Accelerating FUN3D v14.0 Solutions Using GPU Hardware

Aaron Walden, Eric Nielsen, and Gabriel Nastac

Computational AeroSciences Branch

NASA Langley Research Center

Public Community Questions: fun3d-users@lists.nasa.gov
Private/Proprietary Questions: fun3d-support@lists.nasa.gov

March 23, 2023

Background

- FUN3D GPU implementation (FLUDA) over a decade in the making
  - Thanks to everyone who made that possible!
- Breakthrough performance when V100 debuted in 2017
- Conducted simulations of unprecedented spatial and temporal fidelity on the world's largest supercomputer at the time (OLCF Summit, see Ref. [1])
- Originally written in CUDA C++
  - Now runs on CPUs and NVIDIA, AMD, and Intel GPUs

Performance

- FUN3D is memory bound, so performance scales roughly with main memory bandwidth
- Typically see > 4x improved performance comparing one GPU to a dual-socket CPU node of the same generation
  - Compare V100 vs. Skylake and A100 vs. EPYC (4.65x and 4.04x, resp.)
- See below (case is 3.7M CRM SA-QCR2000 EBV from Ref. [2])
  - Speedups given are relative to Skylake 6148

Architecture	Speedup	MBW Ratio	MBW [GB/s]
Intel Skylake 6148 (2 sockets, 40 cores)	1.00	1.00	256
AMD EPYC 7742 (2 sockets, 128 cores)	1.93	1.60	410
NVIDIA 16 GB SXM V100	4.65	3.52	900
NVIDIA 40 GB SXM A100	7.80	6.05	1555

Performance Pitfalls

- Insufficient vertices/GPU – aim for at least 500k / V100 (fewer will work fine)
- Some operations, such as preprocessing and I/O, occur on the CPU
  - Short runs – may be dominated by CPU preprocessor
    - Can be mitigated by dumping partition files (`write/read_partitions=.true. in &partitioning`)
  - Frequent output – slicing/sampling may be slow on CPU
    - Can be mitigated by postprocessing native volume output (but be warned, it can be a lot of data)
  - Mitigate the above by running more MPI ranks / GPU (MPS, detailed later)
- Atypical machine configurations
  - FUN3D by default assigns GPUs to MPI ranks in ascending order
  - If GPU 0 is not physically near CPU 0, performance may degrade
  - Several ways to deal with atypical configs (contact us with specifics)
  - `use_auto_device_select=.false.` in `&gpu_support` is the first step
    - Allows you to bind GPUs manually; renumber with CUDA_VISIBLE_DEVICES
- **Contact us (see title slide) if performance is not as expected**

Implementation

- Original implementation is a straight port of FUN3D's Fortran compute kernels to CUDA C++
  - Retains the same precision; it is computing exactly the same thing as the CPU
- FUN3D retains its Fortran driver; calls CUDA kernels instead of Fortran kernels when running in GPU mode
- All nontrivial computation during time-stepping is done on the GPU
  - Data movement between CPU and GPU is minimal
- The CUDA kernels have recently been abstracted so that the code retains a CUDA-like structure but can be run on CPUs and NVIDIA, AMD, and Intel GPUs
- However, source is not (yet) distributed and the v14 binaries are only available for NVIDIA
- Wrinkle: v14 binaries were miscompiled and so they will only run on V100 and A100
  - Normally, can forward-compile for newer GPUs (H100)
- Will be fixed in next minor release; contact us if you need an H100 binary

Supported FUN3D Options

- The options listed here are generalized for brevity, for a complete listing of capabilities, **please refer to the FUN3D manual**
  - **https://fun3d.larc.nasa.gov/papers/FUN3D_INTG_Manual-14.0.pdf#subsection.10.4**
- Perfect gas, thermochemical nonequilibrium, and incompressible modes are supported
  - SA and SST turbulence models (various flavors)
- Specified rigid grid motion supported
- Aeroelastic analysis using internal modal solver with modal mesh deformation supported
- The code should die when an unsupported GPU options is specified
- **Caveat: no `twod` mode (and not planned) and code doesn't properly die until next minor release**
- Contact us to suggest options for GPU mode

Helpful Options

- **`ebv`** – `&code_run_control` option that uses an edge-based viscous (EBV) method for computation of viscous terms (see Ref. [3])
  - Still considered experimental
  - Greatly speeds up viscous computations (a major bottleneck)
- **`speedup_mpi_override`** - `&gpu_support` option required for GPU to respect stop.dat
  - Removed next minor release, GPU will behave as CPU
- **`time_timestep_loop`** - `&global` option that outputs iteration times
  - Useful for observing GPU performance
- **`print_crude_dev_mem`** - `&gpu_support` option that displays device memory status as memory allocation occurs
  - Helpful for diagnosing various device issues such as running out of memory
- **`estimate_remaining_time`** - self-explanatory new `&global` option

Advanced GPU Options

- **`cuda_start_mps`** - `&gpu_support` option that automates the requirements to run multiple MPI ranks / GPU (for NVIDIA)
- **`use_cuda_mpi`** - `&gpu_support` option that uses device data when making MPI calls, may be faster but requires a very specific software stack
- **`use_half_precision`** - experimental `&gpu_support` option that uses 16-bit floating point to store portions of program data (while retaining 64-bit accuracy, see Ref. [4]), may speed up entire simulations as much as 20%
    - Use at your own risk (and tell us what happens)
    - Truncations may affect stability of solution algorithm
    - Not recommended for flows with chemistry

System Requirements

- Version 10.2 or higher of the NVIDIA CUDA Toolkit must be installed
- Distributed FLUDA binaries will only execute on NVIDIA GPUs (for v14, only V100 and A100)
    - Other binaries (e.g., AMD) may be requested
- NVIDIA has Tesla (HPC-grade) and consumer-grade GPUs
    - Though FUN3D may execute on consumer GPUs, they are not endorsed
        - Many lack adequate double-precision support and will execute at 1/4 the expected speed **or worse**
    - Tesla GPUs earlier than P100 will perform poorly due to lack of hardware support for atomic operations; **do not use them**
- Requirements are otherwise the same as the CPU code

General Setup Guidance

- Build your copy of FUN3D with GPU support
    - See Appendix A of User Manual; configure with:

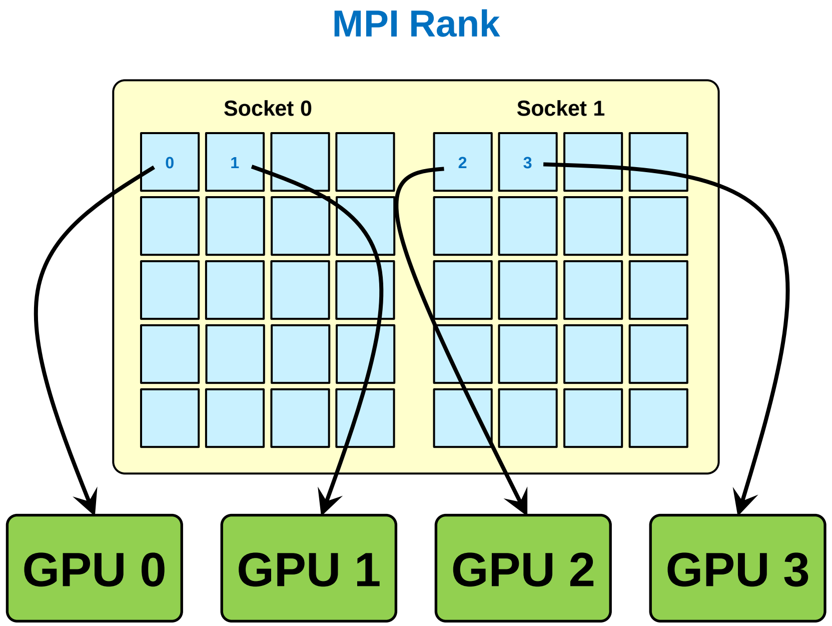
```
--with-cuda=/path/to/CUDA              Path to CUDA installation
--with-libfluda=/path/to/FLUDA         Path to fluda_binaries/nvidia/x86_64/[architecture] in the fun3d tarball
```

- For perfect gas flows, need apx. 1 GB of GPU memory per 200k vertices
- Herein we assume use of homogeneous nodes with 4 GPUs each
- See your system documentation for guidance on how to submit jobs to GPU-enabled resources
- Please contact us for assistance; we are happy to help

Node-Level Partitioning, MPI, and GPUs

- When using GPUs to accelerate FUN3D, the simplest and most efficient strategy is to assign a single MPI rank to each GPU, with the ranks spread out evenly over the sockets
  - Recall that no time-stepping kernels execute on the host when FUN3D is using GPU acceleration, so these CPU cores are solely used to launch CUDA kernels: they serve only to direct traffic
  - Note that most CPU cores sit idle in this paradigm: if host-based kernels such as preprocessing and visualization support are major contributors to the workflow, this arrangement may yield poor performance for such kernels

Consider a dual-socket Xeon Skylake with 20 cores/socket and a 4xV100 arrangement:

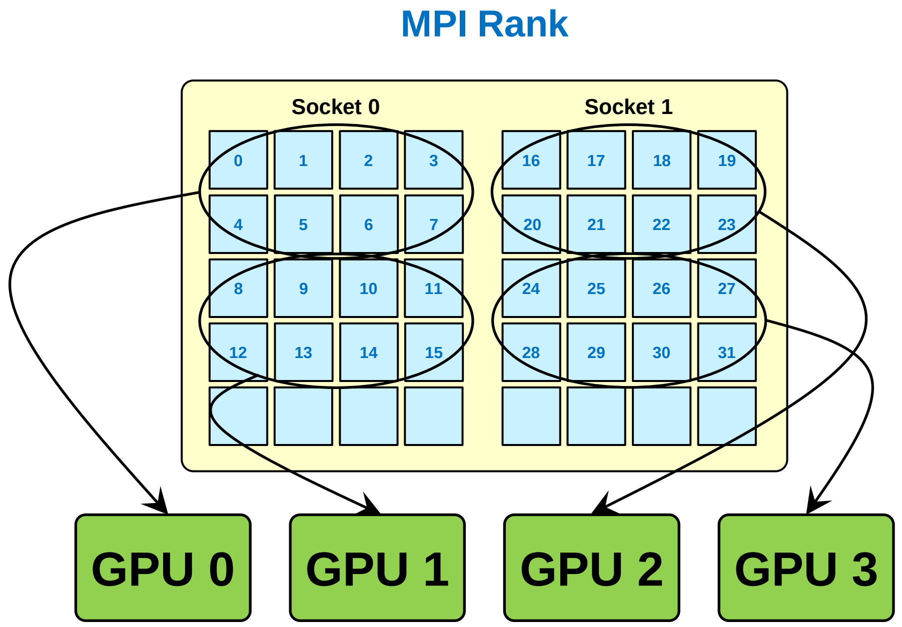


Langley Research Center

Running with Multiple MPI Ranks per GPU

- If host-based kernels such as preprocessing and visualization support are critical to overall performance, we can instantiate more MPI ranks to distribute over the CPU cores
- In this case, launch the MPI job with an integer multiple of the number of GPUs present, and FUN3D will assign multiple MPI ranks to each GPU
- FUN3D will require the use of NVIDIA's Multi-Process Service (MPS), which allows multiple CUDA kernels to be processed simultaneously on the same GPU
- Host-based kernels will now scale accordingly, while GPU performance should not degrade more than 5-10% when using up to 8 MPI ranks per GPU

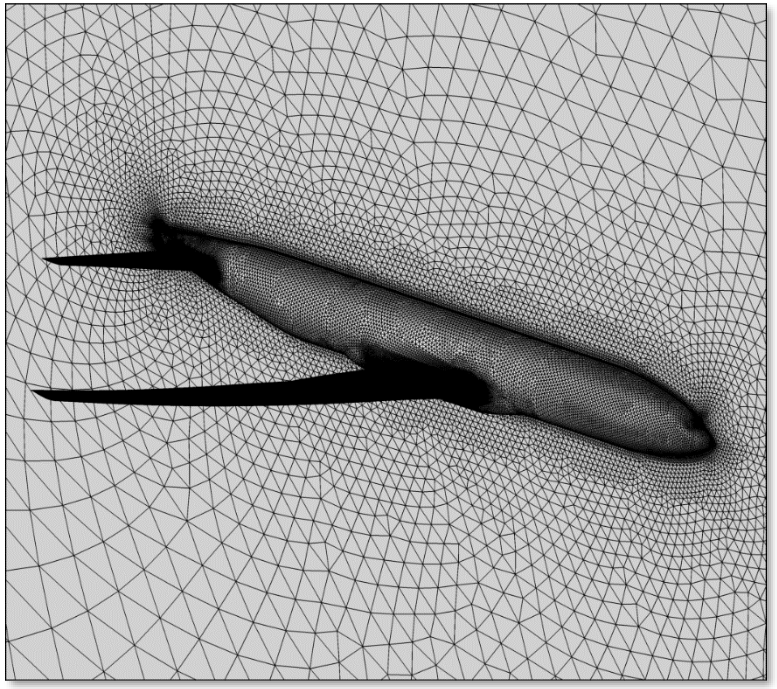
Consider a dual-socket Xeon Skylake with 20 cores/socket and a 4xV100 arrangement:



Langley Research Center

Demo Cases

- Three grids chosen from the 4th AIAA Drag Prediction workshop, based solely on grid size
- Original grids were tetrahedral; merged into mixed elements for this exercise
- Mach 0.85, Re=5M, AOA=1 deg; Spalart-Allmaras turbulence model
- Each case is run for 500 time steps
- Run on NAS sky_gpu GPU 4xV100 nodes



	Grid 1: "1M"	Grid 2: "6M"	Grid 3: "10M"
Points	1,233,948	5,937,410	10,252,669
Tetrahedra	983,281	7,815,201	14,836,294
Pyramids	22,866	71,789	89,642
Prisms	2,068,172	9,006,159	15,154,594

Langley Research Center

Running on a Single GPU

- Get the grid and fun3d.nml: `wget https://fun3d.larc.nasa.gov/FUN3D_v14.0_session5_1M.tgz`
- Here, we are using a single CPU core as a shepherd for a single GPU; **all other CPU cores sit idle**

fun3d.nml

```
&project
  project_rootname = 'dpw-wb0_med-7Mc_5.merged'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&reference_physical_properties
  angle_of_attack    = 1.0
  mach_number        = 0.85
  reynolds_number    = 18129.1
  temperature        = 560.0
  temperature_units = 'Rankine'
/
&force_moment_integ_properties
  area_reference   = 594720.0
/
&nonlinear_solver_parameters
  schedule_cfl       = 10.0 200.0
  schedule_cflturb =  1.0   30.0
/
&code_run_control
  steps               = 500
  restart_read        = 'off'
/
&gpu_support
  use_cuda = .true.
/
```

PBS Script

```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -l select=1:ncpus=36:mpiprocs=36:model=sky_gpu:ngpus=4
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4

module use /swbuild/fun3d/fun3dv14_users/modulefiles                # NASA ONLY
module purge                                                         # NASA ONLY
module load FUN3D_INTG_AVX512                                        # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                        # NASA ONLY

((mpiexec_mpt -perhost 1 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

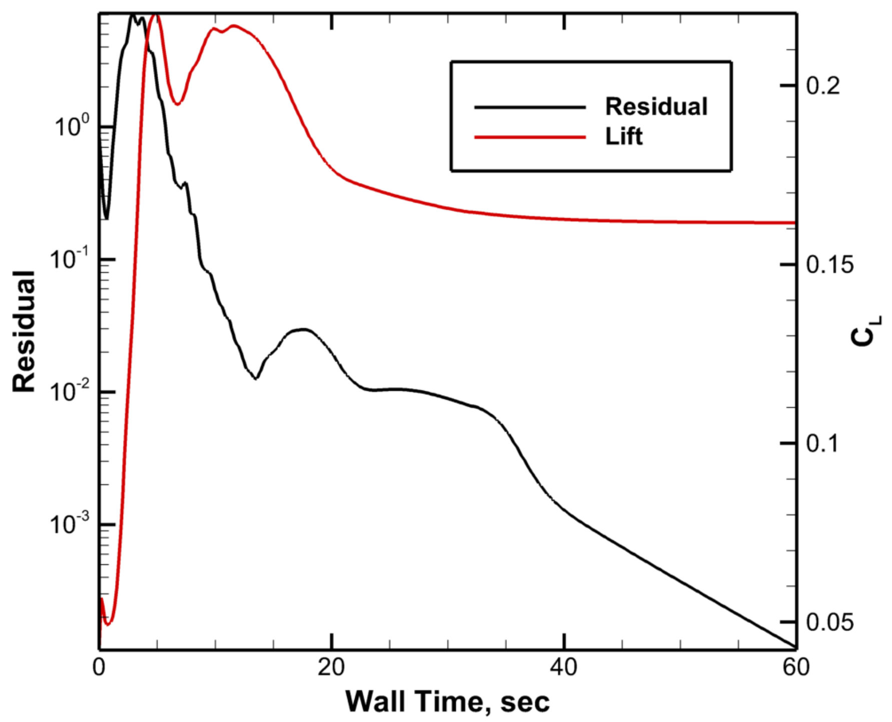
Langley Research Center

Running on a Single GPU

Screen Output

```
FUN3D 14.0-d03712b Flow started 03/22/2023 at 07:30:28 with 1 processes
Contents of fun3d.nml file below------------------------
&project
  project_rootname = 'dpw-wb0_med-7Mc_5.merged'
/
.
.
.
WARNING: CUDA MPS NOT running on r101i0n3.
CUDA MPS status is good: either not needed or running properly on all 1 nodes.
.
.
.
   52  0.216158829571317E+00  0.46052E+02  0.40292E+04  0.41325E+04  0.57527E+04
       0.142220594351411E+01  0.25588E+03  0.98227E+04  0.31565E+04  0.13871E+04
       Lift  0.204387766282627E+00         Drag  0.171265575314454E-01
.16329447 seconds to complete timestep on the master rank.
   53  0.221866818685588E+00  0.47732E+02  0.40292E+04  0.41325E+04  0.57527E+04
       0.146033086295375E+01  0.25335E+03  0.98227E+04  0.31565E+04  0.13871E+04
       Lift  0.205038956434613E+00         Drag  0.168285531266851E-01
.10379885 seconds to complete timestep on the master rank.
   54  0.208703702628507E+00  0.49355E+02  0.72370E+04  0.51382E+04 -0.26047E+04
       0.150686999090736E+01  0.25100E+03  0.98227E+04  0.31565E+04  0.13871E+04
       Lift  0.205646859090078E+00         Drag  0.165874305015577E-01
.16937434 seconds to complete timestep on the master rank.
.
.
.
61.951 seconds to complete main timestep loop on the master rank.
 Done.
```

- Running with a single MPI rank
- MPS is not running (and is not needed; more later)
- Nominal time step costs 0.16 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.10 seconds



Running on a Single Node with 4 GPUs

- Get the grid and fun3d.nml: `wget https://fun3d.larc.nasa.gov/FUN3D_v14.0_session5_6M.tgz`
- Here, we are using four CPU cores as shepherds for four GPUs; **all other CPU cores sit idle**

fun3d.nml

```
&project
  project_rootname = 'dpw_wbt0_fine-35Mc_5.merged'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&reference_physical_properties
  angle_of_attack    = 1.0
  mach_number        = 0.85
  reynolds_number    = 18129.1
  temperature        = 560.0
  temperature_units = 'Rankine'
/
&force_moment_integ_properties
  area_reference   = 594720.0
/
&nonlinear_solver_parameters
  schedule_cfl      = 10.0 200.0
  schedule_cflturb =  1.0   30.0
/
&code_run_control
  steps               = 500
  restart_read        = 'off'
/
&gpu_support
  use_cuda = .true.
/
```

PBS Script

```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -l select=1:ncpus=36:mpiprocs=36:model=sky_gpu:ngpus=4
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4

module use /swbuild/fun3d/fun3dv14_users/modulefiles             # NASA ONLY
module purge                                                     # NASA ONLY
module load FUN3D_INTG_AVX512                                    # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                    # NASA ONLY

((mpiexec_mpt -perhost 4 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

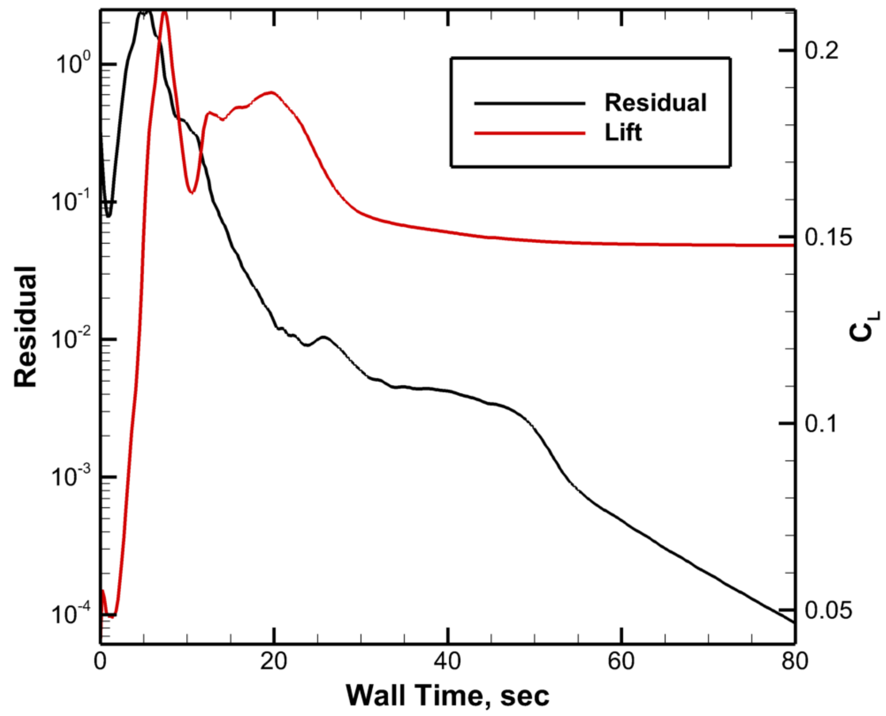
Langley Research Center

Running on a Single Node with 4 GPUs

Screen Output

```
FUN3D 14.0-d03712b Flow started 03/22/2023 at 07:38:36 with 4 processes
Contents of fun3d.nml file below------------------------
&project
  project_rootname = 'dpw_wbt0_fine-35Mc_5.merged'
/
.
.
.
WARNING: CUDA MPS NOT running on r101i0n3.
CUDA MPS status is good: either not needed or running properly on all 1 nodes.
.
.
.
   62  0.951085003205109E-01  0.43358E+02  0.10865E+05  0.20825E+04  0.87569E+03
       0.856370589209556E+00  0.33632E+03  0.10865E+05  0.20825E+04  0.87569E+03
       Lift  0.182665507664676E+00        Drag  0.150035784519962E-01
.21863349 seconds to complete timestep on the master rank.
   63  0.905743990845053E-01  0.41046E+02  0.10795E+05  0.41044E+04  0.34631E+04
       0.852604223912869E+00  0.32636E+03  0.10865E+05  0.20825E+04  0.87569E+03
       Lift  0.182568468979598E+00        Drag  0.151284763517011E-01
.14476674 seconds to complete timestep on the master rank.
   64  0.851303952100990E-01  0.39652E+02  0.10795E+05  0.41044E+04  0.34631E+04
       0.847837283747530E+00  0.31391E+03  0.10865E+05  0.20825E+04  0.87569E+03
       Lift  0.182203611038651E+00        Drag  0.152642508523030E-01
.21870519 seconds to complete timestep on the master rank.
.
.
.
87.605 seconds to complete main timestep loop on the master rank.
 Done.
```

- Running with four MPI ranks
- MPS is not running (and is not needed; more later)
- Nominal time step costs 0.22 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.14 seconds



Langley Research Center

Running on Two Nodes with 4 GPUs Each

- Get the grid and fun3d.nml:
  `wget https://fun3d.larc.nasa.gov/FUN3D_v14.0_session5_10M.tgz`
- Here, we are using four CPU cores as shepherds for four GPUs on each of two nodes; **all other CPU cores sit idle**

fun3d.nml

```
&project
  project_rootname = 'dpw_wbt0_fine-35Mc_5.merged'
/
&raw_grid
  grid_format = 'aflr3'
  data_format = 'stream'
/
&reference_physical_properties
  angle_of_attack   = 1.0
  mach_number       = 0.85
  reynolds_number   = 18129.1
  temperature       = 560.0
  temperature_units = 'Rankine'
/
&force_moment_integ_properties
  area_reference   = 594720.0
/
&nonlinear_solver_parameters
  schedule_cfl     = 10.0 200.0
  schedule_cflturb =  1.0  30.0
/
&code_run_control
  steps               = 500
  restart_read        = 'off'
/
&gpu_support
  use_cuda = .true.
/
```

PBS Script

```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -l select=2:ncpus=36:mpiprocs=4:model=sky_gpu:ngpus=4
#PBS -l place=scatter:excl
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4

module use /swbuild/fun3d/fun3dv14_users/modulefiles                # NASA ONLY
module purge                                                        # NASA ONLY
module load FUN3D_INTG_AVX512                                       # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                       # NASA ONLY

((mpiexec_mpt -perhost 4 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

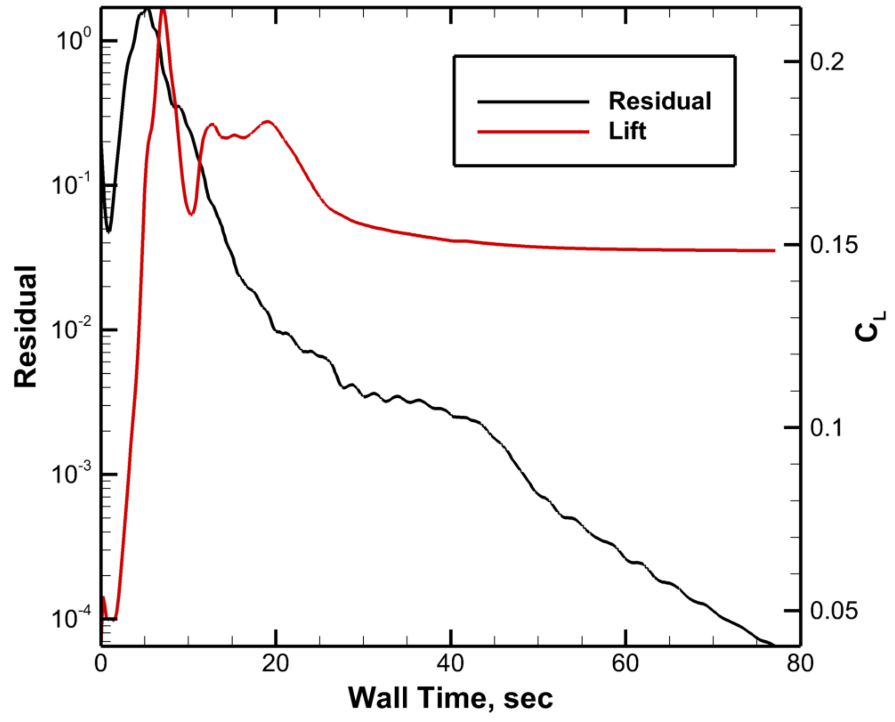
Langley Research Center

Running on Two Nodes with 4 GPUs Each

Screen Output

```
FUN3D 14.0-d03712b Flow started 03/22/2023 at 08:26:56 with 8 processes
Contents of fun3d.nml file below------------------------
&project
  project_rootname = 'dpw-wbt0_med-10Mn_5.merged'
/
.
.
.
WARNING: CUDA MPS NOT running on r101i0n10.
WARNING: CUDA MPS NOT running on r101i0n11.
CUDA MPS status is good: either not needed or running properly on all 2 nodes.
.
.
.
   60  0.132018484698159E+00  0.14594E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.655829947965633E+00  0.51355E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.176613064418888E+00         Drag  0.153978023947141E-01
.20061884 seconds to complete timestep on the master rank.
   61  0.120967899839322E+00  0.11989E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.633808927154000E+00  0.47335E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.178788035714884E+00         Drag  0.148029028171359E-01
.13077858 seconds to complete timestep on the master rank.
   62  0.108924998160599E+00  0.88673E+02  0.16541E+05  0.00000E+00 -0.53707E+02
       0.628431902834406E+00  0.45012E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.180312537215482E+00         Drag  0.143854284486610E-01
.19415972 seconds to complete timestep on the master rank.
.
.
.
79.732 seconds to complete main timestep loop on the master rank.
 Done.
```

- Running with eight MPI ranks
- MPS is not running (and is not needed; more later)
- Nominal time step costs 0.20 seconds
    - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.13 seconds



Langley Research Center

Running Multiple MPI Ranks per GPU

- Recall we have only used a very small number of MPI ranks per CPU so far
  - This severely hampers the performance of CPU kernels such as preprocessing and visualization
- To mitigate these bottlenecks, we may run a larger number of MPI ranks, with multiple ranks sharing a GPU
  - Choose an integer multiple of the number of GPUs available
- To facilitate efficient sharing of each GPU, use the NVIDIA Multi-Process Service (MPS)
  - You may start this daemon yourself, or have FUN3D do it internally
- Here, we are using 32 CPU cores as shepherds for four GPUs (8 MPI ranks each) on each of two nodes; **all other CPU cores sit idle**

fun3d.nml

```
.
.
.
&gpu_support
  use_cuda         = .true.
  cuda_start_mps   = .true.
/
```

PBS Script

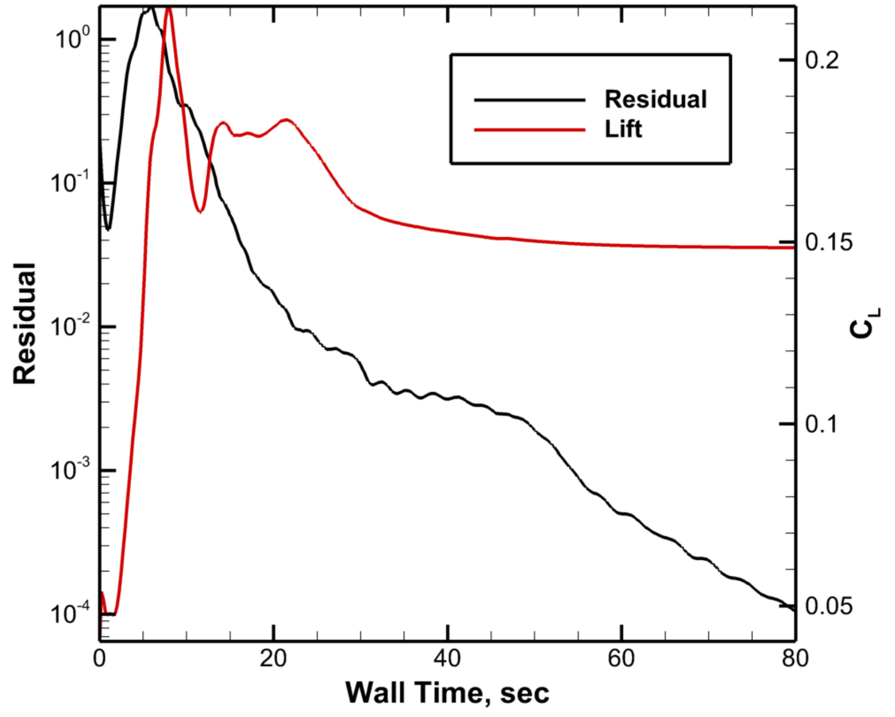
```
#PBS -S /bin/csh
#PBS -N run_test
#PBS -r n
#PBS -l select=2:ncpus=36:mpiprocs=32:model=sky_gpu:ngpus=4
#PBS -l place=scatter:excl
#PBS -l walltime=0:10:00
#PBS -q v100@pbspl4

module use /swbuild/fun3d/fun3dv14_users/modulefiles              # NASA ONLY
module purge                                                      # NASA ONLY
module load FUN3D_INTG_AVX512                                     # NASA ONLY

unsetenv CUDA_VISIBLE_DEVICES                                     # NASA ONLY

((mpiexec_mpt -perhost 32 nodet_mpi --time_timestep_loop ) > test.out) >& error.out
```

Running Multiple MPI Ranks per GPU

Screen Output

```
FUN3D 14.0-d03712b Flow started 03/22/2023 at 08:30:06 with 64 processes
Contents of fun3d.nml file below------------------------
&project
  project_rootname = 'dpw-wbt0_med-10Mn_5.merged'
/
.
.
.
CUDA MPS status is good: either not needed or running properly on all 2 nodes.
.
.
.
   60  0.131023606510773E+00  0.14420E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.657988678847277E+00  0.52227E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.176608990442695E+00         Drag  0.154200246390409E-01
.22486705 seconds to complete timestep on the master rank.
   61  0.120097956449676E+00  0.11736E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       0.636071451756599E+00  0.48195E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.178735835225606E+00         Drag  0.148265470180802E-01
.14993023 seconds to complete timestep on the master rank.
   62  0.108185586213013E+00  0.85668E+02  0.16541E+05  0.00000E+00 -0.53707E+02
       0.630569363594978E+00  0.45850E+03  0.16541E+05  0.00000E+00 -0.53707E+02
       Lift  0.180267154538958E+00         Drag  0.144054614883000E-01
.21819975 seconds to complete timestep on the master rank.
.
.
.
88.757 seconds to complete main timestep loop on the master rank.
Done.
```

- Running with 64 MPI ranks
- MPS is now running on all nodes
- Nominal time step costs 0.22 seconds
  - As we converge, Jacobian evaluations are frequently skipped, reducing per-step costs to 0.15 seconds



General Tips and Guidance

- For many more tips / troubleshooting advice, see the GPU chapter of the FUN3D user manual and/or contact us
- FUN3D GPU may be run on government-approved cloud services
- You may find that FUN3D does not function correctly at first on newly-installed GPU systems
  - We have tried to anticipate a broad range of issues we have encountered before, but please be patient: there can be many details beyond a CPU-only system
  - System administrators are sometimes unfamiliar with subtle details of GPU computing and may have set up the system in an unexpected configuration
  - Please contact us for assistance
  - If we cannot help you identify/solve a problem, NVIDIA is offering tech support to the broader FUN3D community – we can connect you with the appropriate NVIDIA POC

Public Community Questions: fun3d-users@lists.nasa.gov
Private/Proprietary Questions: fun3d-support@lists.nasa.gov

NAS-Specific Guidance

- NAS provides 48 nodes of 4xV100 GPUs (and two 8xV100 nodes)
- For V100 nodes, load the following prebuilt v14 module:

```
module use /swbuild/fun3d/fun3dv14_users/modulefiles
module load FUN3D_INTG_AVX512
```

- For future A100 nodes, load the following prebuilt v14 module (V100 module will work, but run slightly slower):

```
module load FUN3D_INTG_Rome
```

- Here we assume use of entire nodes of 4 GPUs; see online NAS documentation for requesting partial nodes
- GPU jobs should be submitted to the **v100** queue; it is accessed via the PBS server pbspl4 using one of the following methods:
  - Use `#PBS -q v100@pbspl4` in your PBS script
  - Use `–q v100@pbspl4` in your qsub command
  - Log into pbspl4 and submit your job there
- Unlike most other GPU systems, your script must contain the line

```
unset CUDA_VISIBLE_DEVICES    # bash
unsetenv CUDA_VISIBLE_DEVICES # csh
```

- For current guidance on running FUN3D on NAS GPUs, enter the command:

```
module help /path/to/your/FUN3D/module
```

- For more details, see the online NAS documentation

References

1. Mars Retropropulsion campaign

- Paper: https://ntrs.nasa.gov/citations/20210024958
- Presentation: https://ntrs.nasa.gov/citations/20220002950

2. Multi-architecture FLUDA details

- Paper: https://ntrs.nasa.gov/citations/20220016937
- Presentation: https://ntrs.nasa.gov/citations/20220017092

3. Edge-based viscous method

- Paper: https://ntrs.nasa.gov/citations/20220005528
- Presentation: https://ntrs.nasa.gov/citations/20220006376

4. Mixed precision solver

- Paper: https://fun3d.larc.nasa.gov/papers/LowPrecisionSolver.pdf

Backup