



NCAR is sponsored by **National Science Foundation**



Hands-On Session Using OpenACC in MPAS-A

By: Daniel Howard dhoward@ucar.edu, Consulting Services Group, CISL & NCAR

Date: April 28th 2022

In this notebook, we explore the GPU enabled MPAS-A (Model Prediction Across Scales-Atmosphere) to apply techniques learned from MiniWeather and implementing OpenACC to develop for GPUs.

- Review of exercises from prior OpenACC/MiniWeather sessions Part 1 and Part 2
- MPAS-Atmosphere model overview
- Managing GPU data in large software projects
- Assessing performance of extracted GPU kernels in MPAS-A

Head to the NCAR JupyterHub portal and start a JupyterHub session on Casper login (or batch nodes using 1 CPU, no GPUs) and open the notebook in 07_HandsOnMPASA.ipynb. Be sure to clone (if needed) and update/pull the NCAR GPU_workshop directory.

Use the JupyterHub GitHub GUI on the left panel or the below shell commands
git clone git@github.com:NCAR/GPU_workshop.git
git pull

Workshop Etiquette

- Please mute yourself and turn off video during the session.
- Questions may be submitted in the chat and will be answered when appropriate. You may also raise your hand, unmute, and ask questions during Q&A at the end of the presentation.
- By participating, you are agreeing to UCAR's Code of Conduct
- Recordings & other material will be archived & shared publicly.
- Feel free to follow up with the GPU workshop team via Slack or submit support requests to support.ucar.edu
 - Office Hours: Asynchronous support via Slack or schedule a time with an organizer

Notebook Setup

Set the PROJECT code to a currently active project, ie UCIS0004 for the GPU workshop, and QUEUE to the appropriate routing queue depending on if during a live workshop session (gpuworkshop), during weekday 8am to 5:30pm MT (gpudev), or all other times (casper). Due to limited shared GPU resources, please use GPU_TYPE=gp100 during the workshop. Otherwise, set GPU_TYPE=v100 (required for gpudev) for independent work. See Casper queue documentation for more info.

Notebook Setup

Set the PROJECT code to a currently active project, ie UCIS0004 for the GPU workshop, and QUEUE to the appropriate routing queue depending on if during a live workshop session (gpuworkshop), during weekday 8am to 5:30pm MT (gpudev), or all other times (casper). Due to limited shared GPU resources, please use GPU_TYPE=gp100 during the workshop. Otherwise, set GPU_TYPE=v100 (required for gpudev) for independent work. See Casper queue documentation for more info.

In []:

export PROJECT=UCIS0004 export QUEUE=gpudev export GPU_TYPE=v100

Review of MiniWeather Performance Optimization

At the end of last session, it was suggested to use async and predominantly collapse clauses to achieve optimal performance in MiniWeather kernels. Using NX=1024 and NZ=512, the most expensive kernel in terms of compute time was at Line 231 in the semi_discrete_step subroutine, with NVCOMPILER_ACC_TIME statistics highlighted below:

```
/qlade/u/home/dhoward/GPU workshop/05 DirectivesOpenACC/fortran/miniWeather mpi exercise2.F90 # Source file
with OpenACC kernel code
semi discrete step NVIDIA devicenum=0
                                                    # Name of subroutine from which kernel is launched
time(us): 62,147
257: compute region reached 924 times
                                                    # Specific line number for GPU kernel and number times
reached/launched
257: kernel launched 924 times
grid: [16384] block: [128]
                                                    # Arrangement of gang/worker/vector in terms of grids
and blocks
device time(us): total=62,147 max=70 min=66 avg=67 # Timing statistics of the GPU kernel
elapsed time(us): total=76,527 max=87 min=80 avg=82 # Timing statistics of the CPU call (less accurate with
asynchronous execution)
257: data region reached 1848 times
```

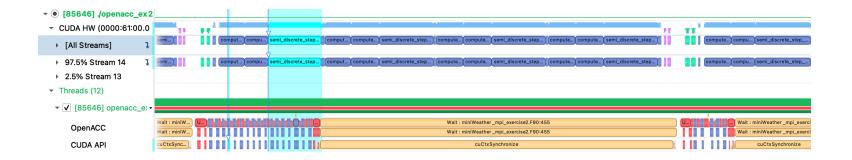
The arrangement of **gang/worker/vector** units is provided by **grid:** [NUM_GANGS] and **block:** [VECTOR_LENGTH x NUM_WORKERS]. The number of workers was 1 in the previous case so is omitted.

Running this version with the NVIDIA NSight Systems Profiler (discussed in later session), we can get a visual representation of the model runtime. You can download and view this profile using the NVIDIA NSight Systems client by downloading (SHIFT + RIGHT-CLICK) MW_baseline.nsys-rep in this folder.

	ne ×								
= Timeline	View -								🔤 Q 1x 🔤 🔥 🔥 🔤 🖓 🔤 🔤
	1s •	7.8ms +847.9ms	+848ms +848.1ms	+848.2ms +848.3m	s +848.4ms	+848.5ms +84	8.6ms +848	3.7ms +848.8ms +848.9ms +	849ms +849.1ms +849.2ms +849.3ms +849.4ms
CPU (72)									
Processes	s (2)								
• • [856	46] ./openacc_ex2								
- CUDA	HW (0000:61:00.0	11					11 - C		11 17 11
► [All S	Streams]	semi	nput compu semi_discrete_step	comput compute semi_discrete_	step) compute) compu	ite) semi_discrete_step	compute compu	tesemi_discrete_step	compu[semi_discrete_step]
▶ 97.5%	% Stream 14 🛛 🖡	semi)	nput compu semi_discrete_step	comput compute semi_discrete_	step) (compute) compu	te) semi_discrete_step	.) (compute) compu	te)semi_discrete_step)	compu[semi_discrete_step]
▶ 2.5%	6 Stream 13								
- Thread	ds (12)								
- V [8	5646] openacc_e: •								
		Wait : miniW			Wait : miniWea	ather _mpi_exercise2.F90:-	155	muidue	Wait : miniWeather_mpi_exercise2 🔲 🚺 🖬 Wait : miniWeather_mpi_exercise2.F
Ope	enACC	Wait : miniW				ather _mpi_exercise2.F90:			Wait : miniWeather _mpi_exercise2) Wait : miniWeather _mpi_exercise2.F
CU	IDA API	cuCtxSync			cu	CtxSynchronize			cuCtxSynchronize cuCtxSynchronize
Pro	ofiler overhead								
		•							
	 Name 								
707					Start	Duration	GPU	Context	Name Description:
	compute_te	endencies_x_310_gpu	J		Start 1.84793s	Duration 51.104 μs	GPU GPU 0	Context Stream 14	
708		endencies_x_310_gpu endencies_x_343_gp							▲ Description: semi_discrete_step_257_gpu Begins: 1.84803s
	compute_te				1.84793s	51.104 μs	GPU 0	Stream 14	Description: semi_discrete_step_257_gpu
709	compute_te	endencies_x_343_gp			1.84793s 1.84798s	51.104 μs 48.864 μs	GPU 0 GPU 0	Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84814s (+108.320 μs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>>
709 710	compute_te semi_discre set_halo_va	endencies_x_343_gp ete_step_257_gpu	u		1.84793s 1.84798s 1.84803s	51.104 μs 48.864 μs 108.320 μs	GPU 0 GPU 0 GPU 0	Stream 14 Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes
709 710 711	compute_te semi_discre set_halo_ve compute_te	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu	u		1.84793s 1.84798s 1.84803s 1.84814s	51.104 μs 48.864 μs 108.320 μs 5.343 μs	GPU 0 GPU 0 GPU 0 GPU 0	Stream 14 Stream 14 Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes
709 710 711 712	compute_te semi_discre set_halo_va compute_te compute_te	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu	u		1.84793s 1.84798s 1.84803s 1.84814s 1.84815s	51.104 μs 48.864 μs 108.320 μs 5.343 μs 54.303 μs	GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Stream 14 Stream 14 Stream 14 Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Registers Per Thread: 50 Local Memory Per Thread: 0 bytes
709 710 711 712 713	compute_te semi_discre set_halo_va compute_te compute_te semi_discre	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_410_gpu	u		1.84793s 1.84798s 1.84803s 1.84814s 1.84815s 1.8482s	51.104 μs 48.864 μs 108.320 μs 5.343 μs 54.303 μs 57.120 μs	GPU 0	Stream 14 Stream 14 Stream 14 Stream 14 Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84804s (+108.320 μs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Tota: 102, 285,160 bytes Local Memory Tota: 102, 285,160 bytes
709 710 711 712 713 714	compute_tet semi_discre set_halo_va compute_tet compute_tet semi_discre set_halo_va	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_410_gpu ete_step_257_gpu	u u		1.84793s 1.84798s 1.84803s 1.84814s 1.84815s 1.8482s 1.8482s	51.104 μs 48.864 μs 108.320 μs 5.343 μs 54.303 μs 57.120 μs 109.407 μs	GPU 0	Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Per Thread: 0 bytes Local Memory Total: 102,236,160 bytes Shared Memory Bank Size: 4 B
709 710 711 712 713 714 715	compute_tt semi_discre set_halo_va compute_tt compute_tt semi_discre set_halo_va compute_tt	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_410_gpu ete_step_257_gpu alues_z_511_gpu	u u u		1.84793s 1.84798s 1.84803s 1.84814s 1.84815s 1.8482s 1.8482s 1.84826s 1.84837s	51.104 μs 48.864 μs 108.320 μs 5.343 μs 54.303 μs 57.120 μs 109.407 μs 5.952 μs	GPU 0	Stream 14 Stream 14 Stream 14 Stream 14 Stream 14 Stream 14 Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Per Thread: 50 Local Memory Per Thread: 0 bytes Local Memory Per Thread: 0 bytes Shared Memory Bank Size: 4 B Theoretical occupancy: 56.25 %
709 710 711 712 713 714 715 716	compute_tt semi_discre set_halo_va compute_tt compute_tt semi_discre set_halo_va compute_tte compute_tte	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_410_gpu ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu	u u u		1.84793s 1.84798s 1.84803s 1.84814s 1.84815s 1.8482s 1.8482s 1.84826s 1.84837s 1.84838s	51.104 μs 48.864 μs 108.320 μs 5.343 μs 54.303 μs 57.120 μs 109.407 μs 5.952 μs 54.528 μs	GPU 0	Stream 14	 Description: sem_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84804s (+108.320 μs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Tota: 102, 236, 160 bytes Shared Memory Tact: 102, 236, 160 bytes Shared Memory Bank Size: 4 B Theoretical occupancy: 56, 25 % Launched from thread: 85646 Latency: <58, 133 μs
709 710 711 712 713 714 715 716 716 717	compute_tt semi_discre set_halo_va compute_tt compute_tt semi_discre set_halo_va compute_tt compute_tt	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_372_gpu	u u u		1.84793s 1.84798s 1.84798s 1.84814s 1.84814s 1.84815s 1.8482s 1.8482s 1.8482s 1.8483s 1.8483s 1.8483s	51.104 µs 48.864 µs 108.320 µs 5.343 µs 54.303 µs 57.120 µs 109.407 µs 5.952 µs 54.528 µs 56.736 µs	GPU 0	Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Per Thread: 0 bytes Local Memory Per Thread: 0 bytes Shared Memory Bank Size: 4 B Shared Memory Bank Size: 4 B Theoretical cocupancy: 56, 25 % Launched from thread: 85646 Latency: +83.133 µs Correlation ID: 4118
709 710 711 712 713 714 715 716 716 717 718	compute_tt semi_discre set_halo_va compute_tt compute_tt semi_discre set_halo_va compute_tt compute_tt semi_discre set_halo_va	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_372_gpu ete_step_257_gpu	u u u u		1.84793s 1.84798s 1.84798s 1.84803s 1.84814s 1.84815s 1.84825s 1.84826s 1.84837s 1.84837s 1.84838s 1.84843s 1.84849s	51.104 µs 48.864 µs 108.320 µs 5.343 µs 57.120 µs 57.120 µs 5.952 µs 5.4528 µs 56.736 µs 110.144 µs	GPU 0	Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> block: <<<128, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Total: 102,236,160 bytes Shared Memory Bark Size: 4 B Theoretical occupancy: 56.25 % Launched from thread: 85646 Latency: <<83.133 µs
709 710 711 712 713 714 715 716 716 717 718 719	compute_tt semi_discre set_halo_va compute_tt compute_tt semi_discre set_halo_va compute_tt semi_discre set_halo_va compute_tt	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu ete_step_257_gpu alues_z_511_gpu andencies_z_372_gpu endencies_z_410_gpu ete_step_257_gpu alues_z_511_gpu	u u u u u		1.84793s 1.84793s 1.84798s 1.84803s 1.84814s 1.84815s 1.84815s 1.8482s 1.8482s 1.84837s 1.84837s 1.84837s 1.84837s 1.84837s 1.84837s 1.84837s 1.84837s 1.84845s 1.84845s 1.84845s	51.104 µs 48.864 µs 108.320 µs 5.343 µs 54.303 µs 57.120 µs 109.407 µs 5.952 µs 54.528 µs 56.736 µs 110.144 µs 5.631 µs	GPU 0	Stream 14 Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s Ends: 1.84814s (+108.320 µs) grid: <<<16384, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Per Thread: 0 bytes Local Memory Per Thread: 0 bytes Shared Memory Bank Size: 4 B Shared Memory Bank Size: 4 B Theoretical accurancy: 56,25 % Launched from thread: 85646 Latency: +83.133 µs Correlation ID: 4118
709 7710 7711 7712 7713 7714 7715 7716 7717 7718 7719 7720	compute_te semi_discre set_halo_va compute_te compute_te semi_discre set_halo_va compute_te semi_discre set_halo_va compute_te semi_discre set_halo_va compute_te	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_410_gpu ete_step_257_gpu alues_z_511_gpu andencies_z_372_gpu	u u u u u		1.84793s 1.84798s 1.84798s 1.84814s 1.84814s 1.84815s 1.8482s 1.8482s 1.84838s 1.84838s 1.84843s 1.84843s 1.84849s 1.8486s	51.104 µs 48.864 µs 108.320 µs 5.343 µs 54.303 µs 57.120 µs 109.407 µs 5.952 µs 56.528 µs 56.736 µs 56.736 µs 51.0144 µs 5.631 µs	GPU 0 GPU 0	Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s Ends: 1.84184 (+108.320 µs) grid: <<<16384, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Per Thread: 50 Local Memory Per Thread: 0 bytes Shared Memory Bank Size: 4 B Shared Memory Bank Size: 4 B Theoretical accurancy: 56,25 % Launched from thread: 56464 Latency: +83.133 µs Correlation ID: 4118
708 709 710 711 712 713 713 714 715 7716 7716 7717 7718 7719 720 721 722	compute_te semi_discre compute_te compute_te semi_discre set_halo_va compute_te compute_te semi_discre set_halo_va compute_te semi_discre set_halo_va compute_te semi_discre	endencies_x_343_gp ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu ete_step_257_gpu alues_z_511_gpu endencies_z_372_gpu endencies_z_410_gpu alues_z_511_gpu alues_z_511_gpu endencies_z_372_gpu andencies_z_372_gpu endencies_z_372_gpu	u u u u u		1.84793s 1.84798s 1.84798s 1.84819s 1.84815s 1.8482s 1.8482s 1.8482s 1.84835s 1.84835s 1.84843s 1.84843s 1.8486s 1.84861s 1.84866s	51.104 µs 48.864 µs 108.320 µs 5.343 µs 57.120 µs 109.407 µs 5.952 µs 54.528 µs 56.736 µs 110.144 µs 5.831 µs 54.815 µs	GPU 0 GPU 0	Stream 14	 Description: semi_discrete_step_257_gpu Begins: 1.84803s Ends: 1.84803s Ends: 1.84184 (+108.320 µs) grid: <<<16384, 1, 1>>> Launch Type: Regular Static Shared Memory: 0 bytes Dynamic Shared Memory: 0 bytes Local Memory Per Thread: 50 Local Memory Per Thread: 0 bytes Shared Memory Bank Size: 4 B Shared Memory Bank Size: 4 B Theoretical accurancy: 56,25 % Launched from thread: 56464 Latency: +83.133 µs Correlation ID: 4118

This timeline shows the kernels running on the GPU runtime in the upper **blue** compute kernels, **pink** device to host transfers, and **teal** host to device transfers segments. The lower segments show the CPU runtime in **blue** compute kernel launches, **red** data directives/regions, and beige **wait/synchronize** sections.

The bright blue highlights the most expensive GPU kernel in the semi_discrete_step subroutine with the associated launch call from the CPU highlighted earlier in the timeline.



Since we used async, the GPU kernels run right after one another without any kernel launch/exit costs.

If we did not use async, the profile would look like this (MW_noasync.qdrep) and time would be lost as the CPU waits between every kernel launch/exit.

	e 🗙 MW_noasync	<u>^</u>									
■ Timeline \	View 👻								📼 Q	1x	2 warnings,14 mes
	1s •	+579ms +579.1ms +579.2ms +	-579.3ms +579.4ms	+579.5ms +579.	6ms +579.7ms +5	79.8ms +579.9ms +580m	ns +580.1ms +	-580.2ms +580	0.3ms +580.4ms	+580.5ms +580.6ms	+580.7ms +580.8ms +580
CPU (72)											
Processes	s (2)										
- 🖲 [9477	71] openacc_ex2										
- CUDA I	HW (0000:61:00.0	· · · · · · · · · · · · · · · · · · ·						'n n '		1 n n 1	
▶ 92.4%	% Kernels 1	semi	semi_dis compu	J [compu] [semi_di	is compu compu	it] [semi_dis] [compu] cor	mput semi_dis	I I	compu compu	mi_dis	comput comp semi_dis
▶ 7.6%	Memory	11 11						11 📕		11 0	
 Thread: 	is (11)										
▼ √ [94	4771] openacc_ex •										
		Com	t Compute C	te	C Comput Comput	e Compute C	nput Compute		Comput Comput) Cor	npute C	Compute Compu Compute C
Ope	enACC	Wait Wait : Wait :	- Wait : mi Wait : r	m Wait : m Wait : m	ini Wait : m Wait : r	n Wait : mini Wait : Wa	it : m Wait : min		Wait : Wait : Wi	it : min	Wait : m Wait : Wait : min
CUE	DA API	cuStCuStreCuStre	e_ cuStream cuStrea	a cuStrea cuStrea	m cuStrea cuStre	a cuStream cuStrea cuS	Strea		cuStrea cuStre cu	Stream	cuStrea cuStre cuStream
Prof	filer overhead										
10 thr	reads hidden – +										_
- 🔿 [9477	78] orted										
 Thread 	is (4)										
√ [94	4778] orted 🝷										
		4									
vents View		•									
vents View	• •									Alama	
		٩								Name •	
	 Name 					Start	Duration	GPU	Context	 Description: 	
949	Name compute_te	andencies_x_310_gpu				1.57844s	52.896 µs	GPU 0	Stream 13	 Description: semi_discrete_s 	
949 950	Name compute_te compute_te	endencies_x_310_gpu endencies_x_343_gpu				1.57844s 1.57851s	52.896 μs 49.791 μs	GPU 0 GPU 0	Stream 13 Stream 13	 Description: semi_discrete_s Begins: 1.5793s Ends: 1.57936s (step_257_gpu +66.495 µs)
949 950 951	Name compute_te compute_te semi_discret	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu				1.57844s 1.57851s 1.57858s	52.896 μs 49.791 μs 68.447 μs	GPU 0 GPU 0 GPU 0	Stream 13 Stream 13 Stream 13	 Description: semi_discrete_s Begins: 1.5793s Ends: 1.57936s (grid: <<<16384, 	step_257_gpu +66.495 µs) 1, 1>>>
1949 1950 1951 1952	Name Compute_te Compute_te semi_discre set_halo_va	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu				1.57844s 1.57851s 1.57858s 1.57866s	52.896 μs 49.791 μs 68.447 μs 4.608 μs	GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Stream 13 Stream 13 Stream 13 Stream 13	 Description: semi_discrete_s Begins: 1.5793s Ends: 1.57936s (grid: <<<16384, block: <<<128, 1 Launch Type: Rep 	step_257_gpu +66.495 μs) 1, 1>>> , 1>>> gular
949 950 951 952 953	 Name compute_te compute_te semi_discre set_halo_va set_halo_va 	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu alues_x_471_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.57877s	52.896 μs 49.791 μs 68.447 μs 4.608 μs 3.968 μs	GPU 0 GPU 0 GPU 0 GPU 0 GPU 0 GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	 Description: semi_discrete_s Begins: 1.57936s Grid: 1.57936s (grid: <<<16384, block: <<<128, 1 Launch Type: Reg Static Shared Me 	step_257_gpu +66.495 μs) 1,1>>> yular mory: 0 bytes
949 950 951 952 953 954	 Name compute_tt compute_tt semi_discre set_halo_va set_halo_va compute_te 	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.57877s 1.5788s	52.896 μs 49.791 μs 68.447 μs 4.608 μs 3.968 μs 51.456 μs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	 Description: semi_discrete_s Begins: 1.57936s Grid: 1.57936s (grid: <<<16384, block: <<<128, 1 Launch Type: Reg Static Shared Me 	step_257_gpu +66.495 μs) 1, 1>>> 1>>> gular mory: 0 bytes Memory: 0 bytes
949 950 951 952 9953 9954 9955	 Name compute_te compute_te semi_discre set_halo_va set_halo_va compute_te compute_te 	endencies_x_310_gpu endencies_x_343_gpu te_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu endencies_x_343_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.5787% 1.5788s 1.5788s 1.5788s	52.896 μs 49.791 μs 68.447 μs 4.608 μs 3.968 μs 51.456 μs 48.704 μs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	▲ Description: semi_discrete_ Begins: 1.5793s Ends: 1.5793s Ends: 1.57936s grid: <<<16384, block: <<<16384, block: <<<16384, block: <<<16384, block: <<<16384, block: <<<16384, block: Shared Me Dynamic Shared Registers Per Th Local Memory Pe	step_257_gpu +66.495 μs) 1,1>>> gular mory: 0 bytes Memory: 0 bytes Memory: 0 bytes read: 39 r Thread: 0 bytes
949 950 951 952 953 954 955 956	Name compute_te semi_discre set_halo_va set_halo_va compute_te compute_te compute_te compute_te semi_discre	endencies_x_310_gpu endencies_x_343_gpu te_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu endencies_x_343_gpu te_step_257_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.57875 1.57875 1.5788s 1.57875 1.57894s	52.896 μs 49.791 μs 68.447 μs 4.608 μs 3.968 μs 51.456 μs 48.704 μs 68.319 μs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	▲ Description: semi_discrete_t Begins: 1.57936s grid: <<<16384, block: <<<16384, block: <<<128, 1, Launch Type: Re Static Shared Me Dynamic Shared Registers Per Th Local Memory Tc Local Memory Tc	step_257_gpu +66.495 μs) 1, 1>>> jular imory: 0 bytes Memory: 0 bytes read: 39
1949 1950 1951 1952 1953 1954 1955 1956 1957	Name compute_te compute_te semi_discre set_halo_va compute_te compute_te compute_te semi_discre semi_discre semi_discre set_halo_va	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu				1.57844s 1.57851s 1.57858s 1.5786s 1.5786s 1.5787s 1.5787s 1.57837s 1.57894s 1.57894s	52.896 μs 49.791 μs 68.447 μs 4.608 μs 3.968 μs 51.456 μs 48.704 μs 68.319 μs 4.544 μs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	➤ Description: semi_discrete_t Begins: 1.57936s grid: <<<16384, block: <<<128, 1, Launch Type: Re Static Shared Me Dynamic Shared Registers Per Th Local Memory Pe Local Memory Po Shared Memory N Shared Memory N	step_257_gpu +66.495 μs) 1,1>>> jlar mony: 0 bytes Memory: 0 bytes read: 39 rr Thread: 0 bytes sexecuted: 0 bytes ank Size: 4 8
1949 1950 1951 1952 1953 1954 1955 1956 1957 1958	Name compute_te compute_te semi_discre set_halo_va compute_te compute_te compute_te semi_discre semi_discre semi_discre set_halo_va	endencies_x_310_gpu endencies_x_343_gpu te_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu endencies_x_343_gpu te_step_257_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.5786s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.57894s 1.57902s 1.57914s	52.896 µs 49.791 µs 68.447 µs 4.608 µs 3.968 µs 51.456 µs 48.704 µs 68.319 µs 4.544 µs 3.712 µs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	▲ Description: semi_discrete_ Begins: 1.5793s Ends: 1.5793s grid: <<<16384, block: <<<128, 1 Launch Type: Re Static Shared Me Dynamic Shared Registers Per Th Local Memory P Local Memory P Coshared Memory P	step_257_gpu +66.495 μs) 1, 1>>> pular mory: 0 bytes Memory: 0 bytes read: 39 r Thread: 0 bytes tal: 94,371,840 bytes sexecuted: 0 bytes Jank Size: 4 B pancy: 75 %
1949 1950 1951 1952 1953 1954 1955 1956 1957 1958	 Name compute_te semi_discre set_halo_va set_halo_va compute_te compute_te semi_discre set_halo_va set_halo_va 	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu				1.57844s 1.57851s 1.57858s 1.5786s 1.5786s 1.5787s 1.5787s 1.57837s 1.57894s 1.57894s	52.896 μs 49.791 μs 68.447 μs 4.608 μs 3.968 μs 51.456 μs 48.704 μs 68.319 μs 4.544 μs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	▲ Description: semi_discrete_t Begins: 1.57936s grid: <<<16384, block: <<<16384, block: <<<128, 1, Launch Type: Re Static Shared Memory To Shared Memory To Shared Memory To Shared Memory Theoretical occu Launched from ti Latency: <6.048	step_257_gpu +66.495 μs) 1, 1>>> 1, 1>>> jular mory: 0 bytes Memory: 0 bytes read: 39 r Thread: 0 bytes tati: 94,371,840 bytes sexecuted: 0 bytes ank Size: 48 pancy: 75 % pread: 94771 μs
1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959	Name compute_tte semi_discre set_halo_ve set_halo_ve compute_tte compute_tte semi_discre set_halo_ve	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu endencies_x_310_gpu endencies_x_310_gpu ete_step_257_gpu alues_x_440_gpu alues_x_441_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.5786s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.5788s 1.57894s 1.57902s 1.57914s	52.896 µs 49.791 µs 68.447 µs 4.608 µs 3.968 µs 51.456 µs 48.704 µs 68.319 µs 4.544 µs 3.712 µs	GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	▲ Description: semi_discrete_ Begins: 1.57936s grid: <<<128,4 block: <<<128,1 Launch Type: Re Static Shared Memory Pe Local Memory Pe Local Memory Pe Local Memory Pi Shared Memory I Shared I Shared Shared Shared Shared Shared Memory I Shared Memory I Shared I Shared Shared Shared Shared Shared Shared Memory I Shared I Shared Sha	step_257_gpu +66.495 µs) 1, 1>>> 1>>> yular mory: 0 bytes Memory: 0 bytes read: 39 vr Thread: 0 bytes stal: 94,371,840 bytes seacutd: 0 bytes Bank Size: 4 B pancy: 75 % read: 94771 µs 303
Events View 1949 1950 1951 1952 1953 1954 1955 1955 1955 1955 1958 1959 1960 1960	 Name compute_tte semi_discret set_halo_vation set_halo_vation compute_tte compute_tte semi_discret set_halo_vation set	endencies_x_310_gpu endencies_x_343_gpu ete_step_257_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu endencies_x_343_gpu alues_x_440_gpu alues_x_440_gpu alues_x_471_gpu endencies_x_310_gpu				1.57844s 1.57851s 1.57858s 1.57866s 1.5787s 1.57888 1.5788s 1.57884s 1.57894s 1.57902s 1.57914s 1.57916s	52.896 µs 49.791 µs 68.447 µs 4.608 µs 3.968 µs 51.456 µs 48.704 µs 68.319 µs 3.712 µs 51.775 µs	GPU 0 GPU 0	Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13 Stream 13	▲ Description: semi_discrete_t Begins: 1.57936s grid: <<<16384, block: <<<16384, block: <<<128, 1, Launch Type: Re Static Shared Memory To Shared Memory To Shared Memory To Shared Memory Theoretical occu Launched from ti Latency: <6.048	step_257_gpu +66.495 µs) 1, 1>>> 1>>> yular mory: 0 bytes Memory: 0 bytes read: 39 vr Thread: 0 bytes stal: 94,371,840 bytes seacutd: 0 bytes Bank Size: 4 B pancy: 75 % read: 94771 µs 303

MiniWeather - Testing different kernel launch configurations and clauses

Recall the final exercise of the prior MiniWeather session where we experimented with various launch configurations in the miniWeather_mpi_exercise2.F90 source file for specific kernels.

Were you able to achieve any significant speed-up?

The next panels shows statistical results from some launch configuration experiments using parameters _NX=1024 , _NZ=512 , and _SIM_TIME=10 and different clauses in place of *** for the semi_discrete_step subroutine kernel. Note that NUM_VARS=4 .

```
!$acc parallel loop *** async
do ll = 1 , NUM_VARS
    !$acc loop ***
    do k = 1 , nz
        !$acc loop ***
        do i = 1 , nx
            state_out(i,k,ll) = state_init(i,k,ll) + dt * tend(i,k,ll)
        enddo
    enddo
enddo
enddo
```

1. Using worker/vector/seq on each loop respectively, the profiler shows

grid: [1] block: [32x4]. Why is this arrangement the least performant?

MiniWeather Kernel L231, semi_discrete_step	Total Device Time (μs)
BaseLine (on V100) - collapse(3) auto vector_length(128)	62,936
clause - gang/worker/vector on each loop resepctively	852,859
<pre>clause - worker/vector/seq (Move NUM_VARS innermost, seq)</pre>	2,271,059
clause - gang/vector/seq (Move NUM_VARS innermost, seq)	72,584

- 1. Did you find any better configurations for this or other kernels in MiniWeather? Explain why it performed better.
- 2. Do you trust the compiler to make relatively optimal choices with minimal direction?

MiniWeather Kernel L231, semi_discrete_step	Total Device Time (μs)
BaseLine (on V100) - collapse(3) auto vector_length(128)	62,936
<pre>clause - collapse(3) vector_length(32)</pre>	100,797
<pre>clause - collapse(3) vector_length(64)</pre>	63,010
<pre>clause - collapse(3) vector_length(256)</pre>	62,990
<pre>clause - collapse(3) vector_length(512)</pre>	63,032
<pre>clause - collapse(3) vector_length(1024)</pre>	66,458

- 1. For tile(), why do you think the (32,1,NUM_VARS=4) clause was closest to the most performant?
- 2. Can you infer the condition that causes the tile() clause to produce incorrect

results? Hint: What is the max warp size?

MiniWeather Kernel L231, semi_discrete_step	Total Device Time (μs)
BaseLine (on V100) - collapse(3) auto vector_length(128)	62,936
<pre>clause - tile(32,32,NUM_VARS) INCORRECT</pre>	26,992
clause - tile(32,32,1)	73,476
<pre>clause - tile(32,8,NUM_VARS)</pre>	77,124
<pre>clause - tile(32,1,NUM_VARS)</pre>	65,040
clause - tile(1024,1,1)	67,393
<pre>clause - tile(128,1,NUM_VARS)</pre>	66,421
<pre>clause - tile(128,2,NUM_VARS)</pre>	74,295
<pre>clause - tile(128,4,NUM_VARS) INCORRECT</pre>	35,999
clause - tile(*,*,*) -> 32,4,32	150,374

MPAS-Atmosphere Overview



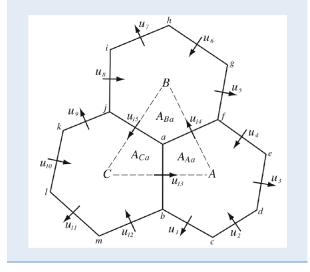
We will now look at a real world production model **MPAS (Model Prediction Across Scales)**, specifically the GPU version of the atmosphere core **MPAS-A** and how this model leveraged OpenACC to refactor towards GPU devices.

So far, **only the v6.x Atmosphere core has been ported to GPUs** and is freely available to review via their website and the stable v6.x or v7.x develop-openacc branches on GitHub. Some work has also been done on the MPAS-Ocean core given this presentation by PhD student Ashwath Venkataraman.

If you'd like a more complete overview of MPAS, how to run the model, and research applications, see the 2021 MPAS Virtual tutorial page or the upcoming 2022 joint WRF/MPAS workshop.

- Fully compressible non-hydrostatic equations written in flux form
- Split-Explicit timestepping via 3rd Order Runge-Kutta, see AMS Paper Klemp, Skamarock, and Dudhia

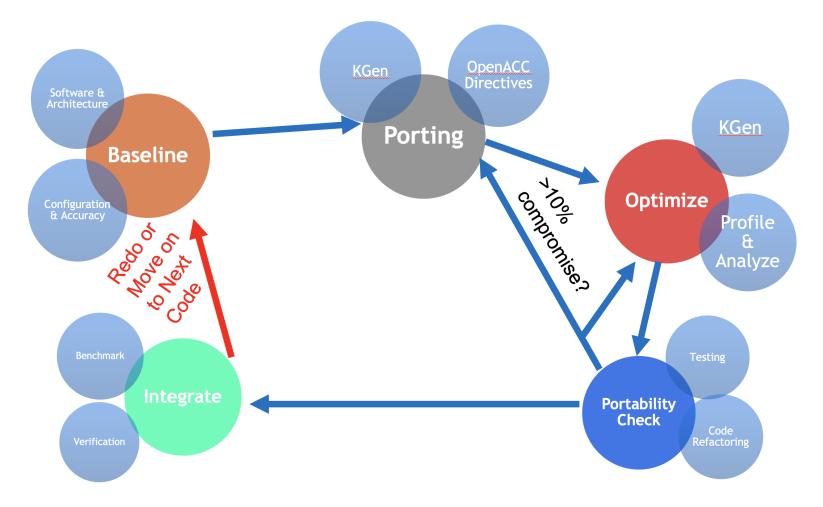
MPAS is based on unstructured centroidal Voronoi (hexagonal) meshes using C-grid staggering and selective grid refinement.



The MPAS-A kernels we will focus on computes coefficients for vertically implicit gravitywave/acoustic computations needed for each Runge-Kutta timestep. The previosly linked paper, specifically section 2 and the appendix, covers this in depth with a broader overview given in the 2021 tutorial Time Integration presentation.

However, understanding the numerical physics at play is not required to port well written code to GPUs.

Development Process of MPAS-A



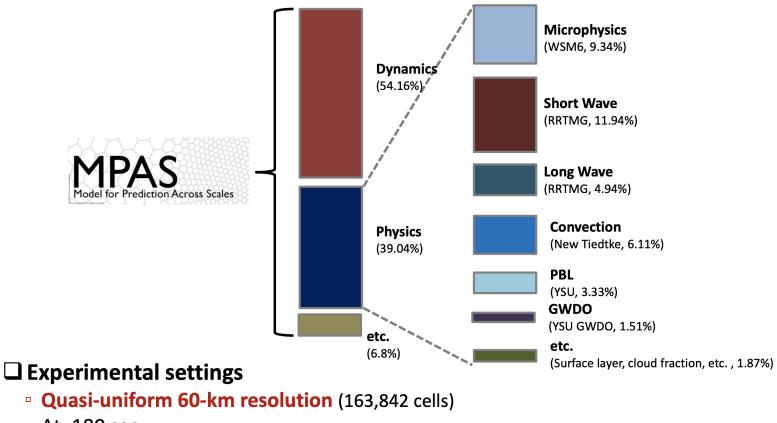
Courtesy of Raghu Raj Kumar, NVIDIA

Identifying an established iterative process for GPU development ahead of work performed significantly eases development cost and increases success outcomes.

- 1. **Establish a baseline**, ensure working and accurate configuration with target hardware and external software.
- 2. **Port the code**, using incremental addition of OpenACC, perhaps using tools for kernel extraction like KGen (Fortran only) to allow separation of concerns.
 - See KGen Guide if interested
- 3. **Optimize computationally expensive kernels** individually via an analysis and profiling iterative process.
- 4. **Check portability expectations** are met and that code satisfies both CPU and GPU unit tests.
 - Look for and eliminate any **GPU anti-patterns** such as linked lists data structures or global memory variables which may cause excessive data movement.
 - Repeat Steps 2-4 as needed.
- 5. **Integrate changes into benchmarks and verification suite**, utilizing version control and ideally a continuous integration process.

MPAS-A Performance Baseline

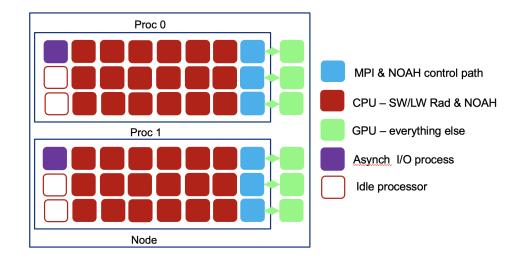
Getting an accurate baseline helps inform where to dedicate development effort. This can be measured using internal timing metrics or your preferred CPU profiler (like TAU, Arm Forge Map, *gprof*, etc), to **identify hotspots** in the code.



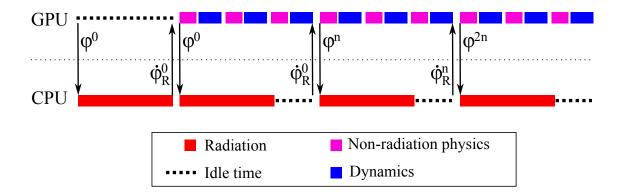
- □ ∆t=180 sec
- 41 vertical layers

From KISTI, Kim, Kang, & Joh GPU Acceleration of MPAS Physics Schemes Using OpenACC

Specific dynamics/physics schemes were prioritized for GPU while some set for CPU.



A lagged computation of radiation was established in order to utilize idle CPUs. Requires **manual tuning of load balancing** between number of CPU and GPU tasks.



Managing GPU Data in MPAS-A

Recall that using **!**\$acc kernels ... and similar directives will generate lists of variables needed to manage data movement for each compute region.

```
153, Generating implicit copyin(rdzu(:)) [if not already present]
Generating implicit copyout(cofwr(:,:)) [if not already present]
Generating implicit copyin(p(:,:)) [if not already present]
Generating implicit copyout(cofwz(:,:)) [if not already present]
Generating implicit copyin(fzp(:),t(:,:),zz(:,:),fzm(:),cqw(:,:)) [if not already present]
Generating implicit copyout(coftz(:,:)) [if not already present]
```

These lists can be used and leveraged for your own data directives as GPU development progresses.

```
do iCell=cellSolveStart,cellSolveEnd
  do i=1,nEdgesOnCell(iCell)
      iEdge = edgesOnCell(i,iCell)
      !DIR$ IVDEP
      do k = 2, nVertLevels
         flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
         w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
      end do
  end do
   !DIR$ IVDEP
                                                                                 !$acc data copy(w_tend, &
  do k = 2, nVertLevels
                                                                                 !$acc edgesoncell, edgesoncell_sign, fzm, fzp,nedgesoncell, u_tend, &
      w_\text{tend}(k, \text{iCell}) = (fzm(k) * zz(k, \text{iCell}) + fzp(k) * zz(k-1, \text{iCell}))
                                                                                 !$acc zb3_cell, zb_cell, zz)
  end do
                                                                                 !$acc kernel
end do
                                                                                do iCell=cellSolveStart,cellSolveEnd
                                                                                   do i=1,nEdgesOnCell(iCell)
                                                                                       iEdge = edgesOnCell(i,iCell)
                                                                                       !DIR$ IVDEP
                                                                                       do k = 2, nVertLevels
                                                                                          flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
                                                                                          w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
  !$acc data present(w_tend, &
                                                                                       end do
  !$acc edgesoncell, edgesoncell_sign, fzm, fzp,nedgesoncell, u_tend, &
                                                                                    end do
  !$acc zb3_cell, zb_cell, zz)
  !$acc parallel num_workers(8) vector_length(32)
                                                                                    !DIR$ IVDEP
  !$acc loop gang worker private(iEdge, flux)
                                                                                    do k = 2, nVertLevels
  do iCell=cellSolveStart,cellSolveEnd
                                                                                       w_tend(k,iCell) = (fzm(k) * zz(k,iCell) + fzp(k) * zz(k-1,iCell))
    do i=1,nEdgesOnCell(iCell)
                                                                                    end do
    iEdge = edgesOnCell(i,iCell)
                                                                                 end do
    !DIR$ IVDEP
                                                                                 !$acc end kernel
       do k = 2, nVertLevels
                                                                                !$acc end data
           flux = edgesOnCell_sign(i,iCell) * fzm(k) * u_tend(k,iEdge)
           w_tend(k,iCell) = w_tend(k,iCell) - zb_cell(k,i,iCell)
        end do
    end do
  !DIR$ IVDEP
    do k = 2, nVertLevels
       w_{tend}(k, iCell) = (fzm(k) * zz(k, iCell) + fzp(k) * zz(k-1, iCell))
    end do
  end do
```

```
!$acc end parallel
!$acc end data
```

Given ported kernels, MPAS-A was designed to create CPU and GPU data copies at initialization via !\$acc declare create(...) and copy data at unstructured data regions via !\$acc enter data copyin(...) prior to each kernel call. Then, each kernel would only require a present(...) clause using the prior variable lists. **Reference counters** would mitigate excessive copies.



Any lingering excessive data copies could be identified by profilers and fixed while other required copies for CPU algorithms & I/O were managed by **!**\$acc update directives.

MPAS-A Kernel Extraction

We will focus on the attracted by Supreeth Suresh, TDD/ASAP in CISL. This is the link, Line 2641 to the source subroutine in the full model codebase and in this workshop directory is the the extracted set of kernels mpas_atm_compute_vert_imp_coefs_work.F90.

Assuming data locality is resolved, this extracted kernel simply utilizes **randomized input data** as we will be **focusing on optimizing the performance** of the subroutine's kernels. The kernel is run in a repeating loop so we can get a relatively consistent average of measured performance. A validation tool has not been included at this time but is typically highly recommended.

For large codebases, building and/or using an automated tool like NCAR's KGen for Fortran codes or Kernel Tuner from NL eScience Center for CUDA/OpenCL codes will likely speed up the development/optimization process.

EXERCISE: MPAS-A Kernel Optimization

Open the mpas_atm_compute_vert_imp_coefs_work.F90 source file and convert the !\$acc kernels loops to optimized !\$acc parallel ... compute constructs. Analyze each set of loops and apply appropriate sets of kernel configuraton clauses to achieve improved performance. Note: !DIR\$ IVDEP tells compiler to ignore loop dependencies for serial vector SIMD compilations.

You are encouraged to reference the initial attempts at optimization done by the !\$acc kernels directive output during the compilation process. Data management has already been done for you using _gpu=managed and present(var-list)/create(varlist) clauses.

Record results of your optimization experiments on a chosen kernel and try to determine optimal configurations for that kernel. Compare your achieved performance with the original at Line 2641. Work on other kernels as time allows. Note that most kernels may benefit from similar clause specifications since they operate on similar domain sizes/variables.

In []:

module load nvhpc/22.2 &> /dev/null
export _OPENACC=true
make

In []:

module load nvhpc/22.2 &> /dev/null
export _OPENACC=true
make

In []:

qcmd -A \$PROJECT -q \$QUEUE -1 select=1:ncpus=1:ngpus=1 -1 gpu_type=\$GPU_TYPE -1 walltime=20 -v NVCOMPILER_ACC_TIME=1 -- \
`pwd`/vert_implicit_coefs.exe

In []:

module load nvhpc/22.2 &> /dev/null
export _OPENACC=true
make

In []:

qcmd -A \$PROJECT -q \$QUEUE -1 select=1:ncpus=1:ngpus=1 -1 gpu_type=\$GPU_TYPE -1 walltime=20 -v NVCOMPILER_ACC_TIME=1 -- \
`pwd`/vert_implicit_coefs.exe

MPAS-A Kernels L###	Device Time (μs)
BaseLine (on V100) - !\$acc kernels	XX
clause - gang/vector	XX
<pre>clause - tile(##,##)</pre>	XX
<pre>clause - tile(*,*)</pre>	XX
clause - vector_length(XX)	XX
clause - num_workers(XX)	XX
	XX

Final Points

- 1. Plan for and **commit to a defined iterative GPU development process** to remove pain points and manage long term goals of your code project
 - Smaller, validated incremental changes are easier to debug
- 2. Start with descriptive !\$acc kernels then add prescriptive !\$acc

parallel ... kernels as needed for expensive kernels

- !\$acc kernels can still achieve meaningful performance alone
- 3. Understand that the GPU development process takes time and effort but **specific tools/techniques can drastically speed up development time**.

Suggested Resources

- 2021 MPAS Virtual tutorial
- Computers & Geosciences, GPU acceleration of MPAS microphysics WSM6 using OpenACC directives: Performance and verification by J. Kim, J. Kang, and M. Joh (KISTI)
- OpenACC.org and NVIDIA managed GitHub, presentations, and learning materials GPU Bootcamps
 - Lab sequence on OpenACC
 - Lab sequence on Profiling Tools with MiniWeather
 - Lab sequence on Various GPU Programming Paradigms (CUDA, OpenACC, stdPar, OpenMP)
 - Lab sequence on Multi-GPU Programming
 - Lab sequences on GPU AI with CFD, PINNs, and Climate models

After this session, we will have three weeks until the next workshop. Order of upcoming sessions will also be adjusted to accomodate availability of a NVIDIA engineer to present on Multi-GPU programming. Look out for upcoming announcements.