding the GPU data copying behind the CPU work

- Kernel not computationally expensive enough



```
Run completed
Total elapsed time: 36.983540058135986

    seconds          %     STEP (rank=       1)
      0.142        0.3     refine
      4.580       11.0     load balance
      1.794        4.3     courant
      0.250        0.6     hydro - set unew
     11.916       28.5     hydro - godunov
      0.209        0.5     hydro - set uold
      0.215        0.5     hydro - upload
     22.689       54.3     flag
     41.795      100.0     TOTAL
STOP
```

Evolution and Strategy

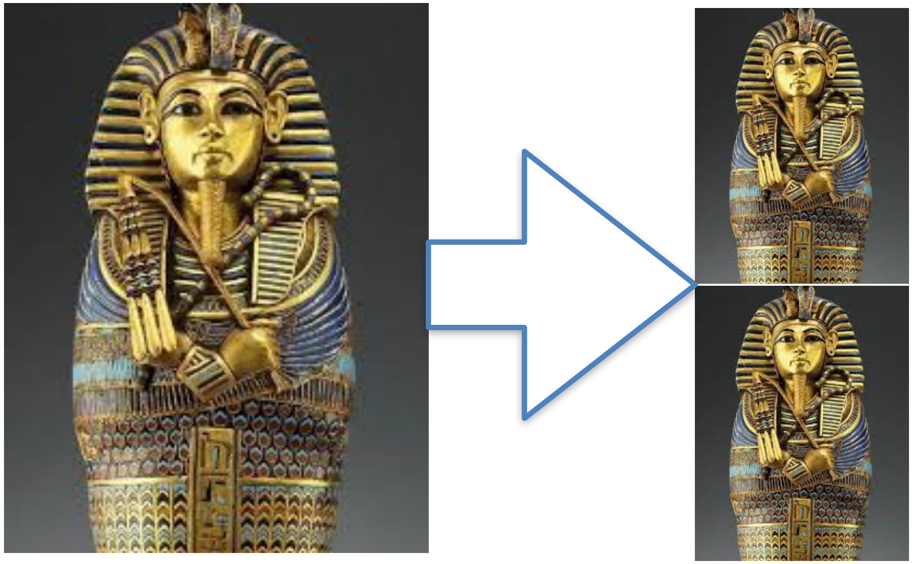
- Let’s do things in parallel! And refine the work by splitting it between group members!
- And then let’s try even more things!

Evolution and Strategy

- Let’s do things in parallel! And refine the work by splitting it between group members!
- And then let’s try even more things!

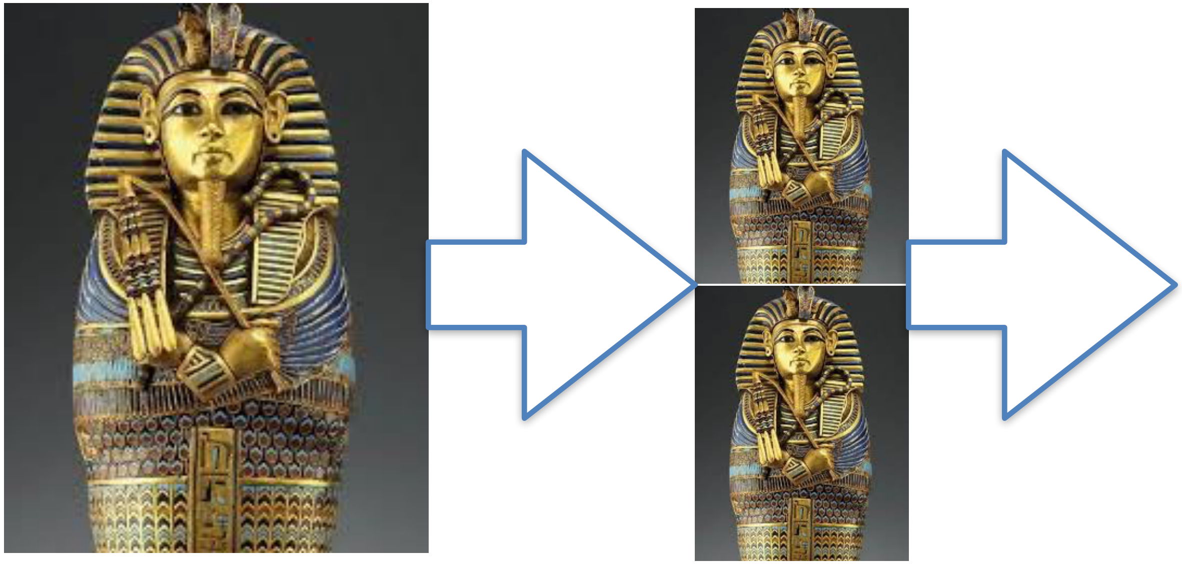
Evolution and Strategy

- Let’s do things in parallel! And refine the work by splitting it between group members!
- And then let’s try even more things!



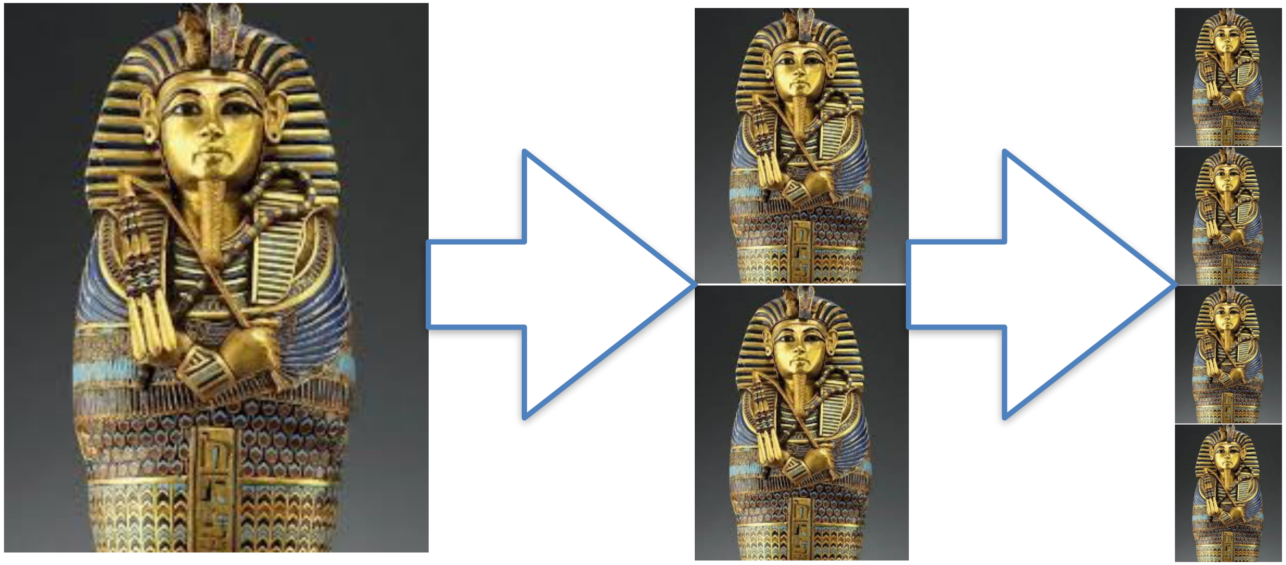
Evolution and Strategy

- Let’s do things in parallel! And refine the work by splitting it between group members!
- And then let’s try even more things!



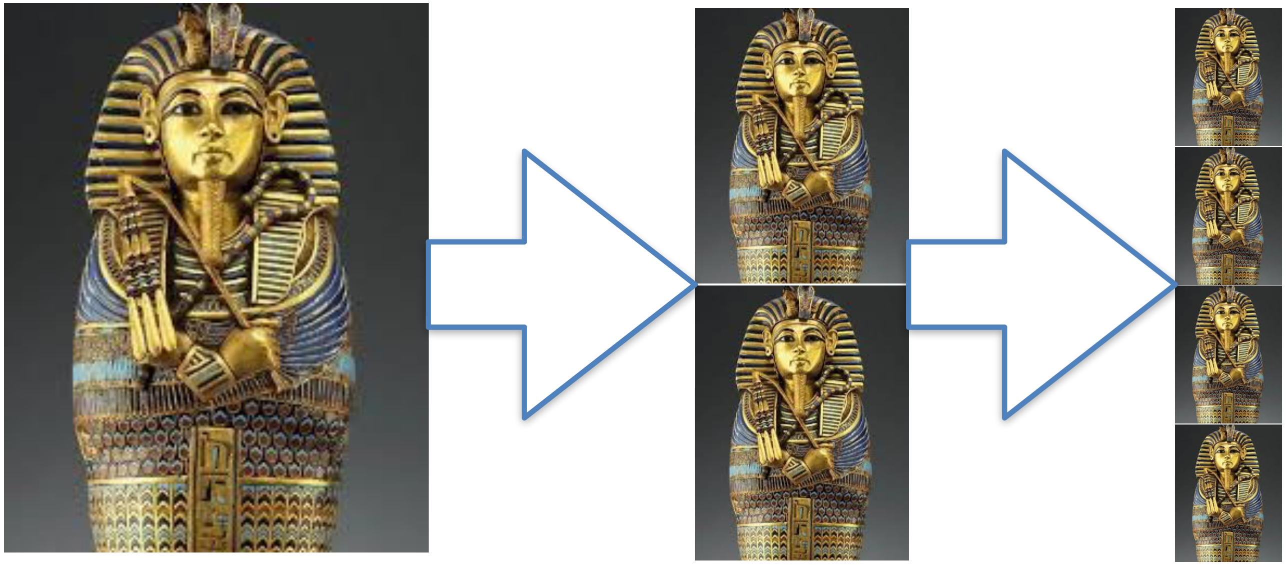
Evolution and Strategy

- Let’s do things in parallel! And refine the work by splitting it between group members!
- And then let’s try even more things!



Evolution and Strategy

- Let’s do things in parallel! And refine the work by splitting it between group members!
- And then let’s try even more things!



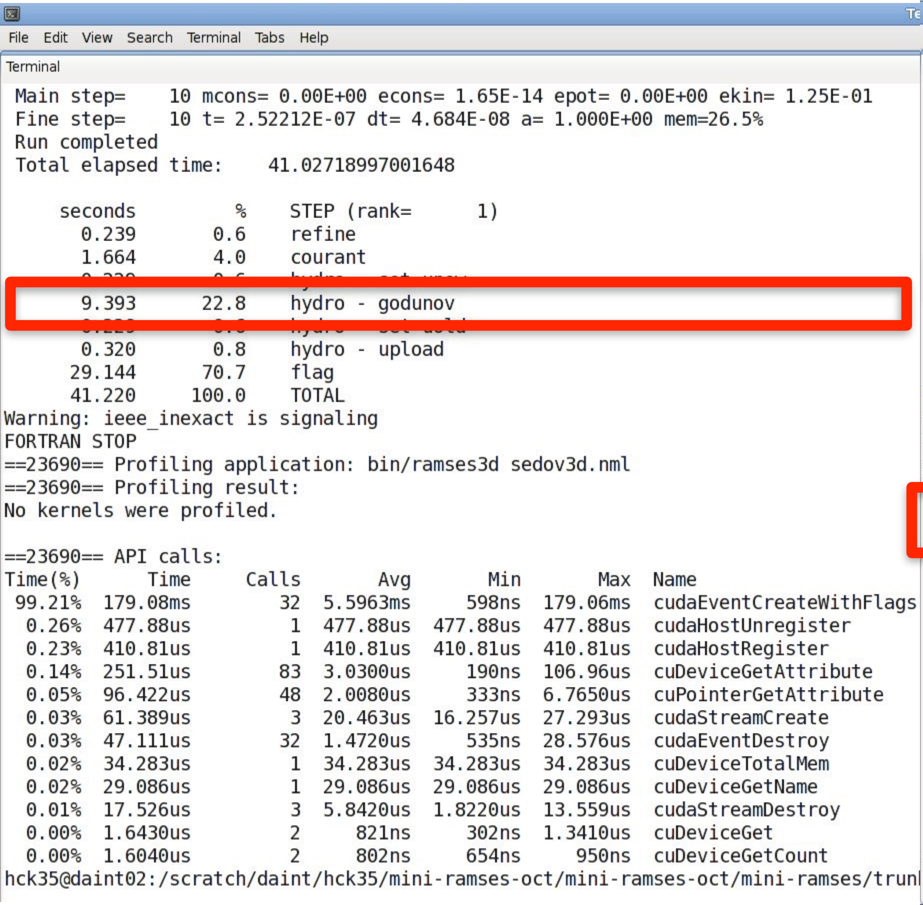
Full old Rad Trans (buggy)

Just new Rad Trans

New app - hydro many octs async

New app - hydro one oct async

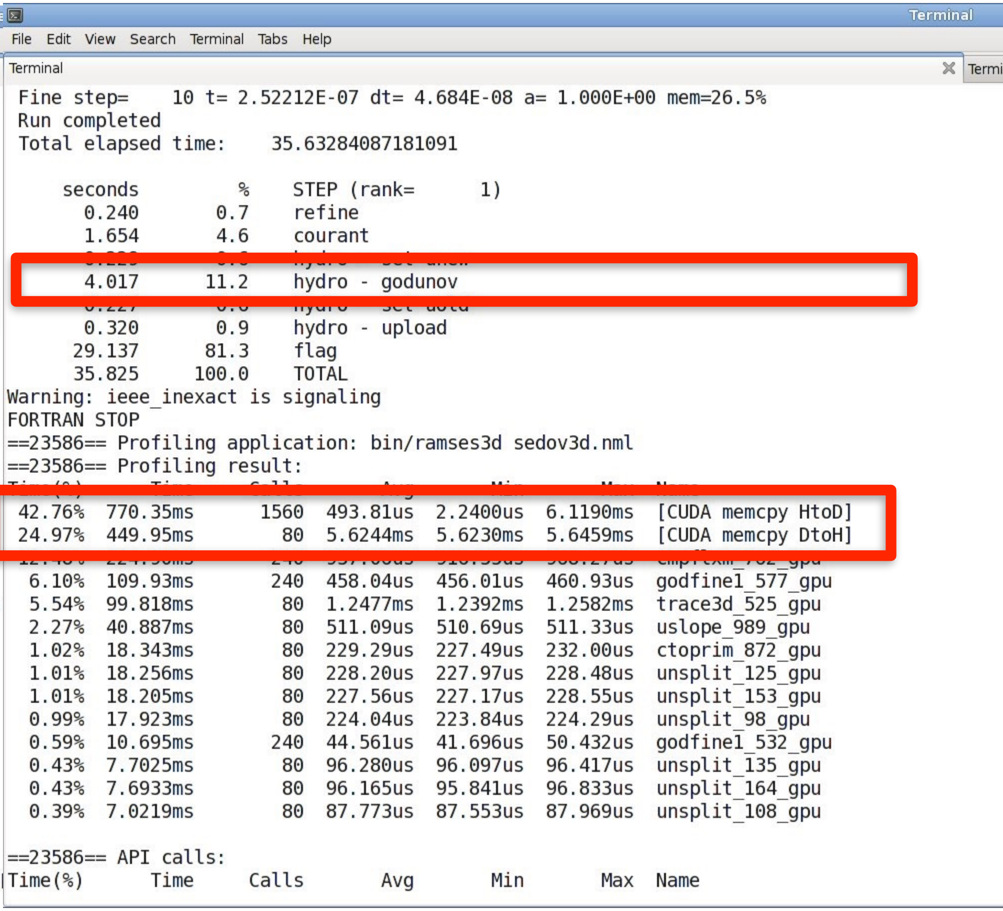
Results and Final Profile



```
 Main step=    10 mcons= 0.00E+00 econs= 1.65E-14 epot= 0.00E+00 ekin= 1.25E-01
 Fine step=    10 t= 2.52212E-07 dt= 4.684E-08 a= 1.000E+00 mem=26.5%
 Run completed
 Total elapsed time:    41.02718997001648

     seconds         %    STEP (rank=      1)
       0.239       0.6    refine
       1.664       4.0    courant
       0.229       0.6    hydro - set unew
       9.393      22.8    hydro - godunov
       0.229       0.6    hydro - set uold
       0.320       0.8    hydro - upload
      29.144      70.7    flag
      41.220     100.0    TOTAL
Warning: ieee_inexact is signaling
FORTRAN STOP
==23690== Profiling application: bin/ramses3d sedov3d.nml
==23690== Profiling result:
No kernels were profiled.

==23690== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
 99.21%  179.08ms        32  5.5963ms     598ns  179.06ms  cudaEventCreateWithFlags
  0.26%  477.88us         1  477.88us  477.88us  477.88us  cudaHostUnregister
  0.23%  410.81us         1  410.81us  410.81us  410.81us  cudaHostRegister
  0.14%  251.51us        83  3.0300us     190ns  106.96us  cuDeviceGetAttribute
  0.05%  96.422us        48  2.0080us     333ns  6.7650us  cuPointerGetAttribute
  0.03%  61.389us         3  20.463us  16.257us  27.293us  cudaStreamCreate
  0.03%  47.111us        32  1.4720us     535ns  28.576us  cudaEventDestroy
  0.02%  34.283us         1  34.283us  34.283us  34.283us  cuDeviceTotalMem
  0.02%  29.086us         1  29.086us  29.086us  29.086us  cuDeviceGetName
  0.01%  17.526us         3  5.8420us  1.8220us  13.559us  cudaStreamDestroy
  0.00%  1.6430us         2     821ns     302ns  1.3410us  cuDeviceGet
  0.00%  1.6040us         2     802ns     654ns     950ns  cuDeviceGetCount
hck35@daint02:/scratch/daint/hck35/mini-ramses-oct/mini-ramses-oct/mini-ramses/trun
```



```
 Fine step=    10 t= 2.52212E-07 dt= 4.684E-08 a= 1.000E+00 mem=26.5%
 Run completed
 Total elapsed time:    35.63284087181091

     seconds         %    STEP (rank=      1)
       0.240       0.7    refine
       1.654       4.6    courant
       0.229       0.6    hydro - set unew
       4.017      11.2    hydro - godunov
       0.227       0.6    hydro - set uold
       0.320       0.9    hydro - upload
      29.137      81.3    flag
      35.825     100.0    TOTAL
Warning: ieee_inexact is signaling
FORTRAN STOP
==23586== Profiling application: bin/ramses3d sedov3d.nml
==23586== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 42.76%  770.35ms      1560  493.81us  2.2400us  6.1190ms  [CUDA memcpy HtoD]
 24.97%  449.95ms        80  5.6244ms  5.6230ms  5.6459ms  [CUDA memcpy DtoH]
 12.48%  224.90ms       240  937.08us  916.95us  968.27us  cmpflxm_702_gpu
  6.10%  109.93ms       240  458.04us  456.01us  460.93us  godfine1_577_gpu
  5.54%  99.818ms        80  1.2477ms  1.2392ms  1.2582ms  trace3d_525_gpu
  2.27%  40.887ms        80  511.09us  510.69us  511.33us  uslope_989_gpu
  1.02%  18.343ms        80  229.29us  227.49us  232.00us  ctoprim_872_gpu
  1.01%  18.256ms        80  228.20us  227.97us  228.48us  unsplit_125_gpu
  1.01%  18.205ms        80  227.56us  227.17us  228.55us  unsplit_153_gpu
  0.99%  17.923ms        80  224.04us  223.84us  224.29us  unsplit_98_gpu
  0.59%  10.695ms       240  44.561us  41.696us  50.432us  godfine1_532_gpu
  0.43%  7.7025ms        80  96.280us  96.097us  96.417us  unsplit_135_gpu
  0.43%  7.6933ms        80  96.165us  95.841us  96.833us  unsplit_164_gpu
  0.39%  7.0219ms        80  87.773us  87.553us  87.969us  unsplit_108_gpu

==23586== API calls:
Time(%)      Time     Calls       Avg       Min       Max  Name
```

Results and Final Profile

```
==11336== Profiling application: /scratch/daint/ajocksch/Hackathon/ramses3d sedov3d.nml
==11336== Profiling result:
Time(%)      Time     Calls       Avg       Min       Max  Name
 43.60%  576.72ms       320  1.8022ms  329.16us  2.5166ms  [CUDA memcpy DtoH]
 20.71%  273.94ms       750  365.25us  2.2400us  2.0191ms  [CUDA memcpy HtoD]
 17.13%  226.64ms       240  944.34us  919.53us  982.99us  cmpflxm_757_gpu
  7.51%  99.327ms        80  1.2416ms  1.2317ms  1.2541ms  trace3d_519_gpu
  3.13%  41.416ms        80  517.70us  517.26us  517.96us  uslope_985_gpu
  1.46%  19.277ms        80  240.96us  230.66us  248.45us  ctoprim_868_gpu
  1.35%  17.894ms        80  223.67us  223.24us  223.97us  unsplit_151_gpu
  1.34%  17.751ms        80  221.89us  221.64us  222.18us  unsplit_96_gpu
  1.34%  17.723ms        80  221.53us  221.28us  221.86us  unsplit_123_gpu
  0.77%  10.200ms       240  42.499us  40.384us  46.753us  godfine1_2_672_gpu
  0.57%  7.5363ms        80  94.203us  93.793us  94.402us  unsplit_161_gpu
  0.57%  7.4813ms        80  93.516us  93.089us  93.730us  unsplit_133_gpu
  0.52%  6.9374ms        80  86.717us  86.401us  87.010us  unsplit_106_gpu
```

Time of memcpy hidden behind the CPU!!!

What problems you encountered

- Problems with legacy app structure
  - too many nested calls and rigid call structure
- Issues with algorithm
  - flexible data format, but problem not computationally expensive enough
- Random access to memory due to the data structure
- [TODO] Communication for boundaries from within the kernel - needs rethinking

Wishlist

- magic compiler flag: **“-faster” ;)**
- nvprof is great, but depends on CUDA version
- cutting-edge tools available at the moment of starting the hackathon (CUDA7.5 and pgprof relation)
- interchangeability of compilers

- BUT: a sneak peek at a new PGI compiler