User interface for a computational cluster: resource description approach

A. Bogdanov[1,a], V. Gaiduchok[1,2], N. Ahmed[2], P. Ivanov[2], M. Kamande[2], A. Cubahiro[2]

[1] Saint Petersburg State University, 7/9, Universitetskaya emb., Saint Petersburg, 199034, Russia

[2] Saint Petersburg Electrotechnical University, 5, Professora Popova st., Saint Petersburg, 197376, Russia

E-mail: [a]bogdanov@csa.ru

Computational centers provide users with HPC resources. This usually includes not only hardware (e.g. powerful computational clusters) and system software (e.g. Linux and some PBS implementation), but also application software too. In case of university computational center system administrators install some scientific applications, provide users with access to them and manage that access. They are usually responsible for updating such software and solving problems. But such access usually implies only the ability to submit user jobs to resources: users are responsible for creating correct job description (in terms of necessary resources) and job start script (conventional and widely used systems like PBS usually require some script which contains commands that actually start the job). But inexperienced users could spend much time to learn the interface. They could do mistakes which could lead to imbalance in system utilization. This article propose an interface that will provide uses with conventional interface. Using this approach computational center users could avoid scripting and concentrate on their job.

Keywords: computational clusters, user interface, cluster management systems, automation, parallel computing, parallel applications.

This work was supported by Russian Foundation for Basic Research (projects N 16-07-01111, 16-07-00886, 16-07-01113) and St. Petersburg State University (project N 0.37.155.2014).

Introduction

Managing computational center resources is a difficult task. Using them should be easy enough but it is not always true. All computational centers triy to provide users with some conventional interface since computing user tasks is the primary goal of such centers. But clusters within such centers usually do not have a conventional interface. They are often managed by PBS-like systems that usually provides only command line interface. As a matter of fact, users should learn scripting (to describe the job) and managing system command line interface (to submit the job).

Such tasks are easy for an average system administrator, but difficult for an average user. Users usually consider job scripting to be difficult. Moreover, some of them can not request the necessary resources correctly. But users should learn scripting language (e.g. bash), some tasks of system administration (e.g. Linux administration), managing system command line interface and learn information about all the hardware in the cluster (in order to specify the necessary resources correctly). Such tasks could be facilitated by creating some template scripts by administrators. But even in this case users should learn scripting in order to modify such scripts for their needs and learn work with command line interface (in order to submit their jobs to the cluster management system). Users are usually not accustomed with such work. And all they need is to run their computations and retrieve the results.

This article is dedicated to this problem: the mentioned above tasks will be discussed in details on an example of a cluster with widely used management system (PBS). Such system includes PBS server and scheduler (information about PBS schedulers could be found in [Bode et al., 2000]).Then the possible solution to this problem will be proposed: a graphical user interface based on resource and task description system. This case implies a language for resource (hardware and applications) description. Such descriptions are created by system administrators. They represent available hardware and software as an abstraction. They are used by graphical user interface in order to represent the available resources in a common way. Such approach eliminates the need to learn command line interface and scripting for users and allow them to access resources in a convenient way. Moreover, this leads to more efficient resource utilization since users will do fewer mistakes when requesting resources.

Typical computational center consist of several clusters with different performance. Typical organization provide users with a variety of software applications, free and open source, as well as proprietary. Typical user is interested in one or two applications. Administrators need to restrict users for different reasons (e.g. security). They need to comply with organization policies and to assure that users do it. They could improve user decisions in terms of performance of applications. But often it is hard to do if users have direct access to the underlying system.

Possible approaches

Creation of a convenient and simple user interface could be considered as one of the most important tasks of an average computational center. Thought-out interface could improve user experience and resource utilization.

But the user interface is often depends on the cluster management system. PBS (Portable Batch System) is one of the possible choices. It is used on many clusters. This system provides users with command line interface which users should learn in order to submit (‘qsub’ command), delete (‘qdel’ command), alter (‘qalter’ command) and check jobs and resources (‘qstat’, ‘pbsnodes’ commands). The mentioned interface usually (PBS has several implementations, for example TORQUE [TORQUE Resource Manager] and PBS Professional [PBS Professional]) has many commands with different options. The job itself should be described (written) as a script (one should specify the necessary commands to run the jobs on cluster resources).

Such systems usually support interactive jobs. But such jobs usually imply just interface to the console on a node of the cluster. So, once again, users should learn PBS interface, system command line interface (to specify commands correctly) and application command line interface (to specify necessary options).

There could be some graphical interfaces to facilitate the tasks. But they usually ease only job management (submission, deleting and changing): users are still responsible to write job scripts and so learn command line interfaces of system and applications (an example of web-based graphical interface of this type is described in [Ma, Lu, 2000]).

While this task itself is not easy for many users, the situation is getting worse because of the diversity of the resources: an average computational center could have several clusters with different characteristics. The cluster nodes could have different architecture. Moreover, they could have different accelerators.

But usage of such accelerators is not a guarantee of good performance (e.g. because of overheads) and one should think whether to use accelerators (e.g. the task big enoug): it makes no sense to use such accelerator for a task that could not be vectorized. So, user should think about the resources to use (user should specify the necessary resources when submitting the job).

Another possible problem is application interface. User should learn the application command line interface in order to describe what is required (what to compute (e.g. input file) and how to compute (e.g. precision)). But such input is not necessarily related to the scientific area (e.g. algorithm to use). For example, user could be required to specify parallel API options (e.g. options for 'mpirun' command in case of MPI).

Finally, user should estimate the application performance since user is responsible to specify the necessary resources. For example, the number of nodes and CPU cores per node to use. In this case user should assess the portion of parallel computations in the application algorithm, how the algorithm was implemented, how much data will be transferred via network (it is especially important in case of slow or overloaded network), the size of input, output and temporary files (in case of limited disk space), whether to use accelerators or not, probably some underlying OS details (e.g. some limitations that could impact the application run), parallel API (e.g MPI) implementation to use (they could have different performance and features (e.g. Infiniband support) and so on. Moreover, all these details are related to each particular task (e.g. there could be different algorithms for different tasks). The wrong estimation could lead to slow application run or system underload. Sometimes some users could assume the more resources they will reserve the faster the application will compute the task. But, of course, this approach is unrealistic because of Amdahl's law and software and hardware limitations.

But it is hard to do the mentioned estimations for users. Only after several runs with different amount of resources one could find out the possible limitations. As a result applications could use more resources that it was necessary. And so the computational center faces with the situation when its resources are used not efficiently.

Administrators could write some scripts in order to facilitate the user work. They could hide some internal aspects of an application (e.g. some command line options). And they could do some advices on resources usage. But such approach is still not convenient.

One could write some graphical user interface for some application and for particular cluster manager. But such interface will solve the problem just for the case of that application running with that particular manager.

New interface

The new approach should solve the mentioned problems. We can highlight the user expectations in the following list.

- Users do not want to go deep into details of the manager system.
- Users do not want to learn the operating system command line interface.
- Users do not want to learn the application command line interface.

- Users are not responsible to learn computational center resources in details.
- Users are not responsible to learn internal application features in details.
- They want the interface to be simple.
- They know their scientific area and want to ‘talk to the system’ in terms they know.
- They need to calculate their tasks fast.
- They assume administrators will solve all problems.

We propose resource description system with graphical interface. Users are given with convenient graphical interface. While administrators could describe the resources.

The mentioned system supports simple and detailed configuration files. Administrators write configurations that describe the resources (e.g. hardware resources), cluster managers and user applications.

We assume that we cluster manager and applications use command line interface. We assume that manager has some command to submit the job (we will discuss only this particular case), we treat that command as a generic command that has some options. We describe that options: they could be required or optional, could have some value, etc. For example, we need to specify the number of CPU cores to use. This value could have some default value (or even constant value in case of serial application – just 1 core) or could be specified by user. We create a new ‘input’ configuration for that option and specify in the ‘value’ of that option that it should be equal to the value of that input. So, user will see the input field in the graphical user interface. If such input value is required user will have to type that value or user will get an error on job submission.

So, we have options for manager that will be chosen and filled automatically depending on the user input. The manager system job submission command will specify the script to submit that will be created. That script is based on task configuration files.

Moreover, such approach is flexible: one can easily change system manager (making changes to the configuration files of the described system only once).

A task could consist of several jobs and that jobs could request different resources (for example, some initialization by some specific application could not be done in parallel – so the first job will always request a single CPU core). The job consist of some preparation work (running some scripts, changing some user-related configuration files, setting some environment variables) and running the application (or several applications within one job).

The application is described with the mentioned application configuration file. The system will specify all required application options with values that user entered. But in order to do it user is given with conventional graphical interface. Moreover, each option is described by administrator (user can even forget how the option is specified in application command line interface). Such description should consist the actual meaning of that option (e.g. the mesh size for some computer simulation task). So user will ‘talk to application’ in terms of a particular scientific area.

All such descriptions are supposed to be written once. New application versions are easy to describe: administrators should specify only the changes (e.g. new path or new option). Administrators could enforce some parameters (e.g. some resource features). There could be many parameters with default values specified by administrators – it could be concerned as advices.

Users are given with conventional graphical interface. They specify the resource, the task, the application version. Then they fill the required input fields. The most of those fields are described in terms that are known very well by user. User could group the options (e.g. list required options first). When user finishes the work, the system will check the input and report errors if any. If everything is correct, the system will form the necessary script(s) and submit the job. Then user could check the status of the job and continue the work.

Such interface will hide all the internal aspects on the underlying system from users. Moreover, it could make security questions easier and ease the monitoring of the jobs: we will know what tasks users run (that is what applications is used often – organization could focus on them e.g. when planning updates). In case of generic system it hard to implements such monitoring since users can write any job and assign any name to it.

So, the main difference from similar systems is the fact that our system not only facilitate the job submission, it eliminates job scripting and allows users to describe the job (task) in terms they understand, that is, it let them avoid learning details of particular cluster manager system and particular application. The possible future work is to describe the input (configuration) files of command line applications and thus move all the application related work to the level of notions of particular scientific area which is convenient for users.

The interface is written in Java (JRE 8 is required). It is assumed that user home directory is accessible from all the nodes of clusters and from the node where the system (interface) is running in the same way (e.g. mounted via NFS).

Conclusions

Command line interface is considered to be inconvenient by the vast majority of users. Direct access to underlying management system is considered to be a bad option by many administrators. Administrators often write scripts to ease task submission, but such approach requires CLI anyway. Existing (even web-based) GUI interfaces usually implies scripting: users work with GUI, but have to write scripts that represent jobs. There are paid proprietary solutions, but such solutions usually have many unnecessary features and are too expensive.

Similar systems usually facilitate the access to the system (GUI instead of CLI), but not the work with the system (users are responsible to write scripts, etc.). The main element of the described system is a specific application build described by administrators. Users will work only with really necessary aspects. The proposed approach meets the user requirements (simple and convenient way to calculate a particular task), as well as administrator needs (impose organization policies and limit the access to the underlying system).

Such approach could improve resource utilization. Proposed system is designed to work with a generic PBS implementation. Web-interface is under development.

References

Ma G., Lu P. PBSWeb: A Web-based Interface to the Portable Batch System // 12th IASTED International Conference on Parallel and Distributed Computing and Systems. – 2000.

Bode B. et al. The Portable Batch Scheduler and the Maui Scheduler on Linux Clusters //Annual Linux Showcase & Conference. – 2000.

TORQUE Resource Manager [Electronic resource]: http://www.adaptivecomputing.com

PBS Professional [Electronic resource]: http://www.pbsworks.com