氣象署-興大應數聯隊

2024 NCHC Open Hackathon

Members

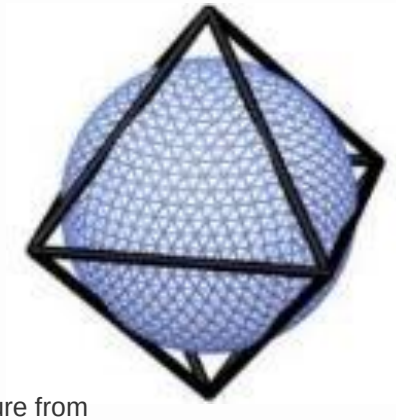
Shang-En Li[1] (李尚恩)
Jen-Her Chen[1] (陳建河)
Lu-Hung Chen[2] (陳律閎)
Ting-An Chen[2] (陳廷安)
Pang-Yen Liu[1] (劉邦彥)
Chun-Hao Teng[2] (鄧君豪)

Mentor

Leo Chen[3]

[1]Central Weather Administration
[2]Department of Applied Mathematics, National Chung Hsing University
[3]NVIDIA Corporation

Introduction



(figure from https://confluence.ecmwf.int/display/FCST/Introducing+the+octahedral+reduced+Gaussian+grid)

Model name: CWAGFS-TCo (global)

Resolution: ~28 km, 72 layers (TCo383L72)

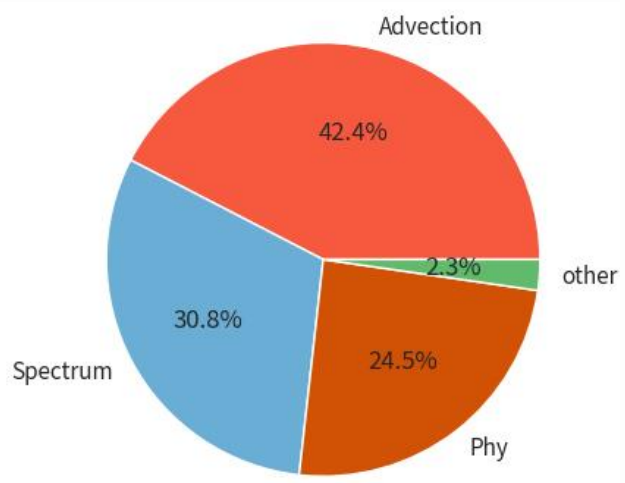
Dy-core: Semi-Lagrangian+Semi-implicit (2 time level)

Grid system: Octahedral reduced Gaussian grid

Programming language: Fortran

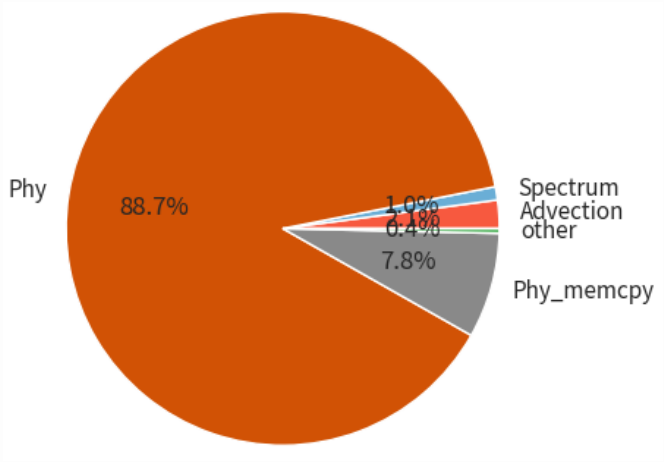
Current Porting Progress

Running on CPUs



	Time spent
Advection	42.4 (%)
Spectrum	30.8 (%)
Physics	24.5 (%)

Running on CPUs + GPUs



	Time spent
Physics	88.7 (%)
memcpy	7.8 (%)
others	3.5 (%)

Evolution and Strategy

Many do-loops in the physics code:

- GWD: 38
- Ozone:14
- PBL: 79

Accelerating path: OpenACC

The size of do-loop iterations:

1440~111744

Speedup ratio: 0.4~3x

```
subroutine diabat()
  do j = 1, latitude
  call ozone()
  call pbl()
  call gwd()
  enddo
end subroutine

subroutine pbl()
  do k = 1, levels
  do i = 1, latitude grids
  enddo
  enddo
end subroutine
```

Evolution and Strategy

Many do-loops in the physics code:

- GWD: 38
- Ozone:14
- PBL: 79

Accelerating path: OpenACC

The size of do-loop iterations:

1440~111744 → 7200000

```
subroutine diabat()
  call ozone()
  call pbl()
  call gwd()
end subroutine

subroutine pbl()
  do j = 1, latitude
  do k = 1, levels
  do i = 1, latitude grids
  enddo
  enddo
  enddo
end subroutine
```

Benchmark (32 CPU cores vs. 8 A100 GPUs)

Speedup ratio:

	Day0	Day1	Day2	final
GWD	40.8	40.8	51.5	56.6
Ozone	67.3	67.3	67.3	417
PBL	7.78	16.1	36.5	39.1

Execution time (execute 15 times, ms):

final	CPU	GPU
GWD	2370	41.82
Ozone	5655	13.56
PBL	5115	130.68

Notice:
No data transformation included in timing.
The first execution is not included.

Optimizing technique

Speedup ratio

	Day0	Day1	Day2	final
GWD	40.8	40.8	51.5	56.6
Ozone	67.3	67.3	67.3	417
PBL	7.78	16.1	36.5	39.1

Before (O(n), 3.87 ms)

```
do k  = 1, levozp-1
  do i  = 1, nxj(j1)
     con1 = (pp(i,kk,jj)-pl_pres(k)) / (pl_pres(k+1)-pl_pres(k))
     if ((con1.gt.0.0) .and. (con1.le.1.0)) then !pp(kk) in pl(k,k+1)
        con2=1.0-con1
        ozplout(i,kk,j,jj) = con2*ozwk2(k,j,jj) + con1*ozwk2(k+1,j,jj)
     endif
  enddo
enddo
```

After (O(log(n)), 0.33 ms)

```
left = 1
right = levozp
do while (right - left > 1)
  mid = (left + right)/2
  con1 = ppr-pl_pres(mid)
  if (con1 .lt. 0.) then
     right = mid
  else
     left = mid
  end if
end do
con1 = (ppr-pl_pres(left)) / (pl_pres(right)-pl_pres(left))
con2=1.0-con1
!$acc loop seq
do j  = 1, pl_coeff
  ozplout(i,kk,j,jj) = con2*ozwk2(left,j,jj) + con1*ozwk2(r
```

Optimizing technique

Speedup ratio

	Day0	Day1	Day2	final
GWD	40.8	40.8	51.5	56.6
Ozone	67.3	67.3	67.3	417
PBL	7.78	16.1	36.5	39.1

Before (acc routine, 30.5 ms)

```
!$acc parallel loop gang
do jj = 1, jlistnum
  call tridin_gpu(ix,myim(jj),km,ntrac,al(1,1,jj),ad(1,1,jj),&
                  au(1,1,jj),a1(1,1,jj),a2(1,1,jj),&
                  au(1,1,jj),a1(1,1,jj),a2(1,1,jj),fk(1,jj),fkk
enddo
```

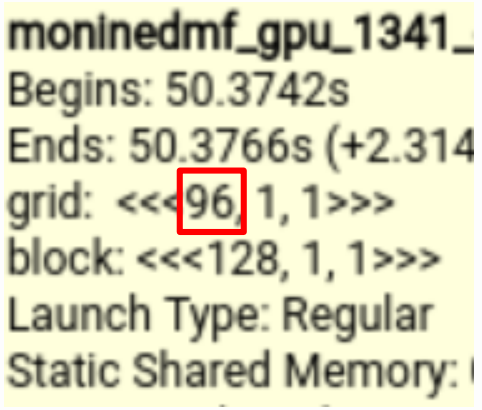
After (cuSPARSE, 2.96 ms)

```
!$acc parallel loop gang collapse(2) private(jj,i,k,accui)
do jj = 1,jlistnum
  do k = 1, km
    !$acc loop vector
    do i = 1, myim(jj)
      accui = nxjp_acc(jj)-1
      if (k .eq. km) then
        aug(accui+i,k) = 0.
      else
        aug(accui+i,k) = aug_backup(accui+i,k)
      endif
    enddo
  enddo
enddo
call cusparseDgtsvInterleavedBatch_async
     (handle,0,km,alg,adg,aug,a2g,nxptot,async_id)
```

Optimizing technique

Speedup ratio

	Day0	Day1	Day2	final
GWD	40.8	40.8	51.5	56.6
Ozone	67.3	67.3	67.3	417
PBL	7.78	16.1	36.5	39.1



Before (gang vector)

```
!$acc parallel loop gang
do jj = 1, jlistnum
  !$acc loop vector
  do i = 1,myim(jj)
```

After (collapse, 4x faster)

```
!$acc parallel loop collapse(2)
do jj = 1, jlistnum
  do i=1, ix
    if (i .le. myim(jj)) then
```

Optimizing technique

Speedup ratio

	Day0	Day1	Day2	final
GWD	40.8	40.8	51.5	56.6
Ozone	67.3	67.3	67.3	417
PBL	7.78	16.1	36.5	39.1

Do-loops fusion:
GWD: 38 → 13
Ozone: 14 → 7
PBL: 79 → 30

Asynchronous operation

Energy Efficiency

Units Average Speedup

70.6x

Node Replacement

70.6x

Node Power Efficiency

12.3x

Metric Tons of CO_2

462

INPUTS	
# CPU Cores	32
# GPUs (A100)	8
Application Speedup	70.6x
Node Replacement	70.6x

GPU NODE POWER SAVINGS			
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Savings
Compute Power (W)	77,660	6,500	71,160
Networking Power (W)	3,278	93	3,186
Total Power (W)	**80,938**	**6,593**	74,346
Node Power efficiency	**12.3x**		

ANNUAL ENERGY SAVINGS PER GPU NODE			
	AMD Dual Rome 7742	8x A100 80GB SXM4	Power Savings
Compute Power (kWh/year)	680,302	56,940	623,362
Networking Power (kWh/year)	28,719	814	27,906
Total Power (kWh/year)	**709,021**	**57,754**	**651,267**
$/kWh	0.18		
Annual Cost Savings	117,228.08		
3-year Cost Savings	351,684.23		
Metric Tons of CO_2	462		
Gasoline Cars Driven for 1 year	100		
Seedlings Trees grown for 10 years	7,633		

(source: Link)

What did we learn?

The usage of OpenACC.

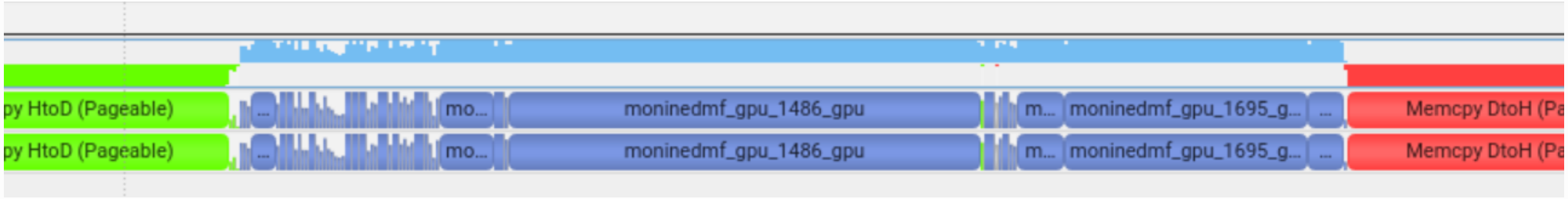
The usage of cuSPARSE.

The usage of Nsight system.

The usage of NVTX.

Techniques to speedup kernels.

Avg	Name
19.846 ms	moninedmf_gpu_1486_gpu
10.608 ms	moninedmf_gpu_1695_gpu
2.449 ms	moninedmf_gpu_1341_gpu
2.179 ms	moninedmf_gpu_1651_gpu
1.467 ms	moninedmf_gpu_1750_gpu
1.044 ms	moninedmf_gpu_450_gpu
673.704 µs	moninedmf_gpu_1531_gpu
610.642 µs	mfpbl_gpu_193_gpu

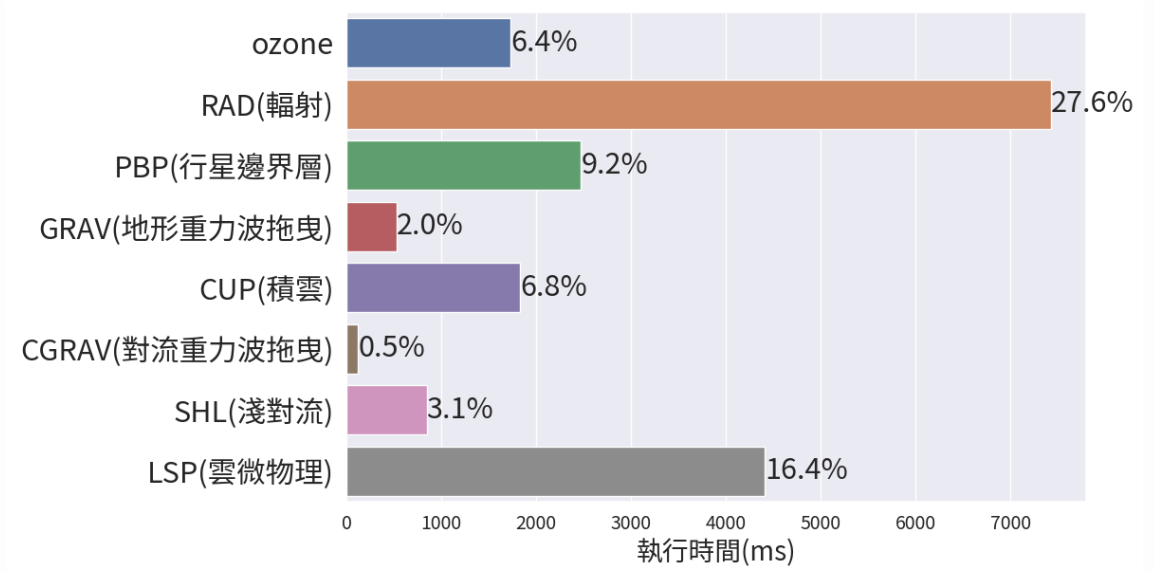


Final Thoughts

- Was this Open Hackathon worth it?
  Yes.
- Will you continue development?
  Yes.
- Next steps, future plans.

  Porting other physics module.

  Integration Test.



V

V

V

- What sustained resources or support will be critical for your work after the event?
  Mentors.

Acknowledgement

- Mentor
- NCHC Open Hackathon

Summary

- Application: CWAGFS-TCo (A weather forecasting model)
- Accelerating path: OpenACC and cuSPARSE
- Physics computation spends 88.7% of total time when TCo runs on CPUs + GPUs.
- We accelerated GWD (56.6x), ozone (417x), and PBL (39.1x) modules.
- Optimizing algorithm, fusing and collapsing do-loops, cuSPARSE, and asynchronous operation are used to speed up codes.
- In the future work, we will accelerate other modules and do integration test.

Thank you~