UDC 004.272.2:004.75
DOI: 10.15587/1729-4061.2021.249462

THE DEVELOPMENT OF THE SYSTEM FOR ARC NORDUGRID BASED GRID-COMPUTING ORGANIZATION USING VIRTUAL ENVIRONMENTS OF THE DOCKER PLATFORM

Olga Prila
Corresponding author
PhD, Associate Professor*
E-mail: olga.prila1986@gmail.com
Volodymyr Kazymyr
Doctor of Technical Sciences, Professor*
Volodymyr Bazylevych
PhD, Associate Professor, Head of Department*
Oleksandr Sysa*
*Department of Information and Computer Systems
Chernihiv Polytechnic National University
Shevchenka str., 95, Chernihiv, Ukraine, 14027

The study of modern frameworks and means of using virtualization in a grid environment confirmed the relevance of the task of automated configuration of the environment for performing tasks in a grid environment.

Setting up a task execution environment using virtualization requires the implementation of appropriate algorithms for scheduling tasks and distributed storage of images of virtual environments in a grid environment. Existing cloud infrastructure solutions to optimize the process of deploying virtual machines on computing resources do not have integration with the Arc Nordugrid middleware, which is widely used in grid infrastructures. An urgent task is to develop tools for scheduling tasks and placing images of virtual machines on the resources of the grid environment, taking into account the use of virtualization tools.

The results of the implementation of services of the framework are presented that allow to design and perform computational tasks in a grid environment based on ARC Nordugrid using the virtual environment of the Docker platform. The presented results of the implementation of services for scheduling tasks in a grid environment using a virtual computing environment are based on the use of a scheduling algorithm based on the dynamic programming method.

Evaluations of the effectiveness of the solutions developed on the basis of a complex of simulation models showed that the use of the proposed algorithm for scheduling and replicating virtual images in a grid environment can reduce the execution time of a computational task by 88 %. Such estimates need further refinement; it is predicted that planning efficiency will increase over time with an increase in the number of running tasks due to the redistribution of the storage of virtual images

Keywords: grid, cloud computing, virtualization, task scheduling, replication

Received date 02.11.2021
Accepted date 03.12.2021
Published date 29.12.2021

***How to Cite**: Prila, O., Kazymyr, V., Bazylevych, V., Sysa, O. (2021). The development of the system for ARC Nordugrid based grid-computing organization using virtual environments of the Docker platform. Eastern-European Journal of Enterprise Technologies, 6 (2 (114)), 117–124. doi: https://doi.org/10.15587/1729-4061.2021.249462*

1. Introduction

The use of distributed computing to perform large-scale tasks in various applied industries is relevant and cost-effective. Grid technologies are used to perform computations based on distributed decentralized computing resources. The problem of using decentralized resources of the grid environment is the complexity of setting up and configuring the environment for executing tasks. Cloud technologies are focused on centralized configuration of the task execution environment. The combination of grid and cloud computing concepts for interoperability and integration began in 2010 [1]. There is no significant incompatibility between the definitions of grid [2] and cloud [3], the main difference is in purpose and scale.

This confirms the possibility of integrating these technologies, which will allow to consider grid resources as distributed and heterogeneous resources for storing data and performing computations, and cloud computing as services deployed on these resources. This principle focuses on the infrastructure aspects of cloud computing, defined as IaaS (Infrastructure as a Service). Solving the problem of interoperability between grid and cloud is a powerful business and scientific model.

The main differences between grid and cloud computing, on which the solution to the problem of interoperability should be focused, are presented in Table 1.

Table 1

Conceptual differences between grid and cloud computing

Concept	Grid	Cloud
Combining or coordinating resources from different domains	Yes	No
Computing/data warehouse abstractions	Executable Task File/Data File	Server/Application/Service/Disk Space
Quickly turn on/off virtual machines (API-based)	No	Yes
Interactivity	No	Yes
Individual runtime	No	Yes
Pricing model	Free use model (investment and maintenance)	Pay per unit of resource use

The combination of grid and cloud computing concepts will effectively solve large-scale problems of various applied industries using distributed computing resources and the ability to flexibly configure the task execution environment.

2. Literature review and problem statement

The study is focused on an overview of solutions for using virtual images in a grid environment based on the ARC Nordugrid middleware [4], which is the most popular for building grid infrastructures, on the one hand. In addition, studies of existing algorithms for scheduling tasks in a distributed environment using virtualization are presented.

ARC Nordugrid introduced the Run Time Environment, which can be defined as Docker and Singularity images [4]. However, the implemented tools do not provide the ability to use dynamic environments when performing tasks in a grid environment, since the installed RTEs must be enabled by the resource administrator and the required RTEs must be enabled before using them.

[5] proposes expanding grid systems with cloud resources, which will allow additional resources to be used when grid resources are overloaded with the HTCondor batch system and ARC NorduGrid middleware. The presented solution allows the deployment of a virtualized grid cluster using ARC middleware in any public or private cloud – the PaaS (Platform as a Service) model for running grid programs. However, the problems of scheduling the execution of tasks and monitoring of virtual machines are not solved. The need for additional research is noted to estimate the costs of sending and storing images of virtual machines.

The RainBow framework provides the ability to install software on a virtual machine using Windows and GNU/Linux. However, for the presented solution, estimates of the efficiency of performing computational tasks are not given, in particular, according to the computation time. Optimizing the placement of virtual machine images on distributed resources is also not considered.

The work [7] presents solutions for the deployment of the computational element Arc Nordugrid on the resources of the cloud infrastructure using the Open Stack software. The presented solution is focused on the use of specific software for performing computational tasks in the field of physics (ATLAS [8]) within a private cloud. The efficiency of using the proposed solutions has been proven, however, the possibility of configuring virtual images for solving other types of computational problems is not provided.

The Nimbus project [9] develops open solutions for building and deploying individual runtime environments in the cloud in order to solve scientific problems. The solutions are being developed based on the OpenStack project, are in a state of active development and maintenance, but are not aimed at providing integration with middleware for grid infrastructures.

There are solutions for mapping virtual machines to the corresponding physical resources (Virtual Machine Placement, VM-PM mapping) of cloud infrastructures with optimization according to various criteria. In particular, regarding criteria for energy conservation, load balancing, ensuring reliability, quality of service, reducing the size of virtual machine images, resource consolidation, etc. Most multi-objective optimization solutions focus on two criteria, only about 10 % of studies take into account several criteria, which is a highly complex task [10].

Methods for scheduling tasks that provide the necessary level of quality of service to users are important for the grid environment. Methods for scheduling tasks should be integrated with the mechanisms for placing tasks on virtual machines and mechanisms for placing virtual machines, respectively.

In [11] a mechanism for replication and deletion of virtual images in the case of a limited storage resource in a distributed environment is presented. It is envisaged to use a weighted algorithm for calculating priorities, taking into account the parameters of the resource cost, the number of requests to the file and the storage time of the file in the storage. This algorithm has been proven to be effective compared to LRU (least recently used) and LFU (least used) in terms of number of successful tasks, unsuccessful task execution, and cost of using storage resources. The use of the algorithm increases the number of successfully completed tasks by 15 % compared to LRU and by 10 % compared to LFU [11]. The mechanism is not focused on integration with planning methods in a grid environment, but it can be used to implement replication and redistribution of virtual machine images in a grid environment.

In [12], an approach is proposed that uses the dynamic programming method and the knapsack problem 0–1 (knapsack problem 0–1) strategy to optimize resource consumption and power consumption in a distributed environment. Physical servers are represented by knapsacks, while VMs are virtual machines – their items. To host the virtual machine, a server is selected whose parameters are most consistent with the parameters of this VM. If the server cannot accept a virtual machine deployment request, the virtual machine deployment requests will be sorted in descending order. Comparison of the proposed algorithm with the Open Stack algorithm, the default algorithm in CloudSim, the random placement algorithm and the first match algorithm is presented. The results of the assessment and comparison showed the advantage of the model developed by the authors for scheduling the placement of virtual machines and requests on the resources of the cloud environment. The proposed approach is focused on optimizing the costs and energy consumption of cloud resources. For optimization according to other criteria, in particular, according to the execution time of computations, it is necessary to adapt the presented algorithm.

In [13], the metaheuristic algorithm MOACS (Multi-objective ant colony system) is proposed, the criteria for optimizing the placement of which is to maximize the number of dismissed physical servers and minimize the number of migrations of virtual machines. Evaluations of the efficiency of the proposed algorithm for cloud storage resources are presented. For optimization according to the criterion of the execution time of calculations, which is determined by the optimization criterion within this study, it is necessary to adapt the presented algorithm.

The study showed that there are no full-fledged solutions for the use of dynamic virtual images in a grid environment, in particular, based on the ARC Nordugrid middleware. In particular, the problems of combining and coordinating resources of different domains for cloud infrastructures, interactivity and setting up a runtime environment for grid infrastructures are unresolved. The scheduling engines presented do not have integration with middleware used in grid infrastructures.

All this suggests that it is advisable to develop tools for using virtualization in a distributed grid environment, as well as methods for scheduling tasks and placing images of virtual machines, taking into account the use of virtualization technology.

3. The aim and objectives of research

The aim of research is to develop a system for organizing grid computing with the implementation of task launch services using virtual environments, mechanisms for scheduling tasks and replication services and saving virtual images. This will allow to perform computational tasks of various applied industries in the grid environment and increase the efficiency of grid computing.

To achieve the aim, the following objectives were set:

– to develop an algorithm for scheduling tasks in a grid environment using virtualization and a mechanism for replicating images of virtual machines;

– to evaluate the effectiveness of the developed algorithm for scheduling tasks using virtualization in a grid environment in comparison with a random algorithm for scheduling tasks by means of simulation;

– to extend the existing framework for performing tasks in a grid environment by using a virtual task execution environment using Docker software.

4. Materials and methods of research

The research and design of the software was carried out using systems analysis methods, object-oriented analysis methods and the UML object-oriented modeling standard. The framework was developed using the NetBeans IDE [16], the git version control system [17], and the maven project build tool [18].

To develop a task scheduling algorithm, a dynamic programming method was used.

To develop a simulation model of the process of performing tasks in a grid environment, the methods of simulation and the GridSim software were used [19]. The adequacy of the model was assessed by carrying out real experiments on the basis of the computing resource grid with the installed back-end of the Arc Nordugrid middleware [4].

5 Results of the development of a grid computing system using virtual runtime environments

5. 1. Development of an algorithm for scheduling tasks and a mechanism for replicating images of virtual machines in a grid environment

To implement the services for scheduling the execution of tasks, an algorithm is used, presented by the authors earlier and based on the method of dynamic programming [20]. The algorithm takes into account the parameters of the problem (dimension of calculations, dimension of data, etc.) and resource parameters (computational capacity, queue size, etc.). The task execution time is defined as an objective function in the algorithm.

The problem in the general case is presented in the form of a directed acyclic graph DAG (Fig. 1), the vertices of which are computational subtasks, the arcs are the dependencies between them. The algorithm assumes the introduction of a level in the structure of the problem. The level is determined by many subtasks that can be performed in parallel. For example, in the structure of the problem shown in Fig. 1, the following levels will be introduced:

1) Level 1: includes Block 1;

2) Level 2: includes Block 2, Block 3.

For dependencies between tasks, the dimension of the transferred data is indicated.

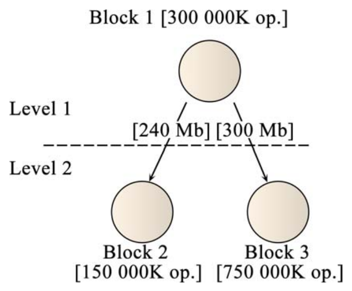


Fig. 1. The structure of the task presented in the DAG form

Sequentially for each level, the planning of the placement of subtasks is performed by exhaustive search. For each placement option, the value of the objective function is calculated, taking into account the links with the subtasks of the previous level and defined as (1):

$$f_1=\sum_{i=1}^{m_1}\left(\begin{array}{l} k_1\cdot\left(\begin{array}{l} time_{wait}+time_{exec}+\\ +time_{data\_transfer}+time_{VI\_transfer}\end{array}\right)+\\ +k_2\cdot cost\cdot time_{exec}\end{array}\right), \qquad (1)$$

$$f_j=\sum_{i=1}^{m_j}\left(\begin{array}{l} k_1\cdot\left(\begin{array}{l} time_{wait}+time_{exec}+\\ +time_{data\_transfer}+\\ +time_{VI\_transfer}\end{array}\right)+\\ +k_2\cdot cost\cdot time_{exec}\end{array}\right)+f_{j-1},$$

$$j=2,..N,$$

where $time_{wait}$ – waiting time in the queue, depending on the number of tasks in the queue for the resource;

$time_{exec}$ – execution time of the computational unit on the selected resource, which depends on the complexity of the computational task and the computational capacity of the resource;

$time_{data\_transfer}$ – time spent on data transfer;

cost – cost of using the resource per unit of time (it can be ignored for non-commercial grid environments);

k_1, k_2 – normalizing factors of the priority of time or cost of performing calculations;

m_j – the number of computing units at the level;

$time_{VI\_transfer}$ – time to transfer the virtual image.

At each level, only solutions are saved that correspond to the minimum value of the objective function of each placement option. Ineffective solutions are rejected and not considered. The optimal placement of the problem is determined by going backwards from the bottom up with the choice of optimal solutions for each placement option on the level.

The optimal solution to the problem includes optimal solutions to subproblems that determine the possibility of using the dynamic programming method.

Thus, the cost of sending the virtual image will be taken into account when choosing the optimal resource for the task. The algorithm for choosing a computational node consists in finding a node for which the value of the objective function is minimal. The method is applicable in the general case both to tasks with the structure of the "workflow" type, and to tasks of the "single computing unit" type (of the simplest type).

Replication of task runner images occurs synchronously and dynamically as to plan and select a resource to run a task. To implement the update and redistribution of the storage of virtual images, the framework services provide asynchronous replication using resource idle time. The choice of virtual images for replication is based on the Most Frequently Used algorithm by storing information about the frequency of using virtual images. Replication service runs decentralized on each server at random intervals. A block diagram of the replication service method is shown in Fig. 2.

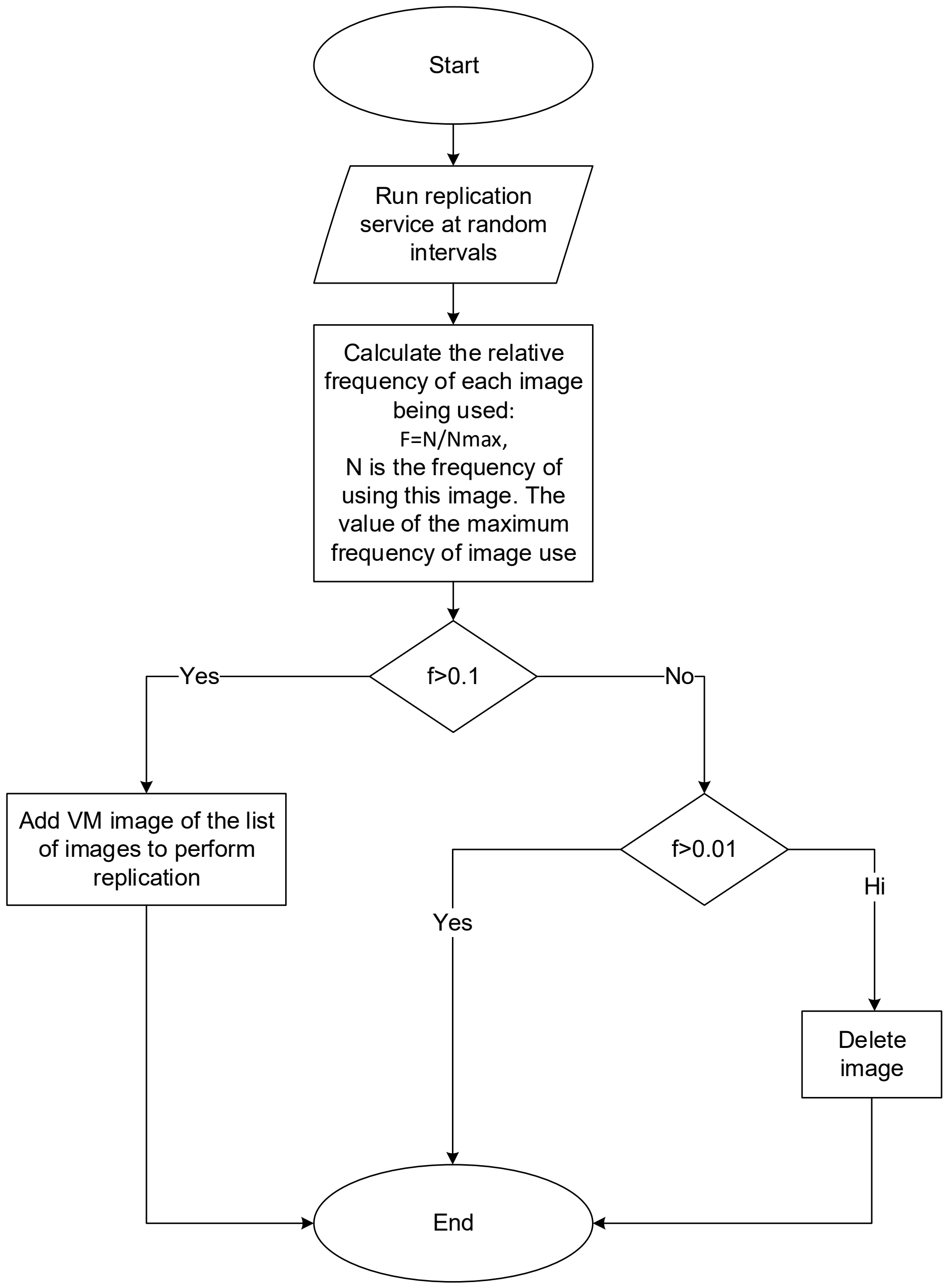
Background tasks distribute virtual images to obtain at least three copies of images added to the list of images for replication. Tasks start at a certain interval, which can be configured.

When choosing a resource to perform a task, provided the available $S_{current}$ memory size is such that $S_{current}$<S, the required number of images will be selected to be deleted from the resource according to the LRU (least recently used) principle (2):

$$S = S_b \cdot 3 + S_e, \tag{2}$$

where S_b – size of the base virtual image;

S_e – size of the executable file.



Start
Run replication service at random intervals
Calculate the relative frequency of each image being used:
F=N/Nmax,
N is the frequency of using this image. The value of the maximum frequency of image use
f>0.1
Yes
No
Add VM image of the list of images to perform replication
f>0.01
Yes
Hi
Delete image
End

Fig. 2. Block diagram of the virtual image replication service algorithm

This approach allows to consider the storage resource constraints of a distributed environment when scheduling a task.

5. 2. Evaluation of the efficiency of the task execution scheduling algorithm using virtual images in a grid environment

Evaluation of the effectiveness of the developed algorithm was carried out by simulating the execution of tasks using the developed algorithm for scheduling tasks and a mechanism for replicating virtual images and a random scheduling algorithm. To determine the parameters of the process of deploying a virtual runtime environment using Docker software (in particular, the deployment time), real experiments were carried out on the basis of a computational resource with the Arc Nordugrid server part installed. Simulation of the execution of 48 tasks with different parameters and requirements for the execution environment was carried out. The problem model is determined by the following parameters:

$$\{N, Memory, \{T\}\}, \tag{3}$$

where N – predicted number of processor instructions required to complete the task;

Memory – requirements for memory use (size, MB);

$\{T\}$ – set of links with other blocks of the task flow (communication between nodes is unidirectional).

Links are characterized by the data capacity parameter – the amount of data transferred between blocks.

Memory requirements for simplicity are omitted in Fig. 1. The predicted number of processor instructions is expressed in abstract units (K_{om}).

The structure of a grid network can be represented by a complete graph, the vertices of which are computing resources, and the arcs characterize the data transmission network between them. The model of the computing node of the grid network is represented by the following parameters:

$$\{\mathrm{CPU}, Memory, QueueSize, \{R\}\}, \tag{4}$$

where CPU – processor power;

Memory – available memory;

QueueSize – size of the queue of jobs to the resource;

$\{R\}$ – set of links with other network nodes (links between nodes are bidirectional).

Links between nodes are characterized by bandwidth.

According to the simulation results, the average time to complete tasks using the proposed scheduling algorithm was 66.4 units of time. The average execution time for the same set of tasks using a random scheduling algorithm was 302 time units. The distributed environment model was represented by 3 computing resources. The number of repetitions of experiments at the level was 10. Thus, according to the results of simulation, it can be concluded that the use of the proposed scheduling and replication algorithm can reduce the execution time of computational tasks by 88 % compared to a random scheduling algorithm.

5. 3. Extension of the existing framework by means of using a virtual environment for performing tasks in a grid environment

The developed system is an extension of the framework for organizing grid computing based on ARC Nordugrid GRID-WMS [21], built on the client-server architecture and consisting of the following modules:

– simplicate-core – contains the entity of the project;

– simplicate-dao – responsible for accessing data;

– simplicate-services – contains a layer of services responsible for the business logic of the application;

– simplicate-webservices – contains REST API and services for sending grid tasks for computation;

– simplicate-gridftp-jca is responsible for the connection between the server and ARC using GridFTP;

– simplicate-frontend provides a user interface for creating tasks, loading Docker images, checking execution status, etc.

To ensure the execution of tasks in a virtual environment, changes were made to the simplicate-webservices, simplicate-services, simplicate-dao and simplicate-core, simplicate-frontend modules of the existing framework. The framework is implemented using JAX-RS to implement the REST API, Java EE CDI to apply dependency injection, and OpenJPA is used to access data. The JavaScript framework Vue was used to develop the user interface.

The framework implements three main use cases, shown in Fig. 3.

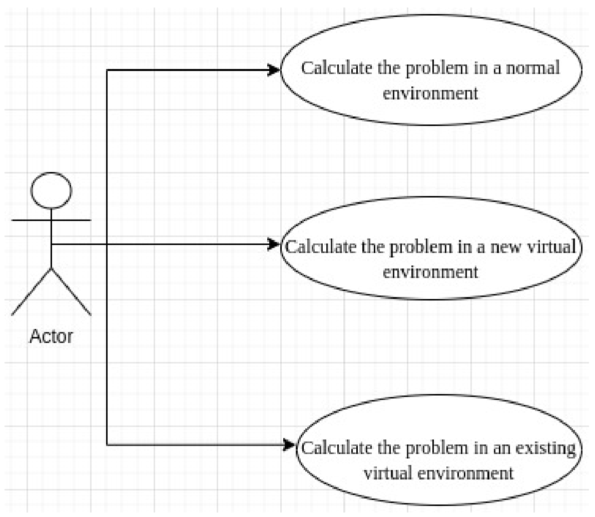

Calculate the problem in a normal environment
Calculate the problem in a new virtual environment
Calculate the problem in an existing virtual environment
Actor

Fig. 3. Scenarios for using the framework to perform tasks in a grid environment

Let's take a closer look at each of these scenarios.

The user computes a grid task without using virtualization:

– the user is authorized in the system;

– the user goes to the tasks section;

– the user creates a task;

– the user creates a node that calculates the task, loads the executable file and defines the necessary parameters;

– the user submits the task for processing.

The user computes the grid task using the new virtual environment:

– the user is authorized in the system;

– the user goes to the tasks section;

– the user creates a task;

– the user creates a node with the virtual type, which will calculate the task, load the executable file, the description of the virtual image and indicate the necessary parameters;

– the user submits the task for processing.

The user computes a grid task using an existing virtual environment:

– the user is authorized in the system;
– the user goes to the tasks section;
– the user creates a task;
– the user views the list of existing virtual images;
– the user creates a node with the virtual type according to the template, which will calculate the task, download the executable file, select the required virtual image and specify the necessary parameters;
– the user submits the task for processing.

The process of deploying images to remote computing resources is carried out using the Docker software platform. ARC Nordugrid makes it possible to use Docker as an RTE by specifying the name of the image that will be used to compute the task [22]. Docker is an operating system-level virtualization system that uses containers. In Docker, applications (processes) are, whenever possible, placed in separate containers – layers, between which there is limited interaction. A container is an isolated Docker environment for executing and running services. The containers are lightweight, do not require the work of the hypervisor, and run directly on the core of the host machine. Thus, it is possible to run more containers than virtual machines on the same physical hardware. When defining a task descriptor file, the RTE parameter specifies the name of the Docker image to use, for example: runtime environment="ENV/DOCKER" "debian".

The container registry stores images – these are immutable templates that contain instructions for creating a container. The image is defined by a config file called DockerFile, each statement in the config file creates a layer in the image. When the configuration file is changed and the process of rebuilding the image is started, only the changed layers will be rebuilt. This particular approach makes containers lightweight and fast when compared to other virtualization technologies.

The Docker Trusted Registry (DTR) is a storage and delivery system that contains named Docker images and their versions with different tags – containerized applications that run on a Docker Universal Control Plane (UCP) cluster. The default storage driver is Posix Local File System, but DTR supports settings for storage technologies such as NFS, Amazon S3, Cleversafe, Google Cloud Storage, OpenStack Swift, and Microsoft Azure [22]. DTR also enables centralized backup (replication) of images.

To compute a task using a virtual image, it is necessary to download the Dockerfile and the executable file required to complete the task. The Dockerfile should contain instructions for transferring files to the container if an executable was specified. In the case of scenario 2, when the user downloads a Dockerfile that will be used to build an image to perform a task, the script that builds the image, notices it with a tag and starts the container, looks like the one shown in Listing 1:

Listing 1 – Script to build a virtual image and run it
```
#!/bin/sh
if [ -z «$1» ]
 then
  echo «No docker image name supplied»
  exit 1;
fi
IMAGE_NAME=$1
echo «Starting build of Docker image: $IMAGE_NAME»
echo «$IMAGE_NAME»
docker build -q -t $IMAGE_NAME .
echo $? > build.txt
echo «Starting image: $IMAGE_NAME:latest»
docker run --tty $IMAGE_NAME:latest || exit
```

A file named build.txt is created in the directory of the current execution session, containing the result of executing the docker build command, containing the result of executing the image build.

If a container is launched based on an existing image (scenario 3), the image launch script will look like the one shown in Listing 2:

Listing 2 – Script for starting a task using an existing image
```
#!/bin/sh
if [ -z «$1» ]
 then
  echo «No docker image name supplied»
  exit 1;
fi
IMAGE_NAME=$1
echo «Starting image: $IMAGE_NAME»
docker run --tty $IMAGE_NAME || exit
```

Fig. 4, 5 show the results of performing computational tasks using a virtual environment using the services of the developed framework.

When defining task parameters using the user interface of the developed framework, the corresponding Dockerfile is specified to define the virtual environment for executing the task.

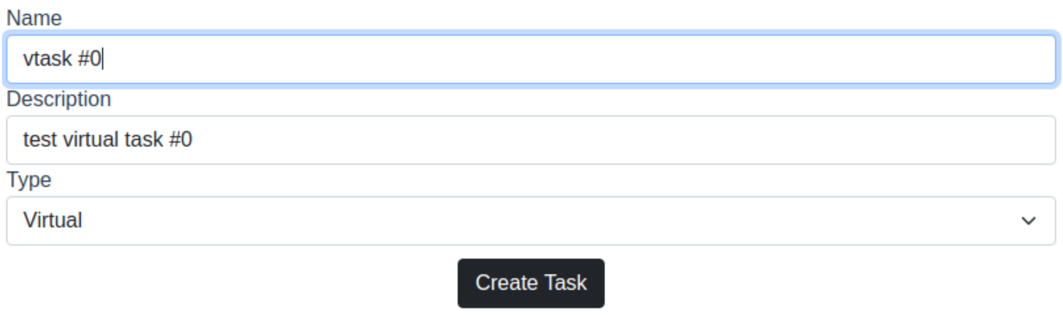
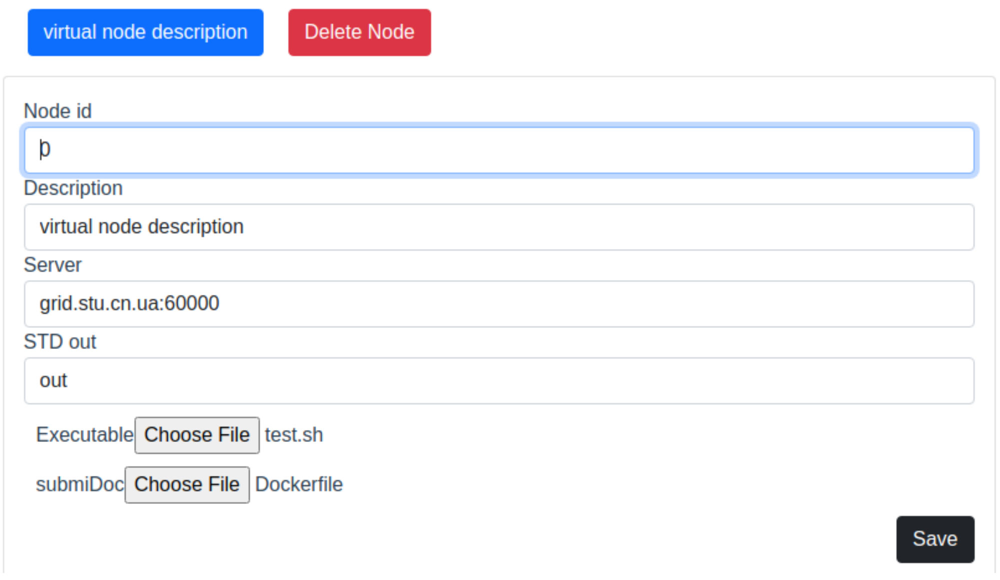

Name: vtask #0
Description: test virtual task #0
Type: Virtual

Fig. 4. Window for creating a task using a virtual image (Type-Virtual)



Node id
p

Description
virtual node description

Server
grid.stu.cn.ua:60000

STD out
out

Executable Choose File test.sh

submiDoc Choose File Dockerfile

Fig. 5. Window for defining task parameters using a virtual image

6. Discussion of the research results of the services of the grid computing system using virtual environments

The developed mechanism for scheduling the execution of tasks, in contrast to the existing ones, is focused on the integration of methods for placing virtual machines and methods for scheduling tasks in a grid environment. The algorithm is based on the dynamic programming method with optimization according to the criterion of the computation time. The architecture of a distributed storage of images of virtual machines is formed by synchronous and asynchronous replication of images using the LRU (least recently used) and MFU (most commonly used) algorithms.

Evaluation of the effectiveness of the developed mechanism for scheduling tasks in the grid environment was carried out on the basis of the GridSim complex of simulation models. The results of the experiments showed that the use of the proposed algorithms for scheduling and replicating virtual images can reduce the execution time of a computational task in a grid environment by 88 % compared to the random scheduling of the developed approaches by the criterion of the execution time of computations Using the replication mechanism of virtual images, shown in Fig. 2 and formula (2), minimizes the cost of sending images of virtual machines. An increase in planning efficiency is predicted with an increase in the number of tasks launched for execution due to reallocation of the storage of virtual images.

Comparison of the effectiveness of the proposed method with other available methods of planning and replicating virtual images is planned in subsequent studies.

The disadvantage of the developed algorithm is the planning time using the algorithm, which is longer than the execution time of a random algorithm, but insignificant compared to the task execution time. The use of the algorithm for scheduling tasks characterized by low dimensionality of computations and data, as well as for homogeneous distributed environments, will be impractical.

The services of the developed system allow to design and perform computational tasks in a grid environment with loading a new or loaded virtual environment for the execution of the Docker platform. The developed framework can be used to plan and execute computational tasks of any applied industries in a grid environment. The developed system, in contrast to the existing ones, makes it possible to configure virtual execution environments for solving computational problems of various types and scheduling the execution of tasks with optimization according to the criterion of the computation time.

One of the directions for expanding the functionality of the services of the developed framework is the implementation of the integration of other environments for using virtual machines (for example, Singularity). This will enable wider use of the framework.

7. Conclusions

1. The algorithm for scheduling tasks in a grid environment using virtualization is implemented based on the dynamic programming method. The mechanism for hosting and replicating images of virtual machines, in contrast to the existing ones, is focused on integration with methods of scheduling tasks in a grid environment.

2. Using the proposed mechanism for scheduling and replicating virtual images can reduce the execution time of computational tasks in a grid environment by 88 %.

3. Services of the developed framework allow to design and perform computational tasks in a grid environment based on ARC Nordugrid using virtual environments of the Docker platform. Computational tasks can be represented by structures of different types, including workflow. The image of the virtual environment can be loaded together with the executable file of the task or selected from among those stored on the resources of the distributed environment.

References

1. Di Meglio, A., Riedel, M., Memon, S. M., Loomis, C., Salomoni, D. (2011). Grids and Clouds Integration and Interoperability: an overview. Proceedings of The International Symposium on Grids and Clouds and the Open Grid Forum – PoS(ISGC 2011 & OGF 31). doi: https://doi.org/10.22323/1.133.0112
2. Foster, I. (2002). What is the Grid? A Three Point Checklist. GRIDToday. Available at: https://www.mcs.anl.gov/~itf/Articles/WhatIsTheGrid.pdf
3. Mell, P. M., Grance, T. (2011). The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. NIST. doi: https://doi.org/10.6028/nist.sp.800-145
4. ARC. NorduGrid. Available at: http://www.nordugrid.org/
5. Krašovec, B., Filipčič, A. (2019). Enhancing the Grid with Cloud Computing. Journal of Grid Computing, 17 (1), 119–135. doi: https://doi.org/10.1007/s10723-018-09472-w
6. Pogorilyy, S. D., Boyko, Y., Salnikov, A. O., Sliusar, Ie. A., Boretsky, O. (2017). Images of virtual machines running as grid tasks provisional configuration and formation. Naukovi pratsi Donetskoho natsionalnoho tekhnichnoho universytetu. Seriya: Informatyka, kibernetyka ta obchysliuvalna tekhnika, 2, 90–97. Available at: http://nbuv.gov.ua/UJRN/Npdntu_inf_2017_2_14
7. Haug, S., Sciacca, F. G. (2017). ATLAS computing on Swiss Cloud SWITCHengines. Journal of Physics: Conference Series, 898, 052017. doi: https://doi.org/10.1088/1742-6596/898/5/052017
8. ATLAS Experiment. Available at: https://atlas.cern/
9. Keahey, K., Riteau, P., Anderson, J., Zhen, Z. (2019). Managing Allocatable Resources. 2019 IEEE 12th International Conference on Cloud Computing (CLOUD). doi: https://doi.org/10.1109/cloud.2019.00019
10. Donyagard Vahed, N., Ghobaei-Arani, M., Souri, A. (2019). Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review. International Journal of Communication Systems, 32 (14), e4068. doi: https://doi.org/10.1002/dac.4068
11. Mohammad, S. G. (2019). A dynamic replication mechanism in data grid based on a weighted priority - based scheme. i-Manager's Journal on Cloud Computing, 6 (1), 9. doi: https://doi.org/10.26634/jcc.6.1.15897
12. Chang, Y., Gu, C., Luo, F. (2016). A novel energy-aware and resource efficient virtual resource allocation strategy in IaaS cloud. 2016 2nd IEEE International Conference on Computer and Communications (ICCC). doi: https://doi.org/10.1109/compcomm.2016.7924911
13. Ashraf, A., Porres, I. (2017). Multi-objective dynamic virtual machine consolidation in the cloud using ant colony system. International Journal of Parallel, Emergent and Distributed Systems, 33 (1), 103–120. doi: https://doi.org/10.1080/17445760.2017.1278601
14. Kazymyr, V., Prila, O., Kryshchenko, M. (2017). The use of dynamic virtual images in a grid environment with replication support. Technical Sciences and Technology, 3 (9), 88–97. doi: https://doi.org/10.25140/2411-5363-2017-3(9)-88-97
15. Prila, O., Kazymyr, V., Kryshchenko, M., Sysa, D. (2018). The technology of reliable task execution in grid environment using dynamic virtual images. 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT). doi: https://doi.org/10.1109/dessert.2018.8409109
16. Apache NetBeans. Available at: https://netbeans.apache.org/
17. Git. URL: https://git-scm.com/
18. Maven. Welcome to Apache Maven. Available at: https://maven.apache.org/
19. GridSim. Available at: https://swmath.org/software/1392
20. Prila, O. A. (2013). The algorithm of job scheduling in Grid environment based on the dynamic programming method. Visnyk Chernihivskoho derzhavnoho tekhnolohichnoho universytetu. Seriya: Tekhnichni nauky, 4 (69), 153–162.
21. Prila, O. (2013). Framework for grid application development with support of different types of large-scale computing tasks. Eastern-European Journal of Enterprise Technologies, 4 (2 (64), 8–14. Available at: http://journals.uran.ua/eejet/article/view/16598
22. About Registry. Available at: https://docs.docker.com/registry/introduction/