OPTISTRUCT SCALABILITY USING MULTIPLE CPUs AND NVIDIA® GPUs

Altair® OptiStruct® is an industry-proven structural analysis solver for static, dynamic, linear, nonlinear, implicit, and explicit structural simulations. The solver is typically deployed in design and optimization workflows that help engineers rapidly produce innovative and lightweight structural geometries. Additionally, OptiStruct provides a wide variety of simulation capabilities for vibration, acoustics, powertrain, impact analysis, and more. Hence, it supplements the physics-based analysis and optimization workflows utilized by various industries.

This document is intended to introduce OptiStruct users to techniques and hardware configurations that obtain efficient and scalable performance for a wide variety of simulations. Some of these assessments include: linear static, nonlinear static, linear transient, nonlinear transient, explicit dynamic, eigen value, and buckling analysis. The document also provides instructions to utilize various OptiStruct scalability methods — that deploy CPUs and GPUs — to quickly assess user specific models. Finally, the document includes scalability test data and some corresponding hardware recommendations.

OPTISTRUCT CPU AND GPU COMPUTING

OptiStruct can implement multiple parallelization techniques, including shared memory parallelization (SMP) and distributed multiprocessing (DMP) implemented via domain decomposition (DDM). Graphics processing units (GPUs) can also be used to scale the performance of OptiStruct. The following sections serve as a quick introduction to these different parallelization techniques.

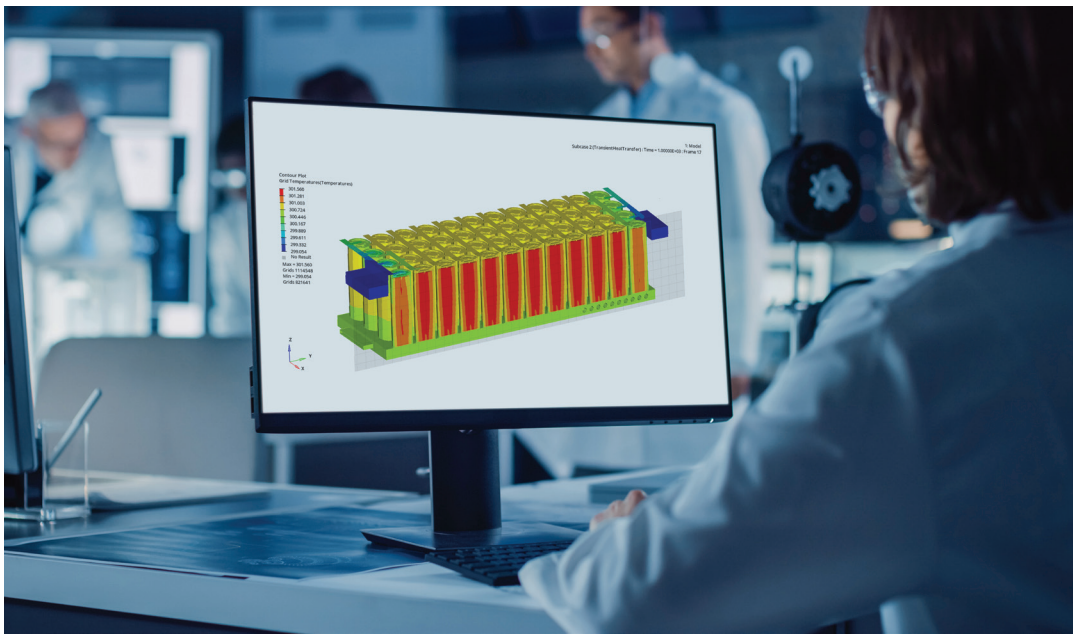

Fig 1. An engineer assesses the thermal management of a battery using OptiStruct.

Shared Memory Parallelization (SMP)

SMP is a parallelization technique that incorporates the use of multiple threads to run programs. It typically involves the program sharing a single memory space while the execution follows different paths through multiple cores. The aforementioned threads are lightweight processes used to parallelize program components for faster execution.

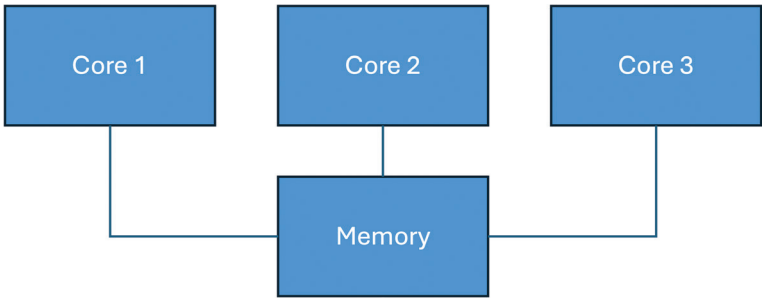

Fig 2. Shared memory parallelization (SMP) scalability diagram.

Running OptiStruct SMP

OptiStruct SMP is available as part of the standard installation. It is typically effective when running models on single node workstations with single or multiple processors, each with multiple cores.

When initiated, the OptiStruct SMP script picks the required executable for the run and simultaneously sets all the required environment variables. SMP runs can be activated using the "-nt" run option either via the Altair Compute Console (ACC) graphical user interface (GUI) or from the Command line via OptiStruct script. The command to run OptiStruct SMP with four threads on Linux and Windows machines is:

$ALTAIR_HOME/scripts/optistruct file.fem -nt 4

The "-nt" option variable denotes the number of threads used in the run.

Supported Solutions for OptiStruct SMP

OptiStruct SMP is generally supported for all analysis solutions and optimization types. For additional details on how to use SMP, refer to the OptiStruct User Guide.

Distributed Memory Parallelization (DMP)

DMP utilizes heavyweight processes called message passing interfaces (MPIs) to achieve parallelization. Each MPI process has its own memory space and communicates with other parallel processes. Efficient DMP runs are possible if sufficient RAM is available on the machine, or clusters, to handle all distributed memory spaces. If this is not the case, it may be more efficient to use SMP-based parallelization.

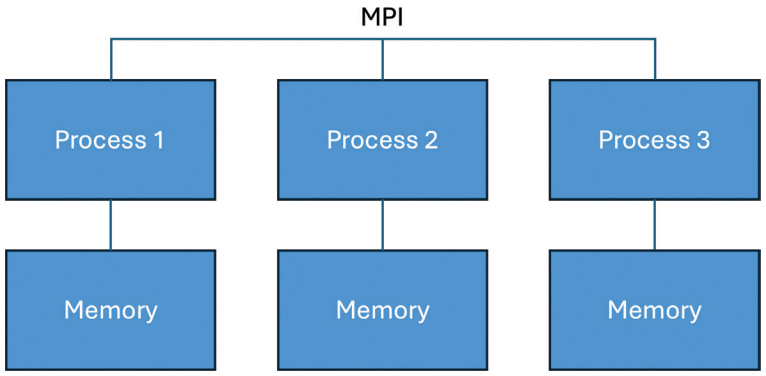

Fig 3. Distributed Memory Parallelization (DMP) scalability diagram.

Running OptiStruct DMP

OptiStruct DMP is available as part of the standard installation. It is typically effective when running models on multiple node workstations or single node workstations with multiple processors. It is also necessary to have sufficient memory to handle all distributed memory requirements.

When initiated, the OptiStruct script/ACC picks the required executable for the run and simultaneously sets all the required environment variables. DMP runs can be activated using the "-np" run option either via the ACC GUI or from the Command line via OptiStruct script.

DDM is a DMP-based parallelization process supported in OptiStruct. It involves multiple levels of parallelization and is typically supported across a wide range of analysis and optimization types. MPIs are used to communicate between multiple tasks, loads, subcases, partitions, levels, and domains.

The following example shows how a typical DDM run, with four MPI processes, is conducted on Windows and Linux machines:

```
$ALTAIR_HOME/scripts/optistruct file.fem -np 4 -ddm
```

The "-np" option variable expresses the number of MPI processes that are used for the solution. For additional details on how to use DMP and DDM, refer to the OptiStruct User Guide.

Supported Solutions for OptiStruct DMP

DDM is supported for a wide range of analysis and optimization types. However, support varies for task-based (level 1) and geometric (level 2) partitioning. OptiStruct automatically determines the required level of parallelization for a solution and then records the appropriate information in the .out file. Users can manually control this process using either "PARAM," "DDMNGRPS," or "–ddmngrps" run option. If DDM level 1 is selected but is not supported for the user's model, then OptiStruct adds a message to the .out file and automatically switches to DDM level 2. However, if DDM is requested and both levels are not supported, then OptiStruct will return an error message.

Table 1. DDM Level 1 Support

DDM Level 1 Support	DDM Task-based Parallelization (Level 1)	
	Parallelizable Tasks	Non-Parallelizable Tasks
Linear static analysis	Two or more static boundary conditions are parallelized.[a]	
Buckling analysis	Two or more buckling subcases are parallelized.[a]	
Direct and modal frequency response	Loading frequencies are parallelized in the frequency-response solution.[b] Eigenvalue extraction is parallelized for modal frequency-response functions (FRFs) when multiple boundary conditions are present.	
Modal linear transient analysis	Multiple subcases are parallelized.	
Global search (DGLOBAL)	Starting points are parallelized.	
Multi-model optimization (MMO)	Models listed in the master file are parallelized depending on the total "-np" specified and/or the number of "-np" defined for each model in the master file.	Task-based parallelization is not applicable within each MMO model. Only geometric partitioning (DDM Level 2) is conducted for each individual model within MMO.
Internal superelements	Internal superelements are generated in parallel.	

a. Sensitivities are parallelized for optimization
b. Optimization is not parallelized

Table 2. DDM Level 2 Support

DDM Geometric-based Parallelization (Level 2) Support		
Linear static analysis and optimization	Nonlinear static analysis for small displacement (SMDISP) and large displacement (LGDISP)	Structural direct frequency response
Normal modes analysis (with Lanczos)[c]	Linear buckling analysis and optimization	Modal frequency response (with Lanczos)[d]
Preloaded modal frequency response (with AMLS/ AMSES) where only the preloading static subcase is parallelized	Acoustic direct frequency response	Direct frequency response with MFLUID
Nonlinear direct transient response	Direct linear transient analysis	Explicit dynamic analysis (NLEXPL)
Structural fluid structure interaction (SFSI)	Fatigue analysis	Nonlinear steady-state heat transfer analysis
Nonlinear transient heat transfer analysis		

c. DDM Level 2 for eigenvalue analysis is based on geometric partitioning only for Lanczos. DDM for AMLS and AMSES eigenvalue extraction is not recommended (if multiple boundary conditions exist, DDM level 1 may offer some speedup for AMLS and AMSES eigenvalue extraction).

d. DDM Level 2 for the frequency response part of the modal FRF solution is not supported. If multiple boundary conditions exist for the eigenvalue analysis part of the modal FRF solution (with AMLS/AMSES) then DDM level 1 may offer some speedup.

Table 3. DDM Level 2: Noteworthy Solutions That Are Unsupported

DDM Geometric-based Parallelization (Level 2) Noteworthy Solutions That Are Unsupported		
Component mode synthesis (CMS) superelement generation	AMLS eigensolution	Failsafe topology optimization (FSO)
Component dynamic synthesis (CDS) superelement generation	Modal complex eigenvalue analysis	Modal linear transient analysis

For more DDM support details, refer to the OptiStruct User Guide.

Graphics Processing Unit (GPU)

GPUs can be used to improve the performance of computationally intensive engineering applications. Generally, OptiStruct offloads most of the computations from the sparse direct equation solver to the GPU and conducts communication and data transfer between the GPU and CPU cores.

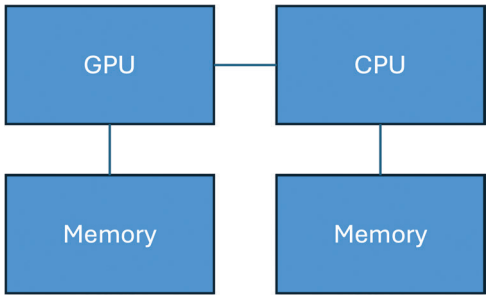

Fig 4. GPU-based scalability diagram.

Running OptiStruct GPU Processes

OptiStruct's GPU scalability processes are available as part of the standard installation. It is typically effective when running models on single node workstations with single, or multiple, GPUs along with single, or multiple, CPU processors.

The OptiStruct script picks the required GPU executable for the run and simultaneously sets all the required environment variables. GPU runs can be activated using the "-gpu" option via the ACC GUI or from the Command line via OptiStruct script. The following example shows how to run OptiStruct with two GPUs on Linux and Windows machines:

```
$ALTAIR_HOME/scripts/optistruct file.fem -gpu -ngpu 2
```

The "-ngpu" run option is optional and is used to denote the number of GPUs being used. Otherwise, a single GPU is its default value. The "-gpu" run option is mandatory as it informs OptiStruct that a GPU run is requested.

Supported Solutions for OptiStruct GPU Processes

OptiStruct GPU processes are supported for a wide range of analysis and optimization types as depicted in Table 4.

Table 4. GPU: Supported Solutions

GPU: Supported Solutions		
Linear static analysis (using MUMPS) and optimization	Nonlinear static and nonlinear transient analysis (using MUMPS)	Eigenvalue analysis (using AMSES or Lanczos eigensolvers) for normal modes analysis and within corresponding modal frequency response or modal transient response analyses.
Direct frequency response analysis (using MUMPS)	The frequency response part of modal frequency response analysis or modal fourier transient analysis when PARAM, FASTFR, YES is used.	Complex eigenvalue analysis.

For more GPU support details, refer to the OptiStruct User Guide.

Hybrid Parallelization

OptiStruct's SMP, DMP, and GPU scalability can be combined into hybrid approaches. This can be necessary when assessments benefit from multiple scalability options. The following sections describe how this can be done.

Hybrid SMP and DMP

Combining SMP and DMP is a commonly used, and highly efficient, technique to reduce runtime. If sufficient RAM is available for all MPI processes simultaneously, in parallel, then both "-np" and "-nt" can be specified to run hybrid parallelization. However, the product of the "-np" and "-nt" values should not exceed the total number of cores on the machine or cluster. The following example shows how to run OptiStruct SMP and DMP on Linux and Windows machines using eight MPI processes, four threads per MPI process, and a total of 32 cores (as 8 x 4 = 32):

```
$ALTAIR_HOME/scripts/optistruct file.fem -np 8 -nt 4
```

Hybrid SMP and GPU

If dedicated GPU cards are available on a node, combining SMP and GPU can provide significant acceleration for supported solutions. If sufficient RAM is available for SMP on the CPU and a dedicated GPU is available, then the "-gpu" option can be used alongside "-nt" to process the model. The following example shows how to run each model using 16 threads in SMP and a single GPU within Linux and Windows machines:

```
$ALTAIR_HOME/scripts/optistruct file.fem -nt 16 -gpu
```

OPTISTRUCT SCALABILITY BEST PRACTICES

There are four computations which contribute the most to runtimes (depending on the solution sequence). They are:

- Matrix factorization in the solver
- Eigenvalue extraction
- Frequency response solution
- Result recovery and output

The following best practice sections address these runtime contributors by providing some guidelines to extract efficient performance when using OptiStruct for highly scalable solutions.

Process and Thread Number-based Best Practices

- If the solution sequence is supported in both SMP and DMP, then one of the critical aspects to determine is whether to use SMP or a Hybrid SMP and DMP method. The answer is based on the memory available. To run parallel MPI processes, distributed memory (with parallel access) is essential. If sufficient distributed RAM memory is not available, then it is typically more efficient to use SMP instead of DMP.
- Run option "-cores" is available as an alternative to "-np" and "-nt". It requires the total number of cores available on the machine, or cluster, as input. For example, "-cores 32," informs OptiStruct that there are 32 available cores for the run. OptiStruct will then automatically set the number of MPI processes, "-np," equal to 8 and the number of threads per MPI process, "-nt," equal to 4.
- For a DMP run, a generally cautious method is to specify the number of threads per MPI process "-nt" equal to the number of cores per socket, and to set the number of MPI processes per machine, or node, equal to the number of sockets. This can be extrapolated for a cluster environment with multiple nodes.

Memory Allocation Best Practices

- Memory allocation is a critical aspect in determining the scalability of a solution. Hardware considerations can limit the amount of available memory and disk. This requires adjustments to make sure the available resources are used efficiently. For most cases, OptiStruct will automatically determine the best memory allocation options for the job without any user intervention. For some specific cases, if user intervention is required, then they should use a "-check" run. If sufficient physical RAM is available, then it may be efficient to enforce an in-core run by using the "-core" run option. This will allow most of the solver computations to run in memory and lead to a more efficient solution.
- If available memory is not sufficient, and if a DMP run is tried, the number of MPI processes "-np" can be reduced. Otherwise, users can switch the run to SMP mode. OptiStruct can automatically and efficiently handle memory allocation in most cases without any requirement for user interaction as long as the solution memory requirement falls within the parameters of the available memory on the machine or node. In addition, a variety of memory handling options are available, for example, "-len," "-minlen," "-maxlen," "-uselen," "-core," etc.

Solution-based Best Practices

The following table denotes the general HPC setup for various solutions.

Table 5. Solution-based Best Practices

Solution-based Best Practices and General Recommendations	
Linear static analysis	Hybrid: SMP and DDM
Normal modes analysis (AMSES)	SMP
Linear modal transient analysis (Single subcase / boundary conditions [BC])	SMP
Linear modal transient analysis (Multiple subcase/BCs)	Hybrid: SMP and DDM (Level 1)
Linear direct transient analysis (single/ multiple subcase/BCs)	Hybrid: SMP and DDM (Level 2)
Nonlinear implicit analysis	Hybrid: SMP, DDM, and GPU
Explicit dynamic analysis (NLEXPL)	Hybrid: SMP and DDM (with as many MPI cores as possible)
Modal frequency response analysis	Hybrid: SMP and DDM (when AMSES is used for eigensolution)
Direct frequency response analysis	Hybrid: SMP and DDM
Complex eigenvalue analysis	Hybrid: SMP and GPU
Response spectrum analysis	SMP
Linear heat transfer analysis	SMP
Nonlinear heat transfer analysis (steady-state and transient)	Hybrid: SMP and DDM
Fatigue analysis	SMP (though certain modules can benefit from DDM)
Optimization	SMP (with hybrid SMP and DDM for specific solutions)

Hardware Resource-based Best Practices

- One of the critical hardware considerations for finite element analysis (FEA) is input/output (I/O), read/write speeds to the disk. Therefore, it is recommended to always utilize a non-network drive for scratch files. If your run is initiated on a network drive, then you can use the TMPDIR I/O option or the "OS_TMP_DIR" environment variable to relocate the scratch file read/write to a local drive.
- For GPU runs or Hybrid SMP and GPU runs, the NVIDIA Pascal™, Volta™, Turing™, Ampere™, and Hopper™ architecture-based graphics cards are supported. The recommended cards are Ampere A100 and Hopper H100. A useful best practice is that 1.0 TFLOPS or more processing power in FP64 operations are needed to observe speedup using GPU or hybrid (SMP and GPU) runs.
- For SMP, DMP, or Hybrid (SMP and DMP) runs, it is important to scale hardware resources based on the type of jobs expected to be run on the machine or cluster. Typically, large FEA models are memory and disk-space intensive. General recommendations include the installation of fast disks for swap space (for example SSDs) local to each node in the cluster and having sufficient RAM so that a majority, if not all, of the matrix solutions run in-core.

Environment Variables

All required environment variables are automatically set when jobs are run using the ACC. These variables are sufficient to extract optimum performance for a wide range of use cases. For some specific use-cases, however, there may be certain environment variables which are required to be set, or changed, from the corresponding defaults. For example, the environment variable "KMP_AFFINITY" can be set to "DISABLE" in cases where more than one model is run on the same machine.

OPTISTRUCT SCALABILITY EXAMPLES

The following scalability curves highlight the performance of OptiStruct for selected solution sequences.

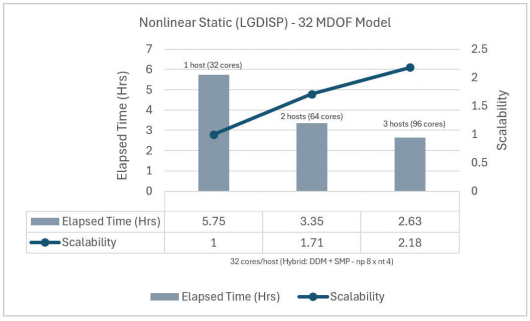

Fig 5. Nonlinear static large displacement analysis on a 32 MDOF model with solid, shells, and 1D elements. CPU: Intel® Xeon® Gold 6226R.

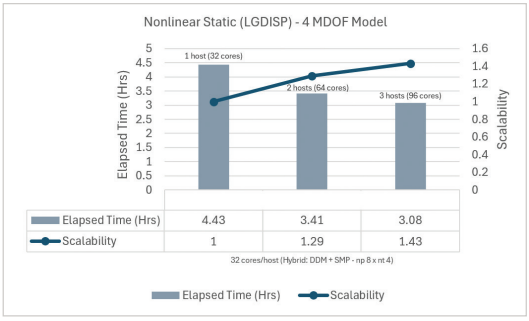

Fig 6. Nonlinear static large displacement analysis on a 4 MDOF model with solid and 1D elements. CPU: Intel Xeon Gold 6226R.

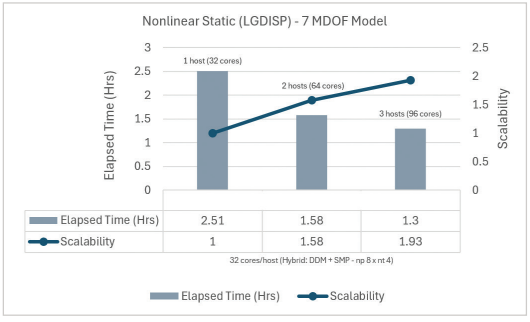

Fig 7. Nonlinear static large displacement analysis on a 7 MDOF model with solid and 1D elements. CPU: Intel Xeon Gold 6226R

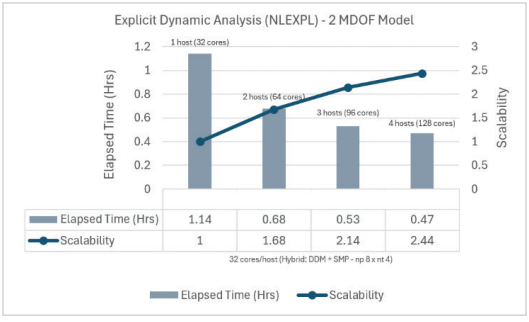

Fig 8. Explicit dynamic analysis on a 2 MDOF model with solid elements. CPU: Intel Xeon Gold 6226R.

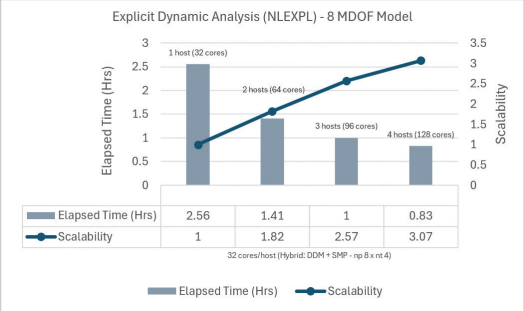

Fig 9. Explicit dynamic analysis on an 8 MDOF model with solid, shell, and 1D elements. CPU: Intel Xeon(R) Gold 6226R.

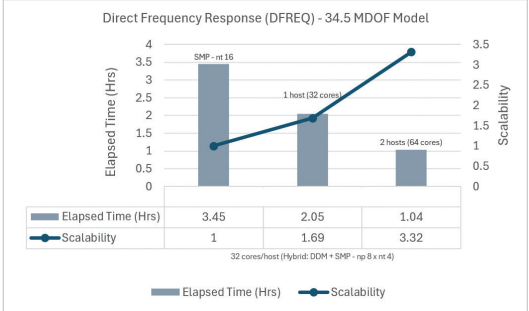

Fig 10. Direct frequency response on a 34.5 MDOF model with solid, shell, and 1D elements. CPU: Intel Xeon Gold 6226R.

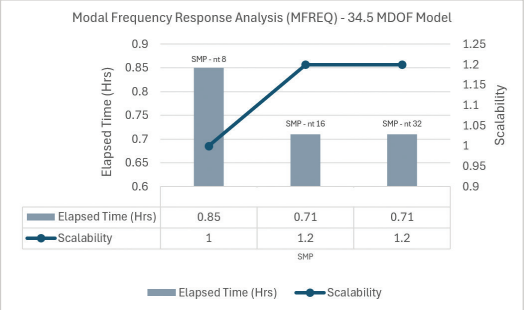

Fig 11. Modal frequency response on a 34.5 MDOF model with solid, shell, and 1D elements. CPU: Intel Xeon Gold 6226R.

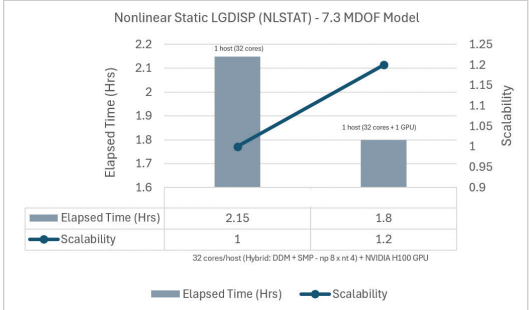

Fig 12. Nonlinear static (NLSTAT) LGDISP on a 7.3 MDOF model with GPU scalability. CPU: Intel Xeon Gold 6326. GPU: NVIDIA H100.

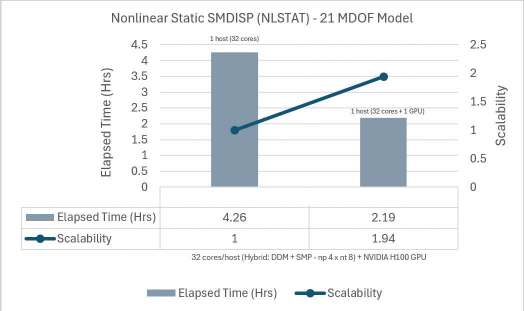

Fig 13. Nonlinear static SMDISP (NLSTAT) on a 21 MDOF model with GPU scalability. CPU: Intel Xeon Gold 6326. GPU: NVIDIA H100.

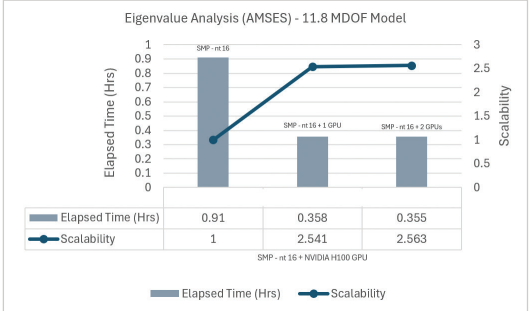

Fig 14. Eigenvalue analysis (AMSES) on a 11.8 MDOF model with GPU scalability. CPU: Intel Xeon Gold 6326. GPU: NVIDIA H100.