Some Approaches to Managing Computing Resources of a Hybrid High-Performance Cluster in a Cloud Environment

Konstantin Volovich[a], Vadim Kondrashev[a] , Mikhail Posypkin[a] and Sergey Denisov[a]

[a] *Federal research center "Computer Science and Control" of the Russian Academy of Sciences, Vavilova st. 44-2, Moscow, 119333, Russia*

Abstract

The article proposes approaches for providing scientists and research teams with computing resources of hybrid HPC clusters as cloud services. A technique for migrating user software to an HPC cluster environment with GPU is proposed. Solutions for adaptation of programs in high-level languages to computing facilities of a hybrid computing cluster are considered. Algorithms for providing tasks with computing resources in a multitasking environment are proposed, as well as methods for creating an adapted task execution environment using container technology.

Keywords 1

Cluster, virtualization, workload manager, distributed computing, parallel computing, graphics accelerator, HPC

1. Introduction

Currently, there is a tendency to provide computing resources for solving fundamental and applied problems in the form of cloud services [1-4].

In accordance with the classical approach to cloud computing, such services can be provided as software - SaaS, platform - PaaS or infrastructure - IaaS. In [5,6], a variant of providing scientific research as a service, RaaS, is also proposed.

It is advisable to apply the cloud approach to high-performance computing services for research teams in various areas of applied and fundamental sciences. This will make it possible to centralize resources and ensure the workload of computing systems with computational jobs with greater efficiency than with the exclusive provision of computing resources to each research team, and will also provide greater flexibility in the operation of the computing cluster.

However, when providing high-performance computing services, it is necessary to take into account the specifics of the organization of the computing process and the architecture of a high-performance cluster.

Modern computing systems designed to solve scientific problems are built, as a rule, on the basis of hybrid architectures, including both general-purpose central processors and specialized accelerators. So, as of November 2020, seven out of ten first supercomputers from the Top 500 rating have a hybrid architecture [7], which can include central processing units (CPUs) of various architectures (for example, Intel x86_64, IBM Power), as well as computing accelerators (GPU) of various architectures (e.g. Nvidia Tesla, Matrix coprocessor).

To use the computing capabilities on such clusters in the concept of cloud computing, it is necessary to apply certain methods of organizing the computational process that provide the sharing of graphics accelerator resources between the clients of the computing cluster.

VI International Conference Information Technologies and High-Performance Computing (ITHPC-2021), September 14–16, 2021, Khabarovsk, Russia
EMAIL: kvolovich@frccsc.ru (A. 1); vkondrashev@frccsc.ru (A. 2); sdenisov@frccsc.ru (A. 3)
ORCID: - (A. 1); 0000-0002-1224-1392 (A. 2); - (A. 3)

Any research team that plans to use a hybrid computing cluster to solve their problems has software designed to solve its scientific issues as a rule. In the general case, such software may not be compatible with the architecture and software environment of the hybrid computing cluster. The issues of classification of application software for compatibility and mobility are considered in [8].

This article proposes systemic approaches to the problems of adapting the program code of scientific teams to the environment of a hybrid high-performance cluster, creating an individual task execution environment using virtualization technology based on software containers, as well as developing an algorithm for managing the resources of a computing cluster [9-11].

2. Program code adaptation to the hybrid high-performance cluster architecture

If the software available to scientific teams for solving fundamental and applied scientific problems cannot be carried out on a hybrid high-performance cluster, it needs specific preparation. To adapt the software available to the research team for solving an applied or fundamental problem, it is necessary to create several tools that allow loading, processing, testing, and debugging, and, finally, executing the program code to solve a scientific problem. The final stage of this process is to assess the improvement or degradation of performance when solving this problem compared to previously used computing systems.

The initial stage of adaptation of the program code for execution on a hybrid high-performance cluster is the classification of the program code according to the degree of mobility [8]. This classification can be performed either directly by the user or using automation tools. In particular, a neural network trained on a pre-labeled dataset can be used as such a tool. User classification, at first glance, is more than sufficient for deciding which class the adaptable task belongs to. At the same time, it is necessary to consider the hidden nuances that may not be available to the user at the time of work. To view such hidden information, it is proposed to use artificial intelligence components - a neural network that is preliminarily trained on a starting set of labeled initial data and further continues its training in the process of the system operation. Thus, at the initial (start-up) stage, the neural network training on the starting data array and adjusts its state based on information obtained in users' jobs on adapting their scientific tasks for execution on a hybrid high-performance cluster.

Figure 1 shows a scenario for functioning of a neural network for classifying program code and adjusting its state based on data obtained during the work of users to adapt the program code.

Note that a dataset for the primary training of a neural network can be built both on the basis of real examples of program code and on the basis of artificially generated (synthetic) examples. The markup of such examples for the primary training of the neural network should in any case be performed by a human.

The dataset for primary training should include examples for all high-level programming languages (HLL) used to develop application code in a hybrid high-performance cluster (for example, Python, TensorFlow). However, the architecture of the classification unit can be different.

It is possible to train a universal neural network on a complete set of primary training data, including examples in various programming languages (network "NN U" in Figure 1).

Another variant for constructing a classification block assumes the presence of a group of neural networks, each of which classifies the program code for one programming language. In Figure 1 shows neural networks "NN 1" - "NN 3" for each high-level language.

From the point of view of universality and the construction of a fully automatic module for classifying the program code, the first variant is preferable, since it does not require preliminary sorting of the program code based on the language used. In addition, this approach allows for the classification of program texts developed using several programming languages, which is a common practice when creating large projects. The disadvantages of this option are the duration of training and high requirements for the resources expended.

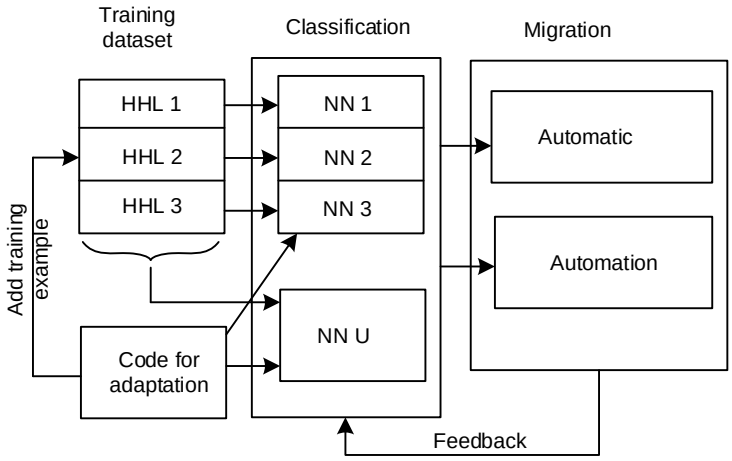

Figure 1: Functioning of a neural network

The variant with dividing the classification block into analysis components of one selected programming language seems to be more expedient for building layouts and testing technology due to less demanding computational resources.

In addition to the starting set of source program texts for training the neural network, in the process of work, texts are used that have passed the classification and further steps to adapt the program code to the conditions of execution in high-performance cluster. The input data for such secondary learning is the program text itself (in Fig. 1 - the code for adaptation), as well as data on further steps for adaptation (feedback), while the program text is added to the original set of training examples as an additional marked-up component.

The business process for adapting the existing program code and its subsequent execution on a hybrid high-performance cluster includes the following stages [8]:

1. classification of applications;
2. preparation of the executable code;
3. creating tasks for calculators;
4. organization and maintenance of the queue of tasks;
5. preparation of the computing environment;
6. execution of the calculation;
7. provision of calculation results.

The adaptation of the program code is carried out at stages 1 and 2. During the execution of these stages, both the preparation of the executable code itself and the preparation of a description of the necessary computing environment for the execution of this code should be performed. The environment itself according to this description is formed at stage 5 and will be discussed below.

To adapt the compiled program code, the following sequence of actions should be performed depending on the classification results performed at the first stage.

For architecture independent applications, you need to define:

- a list of modules and software components, runtime environments to support the application language (for example, Python, TensorFlow);
- a list of system libraries required to support the software components of the application software and environment;
- a list of required device drivers and software components (for example, GPU and CUDA drivers).

Since applications do not depend on the computing platform, the actions for their adaptation are reduced to determining the lists of the specified components and forming on their basis the necessary execution environment using virtualization mechanisms (see below).

For applications that are mobile within the computing architecture, it is necessary:

- to compile the source code into the target architecture code;
- to link the target code with system and application software libraries;

- to determine the composition of the necessary software modules and system libraries for the formation of the execution environment of the adapted code.

Note that for both types of applications, one of the results of the adaptation process is the description of the composition of the runtime environment in the form of a set of libraries, program modules and drivers. This description is used to create a runtime for a custom application in a cloud infrastructure.

For architecture-dependent applications, adaptation to the hybrid computing cluster is determined by an individual approach and comes down to replacing the architecture-dependent code with its functional analogue within the target architecture. In fact, such work is a new development of the program code or its components and cannot be considered an adaptation. Therefore, the issues of adapting such applications to operating conditions on a hybrid high-performance computing system are beyond the scope of this article.

The list preparation of the required software libraries and components can be performed both automatically and in an automated mode with human participation.

Automatic mode is preferable to use for adaptation of tasks using common software, for which there is a description of the dependencies of program modules and libraries. In this case, runtime tools (for example, high-level language interpreters) have the ability to build a tree of required components and automatically generate a complete list of dependencies.

In the case of automated (with human participation) compilation and linking of applications, when there is no exact description of the composition of software modules and dependencies, the actions for the selection of the necessary software libraries are carried out by the developer during the interactive process of building, linking and debugging.

In both cases, an executable code is created in the target language of the executing system, as well as a description of the full composition of the environment (program modules and libraries). During the execution of a custom application, based on this description, virtualization tools create an individual application execution environment.

3. Adaptation of the program code to the architecture of a hybrid high-performance cluster

The preferred mode of operation of a hybrid computing cluster in a multi-user mode is parallel execution of user scientific tasks. The practicality of using this mode is explained by the fact that the cluster has heterogeneous computing resources available for user applications. However, as a rule, the applications themselves use only one type of resource with maximum efficiency, while others are weakly loaded or generally idle. Such a picture is observed when using central processors and computing accelerators to solve scientific problems. As a rule, algorithms of applied tasks are designed for parallel computing only on a CPU or GPU, and the simultaneous full load on both types of processors is not achieved. Therefore, it makes sense to organize the computational process to ensure the parallel execution of user tasks, which leads to the utilization of all the resources of the hybrid cluster.

As shown above, for each user application, a complete list of program modules and runtime libraries is created. It should be noted that the composition of the modules required for the functioning of various user applications may be incompatible with each other, which makes it impossible for several user applications to function within one runtime environment.

The simultaneous formation and execution of a group of such environments within a single computing cluster can be carried out using virtualization tools. In this case, for each application, its own computing environment is formed, within which the isolation of software modules and libraries is provided.

If we consider various approaches to virtualization, then it can be noted that the least resource-intensive technology is the technology of containers, which allows you to create an environment for an application process according to a description file.

In fact, the container can be created dynamically when the job is loaded for execution. The basis for the container description file is the description of the set of software modules obtained during the adaptation of the application.

Based on the description, the structure of the runtime environment is formed, which includes sequentially: the base OS, specialized device drivers, interface libraries of software components for parallel computing, integrated development and execution environments, specialized software packages for applied scientific research. Figure 2 shows an example of containers that include various software stacks designed to run different applications.

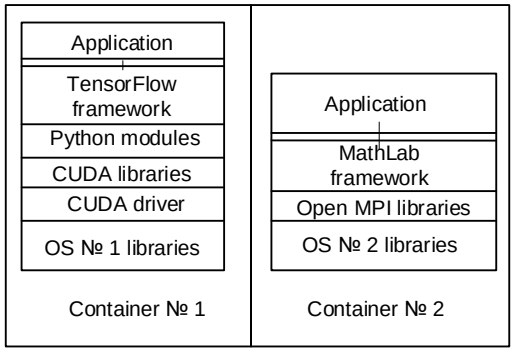

Figure 2: Examples of containers for running custom applications

In a hybrid compute cluster, multiple instances of each container type can run in parallel, utilizing resources more fully. It should be noted that for the parallel execution of several instances of the same task within the framework of a scientific calculation, the task code must be adapted to the conditions of parallel processing. For example, use the MPI interface or other mechanisms to implement parallel computing.

4. Algorithms for resource management of a hybrid high-performance cluster

To ensure the parallel execution of various applications using virtualization based on container technology, the cluster should have a computational task management system that implements the task queues and service policies, taking into account the computational resources used.

The presence of such computing resources as graphics accelerators in the cluster imposes additional requirements on the operation of the task management system in comparison with the classical architecture on the CPU. The graphics accelerator is exclusively allocated to a task for a certain time slot, and switching between tasks using the GPU is a long process and reduces the performance of the cluster.

Another feature of the operation of the hybrid cluster is that tasks that perform real calculations and short test jobs necessary for debugging and testing the code, but also requiring GPU resources, must be simultaneously executed on its resources. Such tasks require a higher priority for execution, since they function almost interactively with the users of the cluster.

Based on the listed conditions, an algorithm for managing the resources of the hybrid complex is proposed (Figure 3). The key point in this procedure is to evaluate the estimated execution time. The initial estimate can be based on the estimated time of program execution declared by the user, however, in addition to such an estimate, it is useful to use statistical data on the launch time of the container of a particular task in real and test debugging runs.

In the case of launching several instances of container, each of them is exclusively allocated a different GPU in order to avoid switching between different instances of a container of the same type.

The use of such an algorithm for determining the availability of resources in combination with policies for servicing queues with different priorities will ensure the most complete utilization of both CPU and GPU. It also provides an acceptable response time to the launch of short test and debugging tasks. Simultaneously with this, jobs are performed that perform real calculations.

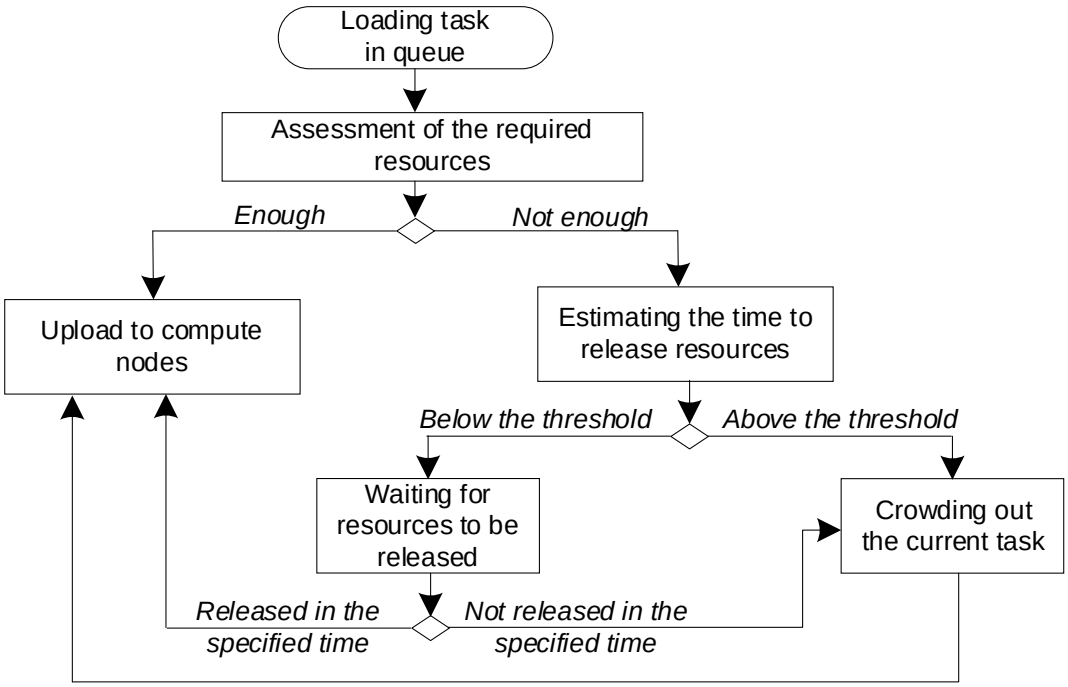

Figure 3: Algorithm for managing the resources of the hybrid cluster

5. Conclusions

The article discusses the issues of adapting software intended for scientific calculations to functioning in a hybrid high-performance computing cluster, as well as creating a virtual environment for executing tasks and managing the resources of this cluster.

It is proposed to use scripts for the automated adaptation of the program code, including classifications of the program code by the degree of mobility using a pretrained neural network. On the basis of the classification carried out, scenarios of automatic and automated adaptation of the program code to the operating conditions on a high-performance computing cluster are proposed.

Approaches have been developed to create an individually configured execution environment for applied tasks using a computing virtualization mechanism based on container technology. Containers are created dynamically at the time of loading tasks for computation according to the descriptions obtained at the adaptation stage.

Methods and an algorithm for managing the resources of a computing cluster, including graphics accelerators, in the conditions of simultaneous parallel execution of applied scientific tasks of various types, as well as testing and debugging tasks, are proposed.

The approaches and algorithm considered in the article can be used as a basis for building a high-performance hybrid computing cluster, which provides research teams as a cloud service with tools for adapting the program code to the conditions of this cluster, an individual virtual execution environment and computing resources for scientific calculations.

6. Acknowledgements

The research is partially supported by the Russian Foundation for Basic Research (projects 18-29-03091, 18-29-03100).

Experiments on the management of individual virtual environments in hybrid HPC cluster were carried out using the infrastructure of the Shared Research Facilities «High Performance Computing and Big Data» (CKP «Informatics») of FRC CSC RAS (Moscow) [12].

7. References

[1] Ding F., Mey D., Wienke S., Zhang R, Li L. A Study on Today's Cloud Environments for HPC Applications // Cloud Computing and Services Science: Third International Conference, CLOSER 2013 (Aachen, Germany, May 8–10, 2013). – Berlin: Springer, 2014. P. 114–127.

[2] Volkov S., Sukhoroslov O. Simplifying the Use of Clouds for Scientific Computing with Everest. Procedia Computer Science, 2017. Vol. 119. P. 112–120.

[3] Wu W., Zhang H., Li Zh., Mao Ya. Creating a Cloud-based Life Science Gateway // e-Science and the Archaeological Frontier: 2011 Seventh International Conference on eScience. – Piscataway, USA: IEEE, 2011. P. 55–61.

[4] Frenkel S., Khankin D., Kutsyy A. Predicting and Choosing Alternatives of Route Updates per QoS VNF in SDN // Proceedings of 16th IEEE International Symposium on Network Computing and Applications (NCA 2017). – Piscataway, USA: IEEE, 2017. P. 423–428.

[5] K. I. Volovich, A. A. Zatsarinnyy, V. A. Kondrashev, A. P. Shabanov. Scientific research as a cloud service. Systems and Means of Informatics, 2017. Vol. 27. Issue 1. p. 73–84.

[6] Zatsarinny A.A., Gorshenin A.K., Kondrashev V.A., Volovich K.I., Denisov S.A. Toward high performance solutions as services of research digital platform. // 13th International Symposium on Intelligent Systems, INTELS 2018; St. Petersburg; Russian Federation; 22 October 2018 through 24 October 2018 // Procedia Computer Science. Volume 150 (2019). p. 622-627.

[7] Top 500. The List. November 2020. https://top500.org/lists/top500/2020/11/.

[8] Zatsarinny A.A., Gorshenin A.K., Kondrashev V.A., Volovich K.I., Denisov S.A. Toward high performance solutions as services of research digital platform. // Procedia Computer Science. Volume 150 (2019). p. 622-627.

[9] Zatsarinnyy A. A., Stepanov P. V., Gorshenin A. K., Volovich K. I., Kondrashev V. A. Management of scientific services as the basis of the national digital platform "Science and Education". Strategic priorities, 2017. Vol. 2 (14). P. 103-113.

[10] Zatsarinny A.A., Kondrashev V.A., Sorokin A.A. Approaches to the organization of the computing process of a hybrid high-performance computing cluster in the digital platform environment // CEUR Workshop Proceedings. Volume 2426 (2019). p. 12-16

[11] Zatsarinny A.A., Kondrashev V.A., Suchkov A.P. The system of scientific services as an actual component of scientific research // Systems and means of informatics. 2019. Vol. 29, Issue 1, p. 25-40

[12] Volovich K.I., Denisov S.A. and Malkovsky S.I. Formation of an Individual Modeling Environment in a Hybrid High-Performance Computing System // Russian Microelectronics, Volume 49, Issue 8, December 2020, P. 580-583