VILNIUS UNIVERSITY

Albertas
JURGELEVIČIUS

Hybrid distributed computing sharing platform

SUMMARY OF DOCTORAL DISSERTATION

Technological Sciences,
Informatics Engineering (T 007)

VILNIUS 2021

This dissertation was prepared in 2016 – 2020 at Vilnius University. The research was supported by the Research Council of Lithuania.

Academic supervisor:
Prof. Habil. Dr. Leonidas Sakalauskas (Vilnius University, Technological Sciences, Informatics Engineering – T 007).
Academic consultant:
Dr. Virginijus Marcinkevičius (Vilnius University, Technological Sciences, Informatics Engineering – T 007).

This doctoral dissertation will be defended in a public meeting of the Dissertation Defence Panel:

Chairman – **Prof. Habil. Dr. Julius Žilinskas** (Vilnius University, Technological Sciences, Informatics Engineering – T 007).
Members:
Prof. Dr. Saulius Gudas (Vilnius University, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Vacius Jusas (Kaunas University of Technology, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Dalius Mažeika (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – T 007),
Prof. Dr. Pilar Martinez Ortigosa (The University of Almeria, Spain, Technological Sciences, Informatics Engineering – T 007).

The dissertation shall be defended at a public meeting of the Dissertation Defence Panel at 12 p.m. on 16th of December 2021 in Room 203 of the Institute of Data Science and Digital Technologies of Vilnius University. Address: Akademijos g. 4, LT-04812, Vilnius, Lithuania.
The summary of the doctoral dissertation was distributed on 15th of November 2021.The text of this dissertation can be accessed at the library of Vilnius University, as well as on the website of Vilnius University: www.vu.lt/lt/naujienos/ivykiu-kalendorius

VILNIAUS UNIVERSITETAS

Albertas
JURGELEVIČIUS

Hibridinių paskirstytų skaičiavimų dalijimosi platforma

DAKTARO DISERTACIJOS SANTRAUKA

Technologijos mokslai,
Informatikos inžinerija (T 007)

VILNIUS 2021

Disertacija rengta 2016–2020 metais Vilniaus universitete.
Mokslinius tyrimus rėmė Lietuvos mokslo taryba.

Mokslinis vadovas:
prof. habil. dr. Leonidas Sakalauskas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Mokslinis konsultantas:
dr. Virginijus Marcinkevičius (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).

Gynimo taryba:
Pirmininkas – **prof. dr. Julius Žilinskas** (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007).
Nariai:
prof. dr. Saulius Gudas (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Vacius Jusas (Kauno technologijos universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Dalius Mažeika (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – T 007),
prof. dr. Pilar Martinez Ortigosa (Alamerijos universitetas, Ispanija, technologijos mokslai, informatikos inžinerija – T 007).

Disertacija ginama viešame Gynimo tarybos posėdyje 2021 m. gruodžio mėn. 16 d. 12:00 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų instituto 203 auditorijoje.
Adresas: Akademijos g. 4, LT-04812 Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2021 m. lapkričio 15 d. Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir VU interneto svetainėje adresu: https://www.vu.lt/lt/naujienos/ivykiu-kalendorius

INTRODUCTION

Research field and relevance of the problem

Many modern companies and organisations are interested in collecting and processing all available data to find various solutions to business problems and make the best business decisions. Due to the large amounts of data, it is no longer possible to apply traditional data management methods and data analysis using high-performance computational resources. Connecting to many various data resources may eventually exceed the company's internal IT infrastructure capacity. Companies are usually required to further increase investments for their internal IT infrastructure development or to purchase and integrate third-party cloud computing solutions. Since internal IT infrastructure development is an expensive solution, it is often easier to find external companies with the necessary infrastructure and buy their offered solutions for an affordable price.

Distributed computing solutions allow integration of internal IT resources for a distributed computing platform easily. However, companies usually choose other alternatives, such as cloud computing services. Cloud computing services provide easy to manage high-performance computing resources and are offered for affordable prices by well-known companies, such as:

- Microsoft[1],
- Amazon[2],
- Google[3],
- Rackspace[4].

---

[1] http://azure.microsoft.com
[2] http://aws.amazon.com
[3] https://cloud.google.com
[4] http://www.rackspace.com

Many public distributed computing projects allow the allocation of available computing resources. However, studies show that small and medium-sized enterprises (SMEs) consider cloud computing services provided by third parties to be more secure than internal solutions. Furthermore, there are no suitable platforms that would allow SMEs to use internal IT resources to meet their business needs. There are no solutions to combine internal IT infrastructure (including employee PCs) into a reliable distributed computing platform.

One of the main reasons preventing companies from using internal IT infrastructures for distributed computing tasks is caused by the straggling task problem (also known as a long-tail problem). Distributed computing tasks consist of several smaller tasks. Such tasks are distributed in parallel to different machines to reduce the time required to execute all the tasks. However, a straggling task problem may occur during task execution in distributed hybrid computing networks. The straggling task problem is caused by unusually slow tasks compared to the average task execution time. Such tasks are called stragglers. An unusually slow task is usually identified as any task that has a 50% longer completion time than average. Stragglers significantly impact the overall task completion time, increase resource utilisation and decrease the system efficiency. As a result, this problem reduces the system availability and adds additional operating costs.

The straggling task problem has led to an increase in the number of studies to identify causes, research task straggler prediction, and avoidance methods. Straggler management methods can be divided into two groups: detection and avoidance. Such methods are based on simulations, replication, load balancing, and task scheduling. However, most methods are designed for specific tasks or systems.

The aim and objectives of the research

The aim of the research is to create a hybrid distributed computing platform that solves the straggling task problem in heterogeneous computing networks more efficiently than the existing solutions. The research objectives are:

- Assess the need and application possibilities of the public distributed calculation model.
- Evaluate the existing public distributed computing model-based platforms.
- Evaluate the existing task scheduling algorithms for distributed hybrid computing platforms.
- Create the architecture of a distributed hybrid computing platform and experimentally evaluate the efficiency of the platform.

The research object and methods

Research objects:

- Public distributed computing.
- Straggling task problem.
- Task scheduling algorithms.

The main research methods are used in the dissertation – review and experiment. The research involved using generated data sets, monitoring computers used for calculations and their actions using systems surveillance and computer modelling methods.

Scientific novelty

The novelty and relevance of this work:

1. Systematically examined the application of task distribution algorithms in distributed hybrid heterogeneous computing platforms.
2. New architecture of the distributed hybrid computing platform proposed.
3. New task scheduling method for distributed hybrid computing platforms proposed.

4. Proposed task scheduling method allows using task stalling buffer in distributed computing systems more than two clusters.

Practical significance of the work

The practical significance of the work is:

1. Task stalling buffer that solves the straggling task problem in a hybrid distributed computing model was applied.
2. New task scheduling algorithm was developed and integrated it into a distributed hybrid computing platform. This method reduces the total time required to complete task queues. Based on this method, a platform was developed that combines a public cluster and a private cluster into a single hybrid computing platform.
3. Hybrid distributed computing platform that operates in heterogeneous computing networks was developed.

Approval of the results

The results of the dissertation research have been published in 3 scientific articles:

1. Jurgelevičius, A., Sakalauskas, L. BOINC from the View Point of Cloud Computing, CEUR Workshop Proceedings, 1973, 2017, 61–66.
2. Jurgelevičius, A., Sakalauskas, L. Big data mining using public distributed computing, Information technology and control. ISSN 1392-124X, eISSN 2335-884X, 2018, vol. 47(2), p. 236–248, DOI: 10.5755/j01.itc.47.2.19738.
3. Jurgelevičius, A., Sakalauskas, L., Marcinkevičius, V. Task stalling for a batch of task makespan minimisation in heterogeneous multigrid computing. Computational Science and Techniques, 2021, 8, 631–638, DOI: 10.15181/csat.v8.2103.

The author participated and presented the results in 2 national and 4 international scientific conferences:

1. Republican conference on “Challenges of Information Technologies in the Creative Economy” (2017, Lithuania). Report topic: “Application of BOINC framework for big data research”.
2. International conference on “BOINC: Fundamental and Applied Science and Technology (BOINC: FAST 2017)” (2017, Petrozavodsk, Russia). Report topic: “BOINC from the view point of Cloud computing”.
3. XVIII international scientific conference “Computer Scientists Days - 2017” (2017, Kaunas). Presentation topic: “Big Data mining using public distributed computing”,
4. Republican conference on “Data Analysis Methods for Software Systems” (2017, Druskininkai). Poster topic: “BOINC Based Enterprise Desktop GRID”.
5. VII international conference on “Open International Conference on Electrical, Electronic and Information Sciences” (2020, Lithuania). Report topic: “Distributed Hybrid Cloud Controller for Runbook Operations”.
6. 4th international conference on “Innovations and Creativity” (2020, Latvia). Report topic: “Task Stalling Buffer Application in Grid Computing”.

Structure of the dissertation

Section 1 provides an overview of the distributed computing model, existing private and volunteer computing solutions, and cloud computing alternatives. Furthermore, this section presents the cloud computing service issues, outlining the need for alternative solutions such as public distributed computing. Section 2 overviews the existing distributed computing platforms and their applications. This section presents and examines the potential use of the BOINC framework to replace cloud computing services. Finally, this section outlines distributed computing platform adoption issues and possible improvements. Section 3 presents the results of the dissertation: a platform for hybrid distributed computing, and a modification of the task distribution algorithm. In addition, hierarchical and non-hierarchical task distribution algorithms are reviewed, which are suitable to use in a hybrid distributed computing platform and solve the straggling task problem. Section 4 presents the computer modelling and experiment results. Finally, the conclusions of the dissertation are presented.

This work contains 116 pages that include 43 figures and 12 tables; the list of references consists of 114 sources.

1. DISTRIBUTED COMPUTING MODEL

1.1 Distributed computing

Distributed computing is a computer processing method in which different parts of a program run simultaneously on multiple computers communicating over a network. Distributed computing is a type of partial or otherwise parallel computing. However, the latter term better defines computing in which different parts of a program run simultaneously on multiple processors that are part of the same computer. Although both types of computation require the program to be divided into segments (parts that run simultaneously), distributed computing still needs to evaluate the different environments in which the individual parts of the program operate. For example, individual computers will likely have different file systems and different hardware components.

1.2 Public distributed computing

Public distributed computing is a computational method that uses multiple public computers in parallel to perform computations. Communication is maintained over a network using a client-server architecture, where client nodes offer their available resources to receive new tasks from the server. The calculations are usually performed in parallel, without affecting calculations performed on other nodes. A distributed computing system’s primary goal is to connect computing resources into a dynamic, open network. One of the advantages of this model over cloud computing is that the infrastructure required for public distributed computing can be fully deployed and fully managed by an organisation with an IT infrastructure. In this way, costs can be reduced by freeing businesses from the need to purchase cloud computing services. Also, it protects against some of the problems that arise in protecting data privacy when using cloud computing services.

The Berkeley Open Infrastructure for Network Computing (BOINC) is an open-source middleware software system that supports volunteer computing and grid computing. BOINC is an excellent example of such a computational method, where tasks that require significant computational resources are divided into smaller tasks, which are then distributed across multiple computers.

1.3 Volunteer computing

Volunteer computing is based on the public distributed computing model. This method uses a client-server model, where client nodes provide their available free resources to the server managing the project tasks. This model depends on people and organisations donating CPU time, networking, and storage capacity from the computers that belong to them. In this way, computing resources can be combined into an open, dynamic network where new nodes can be easily added and the existing ones removed. Such infrastructure has no additional costs and can even reduce the existing ones. Thus, this model offers significant advantages. Also, an infrastructure based on a distributed computing model can be created in a closed network. In this way, the infrastructure is limited to nodes stored on the organisation’s premises. If necessary, the calculation network can be fully controlled and developed by the company's employees. One of the most important aspects of volunteer computing-based systems is attracting and persuading volunteers to participate. The most popular way to attract volunteers is by rewarding them with points for participation. The majority of these projects use the BOINC platform, which will be reviewed in Section 2.

1.3.1 Wikinomics

In the business environment, volunteer or crowd computing can be seen as outsourcing business processes to third parties. This process is also called Wikinomics. The difference between traditional business processes and such mass collaboration is that instead of having an organised business body explicitly designed to perform a unique function, mass collaboration depends on individual free agents brought together to improve business operations. This shows that mass collaboration and volunteer computing in the business environment are not new concepts. These methods are applied in business, and there are platforms for this, which will be reviewed in the following sections.

1.3.2 Extension of the volunteer computing model

Numerous research projects have been carried out to investigate how to estimate and obtain the necessary amount of volunteer computing resources to perform the desired computations. One of them was presented in the EU FP7 EDGI project. The project combines BOINC and XtreemWeb desktop clusters with cloud computing services. This expands clusters with new on-demand resources, making this solution similar to a SaaS cloud. System owners do not have to incur costs for the additional resources as the volunteers provide them. Also, such a solution improves response time in systems using a volunteer computing model. There are more similar studies, where the volunteer computing cluster resources are supplemented with resources from cloud computing services.

Clouds@home is another cloud computing system based on volunteer computing. This system is considered a new form of cloud computing. The project's goal was to create a small-scale and low-cost cloud computing infrastructure by combining cloud and volunteer computing models. From volunteer computers, it creates an infrastructure similar to that of cloud computing services. The idea is based on virtualisation technology in volunteer computing resources, a method called "application sandboxing". It isolates the program inside the virtual machine (VM) and uses a wrapper to run the VM and manage the programs running on it.

Despite efforts to improve and expand this technology, volunteer computing is still not always considered the best way to solve resource demand problems. However, the decision does not necessarily have to be based on volunteers. It may be difficult or even inappropriate to try to attract computational resources from the outside in some cases. Datasets may contain confidential data, meaning that resource utilisation solutions should not rely solely on volunteer computing and should either integrate with other platforms or apply some data protection techniques.

1.4 Private distributed computing

The private distributed computing model uses client-server architecture and is designed for achieving high utilisation and performance using internal resources. Private distributed computing is the most suitable model for enterprises and organisations because it provides high-quality service, performance, and data security.

This model is also used in cloud computing. Data security in cloud computing is more complicated than data security in traditional information systems. Research shows that cloud computing provides an increased risk level because essential services are often purchased from third parties, making it more difficult to ensure data security and privacy, ensure data and service availability, and verify compliance. Emerging new technologies, such as the cloud of things, will create new opportunities for business and increase the risk of attacks. It is necessary to protect data from unauthorised access or attacks by users, such as denial of service, data modification or forgery. First and foremost, user security issues must be addressed to make cloud computing suitable for private users and businesses, making the cloud computing environment reliable. This is the most important condition for winning user confidence for adopting this technology.

1.5 Hybrid distributed computing

Data has become one of the most critical and valued assets in today's rapidly changing business world. Organisations collect and use data to evaluate key performance indicators, make informed decisions, and set their own goals. Useful data can help find problems, increase business efficiency, find new opportunities, and stay ahead of the competitors. Due to the ongoing transformation of industrial production through digitalisation (Industrial 4.0 Strategic Initiative), the amount of data tends to increase.

Large companies typically solve hardware capacity issues by upgrading existing or purchasing new servers and hiring additional staff to maintain the systems. Small and medium-sized enterprises generally do not have the financial capacity to make such investments. Small companies purchase external cloud computing services through various service subscriptions or on-demand service schemes in many cases. Studies show that this makes the volunteer computing model seem unreliable and too difficult to adopt.

While the volunteer computing model can reduce service costs, it also lacks reliability. The required number of volunteers may not always be available, or they will not always perform the assigned tasks. Also, private data protection can cause additional problems. Personal data privacy issues are particularly relevant now. As of May

25, 2018, companies and organisations must comply with the rules of the General Data Protection Regulation in the European Union.

That is why at present we encounter the concept of distributed cloud, which is one of Gartner's top 10 strategic technology trends for 2020. Distributed cloud is the distribution of public cloud services across different geographic locations. Although such services are not located in physical data centres, they are still controlled and supervised by the provider. In addition to the services provided by a private cluster, this technology also offers the advantages of public cloud computing services.

Despite the advantages, the distributed hybrid cloud computing model poses a variety of challenges. One of such problems is task scheduling and execution since it is necessary to maintain an optimal workload between clusters. The existing task scheduling algorithms cannot balance the workload without any additional information about the tasks (e.g., task size, quantity, and incoming task flow). Ensuring the security of the data and systems is also a significant challenge as data is distributed between different devices, including servers, personal computers, and various mobile devices such as wireless sensor networks and smartphones.

1.6 Conclusions

The following conclusions were reached:

1. Literature review showed that the BOINC platform is widely used and is a recommended platform for studying big data using public distributed computing.
2. Literature review showed that to solve public distributed computing platforms' problem of heterogeneity in computational resources, it is necessary to examine the architectures and applications of hybrid distributed computing platforms.

2. DISTRIBUTED COMPUTING PLATFORMS AND APPLICATIONS

2.1 Public distributed computing platforms

There are various publicly available computing solutions: CharityEngine, GridMP, Xgrid, XtremWeb. However, the most widely and actively used solution is called BOINC. BOINC is a high-performance, large-scale computing platform (thousands or millions of computers can participate in computations). It can run virtualised, parallel, GPU-based applications. Also, it can perform data intelligence tasks employing user devices or enterprise servers. BOINC performs calculations only when the central processing unit (CPU) is not in use. This solution allows organisations to use existing employee computer resources without disrupting ongoing work processes. As the computer processors of the company's employee computers are underused 99% of the time, this solution can solve the need for additional computing resources.

2.1.1 Public distributed computing platform BOINC

BOINC is the most popular and considered almost standard software for volunteer computing projects. BOINC is a platform designed for volunteer and cluster computing. This solution is ideal in cases where affordable access to significant computing resources is required, and projects have high public interest. BOINC projects are usually designed to solve complex scientific problems. Such projects use the resources provided by volunteers, so project owners need to gain public trust and interest. This is usually done by presenting that the project is trustworthy. BOINC is based on a client-server architecture and can be used as an example to show how a publicly distributed computing model works.

2.1.2 BOINC adoption issues

Unlike cloud computing, services based on the public distributed computing model are less reliable. This is due to the nature of the model. The computers involved in the computations may become unavailable at any moment, and there is no way of knowing how many computational resources will be available. Therefore, it is not possible to determine how long a particular task will take. It is also not known whether the resources available at one time are sufficient to complete the work in the required time. This type of work in a business environment with strict project schedules is not acceptable.

Numerous studies have been published addressing these issues and examining the combination of volunteer and cloud computing as possible improvements or extensions to existing projects, such as virtualisation, resource capacity and cost estimation methods.

2.1.3 Study of resource availability and energy consumption

A 28-day experiment using two randomly selected computers from different organisations (organisation A and B) was performed to prove the resource availability and cost reduction claims. BOINC clients were installed on each computer and completed the tasks assigned to them by the SETI@home project for two weeks. During that time, power consumption and CPU idle measurements were taken using the Performance Monitor (distributed with the Microsoft Windows 10 operating system) and an electronic power meter. The process was then repeated without any BOINC projects. During the experiments, workers used both computers to perform work-related tasks. As shown in Table 1, Figures 1 and 2, computers perform very little computation and waste available resources.

Table 1 Resource and power consumption statistics.

#	BOINC project	CPU idle time	Power consumption
A	Not running	98.77%	16.61 kWh
	SETI@home	65.23%	22.03 kWh
B	Not running	83.49%	1 kWh
	SETI@home	26.86%	2.09 kWh

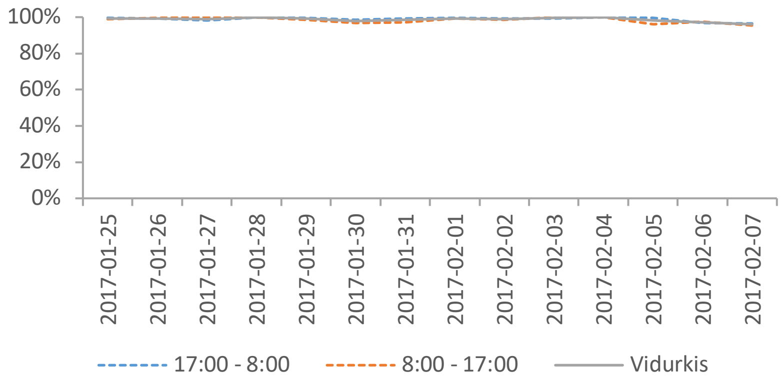

Fig. 1 CPU idle time without any BOINC project tasks running in organization A.

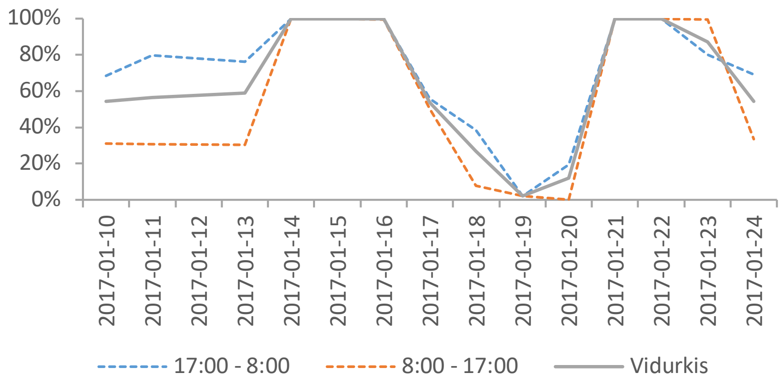

Fig. 2 CPU idle time with SETI@home project tasks running in organization A.

Although few computers were involved in the experiment, the results suggest that such a solution has advantages. Employees did not use computer resources, which allowed for additional calculations at a low cost. Also, the calculations performed did not disrupt any work in progress. This allows us to conclude that organisations can perform additional big data mining and other distributed computing tasks using a public distributed computing platform.

2.2 Private distributed computing platform Apache Mesos

Apache Mesos is one of the cluster resource management platforms. It can manage up to 50,000 (emulated) nodes and has a surplus of less than 4%. Apache Mesos supports a variety of task schedulers, such as Apache Chronos. Apache Chronos is responsible for distributing tasks based on their schedules and dependencies. However, more and more raw tasks can cause a task scheduler error. The proposed hybrid distributed computing platform solves this problem by limiting the number of new incoming tasks to the Apache Mesos cluster's number of available resources. It should also be noted that Apache Chronos and Apache Mesos require a secure network environment.

2.3 Software virtualisation

Software virtualisation is a technology that hides the physical resource layer of the system from the operating system. In a heterogeneous environment, software virtualisation allows one to perform the same tasks on multiple computer architectures and different operating systems.

There are many different software virtualisation technologies: Docker, Kubernetes, Oracle VM VirtualBox, QEMU, VMware, and many more. The proposed hybrid distributed computing platform uses Docker and Oracle VM Virtual Box for software compatibility with Apache Mesos, Apache Chronos and BOINC.

2.4 Security issues

Both public distributed computing and cloud computing models raise similar security concerns, which include:

- system availability,
- data and system integrity,
- user authentication,
- data backup and recovery,
- ensuring data confidentiality,
- privacy and access control.

Distributed computing and cloud computing models have many common problems. Many of them are already addressed in the cloud computing model. However, security is still the main challenge in volunteer computing. BOINC uses two less privileged accesses. However, malicious programs can evade this surveillance.

Research shows that businesses and organisations are interested in using external services, such as cloud computing, as long as there is no serious threat to data security. However, services should be located in the same country or trusted geographical area as the service user.

2.5 Conclusions

The results of the performed research and literature review allow us to draw the following conclusions:

1. Big data mining tasks can be carried out at a very low cost using the available IT infrastructure.
2. Public distributed computing model is capable of processing large amounts of data without interfering with other ongoing work processes.
3. Development of a public distributed computing platform can be summarised and linked to the solution of the following problems:
    a. the computers involved in the calculations have different architectures,

b. the computers involved in the calculations have different performances,
c. the computers involved in the calculations may become inaccessible at any time,
d. the list of tasks and the times of the tasks are not always known in advance.

4. No public distributed computing platform processes big data without using task replication or task size information to solve the straggling task problem in heterogeneous distributed computing networks.

3. STRAGGLING TASK PROBLEM AND SOLUTIONS

3.1 Straggling task problem causes

The leading cause of straggling task problem jams is task resource contention. This reason becomes particularly relevant when resources are heterogeneous. However, tasks can also get stuck due to resource failure and other reasons. There are 8 reasons for causing the straggling task problem:

1. data abstraction,
2. CPU utilisation,
3. scheduling,
4. inaccessible local disk,
5. data skew,
6. resource contention,
7. task execution,
8. faults.

3.2 Straggling task problem solutions

Straggling task problem solutions can be divided into two categories: methods for detecting and avoiding stragglers (Fig. 3).

Straggler detection methods include various methods of modelling, historical data analysis, and real-time system monitoring. These methods are not suitable when there is no information about the tasks to be performed. Task replication itself can cause the straggling task problem in heterogeneous computing networks. Task replication and load management techniques also require information about the average task execution time. Therefore, the task scheduling method is used for the proposed hybrid distributed computing platform to avoid the straggling task problem.

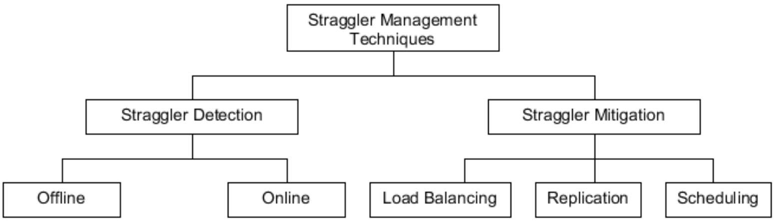

Fig. 3 Straggling task problem solutions.

3.3 Task scheduling algorithms

3.3.1 Hierarchical task scheduling algorithms

There are many existing hierarchical task scheduling algorithms. However, they assume that the task flow and system-wide service capability always remain stable, always available to run at the scheduled task run time or use task replication. The proposed method of task scheduling using a task stalling buffer differs from current solutions since it is designed to operate in a heterogeneous environment without any simulation results or task replication. Furthermore, the proposed method can operate without requiring any additional information about the tasks.

3.3.2 Distributed task scheduling algorithms

This section provides an overview of widely used, mutually independent task scheduling algorithms. Some of them are also used for big data research tasks on Facebook, Yahoo, Hadoop. There are at least five well-known task planning algorithms:

- Fair-share scheduling,
- FIFO,
- Capacity scheduler,
- LATE,
- Round-robin.

The only algorithm that can distribute the stream of dynamic tasks in a heterogeneous environment between two clusters is FIFO. Other well-known task scheduling algorithms, such as Min-min, Min-max, MCT, and Suffrage algorithms are not suitable because these algorithms require a list of all tasks and nodes in advance. Algorithms such as User Defined Assignment are also unsuitable because tasks are assigned in a strict order to the machines where the tasks are expected to be performed the quickest, without estimating whether or not those resources are available.

3.3.3 Task stalling buffer

The task stalling buffer (Fig. 4) improves the task list's execution time in queuing systems with two heterogeneous servers. The task stalling buffer reduces the load on the slow server by redirecting more tasks to the fast server. If the fast server is busy, new tasks are added to the stalling buffer. If the buffer is full, then the slow server receives the task.

This method can be applied to improve the distribution of tasks between the two clusters. It can be assumed that a private cluster will always complete the tasks faster than a public cluster. Therefore, a task stalling buffer can be used to reduce the number of tasks assigned to a public cluster. This reduces the time required to complete tasks and improves the reliability of the services provided by the platform.

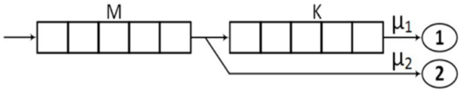

Fig. 4 Queuing system with task stalling buffer.

Here, M is the buffer size for storing new tasks, K is the task stalling buffer size, μ_1 is the fast channel efficiency, μ_2 is the slow channel efficiency, 1 is the fast channel, 2 is the slow channel. Then the task stalling buffer length K can be expressed as:

$$K \approx r \cdot (1 - q), \tag{1}$$

where r is the ratio between fast and slow channel efficiencies and q is the task execution efficiency:

$$r = \frac{\mu_1}{\mu_2},$$

$$q = \frac{c}{t \cdot m \cdot \mu_1},$$

where c is the number of completed tasks, t is the sum of the task execution times, m is the number of fast channel nodes, and μ_1 is the fast channel efficiency:

$$\mu_1 = \frac{a_1}{b_1}, \tag{2}$$

$$\mu_2 = \frac{a_2}{b_2}, \tag{3}$$

where a_1 is the number of tasks performed using the fast channel and b_1 is the time required to complete those tasks; a_2 is the number of tasks performed using the fast channel and b_2 is the time required to complete those tasks.

3.3.4 Task stalling buffer in multichannel systems

The task stalling buffer can be applied in a very similar way for multichannel systems. The task stalling buffer shortens the task execution time the most when the efficiency difference between the two task processing systems is the largest. Therefore, for the distributed computing platform with *n* clusters ($n > 1$), the tasks must be distributed recursively between the slowest cluster (slow channel) and the remaining *n*-1 clusters (fast channel). An example of the application of a task stalling buffer in a three-channel system is shown in Figure 5.

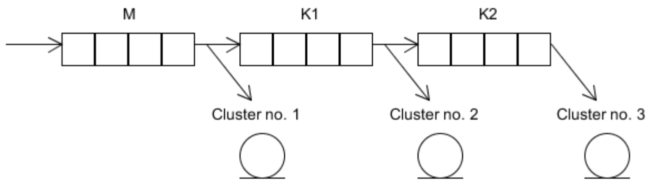

Fig. 5 Task stalling buffer in a 3-channel system.

Like in a two-channel system, *M* is the length of the waiting buffer, *K1* and *K2* are the lengths of the task stalling buffers. The clusters correspond to different performance channels, where Cluster No. 1 has the lowest efficiency, and Cluster No. 3 has the highest performance efficiency.

3.3.5 Algorithms

The following functions are required per each cluster to estimate the length of the task stalling buffer in a hybrid distributed computing platform:

- *is_available* – indicates whether the cluster has free resources that can accept the new task,
- *get_client_count* – returns the number of clients in the cluster;

- *get_busy_client_count* – returns the number of clients in the cluster that are assigned tasks,
- *get_completed_task_count* – returns the number of tasks performed in the cluster,
- *get_completed_task_exec_time* – returns the time it took for the cluster to complete the tasks.

All of these functions can be implemented for BOINC and Apache Mesos clusters.

3.4 Distributed hybrid cloud computing architecture

This section presents a platform architecture based on a distributed hybrid computing model that allows performing various tasks using internal servers (or cloud computing services) and personal computers. The proposed platform combines public and private computing clusters into a hybrid distributed computing network. This platform uses a task distribution method to manage the workload between two clusters without any additional task information. Service reliability and task straggler problems are solved using the proposed task distribution method based on the task stalling buffer. This way, the described platform enables companies to reduce the cost of services and still maintain services' reliability.

The described platform uses open source and interoperable technologies (all support the same software virtualisation solution). However, it is essential to note that other compatible alternatives may be used.

As shown in Figure 6, the proposed hybrid distributed computing platform architecture has a two-level hierarchy with physically distributed (hierarchical) cooperating task schedulers. At the top level is a master scheduler that distributes tasks among lower-level clusters. This architecture provides a dynamic and flexible framework for performing tasks and provides more opportunities to manage service quality. Two clusters are used to distribute tasks between company servers and employee computers: private (managed by Apache Mesos) and public (managed by BOINC). Each cluster is managed by a scheduler specifically designed for a specific cluster environment.

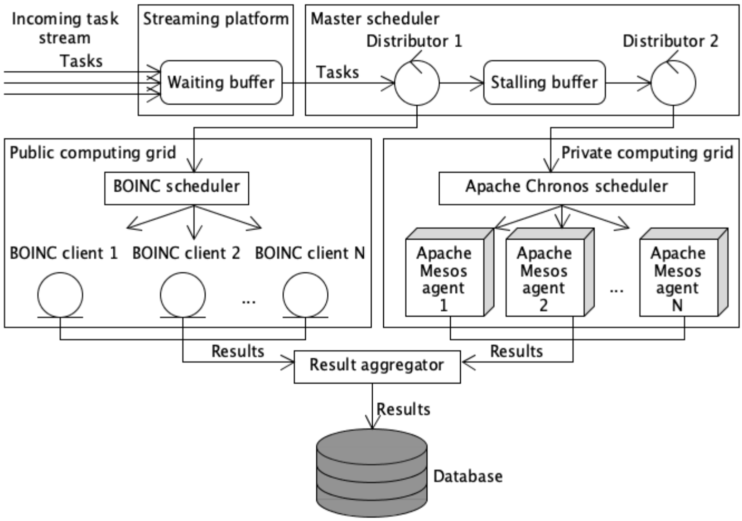

Fig. 6 An architecture that connects two clusters using a two-level scheduler hierarchy.

The top-level (master) task scheduler is the main object of research. The reasons for selecting this particular task scheduling algorithm are presented in Section 3.3.1.

3.5 Conclusions

This section allows us to make the following conclusions:

1. Distributed computing tasks can be performed using the company's internal servers and employees' personal computers without requiring any additional information about the tasks.
2. It is possible to use the task stalling buffer in hybrid distributed computing systems.

4. COMPUTER EXPERIMENTS AND COMPUTER MODELLING

4.1 Computer modelling of a two-channel system

The purpose of the two-channel system computer modeling study is to test the hypothesis that the proposed task scheduling algorithm used in the hybrid distributed computing platform with the task stalling buffer improves task execution time compared to the standard FIFO algorithm.

The experiments are done using a virtual environment created using the PHP programming language. Task execution time is measured using the number of iterations required to complete all tasks. Unlike real-platform experiments (presented in Section 4.3), a virtual environment mimics an infrastructure with a more significant number of compute nodes and can perform large amounts of experiments in a reasonable amount of time. This environment simulates private and public computing resources' behaviour but does not take into account data transmission time and changes in network load.

4.1.1 Computer modelling scenarios

The following scenarios are used to examine each task scheduling algorithm:

- TS_STS: same size tasks are sent to the platform at equal intervals.
- TS_DTS: same size tasks are sent to the platform at varying intervals.
- TD_STS: dynamic size tasks are sent to the platform at regular intervals.
- TD_DTS: dynamic size tasks are sent to the platform at varying intervals.

Each scenario is executed using all possible numbers of tasks, ranging from 40 to 400 tasks. The results gathered from each scenario are aggregated. Sixteen agents serve the slow channel, and 8 agents serve the fast channel. Slow channel agents have a 1000 iteration delay before running a new task (simulating the load time of a virtual machine) and execute tasks 10 times slower than agents serving a fast channel.

4.1.2 Computer modelling results

The results of computer modelling are summarised in Figure 7. The results consist of 21,660 experiments using different scenarios with different task counts. The results show that the task scheduling algorithm using the task stalling buffer is most efficient at the same task flow intensity. The best result is achieved in the TS_STS scenario – an improvement of up to 13% is obtained compared to the standard FIFO algorithm. The results suggest that the task stalling buffer can be applied to hybrid distributed computing platforms, and in all cases outperform the standard FIFO algorithm.

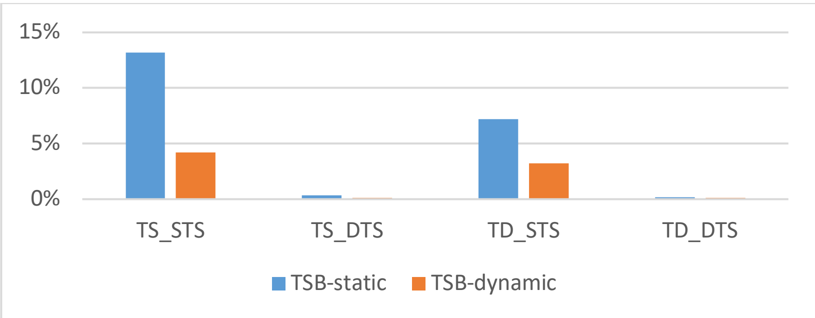

Fig. 7 Reduction of task execution time compared to FIFO.

4.2 Computer modelling of a multichannel system

The aim of the multichannel system computer modelling study is to test the hypothesis that the proposed task scheduling algorithm used in the multichannel hybrid distributed computing platform with task stalling buffer improves task execution time compared to the standard FIFO algorithm.

The research uses the same virtual environment and scenarios as in Section 4.1. The virtual environment used for this study was set to operate in 4 cluster mode. Additional studies were performed to verify the efficiency of the modelled system at different cluster performances. The results are presented in Figures 8 and 9.

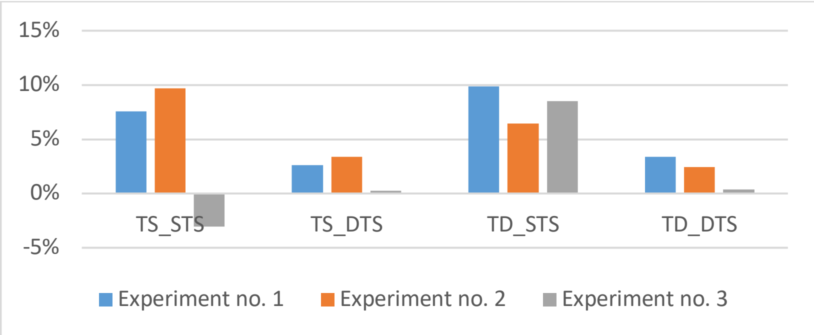

Fig. 8 Reduction of task execution time using static task stalling buffer (compared to FIFO).

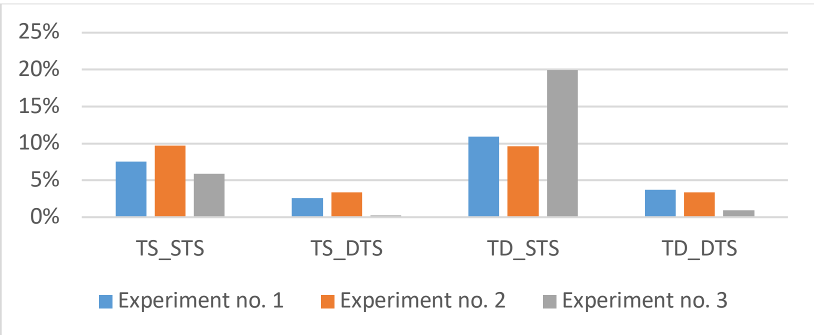

Fig. 9 Reduction of task execution time using dynamic task stalling buffer (compared to FIFO).

The results of all the presented studies show that the proposed task scheduling algorithm used in the multichannel hybrid distributed computing platform with dynamic task stalling buffer improves task execution time compared to the standard FIFO algorithm.

4.3 Platform efficiency study

This section presents the efficiency study of the proposed hybrid distributed computing platform (presented in Section 3.4). The experiment aims to verify that the platform efficiency study results coincide with the experiment results from computer modelling.

4.3.1 Server setups

The research uses two different server setups (server setup A and B). Two environments are used to ensure that the results of the studies will be similar in different environments. The Setup A consists of two separate servers running Docker containers (Fig. 10).

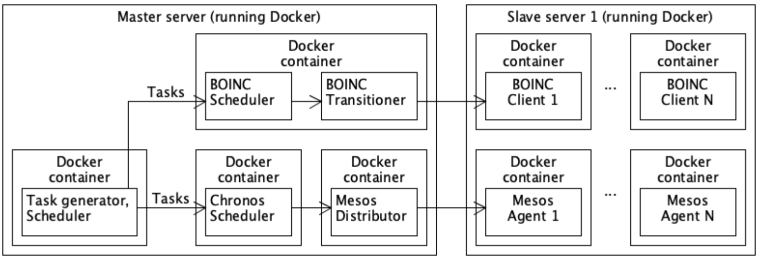

Fig. 10 Server setup A.

The Server setup B is very similar to server setup A. These setups differ in that setup B uses two secondary servers instead of one secondary server (Fig. 11). This separates the two clusters and provides additional resources that are used to emulate more of the computers involved in the computations. In both configurations, the number of emulated nodes is limited by the number of server kernels hosting the virtual machines. Server Setup A emulates two Apache

Mesos agents and two BOINC clients. Server Setup B emulates two Apache Mesos agents and four BOINC clients.

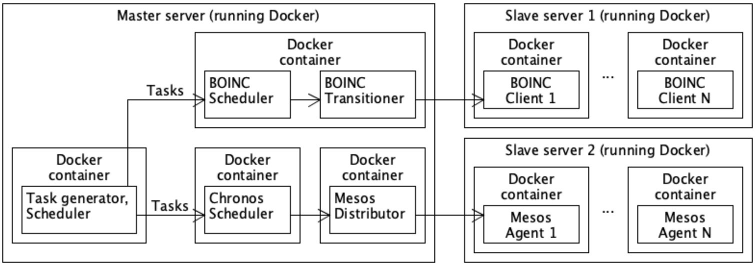

Fig. 11 Server setup B.

4.3.2 Experiment scenarios

Distributed computing tasks were performed to estimate the total task execution time (makespan) for calculating the value of π using the Monte Carlo method. This task requires only CPU capacity, so it allows more accurate time estimates. This eliminates other side effects, including network bandwidth and data storage speed. It is essential to mention that the developed hybrid distributed computing platform can be used to perform various types of distributed computing tasks. The same annotations as described in Section 4.1.1 are used.

4.3.3 Experiment results using server setup A

A hundred tasks were executed using server setup A. The results show that the task stalling buffer improves task execution time (Fig. 12). The use of a static length task stalling buffer improves by an average of 8.875% compared to the standard FIFO algorithm. Meanwhile, the dynamic length task stalling buffer improves by an average of 17.63%.

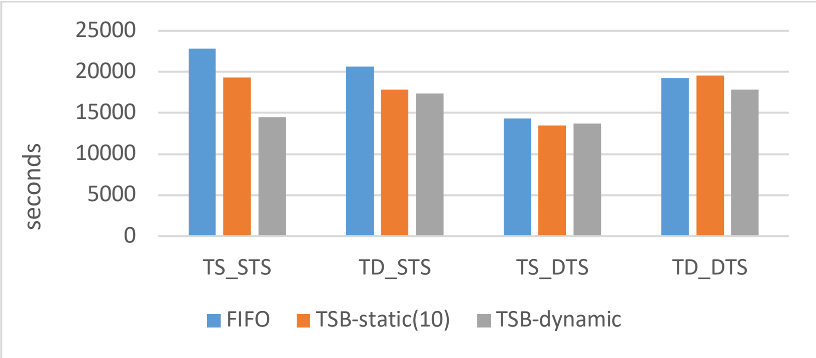

Fig. 12 The total execution time (makespan) of 100 π value assessment tasks generated using the Monte Carlo method.

4.3.4 Experiment results using server configuration B

The purpose of this study is to show that the use of setup B gives similar test results as in the case of setup A. Significantly more experiments were performed during the study to obtain reliable results, using different amounts of tasks for each scenario: 20, 40, 60. Also, each experiment was repeated 5 times to estimate average time deviations and p-value.

This study uses the null hypothesis (two-sided, $\alpha = 0.05$). The aim is to prove that the alternative hypothesis is correct with at least a 95% probability. In this case, the null hypothesis states that the task stalling buffer does not affect the total task execution time.

The aggregated results from 180 experiments are presented in Tables 2 and 3. The scenarios in which the proposed algorithm performed better than FIFO (significance $\alpha = 0.05$) are highlighted in Tables 2 and 3.

The results show a 47.3% improvement using the static-length task stalling buffer and a 20.84% improvement using the dynamic-length task stalling buffer compared to the standard FIFO algorithm.

Table 2 Task execution average makespan (seconds) using TSB-static(10) algorithm (compared to FIFO).

Scenario	Tasks	Average makespan	Standard deviation	Makespan decrease	p-value
TS_STS	20	3503.8	394.22	2.63%	0.646
	40	**6377.8**	**32.8**	**9.71%**	**0.002**
	60	**8978.6**	**297.52**	**10.17%**	**0.0005**
TD_STS	**20**	**3967.4**	**959.76**	**32.09%**	**0.009**
	40	9068.4	172.92	5.76%	0.375
	60	13381.8	393.63	10.22%	0.064
TS_DTS	20	3241.6	203	6.69%	0.063
	40	**6168.8**	**256.39**	**6.19%**	**0.021**
	60	**9254**	**240.38**	**5.13%**	**0.031**
TD_DTS	**20**	**3127**	**824.78**	**47.3%**	**0.001**
	40	8690	301.94	13.43%	0.11
	60	**12293**	**1566.73**	**15.92%**	**0.032**

Table 3 Task execution average makespan (seconds) using TSB-dynamic algorithm (compared to FIFO).

Scenario	Tasks	Average makespan	Standard deviation	Makespan decrease	p-value
TS_STS	20	3493	133.24	2.93%	0.342
	40	**6191.8**	**218.76**	**12.34%**	**0.0002**
	60	**9147.8**	**264.42**	**8.48%**	**0.001**
TD_STS	20	5016.4	117.66	14.13%	0.05
	40	8759.8	257.83	8.96%	0.2
	60	13256.2	480.22	11.06%	0.053
TS_DTS	20	3497.4	133.06	-0.67%	0.765
	40	**6342.2**	**88.81**	**3.56%**	**0.018**
	60	**9204**	**118.35**	**5.64%**	**0.019**
TD_DTS	**20**	**4697**	**607.02**	**20.84%**	**0.008**
	40	9018.8	193.9	10.16%	0.192
	60	13262.2	461.7	9.29%	0.072

A large-scale experiment with 200 tasks was also performed. The results show an improvement of up to 5.86% using TSB-static (10) and an improvement of up to 6.31% using the TSB-dynamic task scheduling algorithm (Table 4).

Table 4 Task execution average makespan (compared to FIFO).

Algorithm	Scenario	Makespan (s)	Makespan decrease
FIFO	TS_STS	31361	-
	TD_STS	47502	-
	TS_DTS	32154	-
	TD_DTS	45159	-
TSB-static(10)	TS_STS	30504	2.73%
	TD_STS	44718	5.86%
	TS_DTS	30431	5.36%
	TD_DTS	44226	2.07%
TSB-dynamic	TS_STS	30700	2.11%
	TD_STS	44504	6.31%
	TS_DTS	30547	5%
	TD_DTS	44963	0.43%

4.4 Conclusions

Computer modelling and empirical research show that the task stalling buffer reduces the total execution time of all tasks. Computer modelling studies show that the proposed distributed hybrid computing platform with a task stalling buffer reduces the total task execution time by 47.3%. The research results allow us to draw the following conclusions:

1. The task stalling buffer can be applied to distributed hybrid computing solutions and improve the workload balance between two and more clusters.

2. Task queue is executed faster using task stalling buffer than using FIFO algorithm.
3. The most significant improvement is achieved by running small batches of tasks in a moderately loaded system.
4. When the system is heavily loaded with more significant amounts of short tasks, the observed improvement is smaller.
5. The TSB-dynamic algorithm performs a sequence of undefined tasks in a multichannel system 2.29% faster than the TSB-static algorithm.
6. The TSB-static algorithm distributes an average of 24.43% fewer tasks to the slow channel than the FIFO algorithm.
7. The TSB-dynamic algorithm distributes on average 3.61% fewer tasks to the slow channel than the FIFO algorithm.
8. The TSB-dynamic algorithm distributes on average 21.57% fewer tasks to the slow channel than the TSB-static algorithm.

CONCLUSIONS

1. The literature review showed that no public distributed computing platform processes big data without using task replication or task size information to solve the straggling task problem in heterogeneous distributed computing networks.
2. Based on the computer modelling results, the straggling task problem in low-performance resources is effectively solved using the developed hybrid distributed computing architecture when the incoming task rate is constant.
   A. The TSB-static algorithm distributes an average of 24.43% fewer tasks to the slow channel than the FIFO algorithm.
   B. The TSB-dynamic algorithm distributes on average 3.61% fewer tasks to the slow channel than the FIFO algorithm.
   C. The TSB-dynamic algorithm distributes on average 21.57% fewer tasks to the slow channel than the TSB-static algorithm.
3. The multichannel system's computer modelling study has shown that a non-predefined sequence of tasks is executed 2.29% faster using the dynamic-length task stalling buffer than the static-length task stalling buffer.
4. The proposed hybrid distributed computing platform using the static-length task hold buffer executes a predefined set of tasks up to 47.3% faster than the FIFO algorithm:
   A. The most significant improvement is achieved by performing small batches of tasks in a moderately loaded system.
   B. Smaller improvement is observed when the system is heavily loaded with large batches of short tasks.

PUBLICATIONS

1. Jurgelevičius, A., Sakalauskas, L. BOINC from the View Point of Cloud Computing, CEUR Workshop Proceedings, 1973, 2017, 61–66.
2. Jurgelevičius, A., Sakalauskas, L. Big data mining using public distributed computing, Information technology and control. ISSN 1392-124X, eISSN 2335-884X, 2018, vol. 47(2), p. 236–248, DOI: 10.5755/j01.itc.47.2.19738.
3. Jurgelevičius, A., Sakalauskas, L., Marcinkevičius, V. Task stalling for a batch of task makespan minimisation in heterogeneous multigrid computing. Computational Science and Techniques, 2021, 8, 631–638, DOI: 10.15181/csat.v8.2103.

BRIEFLY ABOUT THE AUTHOR

Albertas Jurgelevičius graduated from Mykolas Biržiška secondary school (Vilnius) in 2006. In 2010 he graduated from Vilnius University and was awarded a BA in informatics. In 2012 he graduated from Vilnius University and was awarded an MA in informatics engineering. He was a doctoral student at the Institute of Data Science and Digital Technologies from 2016 to 2020. E-mail: j.albertas@gmail.com.

SANTRAUKA

Tyrimų sritis ir problemos aktualumas

Daug šiuolaikinių įmonių ir organizacijų yra suinteresuotos rinkti ir apdoroti kiek galima daugiau su verslo procesais susijusių duomenų, kuriais vėliau būtų galima remtis sprendžiant įvairias verslo problemas ir siekiant priimti geresnius verslo sprendimus. Tam naudojami dirbtinio intelekto, mašininio mokymosi, statistikos ir kiti žinių gavimo metodai. Dėl vis didėjančio duomenų kiekio nebeįmanoma įgyvendinti tradicinių duomenų valdymo ir duomenų analizės metodų, pasinaudojant įprastiniais didelio našumo skaičiavimų ištekliais. Verslo įmonėms ir organizacijoms kaupiant ir prisijungiant prie vis daugiau duomenų turinčių išteklių, skaičiavimo išteklių poreikis pradeda viršyti turimų vidinių IT infrastruktūrų galimybes. Tad organizacijos galiausiai nebeturi pakankamai vidinių skaičiavimo išteklių patenkinti paklausai. Turimi vidiniai aukšto skaičiavimų pajėgumo ištekliai ir panašūs tradiciniai duomenų valdymo sprendimai nebepajėgia susitvarkyti su tokiais duomenų kiekiais, o papildomo aukšto skaičiavimų pajėgumo klasterio diegimas ir priežiūra gali neatitikti finansinių galimybių. Tokiais atvejais įmonės linkusios arba toliau didinti investicijas į savo vidinės IT infrastruktūros vystymą, arba ieškoti trečiųjų šalių sprendimų. Serverių pirkimas ar atnaujinimas yra brangus sprendimas, tad dažnai tenka ieškoti įmonių, kurios turi reikiamą infrastruktūrą ir siūlo savo sprendimus už prieinamą kainą.

Nors ir yra paskirstytųjų skaičiavimų sprendimų, leidžiančių lengvai sujungti vidinius IT išteklius į paskirstytųjų skaičiavimų platformą, organizacijos linkusios rinktis kitas alternatyvas. Vienas iš plačiai naudojamų sprendimų – debesų kompiuterija, kurios paslaugos nuolat plečiamos. Aukštus skaičiavimų pajėgumus siūlančios paslaugos tampa vis pigesnės, o šiuo metu nemažai išorinių kompanijų siūlo debesijos ir didžiųjų duomenų tyrybos sprendimus už prieinamą kainą. Norėdamos suvaldyti gaunamų duomenų srautus ir apdoroti turimus duomenis, įmonės dažnai naudojasi debesų kompiuterijos paslaugomis, kurias teikia tokios gerai žinomos įmonės kaip:

- „Microsoft“ (http://azure.microsoft.com/);
- „Amazon“ (http://aws.amazon.com/);
- „Google“ (https://cloud.google.com/);
- „Rackspace“ (http://www.rackspace.com/).

Tokiomis paslaugomis įmonės dažnai naudojasi tvarkydamos didžiulius turimų duomenų rinkinius. Iš duomenų gautos žinios gali padėti priimti teisingus verslo sprendimus ir suteikti klientams vertingos informacijos.

Tyrimai rodo, kad mažos ir vidutinės įmonės (MVĮ) neretai mano, jog trečiųjų šalių teikiamos debesų kompiuterijos paslaugos yra saugesnės už jų pačių kuriamus techninius sprendimus, naudojančius vidinę įmonės IT infrastruktūrą. Tai verčia mažas ir vidutines įmones domėtis debesų kompiuterijos teikiamomis paslaugomis bei jų taikymo galimybėmis. Debesų kompiuterija nuolat plečiasi, nes ir toliau teikia našias skaičiavimo paslaugas už vis mažesnes kainas. Dėl šių priežasčių debesų kompiuterijos ir viešųjų paskirstytųjų skaičiavimų sprendimai yra aktualūs MVĮ.

Yra daugybė viešųjų paskirstytųjų skaičiavimų projektų, leidžiančių skirti savo turimus skaičiavimo išteklius, tačiau iki šiol nėra tinkamų programų, leidžiančių MVĮ panaudoti vidinius IT išteklius savo verslo poreikiams patenkinti – trūksta įrankių, kurie galėtų paversti vidinę IT infrastruktūrą (įskaitant darbuotojų asmeninius kompiuterius) patikima paskirstytųjų skaičiavimų platforma, atliekančia duomenų analizės ir įvairias kitas skaičiavimo išteklių reikalaujančias užduotis.

Viena iš pagrindinių problemų, neleidžiančių išnaudoti įmonių vidinių IT infrastruktūrų, yra užduočių strigimo hibridiniuose skaičiavimų tinkluose problema. Programos gali būti vykdomos paskirstytųjų skaičiavimų sistemose, tokiose kaip duomenų centrai ir klasteriai, naudojant išteklių valdytoją (YARN, „Apache Mesos“, „Borg“, BOINC ir kt.). Vykdomąją programą sudaro kelios mažesnės užduotys, apibrėžtos kaip mažiausi skaičiavimo vienetai, kurių vykdymą prižiūri išteklių valdytojas. Tokios programos ir užduotys yra lygiagrečiai paskirstomos skirtingiems skaičiavimo ištekliams, siekiant paspartinti darbų vykdymą. Tačiau tokiu būdu vykdant užduotis hibridiniuose skaičiavimo tinkluose pasitaiko užduočių strigimo problemų. Neįprastai lėtai atliekama užduotis, lyginant su vidutine užduoties atlikimo trukme, vadinama įstrigusia. Neįprastai lėta užduotis paprastai identifikuojama kaip bet kokia užduotis, kurios užduoties atlikimo laikas yra 50 % ilgesnis už vidutinį užduoties atlikimo laiką darbo etape. Lėtai vykdomos užduotys (angl. stragglers) daro įtaką viso darbo atlikimo ir užbaigimo laikui, didindamos išteklių naudojimą, mažindamos programų našumą, sistemos prieinamumą ir didindamos papildomas eksploatacines išlaidas. Atlikus didelio masto gamybos sistemų analizę nustatyta, kad maždaug 4–6 % užduoties dalyvių neigiamai veikia daugiau kaip 50 % visų darbo vietų didesnėje sistemoje. Pastebėta, kad šis reiškinys taip pat pasireiškia duomenų centruose ir daro neigiamą įtaką programų veikimui. Užduočių strigimo problemų pasitaiko bet kurioje lygiagrečius skaičiavimus atliekančioje sistemoje, o dar labiau jos išryškėja vykdant darbus, kurie susideda iš daugybės užduočių ir yra atliekami daugelyje kompiuterių tuo pat metu.

Tai lėmė padidėjusį tyrimų skaičių, susijusių su pagrindinių užduočių strigimo priežasčių analize, užduočių strigimo prognozavimu ir užduočių strigimo vengimo metodais, įskaitant simuliacijų vykdymą, replikaciją, apkrovos balansavimą ir užduočių tvarkaraščio planavimą. Kiekviename iš šių darbų daugiausia dėmesio skiriama tik tam tikro pogrupio užduočių ir sistemų taikymui.

Kitų autorių darbuose sprendžiant užduočių strigimo problemą paskirstytųjų skaičiavimų kompiuterinėse sistemose bandyta veiksmingai sumažinti neigiamą strigusių užduočių poveikį. Ši problema spręsta kuriant įvairius įstrigusių užduočių valdymo metodus.

Įstrigusių užduočių valdymo metodus galima suskirstyti į dvi pagrindines klases: aptikimas ir vengimas. Įstrigusių užduočių aptikimas apima metodus, leidžiančius atpažinti įstrigusių užduočių atsiradimą prieš arba po užduoties vykdymo skaičiavimų tinkle, pvz., atliekant įvykdytų užduočių analizę arba įstrigusių užduočių aptikimą naudojant „NearestFit“. Užduočių strigimo vengimo metodais daugiausia dėmesio skiriama bandymui išvengti arba toleruoti aptiktą įstrigusią užduotį ir minimizuoti neigiamas pasekmes, pavyzdžiui, vykdyti užduočių ar išteklių prieigos planavimą, apkrovos balansavimą ir replikavimą. Užduočių strigimo vengimo metodų pavyzdžiai yra „Dolly“, GRASS, LATE ir „Wrangler“.

Darbo tikslas ir uždaviniai

Tyrimo tikslas: sukurti hibridinių paskirstytųjų skaičiavimų platformą, veikiančią heterogeniškuose skaičiavimo tinkluose.
Uždaviniai:

- įvertinti viešųjų paskirstytųjų skaičiavimų modelio taikymo poreikį ir galimybes;
- įvertinti esamas viešųjų paskirstytųjų skaičiavimų modeliu grįstas platformas;
- įvertinti esamus užduočių paskirstymo algoritmus, skirtus hibridinių paskirstytųjų skaičiavimų platformoms;
- sudaryti paskirstytųjų skaičiavimų platformos architektūrą ir eksperimentiniu būdu įvertinti platformos efektyvumą.

Tyrimo objektas ir metodai

Disertacijos tyrimo objektai:

- paskirstytieji skaičiavimai;
- užduočių įstrigimo problema;
- užduočių paskirstymo algoritmai.

Pagrindiniai tyrimo metodai taikomi disertacijoje – analitinė apžvalga, skaičiuojamieji eksperimentai, statistinė duomenų analizė. Tyrime buvo naudojami sugeneruoti duomenų rinkiniai, atliekamas platformos imitavimas, stebimos kompiuterių energetinės ir skaičiuojamosios apkrovos naudojant sistemų stebėjimo ir kompiuterinio tyrimo metodus.

Darbo naujumas

Šio darbo naujumą ir aktualumą sudaro tai, kad:

1. sistemiškai išnagrinėtas užduočių paskirstymo algoritmų taikymas hibridinėse heterogeniškose paskirstytųjų skaičiavimų aplinkose užduotims atlikti;
2. sukurta nauja hibridinių paskirstytųjų skaičiavimų platformos architektūra;
3. pritaikytas užduočių sulaikymo buferis užduotims paskirstyti;
4. sukurtas metodas, leidžiantis užduočių sulaikymo buferį naudoti paskirstytųjų skaičiavimų sistemose, turinčiose daugiau kaip du klasterius.

Darbo rezultatų praktinė reikšmė

Šio darbo rezultatų praktinę reikšmę sudaro tai, kad:

1. pritaikytas užduočių sulaikymo buferis, skirtas užduočių įstrigimo problemai spręsti hibridinių paskirstytųjų skaičiavimų sistemose;
2. sukurtas užduočių paskirstymo algoritmas, integruotas į hibridinių paskirstytųjų skaičiavimų klasterį. Jis leidžia sumažinti suminę užduočių vykdymo trukmę. Remiantis šiuo metodu parengta platforma, sujungianti viešąjį ir privatųjį klasterį į hibridinį skaičiavimų tinklą;
3. sukurta hibridinių paskirstytųjų skaičiavimų platforma, veikianti heterogeniškuose skaičiavimo tinkluose.

Darbo rezultatų aprobavimas

Disertacijos tyrimų rezultatai publikuoti 3 moksliniuose straipsniuose:

1. Jurgelevičius, A., Sakalauskas, L. BOINC from the View Point of Cloud Computing, CEUR Workshop Proceedings, 1973, 2017, 61–66;
2. Jurgelevičius, A., Sakalauskas, L. Big data mining using public distributed computing, Information technology and control. ISSN 1392-124X, eISSN 2335-884X, 2018, vol. 47(2), p. 236–248, DOI: 10.5755/j01.itc.47.2.19738;
3. Jurgelevičius, A., Sakalauskas, L., Marcinkevičius, V. Task stalling for a batch of task makespan minimisation in heterogeneous multigrid computing. Computational Science and Techniques, 2021, 8, 631–638, DOI: 10.15181/csat.v8.2103.

Autorius dalyvavo ir pristatė rezultatus 2-ose respublikinėse ir 4-iose tarptautinėse mokslinėse konferencijose:

1. respublikinėje konferencijoje „Informacinių technologijų iššūkiai kūrybos ekonomikoje“ (2017, Lietuva). Pranešimo tema: „BOINC karkaso taikymas didelių duomenų tyrybai“;
2. tarptautinėje konferencijoje „BOINC: Fundamental and Applied Science and Technology (BOINC:FAST 2017)“ (2017, Petrozavodskas, Rusija). Pranešimo tema: „BOINC from the view point of Cloud computing“;
3. XVIII tarptautinėje mokslinėje kompiuterininkų konferencijoje „Kompiuterininkų dienos – 2017“ (2017, Kaunas). Pranešimo tema: „Big Data mining using public distributed computing“;
4. respublikinėje konferencijoje „Duomenų analizės metodai programų sistemoms“ (2017, Druskininkai). Stendinio pranešimo tema: „BOINC Based Enterprise Desktop GRID“;
5. septintojoje tarptautinėje konferencijoje „Open International Conference on Electrical, Electronic and Information Sciences“ (2020, Lietuva). Skaityto pranešimo tema:

„Distributed Hybrid Cloud Controller for Runbook Operations“;
6. ketvirtojoje tarptautinėje konferencijoje „4th International Conference on Innovations and Creativity“ (2020, Latvija). Skaityto pranešimo tema: „Task Stalling Buffer Application in Grid Computing“.

Ginamieji teiginiai

Disertacijos ginamieji teiginiai:
1. reikalingas naujas užduočių paskirstymo algoritmas, kuris užduočių įstrigimo heterogeniškuose paskirstytųjų skaičiavimų tinkluose problemą spręstų nenaudodamas užduočių replikacijos ir informacijos apie užduočių dydžius;
2. viešųjų paskirstytųjų skaičiavimų platformų skaičiavimo išteklių heterogeniškumo problema mažinama taikant pasiūlytą hibridinių paskirstytųjų skaičiavimų architektūrą, naudojančią užduočių sulaikymo buferį;
3. sukurtoje hibridinių paskirstytųjų skaičiavimų platformoje naudojamas užduočių paskirstymo algoritmas yra efektyvus užduočių paskirstymui daugiakanalėse paskirstytųjų skaičiavimų sistemose – sutrumpina užduočių įvykdymo laiką lyginant su FIFO algoritmu.

Disertacijos struktūra

1 skyriuje pateikiama paskirstytųjų skaičiavimų modelio, esamų privačiųjų ir savanoriškųjų skaičiavimų sprendimų ir debesų kompiuterijos alternatyvų apžvalga. Be to, šiame skyriuje pateikiamos debesų kompiuterijos paslaugų problemos, nurodant alternatyvių sprendimų, tokių kaip viešųjų paskirstytųjų skaičiavimų, poreikį. 2 skyriuje apžvelgiamos esamos paskirstytųjų skaičiavimų platformos ir jų taikymai. Šiame skyriuje pateikiamos ir nagrinėjamos galimos BOINC platformos naudojimo galimybės siekiant pakeisti debesų

kompiuterijos paslaugas. Galiausiai šiame skyriuje aprašomos paskirstytųjų skaičiavimų platformų keliamos problemos ir siūlomi galimi patobulinimai. 3 skyriuje pateikiami disertacijos darbo rezultatai: hibridinių paskirstytųjų skaičiavimų platforma, užduočių paskirstymo algoritmo modifikacija. Be to, apžvelgiami hierarchiniai ir nehierarchiniai užduočių paskirstymo algoritmai, kurie taip pat tinkami naudoti hibridinių paskirstytųjų skaičiavimų platformose, ir sprendžiama užduočių strigimo problema. 4 skyriuje pateikiami kompiuterinio modeliavimo ir eksperimentų rezultatai. Galiausiai pateikiamos galutinės disertacijos išvados.

Disertacijos apimtis: 116 puslapių, 12 lentelių, 43 iliustracija. Disertacijoje remtasi 114 literatūros šaltinių.

IŠVADOS

Darbo išvados:

1. literatūros apžvalgos rezultatai parodė, kad viešųjų paskirstytųjų skaičiavimų platforma didelių duomenų apdorojimui, kurioje būtų išspręsta užduočių įstrigimo problema heterogeniškuose paskirstytųjų skaičiavimų tinkluose nenaudojant užduočių replikacijos arba informacijos apie užduočių dydžius, nėra sukurta;
2. remiantis kompiuterinio modeliavimo rezultatais, užduočių įstrigimo mažo našumo ištekliuose problema veiksmingai sprendžiama taikant sukurtą hibridinių paskirstytųjų skaičiavimų architektūrą, kai užduočių patekimo į sistemą srautas yra pastovus.
    a. TSB-static algoritmas į lėtąjį kanalą nukreipia vidutiniškai 24,43 % mažiau užduočių už FIFO algoritmą;
    b. TSB-dynamic algoritmas nukreipia vidutiniškai 3,61 % mažiau užduočių už FIFO algoritmą;
    c. TSB-dynamic algoritmas nukreipia vidutiniškai 21,57 % mažiau užduočių už TSB-static algoritmą.
3. daugiakanalės sistemos kompiuterinio modeliavimo tyrimas parodė, kad užduočių paskirstymo algoritmas, naudojantis dinaminio ilgio užduočių sulaikymo buferį, iš anksto neapibrėžtų užduočių eilę įvykdo 2,29 % greičiau už algoritmą, naudojantį statinio ilgio užduočių sulaikymo buferį;
4. sukurta hibridinių paskirstytųjų skaičiavimų platforma, naudojanti statinio ilgio užduočių sulaikymo buferį, iš anksto neapibrėžtų užduočių eilę įvykdo iki 47,3 % greičiau už FIFO algoritmą:
    a. reikšmingiausias pagreitėjimas pasiekiamas vykdant mažą užduočių kiekį vidutiniškai apkrautoje sistemoje;

b. pastebėtas mažesnis pagreitėjimas, kai sistema yra labai apkrauta didesniu trumpų užduočių kiekiu.

NOTES

NOTES

Albetas Jurgelevičius

HYBRID DISTRIBUTED COMPUTING SHARING PLATFORM

Summary of a Doctoral Dissertation
Technological Sciences
Informatics Engineering (T 007)
Editor Zuzana Šiušaitė

Albertas Jurgelevičius

HIBRIDINIŲ PASKIRSTYTŲ SKAIČIAVIMŲ DALIJIMOSI PLATFORMA

Daktaro disertacijos santrauka
Technologijos mokslai
Informatikos inžinerija (T 007)
Redaktorė Jorūnė Rimeisytė-Nekrašienė

Vilniaus universiteto leidykla
Saulėtekio al. 9, LT-10222 Vilnius
El. p. info@leidykla.vu.lt
www.leidykla.vu.lt
Tiražas 40 egz.