insideHPC

The insideHPC Guide to

Six Successful Strategies for

Managing High Performance GPU Clusters

BROUGHT TO YOU BY

NVIDIA

Bright Computing

Introduction

As of June 2015, the second fastest computer in the world, as measured by the Top500 list employed NVIDIA® GPUs. Of those systems on the same list that use accelerators 60% use NVIDIA GPUs. The performance kick provided by computing accelerators has pushed High Performance Computing (HPC) to new levels. When discussing GPU accelerators, the focus is often on the price-to-performance benefits to the end user. The true cost of managing and using GPUs goes far beyond the hardware price, however. Understanding and managing these costs helps provide more efficient and productive systems.

THE ADVANTAGES OF GPU ACCELERATORS

The use of NVIDIA GPUs in HPC has provided many applications an accelerated performance beyond what is possible with servers alone. In particular the NVIDIA Tesla® line of GPUs are designed specifically for HPC processing. Offering up to 2.91 TFLOPS of double precision (8.74 TFLOPS using single precision) processing with ECC memory they can be added to almost any suitably equipped x86_64 or IBM Power 8 computing server.

With the support of the NVIDIA Corporation, an HPC software ecosystem has developed and created many applications, both commercial and open source, that take advantage of GPU acceleration. The NVIDIA CUDA® programming model along with OpenCL and OpenACC compilers have provided developers with the software tools needed to port and build applications in many areas including Computational Fluid Dynamics, Molecular Dynamics, Bioinformatics, Deep Learning, Electronic Design and Automation, and others.

Contents

THE CHALLENGES PRESENTED BY GPU ACCELERATORS

Any accelerator technology by definition is an addition over the baseline processor. In modern HPC environments, the dominant baseline architecture is x86_64 servers. Virtually all HPC systems use the Linux operating system (OS) and associated tools as a foundation for HPC processing. Both the Linux OS and underlying x86_64 processors are highly integrated and are heavily used in other areas outside of HPC — particularly web servers.

Virtually all GPU accelerators are added via the PCIe bus. (Note: NVIDIA has announced NVLink™ a high-bandwidth, energy-efficient interconnect that enables ultra-fast communication between the CPU and GPU, and between GPUs.) This arrangement provides a level of separation from the core OS/processor environment. The separation does not allow for the OS to manage processes that run on the accelerator as if they were running on the main system. Even though the accelerator processes are leveraged by the main processors, the host OS does not track memory usage, processor load, power usage or temperatures for the accelerators. In one sense the GPU is a separate computing domain with it's own distinct memory and computing resources.

From a programming standpoint, the tools mentioned above proved an effective way to create GPU based applications. In terms of management, however, the lack of direct access to the accelerator environment can lead to system-related concerns. In many cases, the default management approach is to assume GPUs are functioning correctly as long as applications seem to be working. Management information is often available separately using specific tools designed for the GPU. For instance, NVIDIA provides the `nvidia-smi` tool that can be used to examine the state of local accelerators. Monitoring GPU resources with tools like `nvidia-smi` and NVIDIA's NVML provide administrators with on-demand reports and data, however, information is often extracted using scripts and sent to a central collection location.

Another challenge facing GPU developers and administrators is the ongoing management of the software environments needed for proper operation. There are several reasons for this situation. First, new versions of the NVIDIA CUDA software and drivers may offer better performance or features not found in the previous version. These new capabilities may need to be tested on separate machines within the cluster infrastructure before they can placed into production. Second, some HPC clusters may have multiple generations of GPU hardware in production and need to manage different kernel versions for specific hardware combinations. Both cluster provisioning and job scheduling must take these differences into account. And finally, there may be specific HPC applications that require specific kernel/driver/CUDA versions for proper operation.

Users and developers find managing tools tedious and error prone, while administrators need ways to make sure the applications are running successfully on the right hardware.

These challenges often create administration issues or "headaches" when trying to manage HPC clusters. Each new combination of hardware and software creates both a monitoring and tool management challenge that often reduces the system throughput. Users and developers find managing tools tedious and error prone, while administrators need ways to make sure the applications are running successfully on the right hardware.

CREATING A GPU COMPUTING RESOURCE

The advantages and challenges of GPU accelerators have presented users and vendors with the opportunity to develop a set of best practices for maximizing GPU resources. As will be described in this paper, there are sound strategies that will help minimize the issues mentioned above and keep users and administrators focused on producing scientific results. The goal is to transform a collection of independent hardware components and software tools into an efficiently managed production system.

Best Practices for Maximizing GPU Resources in HPC Clusters

The following best practices can be used to maximize GPU (and all cluster) resources in a production HPC environment. The approaches can be broken down into six strategies that address the issues mentioned above.

STRATEGY 1

Provide a Unified System so Users/ Developers can Focus on Results/Coding

HPC developers want to write code and create new applications. The advanced nature of HPC often requires that this process be associated with specific hardware and software environment present on a given HPC resource. Developers want to extract the maximum performance from HPC hardware and at the same time not get mired down in the complexities of software tool chains and dependencies. For instance the workload scheduler should be GPU-aware and not force the users to figure out how to use and share GPU resources.

To address this need, the entire HPC resource needs to be treated as a "system" and not a collection of hardware stitched together by shell scripts and incompatible cluster management tools. Four factors that should be addressed include:

1. **Provisioning –** The cluster needs to have the flexibility to provision nodes based on specific hardware and software requirements. All valid and tested combinations should be available to users without creating extra work for systems administrators.
2. **Workload Managers –** Cluster workload managers such as Slurm, PBS Pro, Torque/ Maui/Moab, Grid Engine, and LSF should be integrated and be aware of all resources in the cluster - including both CPUs and GPUs.
3. **Comprehensive monitoring –** End users should not have to concern themselves with system monitoring. That is, they should not have to log in to individual nodes, check loads or run tools like `nvidia-smi` to be assured their applications are working properly. Administrators should not be "figuring out" how to collect this data for each type of hardware in the cluster. In addition, pre-job health checks should be automatic so that user jobs are not schedule to run on failing or distressed hardware.
4. **Development Tools –** The entire development tool-chain must be integrated and flexible. Any dependencies must be addressed and not left as "an exercise for the users." These tools include compiler, libraries, debuggers, and profilers that users need to develop code.

Bright Cluster Manager™ provides leading edge and comprehensive cluster management capabilities for any HPC cluster including those equipped with NVIDIA GPU accelerators. The approach is top-down and fully integrates the underlying hardware into a true HPC resource — ready for users.

Without attention to the above four aspects, users can expect delays and increased time to solution. In addition, system administrators must manage ever-increasing complexity as systems change. Fortunately, each aspect mentioned above can be managed automatically with Bright Cluster Manager™. It provides leading edge and comprehensive cluster management capabilities for any HPC cluster including those equipped with NVIDIA GPU accelerators. The approach is top-down and fully integrates the underlying hardware into a true HPC resource — ready for users.

Bright provides both point and click and scriptable command line control of the entire cluster. These capabilities create a flexible provisioning management where any popular Linux distribution (SUSE, Red Hat, CentOS, Scientific Linux) can be loaded onto any node. In addition, specific kernel versions and GPU kernel modules can be easily managed. These capabilities can all be managed using Bright Cluster Manager's "single pane of glass" graphical management console. Every aspect of the cluster, both local and in the cloud is managed in a consistent and intuitive fashion. As an example GPU integration, *Figure 1* shows how administrators can have direct access to the performance-enhancing NVIDIA GPU Boost technology.

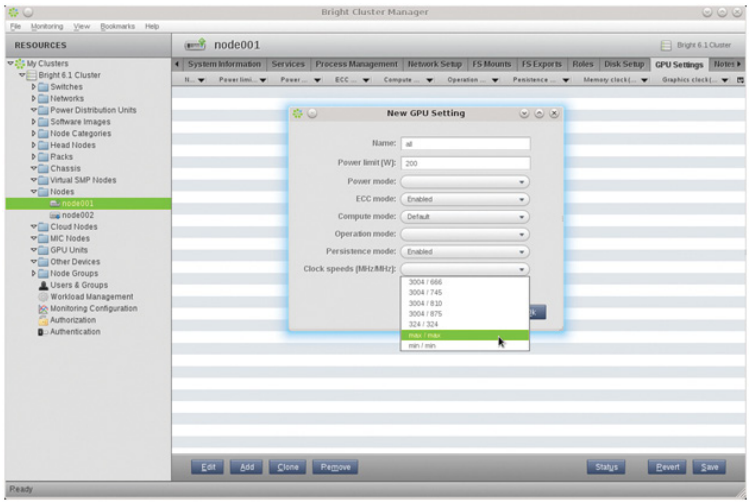

Figure 1: Bright Cluster Manager manages NVIDIA accelerators used in hybrid-architecture systems for HPC. Bright allows direct access to the NVIDIA GPU Boost technology through clock-speed settings.

STRATEGY . 2 .

Automate Updates

Keeping track of available GPU software updates can be a burdensome process. Integrating and testing updated software capabilities can introduce delays and broken development tool chains. When performing these types of updates, functional testing is required to make sure everything that worked previously continues to work correctly. If multiple generations of GPU hardware are present, more detailed testing might be needed to ensure the new environment "just works." Developers appreciate clean updates and dislike broken environments. Simply installing GPU updates and hoping for success is a recipe for disaster.

A good way to address software updates is to install and test in a non-production sandbox environment. The sandbox should have the exact same hardware environment as production nodes. Only when the software updates have been tested should the productions nodes be re-provisioned and any updated or new development tools be made available. The more this process can be scripted or automated the easier it will be to perform for subsequent updates. As part of the scripting process, all kernel modules and drivers should be built with the same production kernel (or kernels) running on the cluster and should include automated testing of the new environment.

An alternative to building and managing a custom local GPU sandbox is to use a comprehensive cluster manager. Bright Cluster Manager provides automatic synchronization with the latest NVIDIA CUDA software (verified for your environment). The synchronization includes a full update and install of any new NVIDIA software versions including the administrative steps required for full operation. This process includes building any kernel modules, rebooting/provisioning cluster nodes, and functional testing with both CUDA and OpenCL test code (Test applications are built and executed on the environment).

This feature actually goes deeper than automatic updates. NVIDIA works closely with Bright to provide early access to drivers, CUDA updates, and new GPUs so that when the Bright updates are released, your cluster is ready with the latest software and hardware from NVIDIA. When a new version of CUDA is released, new Bright CUDA packages are made available for all recent Bright versions as soon as possible, after QA tests. Because Bright follows the CUDA release cycle, the Bright cluster manager will ensure updates are current and work as expected. These updates include full integration of any new capabilities into the monitoring layer of Bright Cluster Manager. The process provides developers with smooth and uninterrupted use of the cluster and eliminates the need for administrators to create and manage a sandbox test environment.

STRATEGY 3

Manage User Environments

In a perfect world, there would be one version of all compilers, libraries, and profilers. To make things even easier, hardware would never change. However, technology marches forward, and such a world does not exist. Software tool features are updated, bugs are fixed, and performance is increased. Developers need these improvements but at the same time must manage these differences. A typical example on many Linux systems is the software libraries on HPC systems. HPC applications are often built with specific compiler versions and linked to other specific library versions that all must work correctly together. The dependency tree can become quite complex.

The most widely used open source environment management tool is called “Modules“. By using the Module tool, a user’s environment “defines“ can be changed with one command.

Allowing users to manage these differences “by hand“ is a poor solution. Managing environments is done with a combination of methods. Source code is often made more robust by using conditional definitions to expose correct code sections for various compilers or software environments (e.g. `#define`, `#ifdef`, `#ifndef`, etc. in C include files). In combination with this approach, the users software environment has defined values that point to tool and library locations, manual pages, and other important *“defines“* that makes sure the development toolset is correctly configured. (e.g. `INCLUDEPATH`, `LD_LIBRARY_PATH`, `CUDA_INCLUDE_PATH`, etc. in the users shell environment.) Administrators can set default environment defines, however, when users need to change a define (or more likely a set of defines) to point to a different version of the software, they often do it “by hand“ in their shell configuration files (e.g. `.bashrc` or `.cshrc`). Changing these settings can be complicated, error prone, and hard to manage for developers. Without some form of “environment management“ the “hand managed“ development environment very quickly becomes a fragile solution for software developers.

Although the Modules package has been a boon to HPC users and administrators, each environment must be configured and tested before the developers can take advantage of the functionality.

The most widely used open source environment management tool is called “Modules“ and is available from http://modules.sourceforge.net. (Another tool called *lmod* performs a similar function and is available from https://www.tacc.utexas.edu/research-development/tacc-projects/lmod). By using the Module tool, a user’s environment “defines“ can be changed with one command. For example, if configured correctly, the command:

```
$ module load cuda60/toolkit/6.0.37
```

will set users development environment to use the NVIDIA CUDA Toolkit version 6.0.37. Once set the developer can be assured that they are using a correctly configured development environment. It would be equally easy to change to a newer CUDA environment, such as version 7, with a similar command.

The use of Modules (or lmod) is a powerful method that can help reduce developer and administrator frustrations. Although the Modules package has been a boon to HPC users and administrators, each environment must be configured and tested before the developers can take advantage of the functionality. This requires considerable thought and planning because module naming needs to be consistent and represent the important differences between tools as the number of options grows.

Unique among cluster management options, Bright Cluster Manager offers fully configured Modules environment for NVIDIA GPU clusters. There is no need to develop and test Module environments for new software. Indeed, in addition to NVIDIA GPU development, Bright Cluster Manager includes many preconfigured module files, for compilers, mathematical, and MPI libraries. Additional modules can be configured centrally by the system administrator, or locally by an individual user. Most popular shells are supported, including `bash, ksh, sh, csh, tcsh.`

Combined with the automatic GPU software updates, the Bright Cluster Manager offers developers instant availability of new development environments. When NVIDIA releases a new CUDA version, a fully tested Module based development environment is made available as part of the automatic update. Developers can begin using the new environment right away. As an example, *Figure 2* shows two CUDA environments existing side-by-side on a cluster. The developer can choose an environment using a single `module load` command.

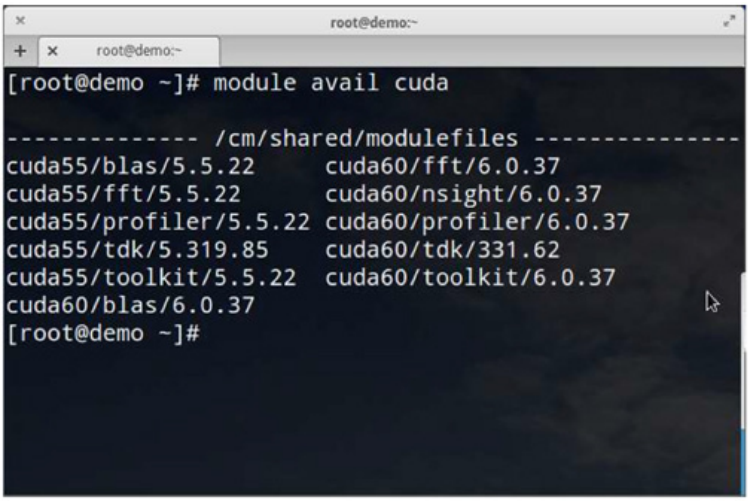

Figure 2: Bright Cluster Manger provides a fully configured development environment using Modules configured for the NVIDIA software environment. Developers can change environments with a single command.

STRATEGY 4

Provide Seamless Support for all GPU Programming Models

In addition to the CUDA environment mentioned above, developers have many options when programming GPUs. In particular, NVIDIA GPUs have an excellent HPC software ecosystem. There are many approaches to GPU programming but most HPC applications utilize one of the following compiler methodologies:

- **NVIDIA CUDA**
  The CUDA programming model was developed by NVIDIA and provides a powerful model to express parallel GPU algorithms in a C/C++ and Fortran. The `nvcc` CUDA compiler is available for NVIDIA hardware.

- **OpenCL**
  A framework for writing programs that execute across heterogeneous platforms such as CPU and GPUs. Based on the C99 standard, the `clcc` compiler will allow code development on heterogeneous processing architectures.

Developers have many options when programming GPUs. In particular, NVIDIA GPUs have an excellent HPC software ecosystem.

- **OpenACC**
  A multi-vendor supported specification based largely on early work by The Portland Group that allows new and existing C and Fortran programs to be modified with comment-based directives. (i.e., the function of the original program is not changed.) A compiler that supports OpenACC is then used to create GPU-ready programs. More information can be found from the http://www.OpenACC.org.

A fourth method using optimized libraries in conjunction with a conventional host compiler can also be used to add GPU capabilities to applications. There are several accelerated libraries available to developers including:

- **cuBLAS**
  An implementation of the Basic Linear Algebra Subprograms package
- **cuSPARSE**
  A set of sparse matrix subroutines
- **cuFFT**
  An implementation of FFT algorithms for NVIDIA GPUs.
- **cuDNN**
  Library of primitives for Deep Neural Networks
- **NPP**
  NVIDIA Performance Primitives is a very large collection of 1000's of image processing primitives and signal processing primitives.

There are also wrappers for other languages (Python, Perl, Java, and others) that allow GPUs to be used. In a typical HPC environment C/C++ and Fortran compilers from GNU, The Portland Group, Intel, and others are used. The Portland Group, Cray, PathScale and GNU gcc 5 compilers now support OpenACC, however, Intel compilers do not. To help with debugging and profiling there are applications such as TAU, Allinea, and TotalView.

Most HPC clusters also include a healthy mix of MPI (Message Passing Interface) libraries including OpenMPI, MPICH, MPICH-MX, and MVAPICH. Finally there are many scientific libraries, including mathematical routines that are made available to developers.

This diverse high performance software tool ecosystem is necessary for today's HPC installations. As mentioned previously, managing these options can be done by using the Modules package. Without some kind of automatic and systematic compiler/library management a developer would need to spend time creating the critical environment to compile and run the applications for both CPUs and GPUs. Any changes, such as new versions or new hardware, can easily break a handmade development environment.

The real power of Modules is when a new compiler or library version is available, a new tool chain can be built and installed next to the existing tool chain. Thus, users can easily move between old and new version with ease. A similar configuration should be created for all GPU tools as well.

The task of creating and testing Module configuration often falls on the system administrator. One common approach is to create an entire tool chain environment including math (e.g., PETSc or ScaLAPACK), MPI libraries, and others. Users can invoke various tool chains when using a specific compiler or library version. The real power of Modules is when a new compiler or library version is available, a new tool chain can be built and installed next to the existing tool chain. Thus, users can easily move between old and new version with ease. A similar configuration should be created for all GPU tools as well.

Bright Cluster Manager comes with many of the popular GPU (and CPU) based tool chains preinstalled with full Module support. That is, there is no need to weave together various tool chains for use with the Modules package. Developers can easily move between virtually all development environments with a simple Modules command. The installed Module configuration is also very flexible and extensible. As mentioned above, administrators can add new Module environments for virtually any installed package. With a fully integrated GPU tool chain, developers can begin working right away, there is no need to configure or construct a development environment.

STRATEGY 5

Plan for the Convergence of HPC and Big Data Analytics

As an open source tool designed to navigate large amounts of data, Hadoop continues to find new uses in HPC. While traditional Hadoop applications do not run on GPU hardware, version 2 of Hadoop supports virtually any computing model or hardware environment. This flexibility allows Hadoop to be used for more than the traditional MapReduce applications and opens up many possibilities. In particular, breakthrough deep learning performance has been achieved with new algorithms and can be accelerated with GPUs. For example, NVIDIA provides the cuDNN library of primitives for Deep Neural Networks. Combined with the Big Data capabilities of Hadoop, GPUs offer a path to accelerated training of deep learning models.

Managing a Hadoop cluster is different than managing an HPC cluster, however. It requires mastering some new concepts, but the hardware is basically the same and many Hadoop clusters now include GPUs to facilitate deep learning. In one sense, a Hadoop cluster is actually simpler than most HPC configurations as they primarily use Ethernet and commodity servers. Hadoop is an open-source project and is often configured "by hand" using various XML files. Beyond the core components, Hadoop offers a vast array of additional components that also need management and configuration.

One solution that provides Hadoop capabilities to HPC clusters is to carve out a sub-cluster within the main cluster. The sub-cluster is a collection of nodes configured to run the various Hadoop services. Hadoop version 2 processing is different than the typical HPC cluster as it has its own resource manger (YARN) and file system (HDFS). Integration with an existing cluster can be somewhat tedious as a completely different set of daemons must be started on Hadoop nodes.

Instead of configuring and managing a sub-cluster or specialized Hadoop system, an automated solution such as Bright Cluster Manager can effortlessly bring Hadoop capability to a cluster. Bright Cluster Manager supports the leading distributions of Apache Hadoop (e.g., from the Apache Foundation, Cloudera, Hortonworks), enabling Bright's customers to choose the one that best fits their needs while taking advantage of Bright Cluster Manager's advanced capabilities. The same management interface covers the Hadoop nodes that can easily be configured to contain GPUs. As shown in *Figure 3* below, Bright Cluster Manager provides an instance of a Hadoop running in the cluster. The view provides an overview of essential Hadoop parameters.

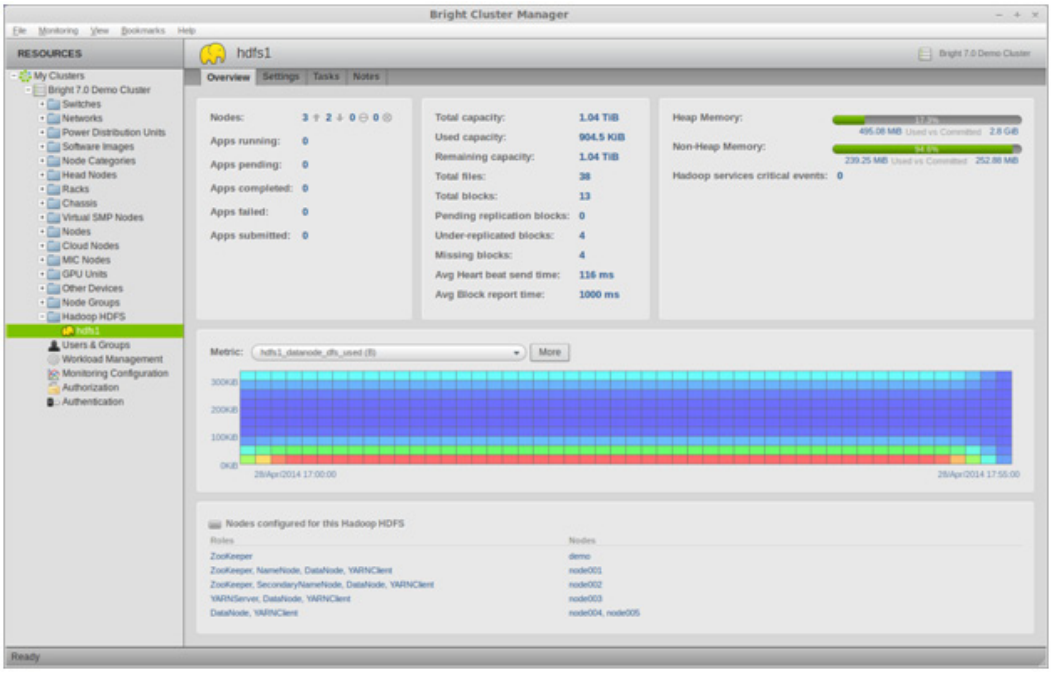

Figure3: Bright Cluster Manager manages multiple instances of Hadoop HDFS simultaneously. Bright's "Overview" tab for Hadoop illustrates essential Hadoop parameters, a key metric, as well as various Hadoop services.

STRATEGY 6

Develop a Plan for Cloud-based GPU Processing

The cloud is here and ready to start running HPC applications. Certainly the cloud is not a one size fits all solution, but many organizations have found great utility in using both public and private clouds as a computing resource.

Trying to "get things working" in the cloud can be an expensive proposition. Wasted time, repeated configuration changes, testing, and frustrated users and developers all reduce the economics that made the cloud attractive in the first place.

The public cloud providers, like Amazon Web Services, that offer systems with NVIDIA GPUs. Thus, it is possible to provide both local and cloud based GPU computing. An in-house private cloud could also be used as a HPC resource. Bright Cluster Manager provides and integrated OpenStack option from which private clouds can be easily created and managed. A cloud HPC strategy can range from a plan to provide local overflow resources to a full-fledged offsite GPU cluster in the cloud.

For some applications, cloud based clusters may be limited due to communication and/or storage latency and speeds. With GPUs, however, these issue are not present because application running on cloud GPUs perform exactly the same as those in your local cluster — unless the application span multiple nodes and are sensitive to MPI speeds. For those GPU applications that can work well in the cloud environment, a remote cloud may be an attractive option for both production and feasibility studies.

Integrating a cloud solution into an in-house cluster can be difficult because the cloud only provides the raw machines to the end users. Similar to an on-site cluster, the GPU cloud cluster needs to be configured and provisioned before use. All of the issues mentioned previously — everything from CUDA drivers and libraries to a managed and integrated development environment — must be available as part of the cloud resource. Expecting developers to use or design for a different software environment creates extra work and expense. Indeed, flexibility and ease of management is now even more important in the cloud due the ephemeral nature of GPU cloud clusters, (i.e., administrators may need to repeatedly set up and tear down cloud instances over time). Long lead times for custom cluster configuration defeats the advantages of an on-demand cloud environment.

In addition, because cloud use is metered, it should be used as efficiently as possible. Trying to "get things working" in the cloud can be an expensive proposition. Wasted time, repeated configuration changes, testing, and frustrated users and developers all reduce the economics that made the cloud attractive in the first place.

The best strategy is to prepare for eventual cloud use and find those providers that support your type of GPU computing hardware. Once identified, determine a method to translate the current local configuration to the cloud hardware and then test to make sure everything works as expected.

The same powerful cluster provisioning, monitoring, scheduling and management capabilities that Bright Cluster Manager provides to onsite clusters extend into the cloud, ensuring effective and efficient use of the virtual cloud resources.

An integrated solution like Bright Cluster Manager can provide a managed and transparent cloud-based GPU cluster. The same powerful cluster provisioning, monitoring, scheduling and management capabilities that Bright Cluster Manager provides to onsite clusters extend into the cloud, ensuring effective and efficient use of the virtual cloud resources. Using Bright Cluster Manager, you can extend into public GPU clouds, such as AWS, with only a few mouse clicks. There's no need for expert knowledge of Linux or cloud computing. The exact same administration interface is used for both local and cluster nodes. Installation/initialization, provisioning, monitoring, scheduling and management are identical. There is no learning curve required for cloud HPC use.

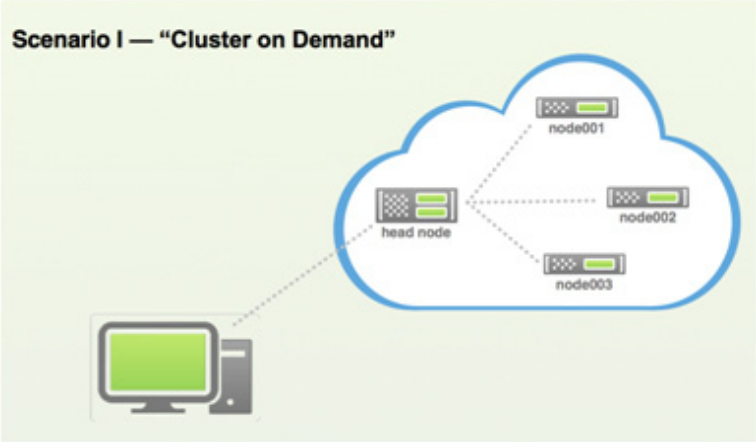

Figure 4: Bright Cluster Manager can create fully functioning ready to run GPU clusters in the cloud.

There are two possible scenarios available with Bright Cluster Manager. The first is a "cluster on demand," shown in *Figure 4* that can be created in the cloud as a fully functioning GPU cluster. Again, depending on the application, this cluster will behave, from a software standpoint, exactly like a local GPU cluster.

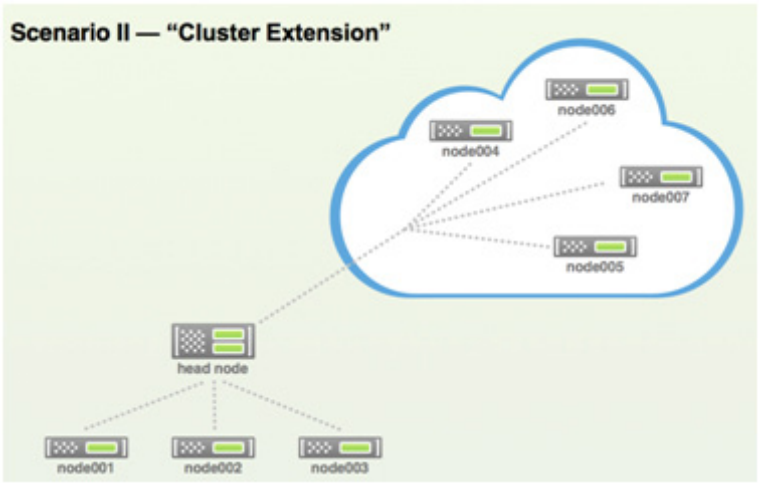

Figure 5: An internal or external cluster extension can be created using Bright Cluster Manager. The new remote cluster resources are manged exactly the same as cluster nodes.

The second use case it that of a "cluster extension," shown in *Figure 5*. In this example, the cloud is used to extend the local resources. In either scenario the cluster is manage directly from the Bright Cluster Manager interface. There is no need to apply any special provision to the cloud nodes.

Accelerated Science: GPU Cluster Case Study

A successful example of how a well-managed GPU cluster allowed scientist to focus on obtaining results comes from the Tokyo University of Agriculture and Technology (TUAT) results. A research group lead by Dr. Akinori Yamanaka develops computation models and simulates engineering materials, for a variety of applications, using HPC. Using Bright Cluster Manager, Dr. Yamanaka and his team were able to immediately focus on algorithm development and not burden the team with cluster administration issues. To learn more about how Bright Cluster Manager accelerated computational research, consult the full case study at: http://info.brightcomputing.com/download-tuat-materials-science-case-study.

Conclusion

To fully take advantage of NVIDIA GPUs requires several sound strategies. The goal of any HPC resource should be to increase the productivity of researchers and engineers because minimizing time to solution is the goal of many leading HPC installations. Keeping users and developers focused on applications is one of the way to increase productivity and minimize wasted time.

The following is a summary of the strategies discussed above. They provide a set of best practices that eliminate many of the delays and challenges facing GPU clusters.

1. Provide a unified system that allows users and developers to focus on applications and coding. Keep infrastructure and tool chain management issues away from those who need to get work done.
2. Keep software current so new features and capabilities are immediately available to users and developers.
3. Manage the users GPU development options with the Modules package to reduce conflicts and misconfigurations.
4. Provide seamless support for all GPU programming models. Developers like options. Provide a well managed and up-to-date development tool chains for all the popular GPU programming methods.

The goal of any HPC resource should be to increase the productivity of researchers and engineers because minimizing time to solution is the goal of many leading HPC installations.

5. Plan for Big Data and Hadoop usage. Breakthroughs in deep learning on GPUs will spawn new applications in this area.
6. Now that GPUs are available in the cloud, develop an approach that allows your organization to take advantage of this resource.

Bright Cluster Manager addresses the above six strategies and provides a comprehensive environment for production GPU cluster computing. There is no reason why a cluster equipped with NVIDIA GPUs cannot operate as an integrated HPC resource. Bright CUDA-ready clusters are GPU developer and GPU user-ready clusters.