# Mid-range Computing

## Summary

Mid-range HPC computation is that which utilizes parallelism to solve larger problems than those which can be handled with desktop systems or campus-level clusters, but which do not utilize the large proportion (>25%) of capability HPC systems as described in the capability use case. This class of user is often in the early stages of scaling up their code from campus-level systems, or may be performing a data analysis task for which a large memory space and high-performance storage system is more important than an enormous number of processes. In addition, these jobs are often components of larger workflows, where the input to the job is the output of another task, or vice versa – this leads to an increased importance for data locality. The recent improvements in parallel and remote visualization techniques have led to the introduction of visualization users to this category as well.

Since these jobs tend to fit on a larger proportion of the systems in national cyberinfrastructures, and are more likely to be part of a workflow, metascheduling and co-scheduling are more likely to be of value to these users. In addition, the vast increase in available cores at the national scale has led to the introduction of “parameter sweeps” for mid-sized parallel jobs, where a whole set of problems with varying parameters may have computational demands which are high enough to require parallel and distributed processing. For this reason, there may be a need to launch many mid-sized jobs on one or more resources simultaneously or sequentially.

## Customers

As with other HPC and HTC use cases, this class of jobs has a wide range of users from many disciplines. The fact that this class of user is often relatively new to national-scale cyberinfrastructure also means that newer disciplines which have not historically utilized HPC resources are disproportionately represented in the mid-range HPC category. User HPC sophistication can range from low to high, and users may be comfortable with interfaces from web portals and gateways to shells and text-based batch job construction.

## Scenarios and applications

The Network for Earthquake Engineering Simulation (NEES) project is an NSF MRE (Microbiological Research Establishment) project that seeks to lessen the impact of earthquake and tsunami-related disasters by providing revolutionary capabilities for earthquake engineering research. A state-of-the-art network links world-class experimental facilities around the country, enabling researchers to collaborate on experiments, computational modeling, data analysis, and education. NEES currently has ~75 users across approximately 15 universities. These users use TeraGrid HPC and data resources for various structural engineering simulations using both commercial codes and research codes based on algorithms developed by academic researchers. Some of these simulations, especially those using commercial FEM codes such as Abaqus, Ansys, Fluent, and LS-Dyna, require moderately large shared memory nodes, such as the large memory nodes of Abe at the Pittsburg Supercomputing Center, but scale to only a few tens of processors using MPI. Large memory is needed because the whole mesh structure is required on each node due to the basic FEM algorithm used. Many of these codes have OpenMP parallelization, in addition to using MPI, and users mainly utilize shared memory nodes using OpenMP for pre/post processing. On the other hand, some academic codes, such as OpenSees which is tuned for specific material behavior, can scale well on many thousands of processors, including on Kraken at NICS, and Ranger at TACC. Due to the geographically distributed location of NEES researchers and experimental facilities, high bandwidth data transfer and data access are vital.

## Involved resources and production Grid infrastructure

This use case has a very broad set of potentially involved resources, including medium-sized clusters and smaller percentages of the largest capability systems, visualization clusters and other data-centric systems, high-performance parallel file systems and archives at one or more sites. In addition, users in this category may need to access a shared community data collection, both to retrieve input data sets before a job begins or to store their output with appropriate metadata and other provenance information once a job has completed. Some jobs may be able to run on any one of several X86-based Linux clusters in particular, while other jobs may have other specialized requirements such as the need for GPUs, shared-memory architectures, or specific data sets local to a computational resource.

The size of data associated with this use case may vary from very small, such as a set of parameters to be passed to the executable, or as large as some multiple of the in-memory size of the total task, for jobs that perform checkpointing.

This use case contains the following functional requirements.

**Resource access**: Users of mid-range computing applications typically submit jobs directly, may make use of workflow tools or may use gateways to submit jobs.

**Authentication, authorization, and accounting (AAA)**: These users require standard Unix and batch AAA as implemented at individual resource provider sites and following the policies those sites adhere to.

**Fault tolerance**: Some mid-range computing applications have built-in periodic checkpointing features to guard against loss of work in the event of a system crash. Some mid-range jobs may employ workflow restart capabilities for fault tolerance.

**Scheduling**: Scheduling the job consists of at least three logical phases: determining where the job can run based on resource and account requirements, selecting a resource on which to execute the job, and preparing the execution environment for executing the job (e.g., staging data, getting the binaries in place, etc.) Determining where the job may run may require an information system that describes execution environments, their capabilities, software installed, etc. The ability to start both single and “vector” batches of jobs should be supported.

**Advance scheduling**: Some mid-range computing applications may require special scheduling capabilities including pre-stage and post-stage data job scheduling or sequential job dependencies.

**Workflow**: Some mid-range computing applications have workflow components. Most of these workflows are managed by the users using homegrown tools or special workflow tools.

**Data storage**: The data sets produced by mid-range computing applications can range from small to quite large. The storage and archival resources available must be both sufficiently large to store such data sets as well as sufficiently fast to handle them in a timely manner

## Security considerations

Authentication to execution is usually necessary. Similarly, if the execution service is to access data elsewhere via another service, that service may require authentication as well. This in turn may require some form of delegation from the client initiating the set of jobs to execution services that will move data in and out of the execution location.

Besides the authentication, access control, and delegation issues, some users have data integrity concerns with their data. This affects both on-the-wire and (more importantly) on-disk storage of input data and results.

In regard to groups and virtual organizations, access to compute resources and data resources may require one or more of the following: individual authentication, authentication as a member of a group or virtual organization, or the assertion of some property or role. Similarly, jobs may need to be managed post-initiation by either the identity that started the job or members of a group or role, or both.

This use case may have a somewhat greater need for the ability to support campus-level authentication, as it is likely the primary use case representing beginning users.

## Performance considerations

Performance considerations generally are similar to those of the capability computing use case – i.e. parallel job execution time and I/O performance are likely to be the most important components of overall job performance. As this use case may include situations where tasks are interdependent, and is less likely to depend on special queueing arrangements such as those used to favor the largest jobs in a queue, the reliable performance of scheduling systems and/or metascheduling systems may be somewhat more important than in the capability use case.

## Usecase situation analysis and PGI expectations

This use case relies on the ability to create and manage jobs on remote resources. In this case, there seems to be very few functions that aren’t handled by existing BES, JSDL, and JSDL related specifications. Some amount of profiling may be desirable to achieve/encourage interoperability such as specifying mandatory supported data staging protocols and/or data staging protocol discovery for BES implementations.

From a security perspective, existing OGSA security profiles and specifications cover most of the use case.

## References

## Author information

Andrew Grimshaw (grimshaw at virginia.edu), Genesis II Project, University of Virginia