ACCELERATING ANSYS FLUENT 15.0 USING NVIDIA GPUS

DA-07311-001_v01 | June 2014

Application Note

DOCUMENT CHANGE HISTORY

DA-07311-001_v01

Version	Date	Authors	Description of Change
01	June 16,2014	VS/CC	Initial release

TABLE OF CONTENTS

LIST OF FIGURES

ACCELERATING ANSYS FLUENT USING NVIDIA GPUS

1. INTRODUCTION

ANSYS® Fluent® 15.0 users can now speed up their computational fluid dynamics simulations using NVIDIA's general purpose graphics processing units (GPGPUs) alongside CPUs. The purpose of this guide is to help Fluent Users make informed decisions about how to -

- Activate the GPU feature for Fluent jobs
- Choose appropriate linear system solver configuration settings for the job and their influence on convergence (residuals), performance (total time) and memory requirements on the GPU
- Evaluate memory requirements and number of GPUs required for the job
- Evaluate GPU performance

2. ACTIVATING THE GPU FEATURE

When running ANSYS Fluent 15.0 interactively, the Parallel Settings tab in the Fluent Launcher panel as shown in Figure 1 allows you to specify settings for running ANSYS Fluent in parallel. This tab is only available if you have selected Parallel under Processing Options. In this panel, you can specify the number of CPU processes using the "Processes" field and specify the number of GPUs using the "GPGPUs per Machine" field. It is assumed that number of GPUs on all machines/nodes is the same.

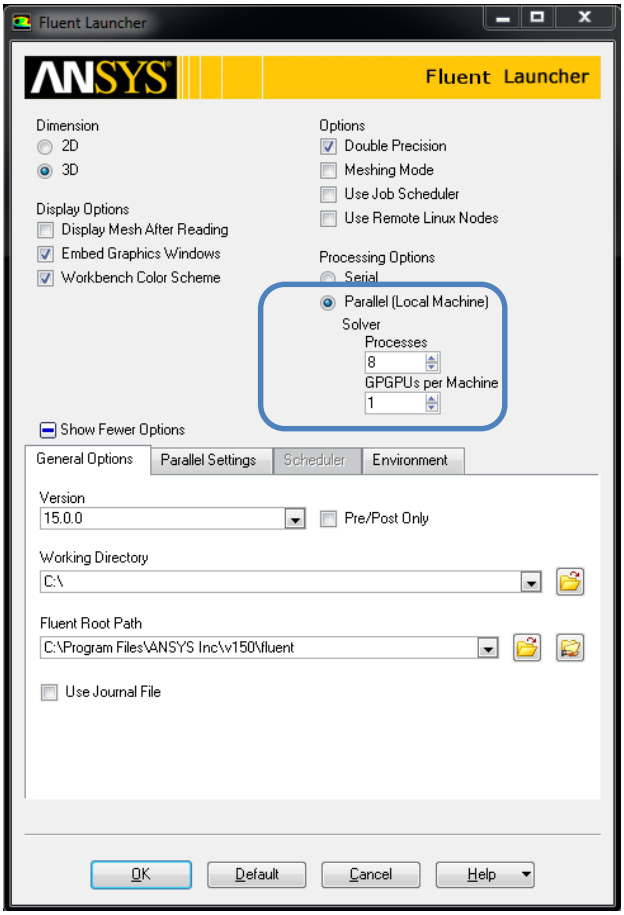

Figure 1. Fluent Launcher Panel in Interactive Mode to Enable and Specify GPUs

For users who are running ANSYS Fluent 15.0 in a shell on a Linux system, the following command invokes and specifies the number of GPUs:

```
fluent <version>  -g  -t<nprocs>  -gpgpu=<ngpgpus>  -i journalfile > outputfile
```

where

version must be replaced by 2d, 2ddp, 3d or 3ddp version of ANSYS Fluent you want to run

nprocs specifies the total number of CPU processors across all machines/nodes

ngpgpus specifies the number of GPUs per machine/node available in parallel mode. Note that t the number of processes per machine must be equal on all machines and *ngpgpus* must be chosen such that the number of processes per machine is an integer multiple of *ngpgpus*. That is, for *nprocs* solver processes running on *M* machines using *ngpgpus* GPUs per machine, we must have:

(nprocs) mod *(M)* = 0

(nprocs/M) mod *(ngpgpus)* = 0

The supported CPU-GPU hardware configuration is described in Figure 2. Unsupported CPU-GPU configurations are described in Figure 3.

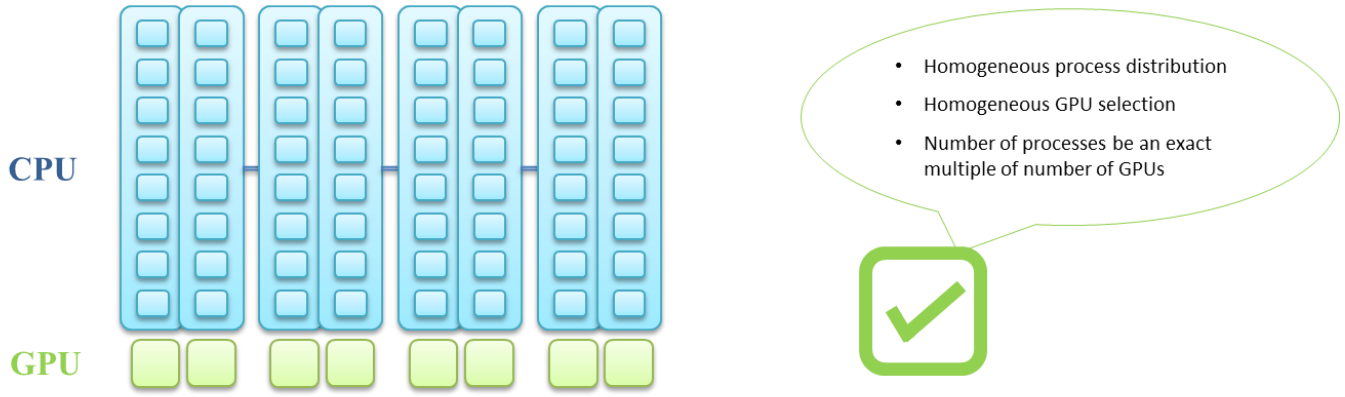

Figure 2. Supported CPU-GPU Hardware Configuration

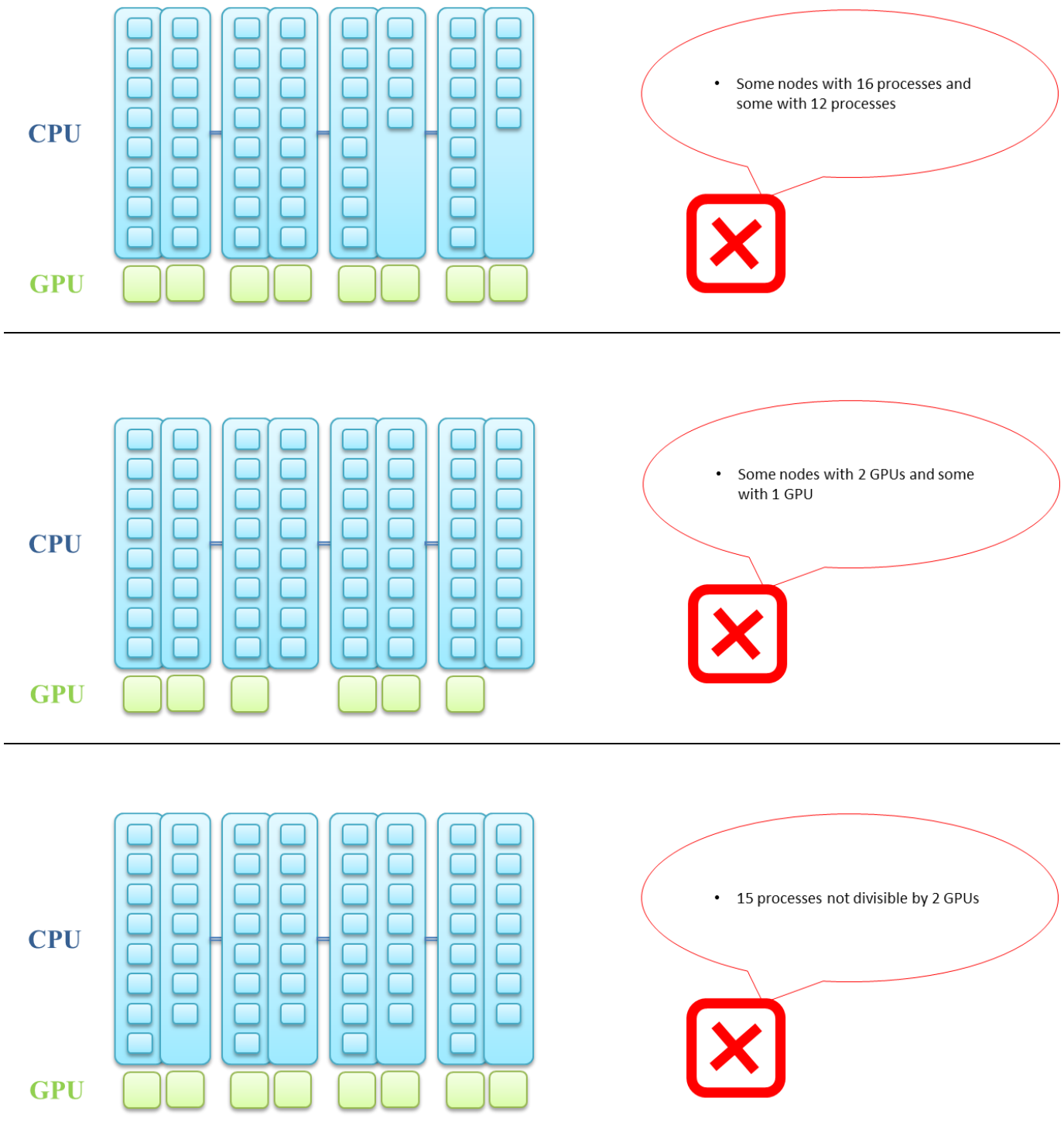

Figure 3. Unsupported CPU-GPU Hardware Configurations

3. CHANGING AMGX CONFIGURATION

In ANSYS Fluent 15.0, the Algebraic Multigrid (AMG) linear system solver used on the CPU is different from that used on the GPU. In the latter case, the AmgX library is used to perform the solution of linear systems. It is a state-of-the-art library that contains implementation of AMG for achieving high performance on the GPUs. The default configuration in Fluent is an outer FGMRES preconditioned by an inner AMG solver.

When running Fluent, one could overwrite the default AmgX configuration settings via journal file commands by specifying the "rpsetvar" command with the appropriate scope setting. The sample Fluent journal file below shows a sequence of ANSYS FLUENT commands, arranged as they would be typed interactively into the program or entered through the GUI or TUI. An example command is highlighted in blue. Lines which start with a semicolon (;) indicate a comment.

```
; Read case and data file
rcd sample.cas.gz
; Amg Verbosity Option
(rpsetvar 'amg/verbosity 4)
; AmgX Configuration Settings
(rpsetvar 'amg/nvamg-config "main:max_iters=20, main:gmres_n_restart=20,
amg:selector=SIZE_2, determinism_flag=1")
; Start Profile
(trace-command "start-profile")
; Run Iterations
it 50
; Stop Profile
(trace-command "stop-profile")
; Print Profile
(print-profile -1)
; Performance Timer Statistics for Iterations
/parallel/timer/usage
; Exit Fluent
exit yes
```

This document does not cover the details of all the configuration settings. However, the following are explanations of some important configuration strings:

selector

This string specifies the algorithm used to select aggregates. The valid options are SIZE_2, SIZE_4, and SIZE_8, which attempt to create aggregates of size 2, 4 and 8, respectively.

max_iters

This string specifies the maximum number of iterations performed before a solver will exit. Setting this to 1, for example, means that only a single iteration of the solver will be applied, regardless of any convergence test. If the convergence test succeeds before max_iters are executed, the solver will exit. Also, in the context of GMRES solver, this parameter specifies the total number of iterations performed, in other words, the number of times GMRES will restart is [(*max_iters/gmres_n_restart)-1].*

gmres_n_restart

This string applies only to the [F]GMRES solver type. This sets the size of the Krylov subspace before a restart is applied. Since GMRES stores all trailing Krylov vectors, the storage requirement of the GMRES solver grows proportionally to this value.

determinism_flag

AmgX often relies on randomized algorithms, therefore the computed results may vary from one run to the next. When this flag is set to 1, the algorithm heuristics will be adjusted such that the results are deterministic and repeatable. This typically results in a small performance penalty, on the order of 10-20%.

3.1 AmgX Verbosity

To turn on the AmgX verbosity for GPU runs, set the following rpsetvar command in the Fluent's Journal file.

```
(rpsetvar 'amg/verbosity 4)
```

This will print the AMG Grid and FGMRES Solve statistics and timings. A sample log file is shown below with important statistics highlighted.

```
AMG Grid:
        Number of Levels: 6
          LVL         ROWS                NNZ    SPRSTY       Mem (GB)
        --------------------------------------------------------------------
          0(D)      3136000           86201600  8.77e-06          0.701
          1(D)       447112           17211616  8.61e-05          0.271
          2(D)        60264            2970304  0.000818         0.0466
          3(D)         8168             451808   0.00677        0.00707
          4(D)         1100              55216    0.0456       0.000865
          5(D)          144               5088     0.245        7.9e-05
        --------------------------------------------------------------------
        Grid Complexity: 1.16479
        Operator Complexity: 1.24007
        Total Memory Usage: 1.02689 GB
        --------------------------------------------------------------------
          iter      Mem Usage (GB)       residual                                                                rate
        ------------------------------------------------------------------------------------------------------------------------------------
           Ini            2.89263   3.131174e-03   1.490097e-02   1.182703e-03   1.990474e-04
             0            2.89263   4.978470e-03   1.340199e-02   1.124433e-03   3.758655e-04         1.5900  0.8994  0.9507  1.8883
             1             2.8926   5.678319e-03   1.041261e-02   9.559884e-04   4.507242e-04         1.1406  0.7769  0.8502  1.1992
             2             2.8926   5.684893e-03   7.408808e-03   7.967974e-04   4.801819e-04         1.0012  0.7115  0.8335  1.0654
             3             2.8926   5.084277e-03   5.459748e-03   6.575208e-04   4.609139e-04         0.8943  0.7369  0.8252  0.9599
             4             2.8926   4.448766e-03   4.086613e-03   5.352960e-04   4.312209e-04         0.8750  0.7485  0.8141  0.9356
             5             2.8926   3.777961e-03   3.094611e-03   4.224051e-04   3.777857e-04         0.8492  0.7573  0.7891  0.8761
             6             2.8926   3.104331e-03   2.307577e-03   3.288549e-04   3.289658e-04         0.8217  0.7457  0.7785  0.8708
             7             2.8926   2.479196e-03   1.653397e-03   2.631800e-04   2.799230e-04         0.7986  0.7165  0.8003  0.8509
             8             2.8926   2.015255e-03   1.254164e-03   2.291125e-04   2.360461e-04         0.8129  0.7585  0.8706  0.8433
             9             2.8926   1.666988e-03   9.978224e-04   1.948702e-04   1.942735e-04         0.8272  0.7956  0.8505  0.8230
            10             2.8926   1.398857e-03   7.831397e-04   1.659515e-04   1.633021e-04         0.8392  0.7848  0.8516  0.8406
            11             2.8926   1.197029e-03   6.412281e-04   1.470132e-04   1.468076e-04         0.8557  0.8188  0.8859  0.8990
            12             2.8926   9.998733e-04   5.254482e-04   1.228390e-04   1.288330e-04         0.8353  0.8194  0.8356  0.8776
            13             2.8926   8.247843e-04   4.513225e-04   1.005395e-04   1.085794e-04         0.8249  0.8589  0.8185  0.8428
            14             2.8926   6.700947e-04   3.671417e-04   8.424725e-05   9.011230e-05         0.8124  0.8135  0.8380  0.8299
            15             2.8926   5.737998e-04   3.007239e-04   7.285303e-05   7.705557e-05         0.8563  0.8191  0.8648  0.8551
            16             2.8926   4.925130e-04   2.312432e-04   6.416459e-05   6.726709e-05         0.8583  0.7690  0.8807  0.8730
            17             2.8926   4.130499e-04   1.670042e-04   5.522863e-05   5.735518e-05         0.8387  0.7222  0.8607  0.8526
            18             2.8926   3.509784e-04   1.306505e-04   4.870342e-05   5.073071e-05         0.8497  0.7823  0.8819  0.8845
            19             2.8926   2.867403e-04   1.021497e-04   4.025316e-05   4.309812e-05         0.8170  0.7819  0.8265  0.8495
        ------------------------------------------------------------------------------------------------------------------------------------
        Total Iterations: 20
        Avg Convergence Rate:               0.8873         0.7795         0.8445         0.9263
        Final Residual:           2.867403e-04   1.021497e-04   4.025316e-05   4.309812e-05
        Total Reduction in Residual:      9.157599e-02   6.855242e-03   3.403488e-02   2.165219e-01
        Maximum Memory Usage:                2.893 GB
        ------------------------------------------------------------------------------------------------------------------------------------
Total Time: 2.0617
    setup: 0.11749 s
    solve: 1.94421 s
    solve(per iteration): 0.0972103 s
    50  5.8385e-03  1.0217e-06  8.1096e-08  1.3648e-08  1.8255e-03  1.5701e-03  0:00:00    0
```

3.2 Choice of Selector Aggregate Size

Aggregation multi-grid is a family of methods where the coarse grid is formed by aggregating values from multiple fine points to form a coarse point. ANSYS Fluent 15.0 has a default selector setting of SIZE_8, which means the algorithm will attempt to aggregate 8 fine points to form a single coarse aggregate. Therefore, the number of AMG levels often varies based on the choice of the selector size. From Figure 4, it is clear that SIZE_8 takes more time to complete the solution because of the need for more FGMRES iterations. Also, if you compare the memory usage, you would notice that SIZE_8 would need more memory at the outer FGMRES Solver because of the need for more no of FGMRES Iterations even though the AMG Grid Memory Usage is less. Other suggested values are SIZE_2 or SIZE_4. Particularly, the choice of selector SIZE_2 seems to be optimum considering both the residual convergence and performance. One could change the current default SIZE_8 Selector in the fluent journal file to SIZE_2 as shown below:

```
 (rpsetvar 'amg/nvamg-config "main:max_iters=20, main:gmres_n_restart=20,
amg:selector=SIZE_2, determinism_flag=1")
```

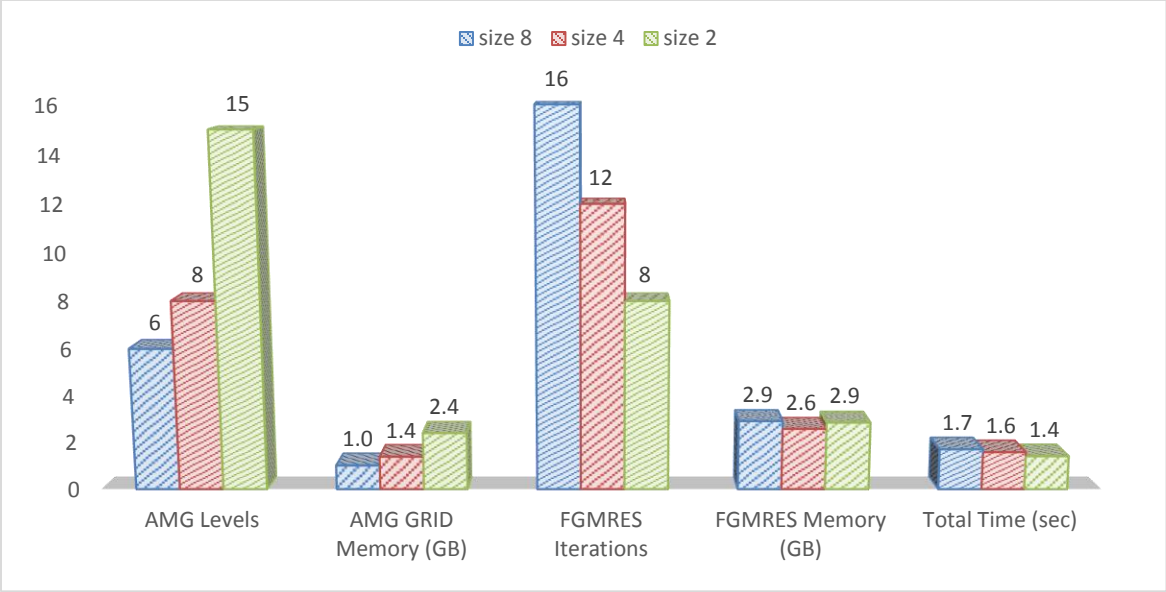

Figure 4. AmgX Aggregate Size Choice and its Effect on Memory Requirements and Performance

3.3 Choice of FGMRES Maximum Iterations

Maximum iterations for the outer FGMRES Solver is currently set at 100. However, it usually takes under 10 iterations for linear equation's solution to converge to the default tolerance. If a particular solution does not converge, it will require all 100 iterations to be computed before the computation is stopped. When this happens, it's often a costly hit on performance as well as memory requirements (and the user might get an Out of Memory Error). Also, it is an indication that the solution is nearly divergent. To avoid these issues, changing the max_iters setting to the fluent default (max cycles=30) or even setting this value to 20 iterations should be sufficient for most cases.

```
  86          3.8393   4.127700e-03   6.529362e-04   1.120735e-04   2.872561e-04        1.0001  0.9984  0.9986  1.0000
  87          3.8393   4.127260e-03   6.517266e-04   1.119438e-04   2.870865e-04        0.9999  0.9981  0.9988  0.9994
  88          3.8393   4.126193e-03   6.506991e-04   1.118468e-04   2.868996e-04        0.9997  0.9984  0.9991  0.9993
  89          3.8393   4.121493e-03   6.481275e-04   1.114808e-04   2.864964e-04        0.9989  0.9960  0.9967  0.9986
  90          3.8393   4.120920e-03   6.481792e-04   1.114673e-04   2.864843e-04        0.9999  1.0001  0.9999  1.0000
  91          3.8393   4.125453e-03   6.481584e-04   1.115336e-04   2.867032e-04        1.0011  1.0000  1.0006  1.0008
  92          3.8393   4.125643e-03   6.477130e-04   1.115315e-04   2.867559e-04        1.0000  0.9993  1.0000  1.0002
  93          3.8393   4.133534e-03   6.477956e-04   1.115373e-04   2.869168e-04        1.0019  1.0001  1.0001  1.0006
  94          3.8393   4.134797e-03   6.485990e-04   1.115834e-04   2.869170e-04        1.0003  1.0012  1.0004  1.0000
  95          3.8393   4.132899e-03   6.479970e-04   1.114664e-04   2.866585e-04        0.9995  0.9991  0.9990  0.9991
  96          3.8393   4.132940e-03   6.479875e-04   1.114670e-04   2.866634e-04        1.0000  1.0000  1.0000  1.0000
  97          3.8393   4.135367e-03   6.485502e-04   1.115557e-04   2.868399e-04        1.0006  1.0009  1.0008  1.0006
  98          3.8393   4.131775e-03   6.490381e-04   1.114598e-04   2.866345e-04        0.9991  1.0008  0.9991  0.9993
  99          3.8393   4.129775e-03   6.483283e-04   1.113239e-04   2.863605e-04        0.9995  0.9989  0.9988  0.9990
-------------------------------------------------------------------------------------------------------------------------------
Total Iterations: 100
Avg Convergence Rate:               0.9828         0.9718         0.9759         0.9807
Final Residual:           4.129775e-03   6.483283e-04   1.113239e-04   2.863605e-04
Total Reduction in Residual:      1.765431e-01   5.748158e-02   8.686889e-02   1.429451e-01
Maximum Memory Usage:               3.839 GB
-------------------------------------------------------------------------------------------------------------------------------
```

One could change the current default max_iters in the fluent journal file to '20' as shown below:

```
(rpsetvar 'amg/nvamg-config "main:max_iters=20, main:gmres_n_restart=20,
amg:selector=SIZE_2, determinism_flag=1")
```

3.4 Choice of gmres_n_restart setting

The **gmres_n_restart** setting could be set to the same value as max_iters. In that way FGMRES stores all trailing Krylov vectors, and the storage requirement of the FGMRES solver grows proportional to this value. This shouldn't be an issue provided you can easily fit everything on the GPU memory. When an Out-of-Memory Error occurs, this parameter could be tuned and reduced by half of max_iters, i.e. 10. For example, if max_iters=20 and gmres_n_restart=10, then 1 restart will be performed.

One could change the gmres_n_restart setting in the fluent journal file as shown below:

```
(rpsetvar 'amg/nvamg-config "main:max_iters=20, main:gmres_n_restart=10,
amg:selector=SIZE_2, determinism_flag=1")
```

4. GPU MEMORY REQUIREMENTS

ANSYS Fluent is a memory-intensive application and it is very important to understand the general memory requirements for a particular job. For this reason it is recommended to use a high memory GPU such as the NVIDIA Tesla™ K40 or NVIDIA Quadro® K6000 which have 12 GB of Memory. One could use the following rule of thumb to estimate the total GPU memory requirements:-

AMG_GRID_Memory_in_GB

= (Precision_Multiplier) x (No_of_Cells_in_million) x (AMG_selector_factor) x 1.6

Additional_FGMRES_Memory_Factor

= (Max_FGMRES_Iterations) x (Precision_Multiplier) x 0.03

Max_GPU_Memory_in_GB

= AMG_GRID_Memory_in_GB + [AMG_GRID_Memory_in_GB x Additional_FGMRES_Memory_Factor]

Precision_Multiplier:-

- For Single Precision (SP) analysis – specify 1
- For Double Precision (DP) analysis – specify 2

No_of_Cells_in_million:-

- Specify the number of cells in million

AMG_selector_factor:-

- For SIZE_2 – specify 1
- For SIZE_4 – specify 0.6
- For SIZE_8 – specify 0.45

Max_FGMRES_Iterations:-

- Specify the main:max_iters value

Example:

Assume the following Inputs:

- Precision_Multiplier = 2 (refers to DP)
- No_of_Cells_in_million=10
- AMG_selector_factor=1 (refers to SIZE_2)
- Max_FGMRES_Iterations=20

GPU Memory requirements and the number of GPUs required for running the job on GPUs for the above Example settings are shown in Figures 5 and 6 based on the choice of AmgX Aggregate Size.

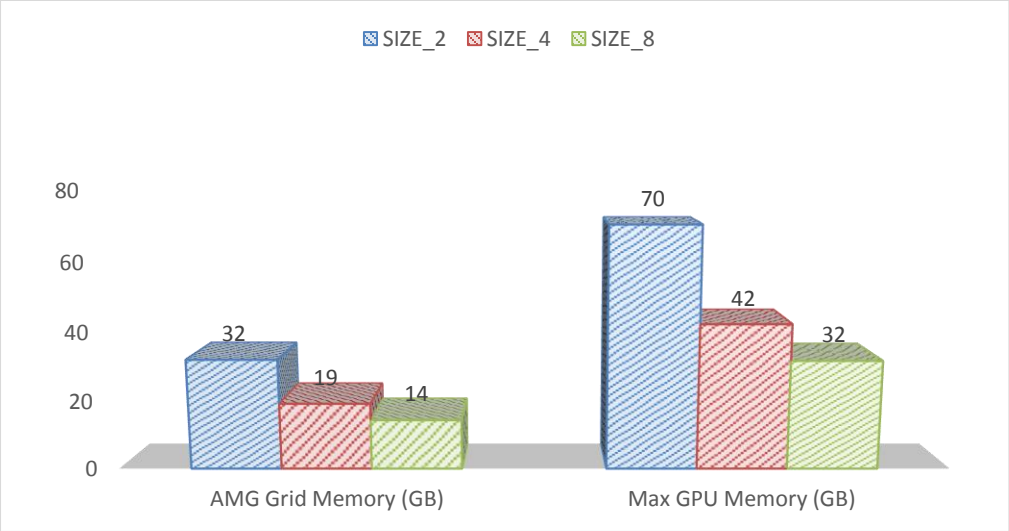

Figure 5. GPU Memory Evaluation Based on the Example

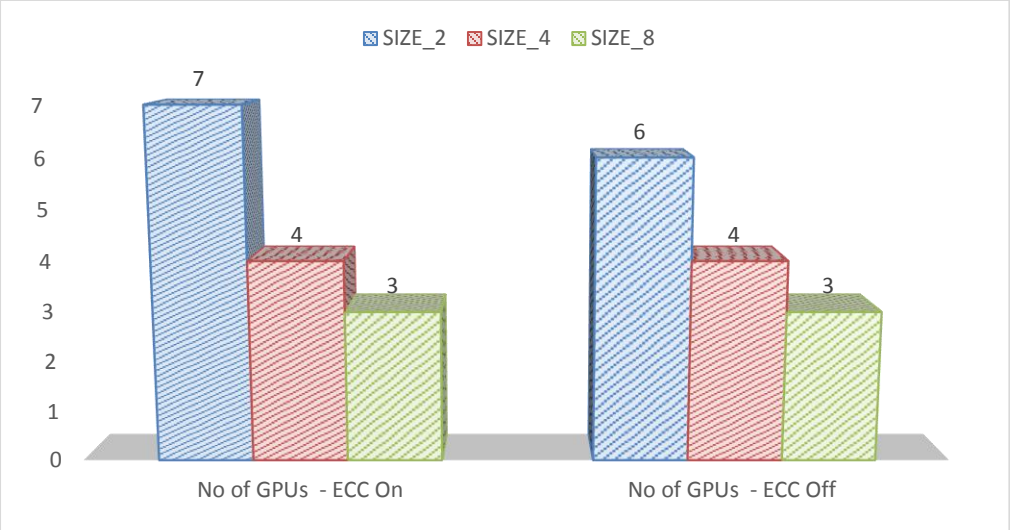

Figure 6. No. of Tesla K40 GPUs Required Based on the Memory Evaluation

5. EVALUATING GPU PERFORMANCE

Understanding and evaluating GPU performance is of utmost importance to many users to maximize the benefits of heterogeneous CPU-GPU systems. As GPUs accelerate the AMG solver or linear solver fraction in a CFD calculation, the speed ups in Fluent depend on the portion of the time spent in the linear solver compared to the total solution time.

Figure 7 shown below helps to evaluate the "speed up" in Fluent based on the linear solver fraction and the related "speed ups" achieved in the AMG solver on GPUs.

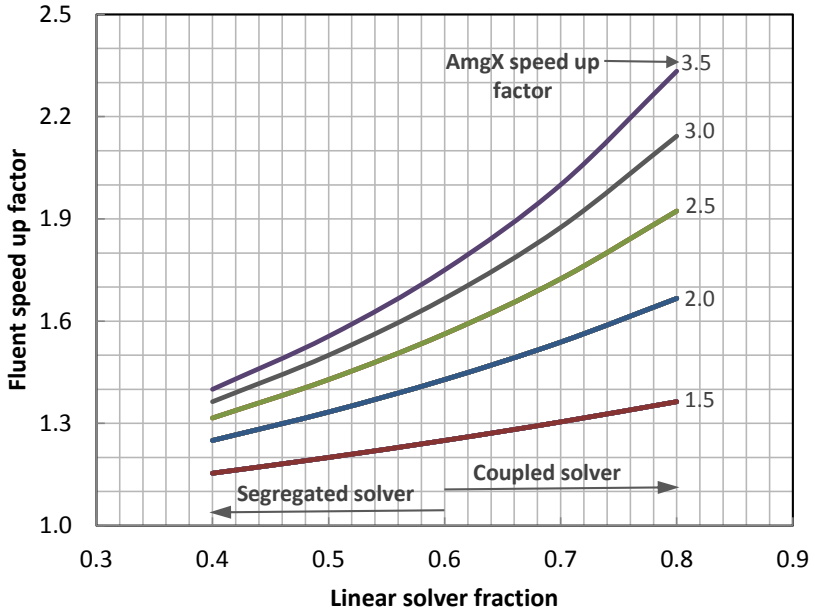

Figure 7. Speed ups in Fluent based on the AMG Performance and Linear Solver Fractions

The linear solver fraction in a CFD calculation can be found from the CPU run when the following command is added to the journal file.

```
/parallel/timer/usage
```

It is reported towards the end of the output file after the successful completion of calculations as shown below, which is nearly 75% or 0.75 in this case.

```
'LE wall-clock time per iteration: 12.299 sec (74.8%)'
```

Both the pressure-based and density-based coupled solvers result in higher linear solver fractions (above 0.6) whereas the segregated solver typically has lower fractions. As a consequence, higher speed ups can be expected from coupled solvers. However, lower linear solver fractions in segregated solver might slow down the calculations because of data transfer overheads, thus not recommended in the current version 15.0.

To calculate the Fluent speed up, find out the total wall-clock times from GPU+CPU and CPU runs

$$\text{Fluent speed up factor} = \frac{Total\ wall{-}clock\ time\ from\ GPU{+}CPU\ run}{Total\ wall{-}clock\ time\ from\ CPU\ run}$$

For example, when the linear solver fraction is around 0.75, a Fluent speed up factor of 2.0 indicates that the AMG portion of the calculation is accelerated by 3x with GPUs referring to the above plot.

By tuning the AMG parameters, users should be able to get better AMG speed ups for high Fluent speed up factors as previously explained.

Notice

Trademarks

Copyright