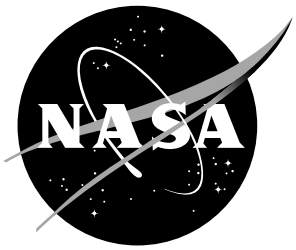
NASA/TM–2019–220251



Computational Performance of Progressive Damage Analysis of Composite Laminates using Abaqus/Explicit with 16 to 512 CPU Cores

A.C. Bergan
Langley Research Center, Hampton, Virginia

February 2019

NASA STI Program. . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI Program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collection of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

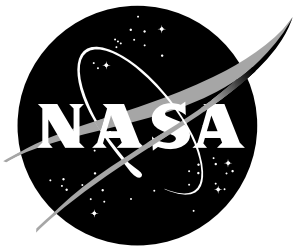
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI Program, see the following:

- Access the NASA STI program home page at `http://www.sti.nasa.gov`
- E-mail your question to `help@sti.nasa.gov`
- Phone the NASA STI Information Desk at 757-864-9658
- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM–2019–220251



Computational Performance of Progressive Damage Analysis of Composite Laminates using Abaqus/Explicit with 16 to 512 CPU Cores

A.C. Bergan
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

February 2019

Acknowledgments

The support provided by Dassault Systèmes Simulia throughout this study is gratefully acknowledged. The author wishes to thank Aharon Mims for executing and processing many preliminary analyses.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

The computational scaling performance of progressive damage analysis using Abaqus/Explicit is evaluated and quantified using from 16 to 512 CPU cores. Several analyses were conducted on varying numbers of cores to determine the scalability of the code on five NASA high performance computing systems. Two finite element models representative of typical models used for progressive damage analysis of composite laminates were used. The results indicate a 10 to 15 times speed up scaling from 24 to 512 cores. The run times were modestly reduced with newer generations of CPU hardware. If the number of degrees of freedom is held constant with respect to the number of cores, the model size can be increased by a factor of 20, scaling from 16 to 512 cores, with the same run time. An empirical expression was derived relating run time, the number of cores, and the number of degrees of freedom. Analysis cost was examined in terms of software tokens and hardware utilization. Using additional cores reduces token usage since the computational performance increases more rapidly than the token requirement with increasing number of cores. The increase in hardware cost with increasing cores was found to be modest. Overall the results show relatively good scalability of the Abaqus/Explicit code on up to 512 cores.

1 Introduction

One of the main barriers to applying progressive damage analysis (PDA) methods to element and component level aerospace structures is computational expense. Improvements in predictive capability of PDA for strength and life of composite coupon scale specimens (e.g., open-hole tension) continue to be reported [1]. However, relatively few attempts have been made to apply the analysis methods demonstrated at the coupon scale to structural elements and components. One reason is that the PDA methods require the use of very small elements, typically less than 0.5 mm [2]. Using these element sizes, run-times for coupon simulations are often several hours or days on desktop workstation machines with between 8 and 24 central processing unit (CPU) cores. Therefore, directly applying the analysis methods to larger structures appears computationally intractable.

A variety of techniques have been proposed for scaling up PDA methods to the element and component level. Perhaps the most common approach is to modify the analysis method to directly reduce the number of degrees of freedom (DoF). Some authors have proposed techniques that simplify the representation of damage as the length scale is increased [3, 4]. Others have proposed approaches for concurrently solving the local damage progression problem within the context of the global structural analysis such that the coupling between the two scales is maintained, e.g. [5]. An alternative approach is to enhance the parallel performance of the solver. Recognizing that most commercial finite element analysis (FEA) software has been slow to embrace parallelization (especially in the case of implicit solvers) and the rapidly reducing cost of parallel high performance computing (HPC) resources with thousands of cores, researchers have developed in-house FEA codes specifically for parallel performance [6–8]. As these research codes demonstrate the benefits of using HPCs, it is expected that the parallel scaling performance of commercial FEA solvers will be improved. Recently, the commercial FEA code Abaqus/Explicit has demonstrated good parallel scaling performance [9]. In the end, the optimal approach to scaling PDA methods to large structures probably uses both simplifying techniques to reduce or limit the number of DoF and parallel computing with HPC resources.

This study aims to quantify the present performance of the Abaqus/Explicit code on several NASA HPC systems. The study was motivated by the Advanced Composites Project interest in predicting element and component level structural response using the NASA Langley Research Center code CompDam [10]. Through characterizing parallel scaling performance, a simple empirical expression is obtained relating run time, number of CPU cores, and number of DoF. The results allow analysts to design models appropriately sized for the available computational resources and acceptable run times.

This report is organized as follows. First, the computing resources used in the study are described. Next, the two finite element models used for the evaluation are summarized. The models were selected to represent a range of complexity in terms of the damage progression behavior. In the following section, the computational performance results are described. Then, an empirical estimate for run time is provided. Finally, the cost of analysis with increasing number of cores is considered.

2 Computational Resources

The performance of Abaqus/Explicit was evaluated using several NASA computing systems. The computing systems were chosen to represent different machine architectures and several recent generations of hardware. The systems included one shared-memory server, denoted H, and four distributed-memory systems, denoted K2, K3S, K3H, and K4. Table 1 summarizes the features of the four systems.

Table 1: NASA compute systems used in this study.

	Compute system designation				
	H	K2	K3S	K3H	K4
Memory architecture	Shared	Distributed	Distributed	Distributed	Distributed
Nodes	1	160	262	135	172
CPU model	Haswell	Westmere	Sandybridge	Haswell	Skylake
CPUs/node	4	2	2	2	2
Cores/CPU	18	6	8	8	20
Total cores	72	1920	4192	2160	6880
Memory/core (GB)	32	2	2	4	2.4
Interconnect speed (Gbps)	N/A	32	56	56	100
Acquisition year	2016	2011	2013-2016	2013-2016	2018

System H is a Hewitt-Packard ProLiant DL850 generation 9 and is representative of a modern system often used for finite element analysis with commercial codes such as Abaqus, Nastran, and Ansys at NASA Langley Research Center. H is a shared memory machine with 4 Intel Haswell E7-8890 v3 CPUs with 18 cores operating 2.5 GHz, yielding a total of 72 cores. Since the machine is often used for solving problems using an implicit solver, a relatively large amount of memory is installed: 2.3 TB for a total of 32 GB/core.

K2 is a Silicon Graphics Altix ICE 8400 distributed memory system acquired in 2011. Each node has 2 Intel X5675 Westmere CPUs with 6 cores each operating at 3.07 GHz. A total of 24 GB of memory is available on each node so that 2 GB of memory is available per core. In contrast to system H, all of the distributed memory machines have a relatively small amount of memory per core.

K3S and K3H refer to nearly identical nodes that comprise a Silicon Graphics Altix ICE X distributed memory machine collectively referred to as K3. Herein, the nodes are treated separately and all analyses are run on a homogeneous set of nodes. The K3S nodes utilize 2 Intel E5-2670 Sandybridge CPUs with 8 cores each, operating at 2.6 GHz and 2 GB of memory per core. The K3H nodes have 2 Intel E5-2640v3 Haswell CPUs with 8 core each, operating at 2.6 GHz and 4 GB of memory per core.

K4 is the newest distributed memory machine used in this study. K4 is an HP

Apollo 6000 distributed memory system acquired in 2018. Each node includes 2 Intel Gold 6148 Skylake CPUs with 20 cores each, operating at 2.4 GHz. The nodes have 96 GB of memory for 2.4 GB per core. The interconnect speed varies among the different generations of distributed memory machines, with K4 being the fastest at 100 Gpbs.

Abaqus/Explicit 2017 build ID 2016_09_27-17.54.59 126836 was used on all systems. The Intel Fortran v16 compiler was used for the models that require user-subroutines. While these two pieces of software were kept constant for all runs, other libraries and software on the systems were patched and updated following normal organizational procedures during the course of several months over which the runs were executed. As a result, updates to underlying libraries may have contributed to scatter in the reported run times.

3 Finite Element Models

Two structural models for progressive damage analysis of composite laminates were used in this evaluation. The models represent a range of complexity of in terms of the damage progression. The first model was a mixed-mode bend specimen which includes the propagation of a delamination at a single interface. The vast majority of the study was conducted using the mixed-mode bend model. The second model considered a $\pm 45^\circ$ unnotched laminate using the CompDam [11] user-subroutine. The output requests used in the models are intended to be typical of what would be utilized in progressive damage analysis, with field data written for many increments. Although not quantified separately here, the disk write performance may play a role in the runtime and scalability.

The two models were selected for this study as representative of typical models used for progressive damage analysis of composite structures. The computational scaling performance is generally expected to be problem dependent. The models include several features that are often required for progressive damage analysis of composite structures including cohesive elements, a continuum damage mechanics model implemented as user-defined material model, large displacements, and interacting damage mechanisms. However, several other model features, for example contact and tie constraints, were not considered. Therefore, while it is expected that the results generated for these two models are representative of the class of models, deviations may occur.

The two models are described briefly in this section with an emphasis on the computational aspects relevant to the scalability of the analysis. The section concludes with a brief review of the domain decomposition method by which the solution is obtained in parallel.

3.1 Mixed-mode bend (MMB)

The mixed-mode bend (MMB) model represents the standardized fracture test used in characterization of the mode mix behavior of interlaminar fracture propagation in unidirectional fiber reinforced polymer matrix composites [12]. A schematic of the test configuration and the model is shown in Fig. 1. The specimen thickness is

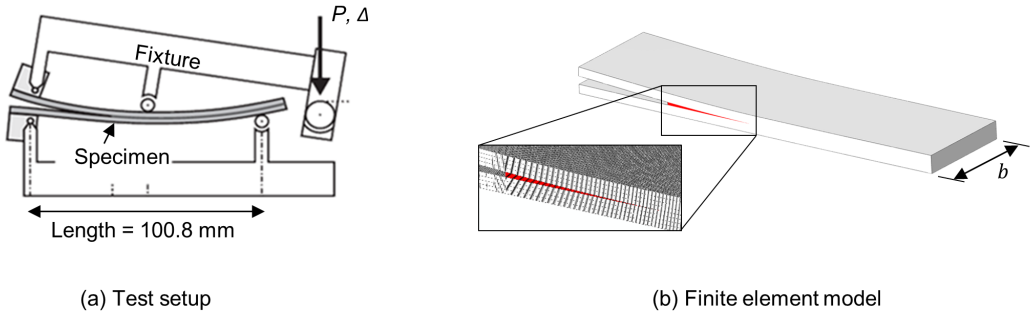


Figure 1. Schematics of the mixed-mode bend specimen.

4.5 mm and the initial delamination length is 25.4 mm. The test fixture shown in Fig. 1a is modeled with multi-point constraints and the specimen is modeled with 3-D solid continuum elements (C3D8). The fracture interface is modeled with a layer of cohesive element (COH3D8), colored in red in Fig. 1b. The typical element size was 0.127 mm. The analysis was run using double precision. Material properties for IM7/8552 carbon/epoxy were used [13]. The model included one step where prescribed displacements were applied using a smooth step amplitude through a duration of one second. Automatic mass scaling was used with a target time increment of 5×10^{-7} such that the total analysis required 2×10^6 increments.

The model is parametric so that the specimen geometry and mesh size can be easily adjusted. Several versions of the model with different widths were analyzed to vary the total number of DoF. The width, b, was selected as the parameter to adjust to change the problem size since the structural response and problem size vary predictably as a function of width. For $b = 25.4$ mm, the model contains 2.7×10^6 DoF and produced a 41 GB output database.

The typical load vs. displacement response produced by the model is shown in Fig. 2 along with the analytical solution [14]. Some vibrations are evident in the post-peak response predicted by the finite element model. The stiffness, strength, and post-peak response are in good agreement. All analysis runs were checked to verify the load-displacement response remained in good agreement with the closed-form analytical solution and thus unaffected by the numerical settings (e.g., number of cores).

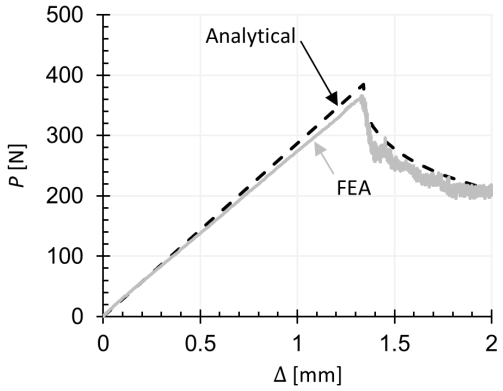
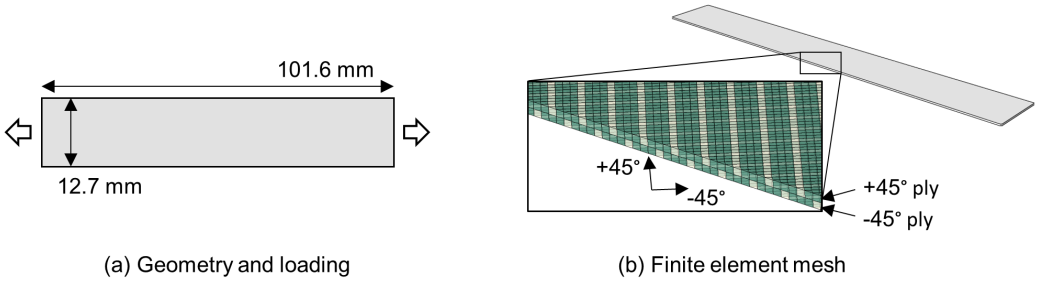


Figure 2. Typical mixed-mode bend load vs. displacement response.

3.2 $\pm 45^\circ$ laminate (PM45)

The $\pm 45^\circ$ laminate (PM45) was selected for a more complex damage process with interaction of multiple damage modes and use of a user-subroutine to assess the impact of these features on computational scaling performance. The specimen is modeled after the ASTM in-plane shear characterization test specimen [15] but is simplified in that it is a reduced size and contains only two plies. An illustration of the model showing the dimensions and the mesh is shown in Fig. 3. The PM45 model includes two plies, $+45^\circ$ and -45°, each modeled with one layer of 3-D continuum solid elements with reduced integration (C3D8R). The CompDam constitutive model is used to model matrix cracks in the two plies. CompDam is a model for progressive damage analysis of laminate composites that is implemented as a VUMAT for use with Abaqus/Explicit [11]. The mesh is fiber aligned with a typical element size of 0.2 mm for a total of about 4×10^5 DoF. A minimum spacing between matrix cracks is enforced using an alternating section definition where matrix cracking is enabled in every fourth row of elements (shown by the colors in Fig. 3) [16]. In the interface between the two plies, a layer of cohesive elements (COH3D8) is used to model delamination. The combination of matrix cracking modeled using CompDam and delamination modeled using cohesive elements is intended to capture the experimentally observed failure process where matrix cracks are linked by a delamination which results in two-piece failure.

The PM45 model was run with the same material properties and mass scaling parameters as the MMB. Parameters used for loading and output requests were also similar to those used for the MMB. The PM45 model produces a linear load vs. displacement response with an abrupt drop when two-piece failure is predicted. Several matrix cracks and extensive delamination are predicted. While a nonlinear shear stress vs. shear strain response is typically observed in tests using $\pm 45^\circ$ laminates, shear nonlinearity was neglected in the model used in this study.

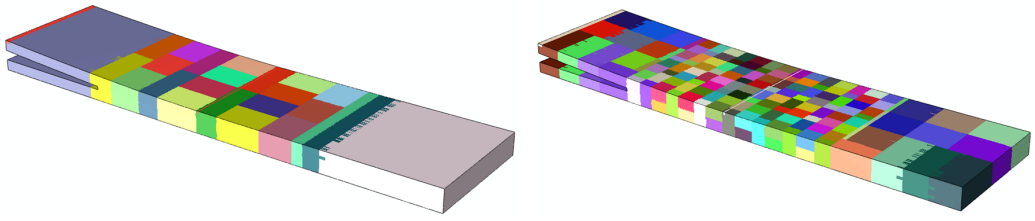


(a) Geometry and loading (b) Finite element mesh

Figure 3. Schematics of the ±45° specimen. In (b), white elements have matrix cracking enabled; green elements are linear elastic.

3.3 Domain decomposition

The basic method by which the solution is obtained in parallel is domain decomposition [9]. Using domain decomposition, the model is divided into a user specified number of parallel domains. The solution is obtained independently for each domain and information is passed to ensure continuity and equilibrium at the domain boundaries. The domains are divided initially so that each has approximately equivalent computation expense. Two examples of the division of the MMB model into domains are shown in Fig. 4. Changing the number of domains used in the solution procedure may affect the solution (as discussed in section 5) and the run time. The effect on run time is due to disparity in computational expense among the domains, with the most costly domain determining the run time. To address this effect, a dynamic load balancing feature is available in Abaqus/Explicit where, when the number of domains is set as a multiple of the number of cores, the solver periodically re-balances the domains among the cores so that computational expense for each core is similar. The dynamic load balancing feature was not used in this investigation. All runs were conducted using one domain per core.



(a) $n_{\text{cores}} = 24$ (b) $n_{\text{cores}} = 224$

Figure 4. Parallel domains shown with a different color for each parallel domain for two different decompositions.

4 Computational Performance

The performance in terms of absolute run time as a function of the numbers of cores used in the analysis is described in this section. Results are shown for the MMB model.

The MMB model was analyzed in a configuration with 2.7×10^6 DoF on all compute systems, with a varying number of cores, n_{cores}. The n_{cores} used for each run was selected to prefer using all cores per node whenever possible. The run time, T, is plotted as a function of number of cores in Fig. 5. One curve is shown for each of the five compute systems with the exception of K4 for which two curves are shown. Runs were conducted on K4 using all 40 cores per node (K4_40) and with 20 cores per node (K4_20) to assess the effect of using fewer cores per node than are available. The results show generally substantial reductions in run time with increasing numbers of cores. Performance improvements with each generation of CPU are also evident. Comparing the results from the shared memory system (H) with the results from the distributed memory systems, there is no meaningful difference in performance. Interestingly, the distributed memory system K3H, which uses the same generation of CPU as the shared memory system H, slightly outperformed the shared memory system in terms of absolute run time using the same number of cores. A notable improvement in performance was obtained by using 20 cores per node on K4 as compared with using all available cores (40) per node. While in most cases, adding additional cores reduced the run time, in a few cases, the opposite occurred. Overall, the results suggest that substantial reductions in run time are achievable using the most recent CPU generation and as many cores as possible.

Improvement in run time performance for the four generations of CPU chips represented by the distributed memory machines was quantified and is shown in Fig. 6. Power law fits were used to determine the expected run time for each HPC system on 64, 128, and 256 cores. The results are plotted as a function of acquisition year in Fig. 6. The reduction in run time performance per year is 8.5% on average. While not insignificant, the reduction in run time from newer hardware is relatively small compared with the speed up from parallelization.

Variability in run time for identical analyses run on the same system was quantified. On K3H, four runs of the MMB model with 2.7×10^6 DoF on 96 cores produced a standard deviation of 34 minutes and a coefficient of variation of 2.8%. On K3S, six runs of the MMB model with 1.5×10^5 DoF on 16 cores produced a standard deviation of 5 minutes and a coefficient of variation of 1.7%.

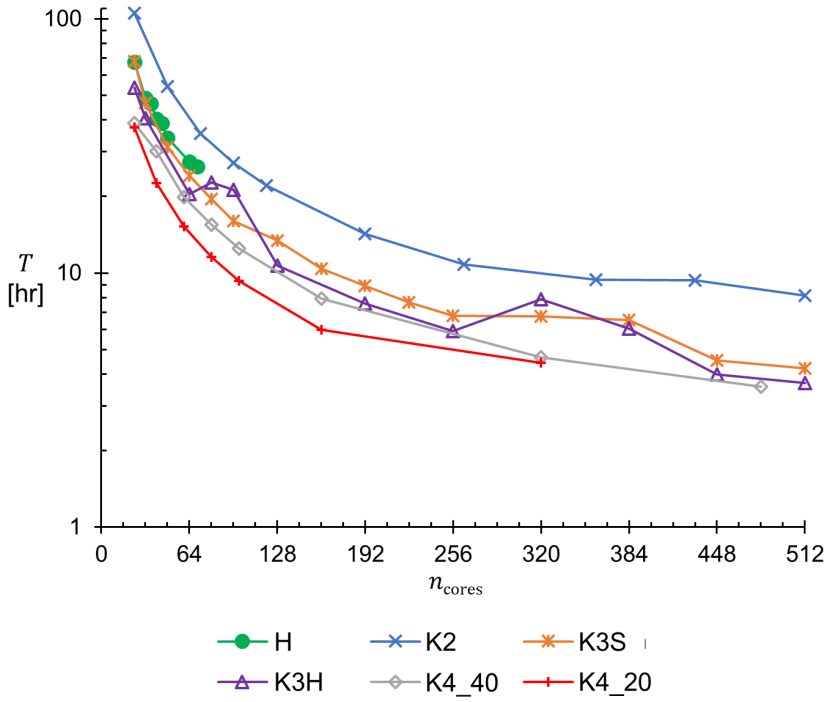


Figure 5. Run times for the MMB on each compute system.

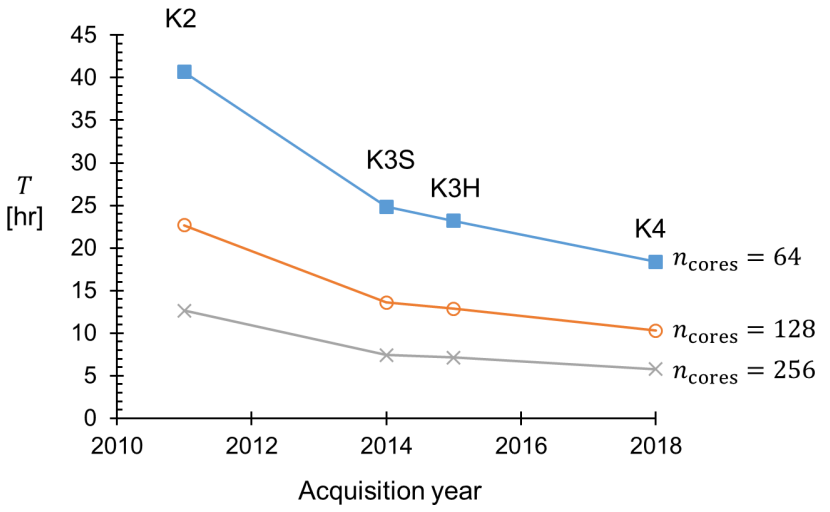


Figure 6. Estimated run times for the MMB as a function of hardware acquisition year.

5 Scalability

The scalability was assessed for the MMB with 2.7×10^6 DoF by using a speed up factor, S, calculated as

$$S = \frac{T_{24}}{T_{n_{\text{cores}}}} \tag{1}$$

where T_{24} is the run time using 24 cores and $T_{n_{\text{cores}}}$ is the run time using n_{cores}. The number of cores used in the numerator in eq. (1) is arbitrarily chosen as 24 since this is a common number of cores used for this class of analysis. The speed up as a function of the number of cores is shown in Fig. 7 for the MMB results from each system. The results are shown along side ideal scaling, S_{ideal}

$$S_{\text{ideal}} = \frac{n_{\text{cores}}}{24} \tag{2}$$

The results show nearly ideal scaling performance up to about 256 cores for most systems. Between 256 and 512 cores, the scaling performance deteriorates from the ideal case showing smaller incremental performance gains. No significant differences in scaling performance were observed across the different machines. While the scaling performance is less than ideal using large numbers of cores, the trends show ever increasing speed up, never completely flattening out. The results show that speed up factors of 10 to 15 times are achievable. From a practical point-of-view, speed up of 10 to 15 times is significant in terms of allowing many more cases to be analyzed or making much larger problems tractable.

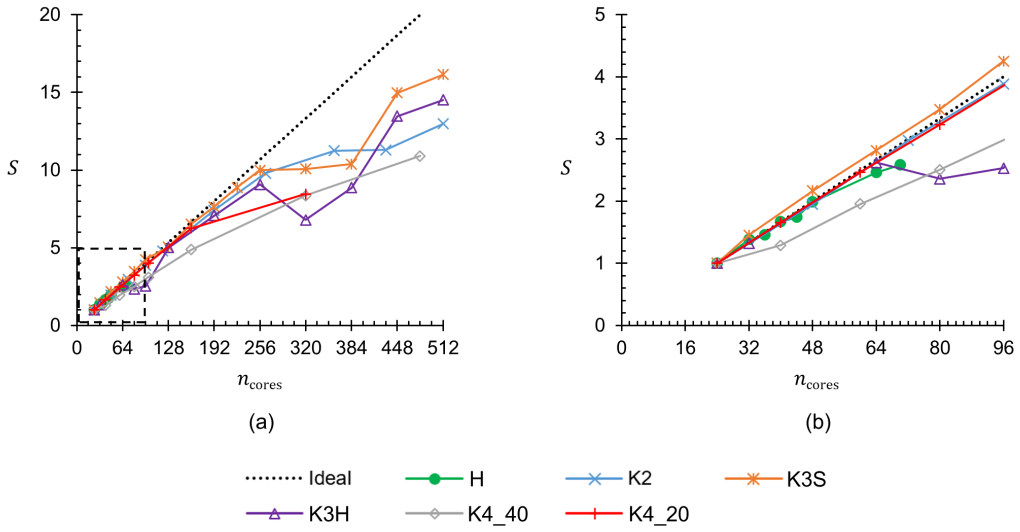


Figure 7. Speed up factor for the MMB on each compute system with (b) showing a zoomed-in view of (a).

A similar result was found for the ±45° laminate, as shown in Fig. 8. As discussed in the next paragraph, some differences in the results were observed, especially in

the damage progression, after the peak load was reached. Therefore, the run time was defined as the time required to reach the peak load. The scaling performance is similar to the MMB, where the trend follows the ideal scaling curve. In general, the performance of the three hardware systems is similar with slightly better performance observed using KS3.

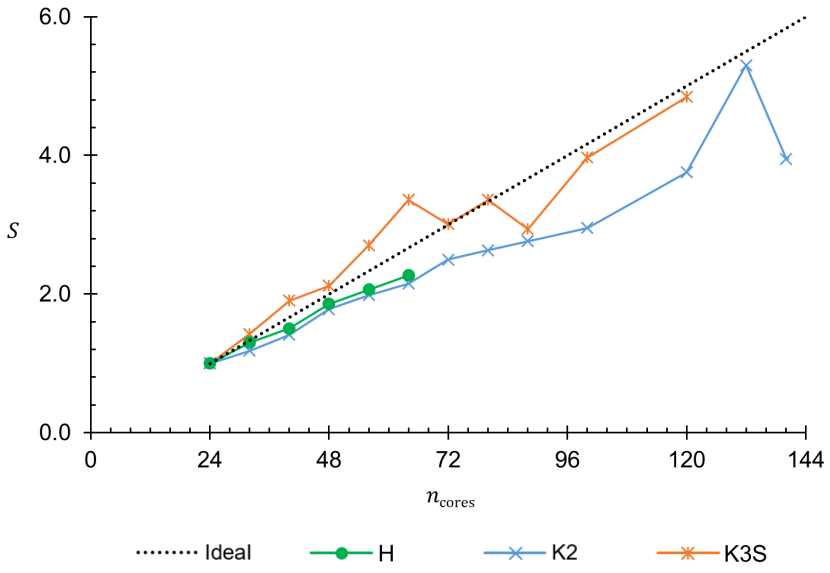


Figure 8. Speed up factor for the PM45 on three compute systems.

Since the cross section and material properties of the ±45° laminate are uniform along the length of the specimen, the stress field is nominally uniform and the predicted location of damage initiation is sensitive to small perturbations in the stress field. As the number of cores is changed, the model is decomposed differently, and hence small differences are introduced into the stress field. As a result, the model predicts different locations of damage for nominally identical analyses where the only difference is the number of cores used in the analysis as shown in Fig. 9. This difference in solution pertains only to the location of the damage and is a direct consequence of the modeling approach. The damage pattern and structural load vs. displacement response up to the peak load are very similar in all cases. For example, the peak loads predicted for the three cases shown in Fig. 9 were within 0.1% of the average peak load. Nonetheless, analysts must be aware of the possibility that the domain decomposition procedure used for parallelization may affect the solution. The effect of domain decomposition on the solution is most significant in cases such as the one discussed here where bifurcation of the solution path is likely in the presence of small perturbations in the stress field.

As the number of compute nodes is increased, at some point the computational expense of passing information between the nodes outweighs the benefits of adding additional nodes. A useful metric in evaluating this trade-off is the number of DoF per core, λ. Since the domain decomposition produces domains with different

Figure 9. Matrix cracking damage variable showing three different cracking patterns for three different number of cores.

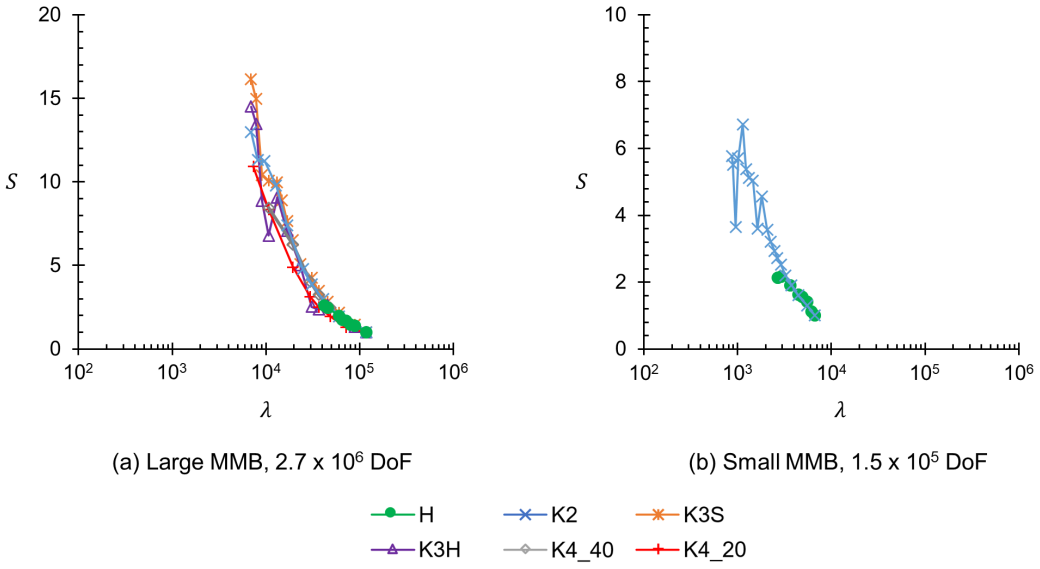


Figure 10. Speed up as a function of the number of degrees of freedom per core, λ, for two versions of the MMB model.

numbers of DoF, the average DoF per core is reported for λ. The speed up as a function of λ for two versions of the MMB model is shown in Fig. 10. In Fig. 10a, the number of cores is increased from 24 up to 512 so that λ is decreased from 1.2×10^5 to 3.5×10^3. The results show no clear indication of eroding performance as λ is decreased. Also, the results show consistency across the different machines. A reduced-width version of the MMB with fewer DoF (1.5×10^5) was used to investigate smaller values of λ. The results for the small MMB, shown in Fig. 10b, suggest that when $\lambda < 2 \times 10^3$ performance gains achieved with additional cores are minimal. Therefore, good scaling performance can be expected with $\lambda > 2 \times 10^3$.

6 Empirical Estimates for Run Time

It is beneficial to be able to estimate the runtime of a model of a given size. Having an estimate for run time allows model builders to use the maximum model size possible with the available compute resources and schedule constraints. Although analyst experience is useful in estimating run times for small jobs, the extrapolation to large jobs is not straightforward since the run time performance scales nonlinearly. Therefore, two series of analyses were conducted on K3S in order to determine an empirical expression for run time as a function of λ and n_{cores}. Two sets of analyses were conducted using the MMB model and 2×10^6 increments, which is the maximum number of increments that should be used in a double-precision Abaqus/Explicit analysis. Thus the estimates for run time provided here will be high in cases were fewer increments are used.

The first set of analyses was conducted on one node ($n_{\text{cores}} = 16$). The model size was scaled from 5×10^4 DoF to 1.6×10^6 DoF, or, equivalently, 3×10^3 DoF/core to 1×10^5 DoF/core. The run time for an explicit analysis is expected to scale linearly with the problem size and the results were nearly linear, as shown in Fig. 11. The run time on 16 cores is approximated with

$$T_{16} = 0.55\lambda \tag{3}$$

which is included in Fig. 11 to show the agreement with the recorded run times.

The second set of analyses assessed the performance penalty when the model size and n_{cores} are increased at the same time. In other words, the model size is increased while λ is held constant. The results for $\lambda = 5 \times 10^3$ DoF/core and $\lambda = 1 \times 10^4$ DoF/core are shown in Fig. 12. The absissca is the run time normalized to the run time on one node. If the simulation time scaled perfectly, increasing the number of cores would always yield $T/T_{16} = 1$. The results show that a performance penalty exists as the model size and number of cores are increased proportionally. The penalty, less than two for all cases that were considered, is relatively small and consistent with the results in section 5. The scaling penalty is slightly lower for $\lambda = 1 \times 10^4$ DoF/core. The expression

$$\frac{T}{T_{16}} = 0.134\ln(n_{\text{cores}}) + 0.733 \tag{4}$$

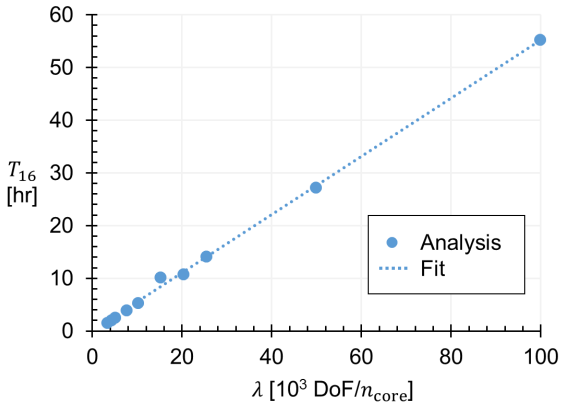


Figure 11. Run time scaling with problem size on one node.

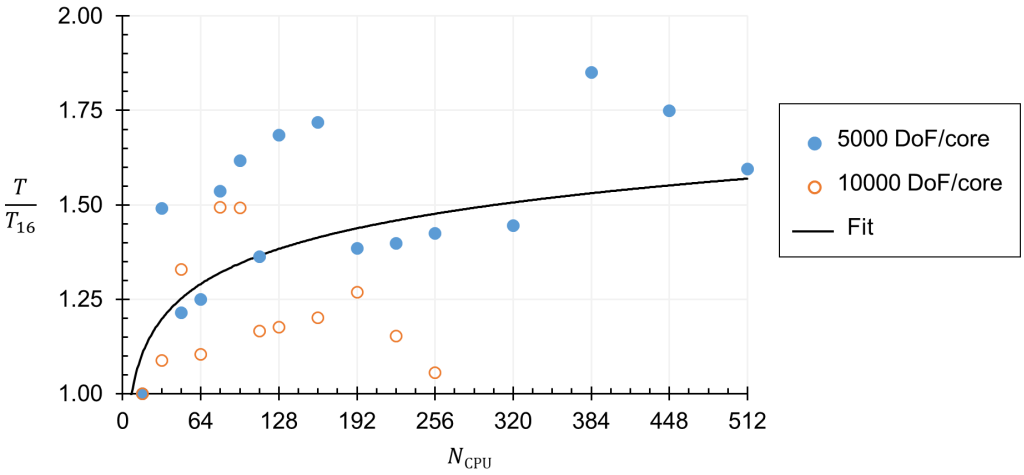


Figure 12. Run time scaling with λ constant.

was fit to the results using a least squares fitting procedure. Equation (4) can be utilized to obtain the performance penalty as the model size increases. For example, using eq. (4), it can be determined that an analysis with $n_{\text{cores}} = 512$ is 57% slower than ideal linear scaling would suggest. Combining eq. (3) and eq. (4), an expression is obtained for run time in terms of λ and n_{cores}

$$T = 0.55\lambda(0.134\ln(n_{\text{cores}}) + 0.733) \tag{5}$$

Equation (5) can be rearranged to determine the maximum model size possible for a given n_{cores} and T

$$DoF = \frac{Tn_{\text{cores}}}{0.55(0.134\ln(n_{\text{cores}}) + 0.733)} \tag{6}$$

Estimated model sizes obtained using eq. (6) for several combinations of n_{cores} and T are provided in Table 2. The data show that model size can be increased by a factor of 20 when scaling from 16 to 512 cores while maintaining the same run time. The estimates for $n_{\text{cores}} = 1024$ are extrapolated and are shown to motivate future investigation into the scalability in this regime.

Table 2: Estimated for model DoF (1×10^6) for combinations of n_{cores} and T.

	n_{cores}						
T [hr]	16	32	64	128	256	512	1024
8	0.21	0.39	0.72	1.35	2.52	4.75	8.96
24	0.63	1.17	2.16	4.04	7.57	14.24	26.89
48	1.26	2.33	4.33	8.08	15.13	28.48	53.77
72	1.90	3.50	6.49	12.11	22.70	42.71	80.66

7 Analysis costs

Although the computational capability may exist to analyze very large models, the cost of the analysis limits the model size in practice. The costs for compute time and the cost for Abaqus tokens are quantified in this section and compared with the run time performance to elucidate run time vs. cost trade-offs.

Analyses conducted with Abaqus use a token system for licensing where the number of tokens required for an analysis varies with n_{cores} as

$$n_{\text{tokens}} = \text{INT}(5(n_{\text{cores}})^{0.422}) \tag{7}$$

Therefore, the license cost, C_a, of a particular analysis is the product of the number of tokens n_{tokens} and the run time T

$$C_a = Tn_{\text{tokens}} \tag{8}$$

Token usage from the MMB with 2.7×10^6 DoF held constant and varying n_{cores} are shown in Fig. 13. The token usage decreases when additional cores are used

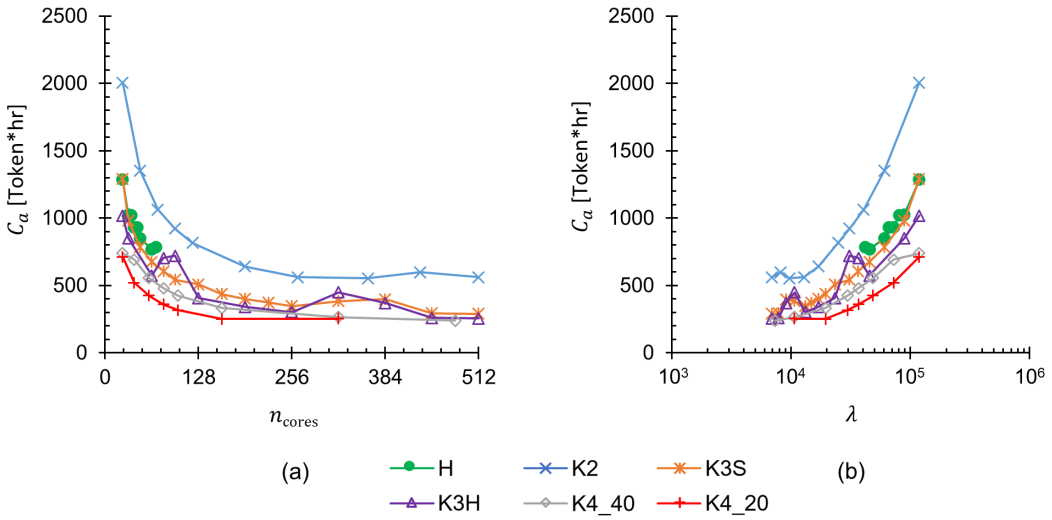


Figure 13. Token usage as a function of (a) n_{cores} and (b) λ.

since the run time scales faster than the token usage. There is a limit around 2×10^3 DoF/core where additional cores do not substantially decrease token usage. The token usage appears asymptotic with increasing n_{cores}; no local minimum is observed. In most cases, using additional cores reduces token usage. These results imply that it is preferable to run multiple analysis in series with all available cores as opposed to parallel execution with a subset of cores allocated to each analysis. Faster (newer) computation systems (K4) reduce token usage compared with slower systems (K2).

The cost for the compute time should also be considered. The compute cost is a function of both the run time and the operational cost of the hardware. The NASA Advanced Supercomputing (NAS) standard billing unit (SBU) is used to quantify the compute cost across different machines. The SBU is a means of normalizing computing time across different architectures, where a conversion factor f is assigned to each compute architecture to account for the differences in performance. Using this framework, the compute cost C_c for a job is calculated as

$$C_c = n_{\text{nodes}} T f r_{\text{SBU}} \tag{9}$$

where r_{SBU} is the cost per SBU. Each year, the NAS assigns the cost for an SBU in dollars. In FY2018, the cost was \$0.16/SBU [17]. The conversion factors f assumed for the hardware used in this study are summarized in Table 3. Using eq. (9), the compute cost is shown as function of n_{cores} for the MMB with 2.7×10^6 DoF in Fig. 14 for the distributed memory systems. The shared memory system is omitted since no conversion factor was available. The results show that the compute cost increases only slightly with n_{cores} in all cases. Since the compute cost is a function of n_{nodes}, there is a relatively significant cost penalty if using half the available cores per node as in the case of K4_20. Since the compute cost increases while the license cost decrease with increasing n_{cores}, apparently an optimal n_{cores} can be determined.

Table 3: Conversion factors f.

K2	K3S	K3H	K4
1.0	1.82	1.82	6.36

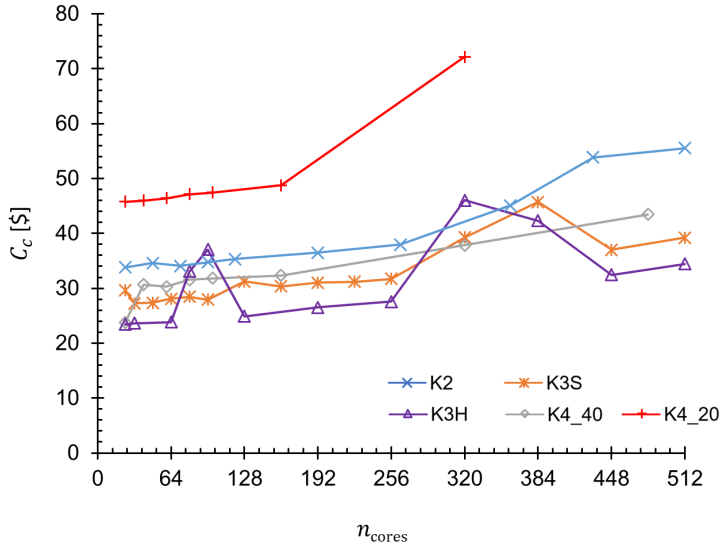


Figure 14. Compute cost as a function of n_{cores} for the MMB model.

8 Conclusions

The computational scaling performance of Abaqus/Explicit was evaluated by conducting several analysis using varying number of central processing unit (CPU) cores. By comparing run times across different numbers of cores, hardware, and models, the computational scaling performance was quantified. Five different high performance computing (HPC) systems were used and analyses were run using between 16 and 512 cores. The HPC systems included both shared and distributed memory architectures. The range of hardware used spans a decade in terms of the CPU generations which provides a sense for the run time improvements with each new generation of CPU chip.

Two models were analyzed: the first model was a mixed-mode bend specimen with a single delamination interface represented by cohesive elements and the second model was a $\pm 45°$ laminate with interlaminar and intralaminar damage interactions. In both cases, the models were configured in a fashion typical for progressive damage analysis of composite laminates. That is to say that the accuracy of the damage prediction was the first priority in the model setup; no special efforts were made to improve computation scaling performance. The models were chosen to represent a range of complexity in progressive damage prediction from a relatively simple case to a relatively complex case.

The results for run time as a function of the number of CPU cores showed signifi-

cant reductions in run time scaling up to 512 cores. Each new generation of hardware accelerated the run time by about 8.5% per year. Speed up of 10 to 15 times was typical for the different hardware architectures investigated in this study. Scaling performance was nearly ideal up to 256 cores. The scaling performance deteriorated but never fully halted as the number the cores was increased to 512. Although the computational scaling performance is problem dependent, similar performance was found for the two models analyzed in this study.

While good computational scaling performance is highly desirable, the cost cannot be ignored. As the number of cores is increased, at some point the performance may deteriorate due the expense of passing information outweighing the benefits of additional cores. This trade-off was studied and good performance is obtained when the number of degrees of freedom per node is maintained above 2×10^3. Monetary costs in terms of license and HPC usage were quantified as well. The computational performance outpaced the requirement for additional licenses as the number of cores is increased. Therefore, license usage decreases when additional cores are used. The compute cost was found to increase modestly when additional cores are used.

Through characterizing parallel scaling performance as a function of model size, a simple empirical expression is obtained relating run time, number of CPU cores, and number of DoF. The results show that, for a given run time and ratio of degrees of freedom per core, the model size can be increased by a factor of 20 scaling from 16 to 512 cores. This expressions allow analysts to design models appropriately sized for the available computational resources and acceptable run times.

References

1. Engelstad, S. P. and Clay, S. B., "Comparison of Composite Damage Growth Tools for Static Behavior of Notched Composite Laminates," *Journal of Composite Materials*, Vol. 51, No. 10, 2017, pp. 1493–1524.

2. Leone Jr., F. A., Dávila, C. G., Mabson, G. E., Ramnath, M., and Hyder, I., "Fracture-Based Mesh Size Requirements for Matrix Cracks in Continuum Damage Mechanics Models," *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Grapevine, TX, Jan. 2017.

3. Bergan, A. C., Dávila, C. G., Leone Jr., F. A., Awerbuch, J., and Tan, T.-M., "An Analysis Methodology to Predict Damage Propagation in Notched Composite Fuselage Structures," *SAMPE Baltimore*, Baltimore, MD, May 2015.

4. Dávila, C. G., Leone Jr., F. A., Song, K., Ratcliffe, J. G., and Rose, C., "Material Characterization for the Analysis of Skin/Stiffener Separation," *American Society for Composites 32nd Technical Conference*, West Lafayette, Indiana, Oct. 2017.

5. Gigliotti, L. and Pinho, S. T., "Multiple Length/Time-Scale Simulation of Localized Damage in Composite Structures Using a Mesh Superposition Technique," *Composite Structures*, Vol. 121, 2015, pp. 395–405.

6. SIERRA Solid Mechanics Team, "Sierra/SolidMechanics 4.22 User's Guide," Tech. rep., SAND2011-7597, Sandia National Lab, Albuquerque, NM, 2011.

7. Casoni, E., Jérusalem, C., Eguzkitza, B., Lafortune, P., Tjahjanto, D. D., Sáez, X., Houzeaux, G., and Vázquez, M., "Alya: Computational Solid Mechanics for Supercomputers," *Archives of Computational Methods in Engineering*, Vol. 22, No. 4, 2015, pp. 557–576.

8. Warner, J. E., Bomarito, G. F., Heber, G., and Hochhalter, J. D., "Scalable Implementation of Finite Elements by NASA _ Implicit (ScIFEi)," Tech. rep., NASA/TM2016-219180, NASA Langley Research Center, Hampton, VA, 2016.

9. Simulia, *Abaqus Analysis User's Guide. Version 2017*, 2017.

10. Eldred, L. B., "Advanced Composites Project Overview: Advanced Air Vehicles Program," *TCC Fall 2015 Technical Meeting*, Columbia, SC, Sep. 2015.

11. Leone, F., Bergan, A. C., and Dávila, C. G., "CompDam - Deformation Gradient Decomposition (DGD)," 2018, https://github.com/nasa/CompDam_DGD.

12. "ASTM Standard D6671. Test Method for Mixed Mode I-Mode II Interlaminar Fracture Toughness of Unidirectional Fiber-Reinforced Polymer Matrix Composites," *Annual Book of ASTM Standards*, ASTM Int., 2006.

13. Wanthal, S., Schaefer, J. D., Justusson, B., Hyder, I., Englestad, S., and Rose, C., "Verification and Validation Process for Progressive Damage and Failure

Analysis Methods in the NASA Advanced Composites Consortium," *American Society for Composites 32nd Technical Conference*, West Lafayette, Indiana, Oct. 2017.

14. Turon, A., Cammanho, P. P., Costa, J., and Renart, J., "Accurate Simulation of Delamination Growth Under Mixed-Mode Loading Using Cohesive Elements: Definition of Interlaminar Strengths and Elastic Stiffness," *Composite Structures*, Vol. 92, No. 3, 2010, pp. 1857–1864.

15. "ASTM Standard D3518. Standard Test Method for In-Plane Shear Response of Polymer Matrix Composite Materials by Tensile Test of a ±45° laminate," *Annual Book of ASTM Standards*, ASTM Int., 2013.

16. Hyder, I., Leone Jr., F. A., Justusson, B., Schaefer, J. D., Bergan, A. C., and Wanthal, S., "Implementation of a Matrix Crack Spacing Parameter in a Continuum Damage Mechanics Finite Element Model," *American Society for Composites 33rd Technical Conference*, Seattle, Washington, Sep. 2018.

17. Lee, T. and Cohen, J., "High-End Computing Program: Standard Billing Units," 2018, https://www.hec.nasa.gov/user/policies/sbus.html.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-02-2019	Technical Memorandum	

4. TITLE AND SUBTITLE
Computational Performance of Progressive Damage Analysis of Composite Laminates using Abaqus/Explicit with 16 to 512 CPU Cores

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Bergan, Andrew C.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
826611.04.07.01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, Virginia 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER
L–20998

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM–2019–220251

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 24
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://ntrs.nasa.gov.

14. ABSTRACT

The computational scaling performance of progressive damage analysis using Abaqus/ Explicit is evaluated and quantified using from 16 to 512 CPU cores. Several analyses were conducted on varying numbers of cores to determine the scalability of the code on five NASA high performance computing systems. Two finite element models representative of typical models used for progressive damage analysis of composite laminates were used. The results indicate a 10 to 15 times speed up scaling from 24 to 512 cores. The run times were modestly reduced with newer generations of CPU hardware. If the number of degrees of freedom is held constant with respect to the number of cores, the model size can be increased by a factor of 20, scaling from 16 to 512 cores, with the same run time. An empirical expression was derived relating run time, the number of cores, and the number of degrees of freedom. Analysis cost was examined in terms of software tokens and hardware utilization. Using additional cores reduces token usage since the computational performance increases more rapidly than the token requirement with increasing number of cores. The increase in hardware cost with increasing cores was found to be modest. Overall the results show relatively good scalability of the Abaqus/Explicit code on up to 512 cores.

15. SUBJECT TERMS
Abaqus/Explicit; Computational scaling; Cohesive elements; Continuum damage mechanics

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Information Desk (help@sti.nasa.gov)
U	U	U	UU	26	19b. TELEPHONE NUMBER *(Include area code)* (757) 864-9658