High Performance Computing in a Shared Virtual Infrastructure

Konstantin Volovich[a], Alexander Zatsarinnyy[a], Sergey Frenkel[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 discusses the issues of providing users with computing resources of a hybrid high performance computing (HPC) cluster. The architecture and algorithms of the workload management system are proposed, which allows you to simultaneously perform tasks on the HPC cluster that require various software and technologies. The classification of user software systems, the workload management system architecture, and the algorithms for its functioning are given. To perform mathematical modeling tasks in a corporate computing facility for shared use, queuing schemes, algorithms for managing computing tasks and methodological approaches to assessing the effectiveness of various methods of task management are proposed.

Keywords 1
HPC cluster; hybrid architecture; graphics accelerator; workload management system; CPU; GPU; MPI

1. Introduction

The classical way to organize high-performance computing is to parallelize a mathematical task on several nodes of a supercomputer. The classical way for high-performance computing running is based on parallelizing a mathematical job on several supercomputer nodes. Advances in microelectronics, virtualization technologies, and hybrid architectures are making it possible to create high-performance computing systems for solving mathematical problems in the framework of cloud computing in the data center. Such data centers represent an evolutionary development of the computer systems of enterprises and scientific centers, which have replaced the separated computer systems of individual departments, previously used to achieve highly specialized goals.

This centralization of enterprise computing resources requires new approaches to task management, since different departments may have tasks that differ in the requirements for computing architectures and components, as well as the composition of the software used. At the same time, the requirements are often so different that combining them in one computing unit turns out to be impossible without serious reconfiguration of the computing cluster. In this case, the task of simultaneously fulfilling the jobs of different scientific groups seems unsolvable.

In [1], the class of tasks requiring the creation of a specific runtime environment designed for parallel execution on a hybrid high performance computing cluster (HHPCC) is referred to as convergent tasks. Such tasks require convergence on one computing unit of mathematical modeling different computing technologies, libraries, frameworks, which may be incompatible with each other or their joint functioning is difficult. Such incompatibility is caused, as a rule, by different versions of the required software products.

One of the directions of ensuring the parallel execution of tasks is the use of virtualization technologies, containerization (dockers), exclusive allocation of resources to one task with control of the duration of execution and resource sharing by means of the operating system. The implementation

VI International Conference Information Technologies and High-Performance Computing (ITHPC-2021), September 14–16, 2021, Khabarovsk, Russia
EMAIL: kvolovich@frccsc.ru (A. 1); azatsarinny@frccsc.ru (A. 2); sdenisov@frccsc.ru (A. 4)
ORCID: - 0000-0003-0227-688X (A. 2)

of such technologies objectively requires the creation of a control system for a converged computing environment.

The purpose of this article is to study methodological approaches to creating a control system based on the proposed methodological approaches to the classification of computational tasks in accordance with the requirements for a computing environment, as well as algorithms for controlling computational tasks when they are executed in parallel in a convergent system.

The methodological approaches and algorithms proposed in the article are based on the experience of creating and using the Shared research facilities CKP "Computer Science" of the FRC CSC RAS [4].

2. Classification of methods for creating a user's computational task

Depending on the task, converged user software can use various software systems in the converged computing environment.

The first use case is to call standard procedures and utilities of the hybrid HPC cluster:

compilers

CUDA graphics accelerator interface;

Means of supporting interprocess interaction MPI of various manufacturers;

other system applications.

Such applications in the converged computing environment will be called simple-application.

The second option for using the computing environment is to install additional software systems that require user rights.

In this case, users can deploy software systems such as anaconda, various Python modules, compilation and code generation tools.

Such applications in the converged computing environment will be called user-application.

The third option is the formation of a completely autonomous runtime environment through the use of containerization technology. This technology allows, on the one hand, to create an individual execution environment, and on the other, it does not require large computing resources for its maintenance. In fact, an individual container from the point of view of the operating system of a compute node is one process. Container technology allows the user to create their own runtime environment into which they can install any root application.

Such applications in the converged computing environment will be called root-application.

In terms of cloud computing, PaaS is provided for all three types of tasks from the HPC cluster [[5]-[6]]. Provides access to the basic computing environment and provides the opportunity to improve it by deploying additional software. However, depending on the type of converged application, the deployment algorithm will be different.

3. Deployment of runtime environment for simple-application

To prepare in the converged computing environment an application of the simple-application type, the user uses software components and libraries already deployed on the computing nodes of the cluster. In this case, the user's actions to prepare a computational task consist of the following stages:

Download software codes in storage;

Compilation of program codes for target system architecture;

Layout with system and application libraries;

Test run calculations online.

Figure 1 shows the deployment algorithm for a simple-application convergent application. The algorithm is performed by the user either interactively or using pre-designed scripts.

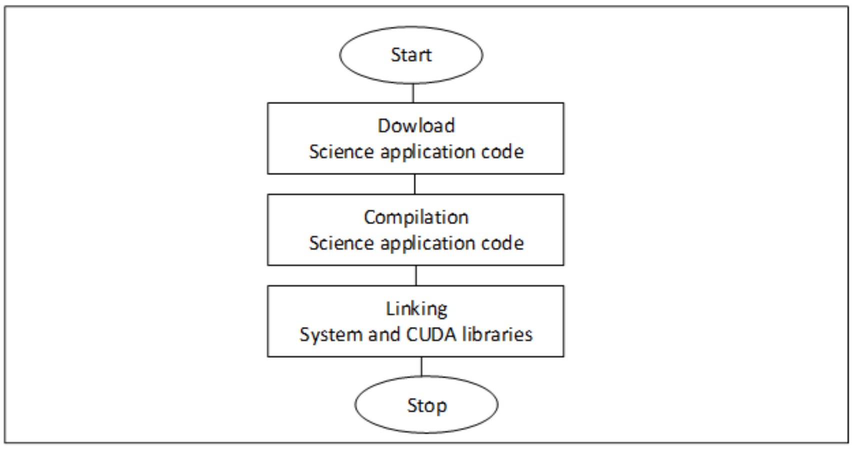


Figure 1: Simple-application deployment

After the deployment and testing in interactive mode are completed, the task is ready to be downloaded to computing nodes using the workload management system.

User-applications in the converged computing environment use both software installed on compute nodes and additional software installed by the user using his privileges.

With this option for deploying the software environment, the user loads software packages and source codes of all necessary libraries, frameworks, and development tools into the cluster repository. After that, he deploys these components in the storage allocated to him. Compiles the source code, performs the necessary settings of the software environment.

After completion of the preparatory procedures, the user compiles and links the application intended for performing science calculations.

Then the user launches the application in interactive mode for testing and debugging. In the case of normal functioning of the application, it can be uploaded by the workload management system to the computing nodes of the hybrid HPC cluster.

Using converged applications of the user-application type is more convenient than simple-application, since it provides a certain flexibility in the formation of the runtime environment and does not create additional burden on the administrator of the hybrid HPC cluster. All settings are performed using user privileges and do not affect the cluster system software environment.

Figure 2 below shows the algorithm for user actions when deploying a converged user-application application.

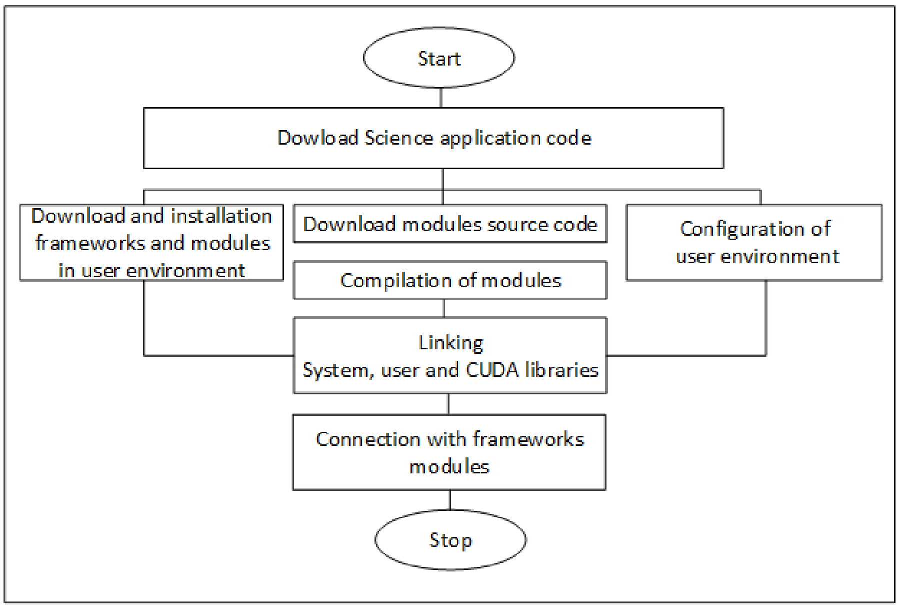

Start
Dowload Science application code
Download and installation frameworks and modules in user environment
Download modules source code
Configuration of user environment
Compilation of modules
Linking
System, user and CUDA libraries
Connection with frameworks modules
Stop

Figure 2: User-application deployment

When implementing this deployment algorithm, it is necessary to pay attention that for the formation of a user environment, it may be necessary to download software and source codes from repositories located on the Internet. Such a download may carry risks in terms of information security.

Applications that require installation of software with root privileges are the most flexible in terms of creating a software environment for mathematic calculations. These applications require the use of additional technologies compared to the basic ones provided by linux family operating systems.

In the HPC hybrid FRC CSC RAS cluster, docker technology is used in combination with the NVidia CUDA software package and the SLURM [7] job management system to create an individual runtime environment for applications of the root-application type [8].

When deploying such a software environment, the user performs the following actions:

- selects the base image of the container from the HPC cluster repository;
- forms the list of necessary soft for installation and deployment of an individual runtime environment;
- creates a command script that describes the deployment of software.
- determines the necessary resources of the computing complex (GPU, RAM, CPU Cores) and requests them from the administrator of the HPC cluster;
- the administrator generates a description of the computing environment of the container for this converged task and includes it in the configuration of the workload management system (the description can be formed new or selected from the template);
- the user makes a test run of the application in interactive mode. In this case, the base container is deployed in the cluster. The container runs a command script that downloads, compiles, and installs the software components. If necessary, the installation is performed with root privileges. In the formed environment, a test launch of a scientific application is performed.
- in the case of successful execution of test launches, the virtual environment of the convergent application of the root-application type can be downloaded for execution to the computing nodes of the hybrid HPC cluster.

Figure 3 shows the algorithm of user and administrator actions when deploying a converged root-application.

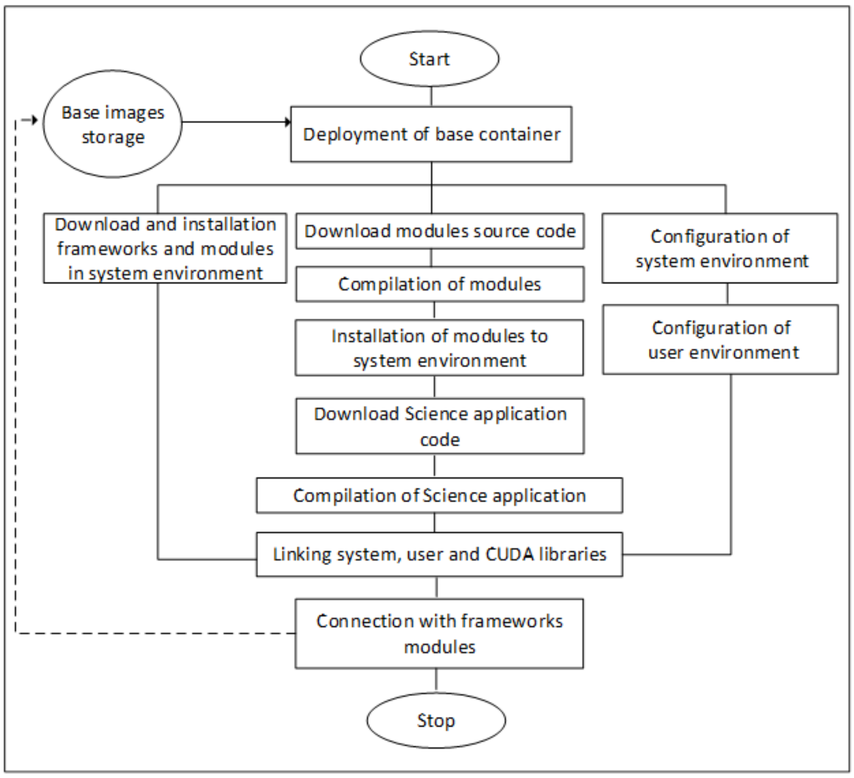

Start
Base images storage
Deployment of base container
Download and installation frameworks and modules in system environment
Download modules source code
Configuration of system environment
Compilation of modules
Configuration of user environment
Installation of modules to system environment
Download Science application code
Compilation of Science application
Linking system, user and CUDA libraries
Connection with frameworks modules
Stop

Figure 3: Root-application deployment

Note that the technology for deploying an individual runtime environment based on virtual containers is the most flexible and allows you to create almost any software environment that provides a custom application. On the other hand, such an algorithm creates a load on the administrator of the hybrid HPC cluster and does not allow to implement it completely automatically.

Full automation of the user interaction process with the job management system of a high-performance cluster may be required if there is a flow of new users and a large number of convergent tasks. In a HPC cluster performing scientific calculations, this trend is absent. Cluster users do not change frequently. Scientific teams gain access to the system, and perform scientific calculations for a long period. Based on the operating experience of the hybrid HPC cluster FRC CSC RAS, it can be concluded that the lack of full automation when creating the runtime environment for the root-application is not an obstacle to the provision of high-performance computing services.

4. Architecture of workload management system

The architectural components of the workload management system should provide the following functions:

- providing the user access with certain privileges for loading data, launching system and application utilities that provide processing of the source code and execution of the applications;
- providing the user with an interface to the controls for computing tasks;
- job queue management;
- ensuring interactive jobs
- deployment of virtual containers and transferring command files to them for tuning an individual runtime environment;
- prioritization of tasks;
- maximizing the load of the hybrid computing cluster.

Figure 4 shows the architecture of the workload management system.

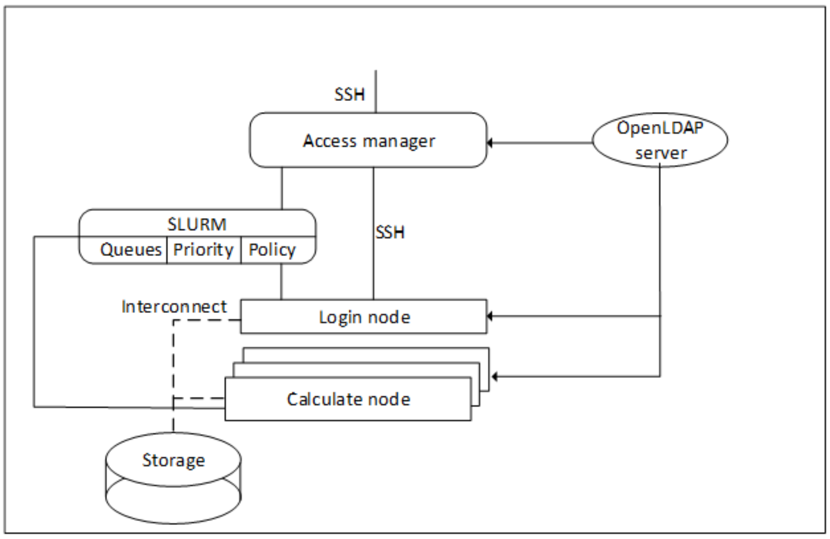


Figure 1: Workload management system architecture

The “Access manager” component provides user access, maintains a single user’s database and based on the basis of the OpenLDAP service. The use of this software allows you to connect the user via SSH to one of the cluster components - login node. This is the only cluster component that the user has direct access to. It deployed all the development and debugging tools for applications, as well as a workload management system that manages computing tasks. The login node is connected to other nodes of the computing cluster and data storage by the high-speed interconnect network.

In the hybrid HPC cluster FRC CSC RAS, one of the computing nodes acts as the login node, but, in the general case, it can be a dedicated server. The OpenLDAP server operates separately in the FRC CSC RAS infrastructure.

“Access manager” also provides end-to-end user authentication between the login server and compute nodes for parallel execution of tasks using MPI technology [9].

Computing jobs are managed by the SLURM component, which provides the user with a command line interface for managing computing jobs. This component allows you to perform tasks both interactively and in the background. Interactive mode intercepts standard input / output of applications and transfers it to the user's command line. It does not matter on which of their cluster nodes the application is running, standard output is transmitted to the login node in the user terminal. Interactive mode allows you to debug and configure applications, their preparation for launch in batch.

The main mode of calculation is the background mode. SLURM allow you to perform tasks in the background, regardless of the state of the user's terminal session. In the background, standard input and output are not intercepted; to obtain the results of calculations, they must be written to the file storage.

SLURM provides an interface for launching applications, viewing a list of computational tasks and their status, and deleting computational tasks.

This component supports all three types of converged jobs.

The tasks of the simple-application and user-application types are executed by directly executing binary codes or scripts on one or several cluster nodes. A single network file storage provides access to the codes, libraries, and modules of each computing node.

The execution of tasks of the root-application type is carried out by deployment one or several instances of containers and passing the script to configure the individual runtime environment to them. For the functioning of such a mechanism, SLURM provides for the allocation of computing resources to a container based on a template that is configured by the administrator. Deployment of containers is possible on any node or group of nodes. Therefore, each node must have access to the repository of basic container images.

The job queue management module is a component of SLURM and supports three queues:

- interactive queue;
- normal queue;
- longtime queue.

The user through the command line interface can place tasks in any queue.

An interactive queue provides the highest priority for a computational job to reduce latency when debugging programs. The queue’s functioning time is limited to 10 minutes. This allows you to reduce the load on the cluster and use its resources more efficiently.

Normal queue provides the basic priority of a computational job. The queue’s functioning time is limited to 4 hours. After that, the task moves to a longtime queue.

Longtime queue provides lower priority for job execution. The task in the queue can be arbitrarily long until the calculations are completed or until termination by the user or administrator.

The task prioritization module allows moving tasks between queues, starting and pausing tasks.

According to the algorithm for changing the priority of the task, the jobs from the «normal» queue are loaded for execution until the cluster resources are exhausted or the «normal» queue is empty. Then tasks are loaded from the «longtime» queue. If at the same time there are tasks in the «normal» queue, they crowd out the previous ones and are put to execution. When waiting for jobs in the «longtime» queue, twice as long as «normal» jobs run, they are transferred to the «normal» queue. Then there is the crowding out of tasks. This ensures priority execution of tasks requiring a short execution time, as well as guaranteed performance of tasks requiring lengthy calculations. It is also possible to use dynamic priorities [10].

The cluster load management module is part of the SLURM software system. It monitors the use of cluster resources. Based on the information received, this module transfers tasks from queues to the computing nodes of the cluster. The algorithm of the module provides for the loading of tasks in the case of sufficient computing resources. This is done in order to avoid sharing resources with the operating system, which leads to serious performance losses, especially when using graphics accelerators.

In general, the architecture of the control system of the computing process makes it possible to successfully perform tasks of scientific calculations in a converged environment of a hybrid high-performance computing cluster. The simultaneous execution of several tasks is ensured by the distribution of tasks between computing nodes and the sharing of resources of one node between tasks.

Improving of the computing resources efficiency

It is natural to strive to find ways to increase the efficiency of using the resources of the computing cluster. Some approaches are discussed below.

Traditionally, in high-performance computing on supercomputers, all the resources of one or several nodes of a computing cluster are allocated for performing a mathematical task. Using a converged computing environment allows you to allocate the required amount of resources for application containers on each of the nodes (CPU cores, GPU, RAM, etc.).

In this regard, it is relevant to stimulate users to tune computational algorithms for efficient workloading of computational resources. For this, it is advisable to use a queuing policy with relative priorities, which does not consume resources for interruption and additional service.

The priority should be a function of the quality of the algorithm and program code. The best practice to stimulate improvement in the quality of the algorithm is to use non-linear dependencies for this, for example, parabolic or exponential. In practice, you can use piecewise-linear approximation of these functions (see Figure 5) with areas understandable for application users: pure, good, excellent.

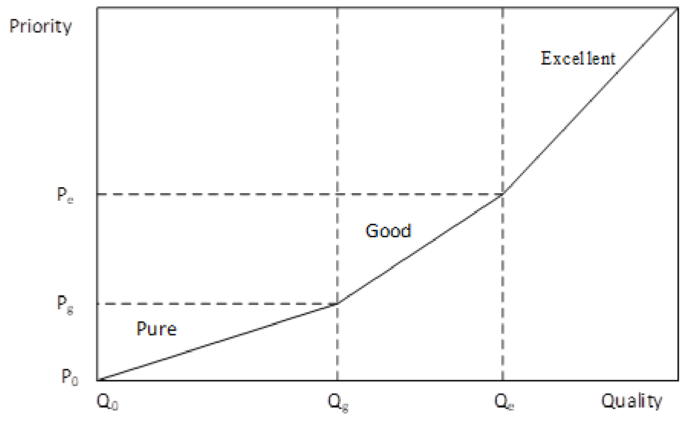


Figure 2: Function of priority

It is useful to represent the quality of using the resources of a computing cluster by an application using the Roofline model [11]. The model allows one to show the performance of the algorithm components (procedures and loops) under the natural limitations of the computing system - memory performance and peak performance of the computing unit.

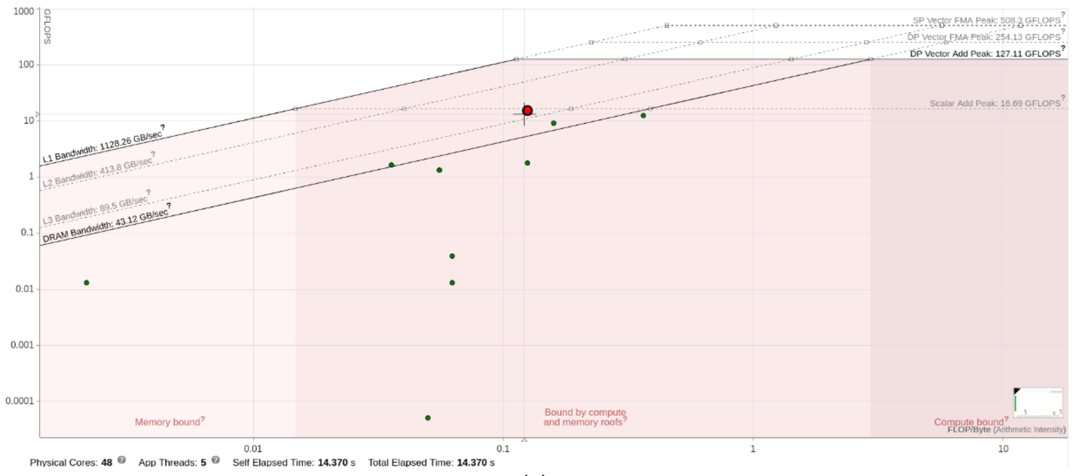
Figure 6 shows examples of Roofline model images for an application using Fourier transform operations (a) and sorting with random memory access (b). The model allows, depending on the technology used, to set different performance boundaries. Experiments with tests from the NPB package [12] showed that both RAM and computational resources (CPU Cores, GPU) are the boundaries for application performance.

So, from the diagram (Figure 6 a) it can be seen that the limiting factors for the main application loop are the processor performance for scalar operations (Scalar Add Peak) and the performance of the L2 memory cache. Therefore, the main direction of optimization is the transition to vector technologies of the computing architecture of the complex and the restructuring of the application in such a way as to use the functioning of the first-level cache. In this case, the main program loop will be shifted vertically up the diagram, which will mean more efficient use of the computing architecture's capabilities in terms of floating-point performance. Another direction of optimization may be to reduce the ratio of floating-point operations and the amount of memory exchange. In this case, the main program cycle will be shifted to the right in the diagram. In this case, the transition to vector technologies (moving up the diagram) can be carried out without using memory caching.

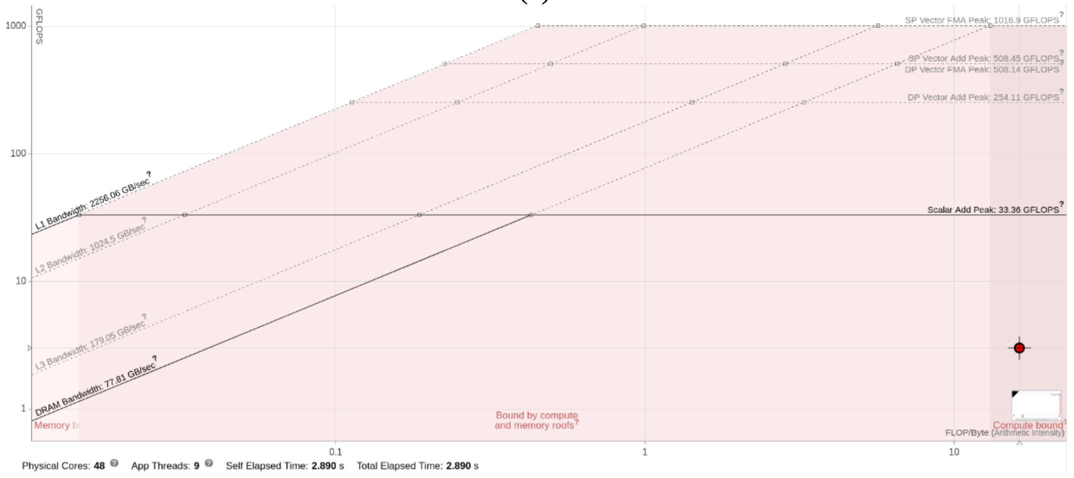
From the diagram (Figure 6 b) it can be seen that the main application loop is in the area in which the ratio between the performance of the memory and the computational unit is such that it is enough to use RAM without caching for any computing technology - scalar or vector. At the same time, the use of the performance of computing units by the application is not optimal. There is a multiple performance margin for scalar technology and even more significant for vector technology.

In general, Roofline models leverage the best algorithms improvement practices published by computational benchmark developers.

The results of evaluating the performance of applications in the Roofline model are the initial data for assigning a priority in the SLURM job management system.



(a)



(b)

Figure 3: Roofline models for evaluating application performance

5. Conclusions

The solution of heterogeneous scientific problems in the converged environment of the HHPCC demonstrates high flexibility and the ability to adapt to the conditions of the applied problem through the use of docker virtualization tools.

As experience shows, the main type of converged applications used in the CKP “Computer Science” is root-application, and when providing mathematic modeling services, it is simple-application. User-application software systems are practically not used, which is associated with the complexity and high requirements for software environments designed for solving scientific and practical problems.

Stimulating efficient custom algorithms for mathematical modeling through the application of a priority policy, as a result, allows servicing a larger number of calculation jobs. In this case, the efficiency of using the computing resources of the computing cluster is increased. This makes it possible to serve a larger number of research teams in various fields of science and technology using a converged high-performance environment.

Thus, on the example of CKP "Computer Science" it can be argued that the main directions of providing hybrid HPC cluster services for converging converged tasks are:

- centralization of resources;
- cloud computing;
- virtualization;
- workload management, converged environment high-performance computing resources share taking into account the priority stimulation of efficient applications;
- providing mathematic modeling as a service.

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) [9].

7. References

[1] Volovich K.I., Shabanov A.P., Malkovsky S.I. Converged computing in a hybrid HPC cluster. Highly Available Systems. 2020. V. 16. № 2. P. 22–32. DOI: 10.18127/j20729472-202002-02 (In Russian).

[2] Abramov S.M., Lilitko E.P. The status and development prospects of computing systems of super-high performance. Information Technologies and Computing Systems. 2012. Moscow. No. 2. p. 6-22.

[3] 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. 2019. Volume 150. p. 622-627.

[4] Regulations of CKP "Computer Science". Available online: http://www.frccsc.ru/ckp (accessed on 18.11.2020).

[5] Zatsarinny A.A., Gorshenin A.K., Volovich K.I., Kolin K.K., Kondrashev V.A., Stepanov P.V. Management of scientific services as the basis of the national digital platform "Science and Education". Strategic priorities. 2017. No. 2 (14). p. 103-113.

[6] V.A. Kondrashev, K.I. Volovich Service management of a digital platform on the example of high-performance computing services. Materials of the International scientific conference. 2018. Voronezh, September 3-6

[7] SLURM Workload manager. Available online: https://slurm.schedmd.com/ (accessed on 18.11.2020)

[8] K.I. Volovich, S.A. Denisov, S.I. Malkovsky. The formation of an individual modeling environment in a hybrid high-performance computing complex. News of higher educational institutions. Materials of electronic equipment. 2019. 22 (3).

[9] Gorchakov A.Yu. Using OPENMP to implement the multithreaded method of uneven coatings. Advanced Information Technologies (PIT 2018) proceedings of the International Scientific and Technical Conference. 2018. p. 613-617.

[10] Volovich, K.I., Denisov, S.A., Shabanov, A.P., Malkovsky, S.I. Aspects of the assessment of the quality of loading hybrid high-performance computing cluster. CEUR Workshop Proceedings. 2019. Volume 2426. p. 7-11

[11] S. Williams, A. Waterman, D. Patterson. Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM. 2009. Vol. 52, No 4. p. 65-76.

[12] NAS Parallel Benchmarks (NPB). Available online: https://www.nas.nasa.gov/publications/npb.html (accessed on 18.11.2020)