HPC Considerations for Scalable Multidiscipline CAE Applications on Conventional Linux Platforms

Author:
Stan Posey
Panasas, Inc.

Correspondence:
Stan Posey
Panasas, Inc.
Phone +510 608 4383
Email sposey@panasas.com

ABSTRACT:

This paper examines HPC workload efficiencies for sample multidiscipline LS-DYNA applications on a conventional HPC Linux platform with proper balance for I/O treatment. Model parameters such as size, element types, schemes of implicit and explicit (and coupled), and a variety of simulation conditions can produce a wide range of computational behavior and I/O management requirements. Consideration must be given to how HPC resources are configured and deployed, in order to satisfy growing LS-DYNA user requirements for increased fidelity from multidiscipline CAE.

Keywords:
High Performance Computing (HPC), Linux Clusters, Parallel I/O, Storage System

INTRODUCTION

Manufacturing industry and research organizations continue to increase their investments in structural analysis and impact simulations such that the growing number of LS-DYNA users continues to demand more from computer system HPC resources. These LS-DYNA workload demands typically include rapid single job turnaround and multi-job throughput capability for users with diverse application requirements in a high-performance computing (HPC) infrastructure.

Additional HPC complexities arise for many LS-DYNA environments with the growth of multidiscipline CAE coupling of structural and CFD analyses, that all compete for the same system resources. Such requirements also drive I/O levels that prevent most system architecture's ability to scale. Yet for today's economics of HPC, the requirements of CPU cycles, large memory, system bandwidth and scalability, I/O, and file and data management – must be satisfied with high levels of productivity from conventional systems based on scalable, inexpensive Linux clusters.

This paper examines HPC workload efficiencies for sample multidiscipline LS-DYNA applications on a conventional HPC Linux platform with proper balance for I/O treatment. Model parameters such as size, element types, schemes of implicit and explicit (and coupled), and a variety of simulation conditions can produce a wide range of computational behavior and I/O management requirements. Consideration must be given to how HPC resources are configured and deployed, in order to satisfy growing LS-DYNA user requirements for increased fidelity from multidiscipline CAE.

GENERAL HPC CHARACTERISTICS OF LS-DYNA

Finite element analysis software LS-DYNA™ from Livermore Software Technology Corporation (www.lstc.com) is a multi-purpose structural and fluid analysis software for high-transient, short duration structural dynamics, and other multi-physics applications. Considered one the most advanced nonlinear finite element programs available today, LS-DYNA has proved an invaluable simulation tool for industry and research organizations who develop products for automotive, aerospace, power-generation, consumer products, and defense applications, among others.

LS-DYNA simulations for the automotive industry include vehicle crash and rollover, airbag deployment and occupant response. For the aerospace industry, LS-DYNA has ability to simulate bird impact on airframes and engines and turbine rotor burst containment, among others. Additional complexities arise from simulations of these classes since they often require predictions of surface contact and penetration, modeling of loading and material behavior, and accurate failure assessment.

From a hardware and software algorithm perspective, there are roughly three types of LS-DYNA simulation "behavior" to consider: implicit and explicit FEA for structural mechanics, and computational fluid dynamics (CFD) for fluid mechanics. Each discipline and associated algorithms have their inherent complexities with regards to efficiency and parallel performance, and also regarding modeling parameters.

The range of behaviors for the three disciplines that are addressed with LS-DYNA simulations, highlights the importance of a balanced HPC system architecture. For example, implicit FEA for static load conditions requires a fast processor and high-bandwidth I/O subsystem for effective turnaround, in contrast to dynamic response, which requires high rates of memory and I/O bandwidth with processor speed as a secondary concern. In addition, FEA modeling parameters such as the size, the type of elements, and the load condition of interest all affect the execution behavior of implicit and explicit FEA applications.

Explicit FEA benefits from a combination of fast processors for the required element force calculations and a high rate of memory bandwidth necessary for efficient contact resolution that is required for nearly every structural impact simulation. CFD also requires a balance of memory bandwidth and fast processors, but benefits most from parallel scalability. Each discipline has inherent complexities with regard to efficient parallel scaling, depending upon the particular parallel scheme of choice. In addition, the I/O associated with result-file checkpoint writes for both disciplines, and increasing data-save-frequency by users, must also scale for overall simulation scalability.

Implementations of both shared memory parallel (SMP) and distributed memory parallel (DMP) have been developed for LS-DYNA. The SMP version exhibits moderate parallel efficiency and can be used with SMP computer systems only, while the DMP version, exhibits very good parallel efficiency. This DMP approach is based on domain decomposition with MPI for message passing, and is available for homogenous compute environments such as shared memory parallel systems or clusters.

Most parallel MCAE software employ a similar DMP implementation based on domain decomposition with MPI. This method divides the solution domain into multiple partitions of roughly equal size in terms of required computational work. Each partition is solved on an independent processor core, with information transferred between partitions through explicit message passing software (MPI) in order to maintain the coherency of the global solution.

LS-DYNA is carefully designed to avoid major sources of parallel inefficiencies, whereby communication overhead is minimized and proper load balance is achieved. Another advantage for LS-DYNA is the use of an MPI that is shared-memory-aware and transparent to the user. This MPI further reduces communication overhead when

scaling to a large numbers of cores on a single node, which is achieved by a reduction in bandwidth and latency that is improved over public-domain MPI such as MPICH.

A NOVEL PARALLEL I/O AND STORAGE APPROACH

Currently, there are two types of network storage systems, each distinguished by its command sets. First is the SCSI block I/O command set, used by storage area networks (SAN), which provides high random I/O and data throughput performance via direct access to the data at the level of the disk drive or fibre channel. Network attached storage (NAS) systems use protocols such as NFS or CIFS command sets for accessing data with the benefit that multiple nodes can access the data as the metadata (describes where the data exists) on the media is shared. To achieve the high-performance and data-sharing benefits that Linux clusters can provide requires a fundamentally new storage design, one that can offer both the performance benefits of direct access to disk and the easy administration provided by shared files and metadata. That new storage design is an object-based storage architecture.

Object storage offers virtually unlimited growth in capacity and bandwidth, making it well-suited for handling large results-data generated by LS-DYNA simulations on Linux clusters. Unlike conventional storage systems, data is managed as large virtual objects. An object is a combination of application (file) data and storage attributes (metadata) that define the data. Managing data as objects, as opposed to traditional storage blocks, means that files can be divided into separate pieces. Such object storage blocks are then distributed across storage media known as object-based storage devices (OSDs). So just as the Linux clusters spread the work evenly across compute nodes for parallel processing, the object-based storage architecture allows data to be spread across OSDs for parallel access. It is massively parallel processing on the front end, matched by massively parallel storage on the back end.

Such an architecture delivers substantial benefits. By separating the control path from the data path, file system and metadata management capabilities are moved to the nodes in the Linux cluster, and provided nodes with direct access to storage devices. By doing so, OSDs autonomously serve data to end-users and radically improve data throughput by creating parallel data paths. Instead of pumping all information through one path, which creates major bottlenecks as data size and number of nodes increase, Linux cluster nodes can securely read and write data objects in parallel to all OSDs in the storage cluster system.

With object-based storage, the Linux compute cluster has parallel and direct access to all of the data spread across the OSDs within the shared storage. The large volume of data is therefore accessed in one simple step by the Linux cluster for computation. While the simulation and visualization data may still need processing for weeks at a

time, the object model of storage drastically improves the amount, speed, and movement of data between storage and compute clusters.

PANASAS PARALLEL STORAGE CLUSTER

The Panasas Storage Cluster has been designed to provide the benefits of a clustered file system. The Panasas File System allows for a single namespace that can be shared across the entire pool of storage and load-balanced dynamically. This system provides the foundation for a single, shared storage infrastructure that can be fully leveraged and utilized for CAE work.

The key differentiation for Panasas from other RAID storage solutions lies in the software architecture. Panasas is leading the development of object-based storage. The core principle of this approach is that data is managed in large virtual objects and not as small blocks or files. This allows for parallel communications and I/O between cluster compute nodes and storage, eliminating the bottlenecks that invariably come with traditional storage architectures.

The Panasas architecture divides files into objects, which are logical units of storage and can be accessed with file-like methods such as open, close, read, and write. This approach is especially effective for large files. Objects have associated application data, attributes, and metadata. Each object is designed to be managed, grown, shrunk, deleted, and dynamically distributed across physical media. As these objects are distributed across storage devices, they can be accessed in parallel by the cluster compute nodes. Meanwhile, the management of the objects is done by a metadata manager. The object attributes allow the system to offload work from the traditional filer head or file server to the storage device.

Object storage enables two primary technological breakthroughs. First, since the system is able to offload work directly to the storage device instead of going through a central filer head or file server, the system is able to deliver parallel performance directly from disk. Secondly, since each object is injected with attributes as well as application data, it can be managed intelligently.

This architecture allows Panasas to scale with performance that grows with capacity. The scaling of performance with capacity is almost linear; Panasas does this with an architecture that uses finely tuned hardware components to optimize the software architecture's capabilities. Panasas DirectorBlades serve as a 'virtual filer' to scale and manage metadata growth and Panasas StorageBlades act as smart disk drives to scale and manage capacity growth.

PERFORMANCE OBSERVATIONS OF LS-DYNA

Performance and parallel efficiency of any CAE software has certain algorithm considerations that must be addressed. The fundamental issues behind parallel algorithm design are well understood and described in various research publications. For grid-based problems such as the numerical solution of partial differential equations, there are four main sources of overhead that can degrade ideal parallel performance: 1) non-optimal algorithm overhead, 2) system software overhead, 3) computational load imbalance, and 4) communication overhead.

Parallel efficiency for LS-DYNA is dependent upon among others, MPI latency, which is determined by both the specifics of system architecture and the implementation of MPI for that system. Since system architecture latency is determined by design of a particular interconnect, overall latency improvements can only be made to the MPI implementation. Modifications to the MPI software to ensure "awareness" of a specific architecture are a way to reduce the total latency and subsequently the communication overhead. Improvements to architectural-awareness of MPI is a common development for several system vendors.

Specifically for structural FEA simulations, they often contain a mix of materials and finite elements that can exhibit substantial variations in computational expense, which may create load-balance complexities. The ability to efficiently scale to a large number of processors is highly sensitive to load balance quality. For example, the crash worthiness of automotive vehicles exhibit these characteristics. An additional consideration for improved load balance is efficient use of the domain decomposer.

Evaluation of an vehicle's crashworthiness is currently the fastest growing application for CAE simulation in the automotive industry. Proper crash management and occupant safety is a mandate by local governments, but it's also viewed as a competitive advantage by many automotive developers. Performance improvements continue with LS-DYNA such that parallel scalability keeps pace with growing demand.

Similarly, an aerospace application for design of gas turbine engines for aircraft, has utilized the parallel scalability of LS-DYNA to reduce the time it requires to complete a 5M-element model for blade-out simulation.

Additional developments between LSTC and Panasas applications engineering include an enhanced I/O scheme that significantly improves overall model turnaround in a mix of LS-DYNA jobs in an HPC production environment. This development is particularly important in production environments that include other applications and disciplines

such as NVH and CFD that might request similar HPC resources as LS-DYNA during a multi-job throughput workload.

SUMMARY AND CONCLUSIONS

A discussion was provided on the HPC technology requirements of LS-DYNA applications, including characterizations of the performance behavior typical of LS-DYNA simulations on distributed memory clusters. Effective implementation of highly parallel LS-DYNA simulations must consider a number of features such as parallel algorithm design, system software performance issues, hardware communication architectures, and I/O design in the application software and file system.

Development of increased parallel capability will continue on both application software and hardware fronts to enable FEA modeling at increasingly higher resolutions. Examples of LS-DYNA simulations demonstrate the possibilities for highly efficient parallel scaling on the clusters in combination with clusters and the Panasas system.

LSTC and Panasas will continue to develop software and hardware performance improvements, enhanced features and capabilities, and greater parallel scalability to accelerate the overall solution process of LS-DYNA simulations. This alliance will continue to improve FEA modeling practices in research and industry on a global basis and provide advancements for a complete range of engineering applications.

REFERENCES

1. Gibson, G.A., R. Van Meter, "Network Attached Storage Architecture," Comm. of the ACM, Vol. 43, No 11, November, 2000.

2. LS-DYNA User's Manual Version 971, Livermore Software Technology Corporation, Livermore, 2005.