An easy way to boost home calculation performance with HGRID

Sergey A. Balabaev, Sergey Lupin

***Abstract*—The paper considers an approach to increasing the computing power of a personal computer by integrating a distributed system of smartphones with it. An overview of the main hardware and software features of mobile devices is given. Based on the analysis of sources, software for integrating mobile devices with a PC has been developed. It allows you to load the calculated task onto nodes, run it, accumulate the results obtained from the nodes, and display the calculation result. Interaction between devices occurs over the network. The used node load balancing algorithm allows combining smartphones of different generations, which differ significantly in performance, into a single environment. It is shown that for efficient operation, it is necessary to use value based on the actual performance of the nodes as the load factor. The paper presents the results of solving the problem of training a neural network in a distributed environment organized using the developed software. They confirm the possibility of increasing the productivity of home computing by including a distributed system of smartphones in this process.**

***Keywords—distributed computing, smartphone, Android OS, grid systems, Raspberry Pi, Smart TV*.**

I. INTRODUCTION

In the modern world, there are various tasks, such as searching for new stars, discovering protein structures, calculating weather forecasts. To solve them effectively, it is necessary to use high-performance systems - supercomputers, clusters, grids. However, it is difficult for a novice researcher to gain access to such systems, so he is forced to use a home personal computer to conduct experiments. To increase the efficiency of calculations, it is possible to combine it with devices that are already in the house - smartphones, TV set-top boxes or smart home elements. A distributed system organized in this way belongs to the Home GRID (HGRID) class - a home grid. Its feature is the high heterogeneity of nodes, so to increase the efficiency of calculations, it is necessary to balance the load depending on the power of each of the devices included in it.

II. RELATED WORKS

All solutions for combining heterogeneous devices into grids can be combined into several classes. The first class involves combining identical devices with different characteristics. This includes grids from personal computers, grids from smartphones, grids from Raspberry Pi microcomputers. Using grids from smartphones to solve complex problems is one of the popular areas of research [1]. Modern smartphones have high characteristics and are capable of performing high-performance computing.

The main features of using smartphones as nodes in a distributed system are: the need for their constant connection to a power supply and additional cooling, limited capabilities for running third-party applications, and heterogeneity of the system [2]. It should be taken into account that when performing intensive calculations, there is high energy consumption and a quick battery discharge and, as a result, slowdown. Similarly, with long-term calculations, the processor temperature rises, which leads to an automatic decrease in the processor clock frequency. To develop applications for devices with the Android operating system, you must use the Java or Kotlin programming languages. Using C/C++ to write code sections that describe the algorithms is only possible with the languages listed above. Due to significant differences in device characteristics, the use of standard methods of parallelizing programs will result in low efficiency. Therefore, it is important to use balancing algorithms that take into account the characteristics of the nodes [3]. The distributed system is developed based on the client-server architecture principle. Tasks are transferred from the main device to the nodes and executed on them. The functionality of applications has no formal limitations, so in the HGRID environment we can solve a variety of problems, such as training a neural network. An example of a project implementing the first approach is Talos, an application designed to train neural networks [4]. In [5,6,7,8], it is proposed to use smartphones as nodes for training neural networks, but the proposed application architecture allows only solving the training problem. Examples of a project with a universal approach are BOINC, IBIS and CWC. The most popular of all existing solutions for using mobile devices for high-load computing is the BOINC project. This is a software package for organizing voluntary distributed computing. It can be used to calculate complex tasks using the resources of personal computers of volunteers who have joined the project. Those wishing to participate in the project must install special software on their PCs that allows them to connect to the system. Then part of the computing task will be sent to the user and calculated in free time [9, 10].

Initially, BOINC was developed in the USA for the SETI@home volunteer computing project, which searches for extraterrestrial intelligence, but later the developers made the platform available for third-party projects.

The system works as follows:

- The client and server launch the application.
- The client selects the project in which he wants to participate.
- The instructions and executable file are sent to the client from the project server.
- The executable file is automatically launched on the client node in accordance with the instructions received.
- After solving the problem, the client sends the solution to the server.

Features of the solution:

- Easy to include volunteer resources in the project.
- Simple interface.
- High entry threshold for developing your own applications.
- Foreign development.

The BOINC project is very convenient for the client - installation and configuration of the system does not involve high costs. However, using the system as your own computing platform may cause difficulties for researchers who do not have high programming skills. It is also noteworthy that, although BOINC is an open-source project, its developers are considering the possibility of boycotting Russian projects [11]. Dutch scientists have developed the IBIS system, which allows combining various devices for distributed computing. The project is written in Java and provides the ability to work with Android devices. The authors propose an effective way for devices to communicate with each other using the developed library and MPI support. However, in 2014, the project was frozen and there is no access to it [12,13]. American researchers have developed the CWC project - Computing while charging. It is proposed to implement high-performance computing in large companies on a grid of smartphones instead of servers and cloud computing. Each employee of the company must provide the power of their device during its inactivity - at night while charging. Experiments have shown the viability of the project. It has been shown that four smartphones united in a network are comparable in power to a personal computer [14, 15]. To solve resource-intensive tasks, you can also use a grid from Raspberry Pi, a popular platform on the basis of which smart home devices, robots, media centers, and DIY projects are developed [16]. There are several versions of microcomputers with different characteristics (Table 1).

TABLE I. CHARACTERISTICS OF DIFFERENT RASPBERRY PI MODELS

Parameter	Raspberry Pi				
	Model 1B	***Model 2B***	***Model 3B***	***Model 4B***	***Model 5B***
Cores	1	4	4	4	4
Clock frequency (GHz)	0.7	0.9	1.2-1.4	1.5	2.4
RAM (Gb)	0.5	1	1	1-4	2-8

The paper [17] describes the use of a Raspberry Pi cluster for machine learning. The cluster developed by the researchers consists of 6 Raspberry Pi Model 4B devices with 4 GB of RAM on each node. The devices were connected to a Gigabit Ethernet switch. The cluster uses the Hadoop (3.2.0) distributed file system and Apache Spark (2.4.3) to implement distributed machine learning algorithms (MLlib Spark). It is shown that the cluster performance is sufficient for machine learning. One of the areas of application of the Raspberry Pi grid is education, when students are trained in practical work with distributed systems [18, 19]. The works emphasize the simplicity of the projects and their relatively low cost. The analysis of existing approaches allows us to state that there is no universal solution on the market, and the task of creating software for organizing HGRID is relevant, especially in the context of sanctions.

III. PROPOSED SOLUTION

The software for organizing HGRID is being developed considering that the following conditions will be met:

- The devices are connected to the local network using WiFi points;
- All devices are connected to the power supply network;
- The devices cannot be disconnected or moved outside the WiFi zone during the calculation.

The functionality of the integrating software must ensure the ability to combine devices with significantly different characteristics. A platform consisting of a personal computer and three devices - a smartphone, a Raspberry Pi microcomputer, and a Smart-TV set-top box - was chosen as a test platform. (Fig. 1)

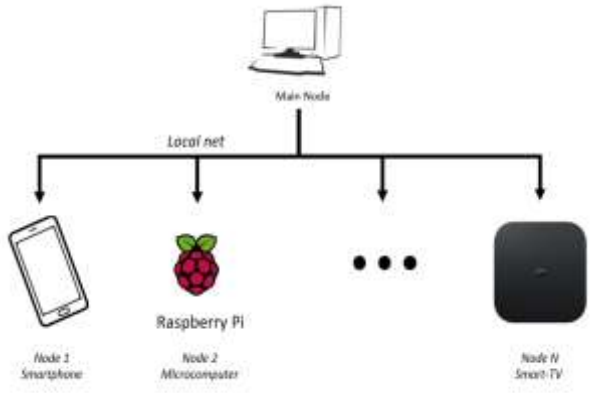

Fig. 1. HGRID structure.

After the task is sent from the server, it arrives via the local network to the client application. If the client is an Android smartphone or a Smart TV set-top box, then the main process receiving messages from the server is the Android application written in Java. The main tasks of the application are to determine the type of task, launch it, and monitor the characteristics of the device. If the task was designed as an APK application, then it is launched using library calls of the application itself. If the task is a script developed, for example, in Python, then the data is transferred to the application developed for the Termux command line emulator. Inside it,

the program is launched and the result is sent back to the client application, which in turn returns the calculation result to the server.

The Termux application is freely licensed and can be installed on most modern mobile devices. It is convenient to use for running simple applications in C, C++, Python, and other programming languages. The disadvantage is the lack of many common libraries. To solve the problem, it is possible to install an ubuntu distribution image inside the application, onto which the necessary software is loaded. Table 2 shows a comparison of the availability of the main libraries required for machine learning on Termux and in the ubuntu virtual machine. The disadvantage of using ubuntu is a strong slowdown in the program, and the inability to use all processor cores.

TABLE II. AVAILABILITY OF MACHINE LEARNING LIBRARIES

Library	Termux	Ubuntu on Termux
numpy	python-numpy	+
tensorflow	-	+
keras	-	+
pytorch	-	+

In the case where the client is a microcomputer, the transferred task can be compiled, processed and launched, as on any ordinary personal computer. At the same time, it is possible to use containerization to launch applications.

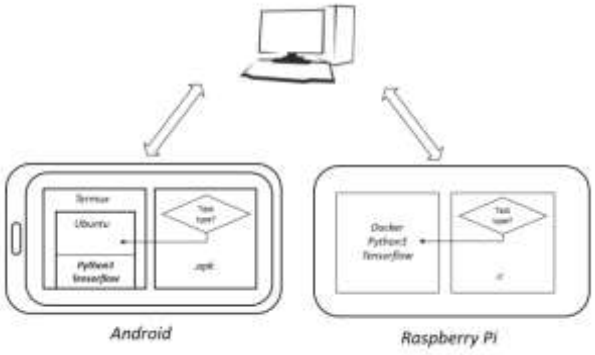

Fig. 2. Fig. 2. HGRID system architecture.

IV. EXPERIMENTAL TASK

To evaluate the computational capabilities of HGRID, experiments were conducted on federated training of a neural network for recognizing advertising messages when sending emails. The initial dataset was 22,032 valid messages and 9,000 spam emails. The characteristics of the HGRID nodes selected for the experiments are presented in Table 3.

TABLE III. CHARACTERISTICS OF HGRID NODES.

Device	Characteristics
PC-1	Intel core i5-11400H @2.70 GHz 6 core 12 threads 24 Gb RAM
PC-2	Intel core i5-5200U @2.2 GHz 2 core 6 Gb RAM
RPI	Raspberry Pi 4 Model B Cortex-A72 @1.8 GHz 4 core 6 Gb RAM
Smartphone	Xiaomi Readme ARM 2× Cortex-A78 @2.6 GHz, ARM 6× Cortex-A55 @2.0 GHz 8 cores, 8 Gb RAM
Smart-TV	Rgeeed Cortex-A53 @1.51 GHz 4 cores, 4 Gb RAM

In the first experiment, the network training time was measured separately on each node in single-threaded and multi-threaded modes using the load balancing algorithm on the nodes. The developed application did not use external libraries, so it could be launched without using the Ubuntu emulator. The results of the experiment are given in Table 4.

It can be noted that the task execution time on the smartphone and Raspberry Pi is lower than on the PC-2 personal computer. This is explained by the fact that it has more computing cores than the computer and its low performance. Due to the fact that the Smart-TV processor was a 32-bit version, it was not possible to fully launch Ubuntu on it. The task execution time in single-threaded and multi-threaded modes on Ubuntu on the smartphone coincided, which confirms the theory about the impossibility of using multiple cores. In the second experiment, a similar task was launched on all HGRID nodes, which included PC-1 and PC-2 in turn. When using HGRID resources, the training time can be reduced by 1.34 times compared to training only on PC-1. In the case of PC-2, using a distributed system accelerates the training process by 4.67 times (Table 5).

TABLE IV. RESULTS OF THE FIRST EXPERIMENT

Computation time (sec)	PC-1	PC-2	Raspberry Pi	Smartphone		Smart-TV	
				Ubuntu On Termux	*Termux*	*Ubuntu On Termux*	*Termux*
Max threads	4.2	48.1	46.36	96.6	19.6	-	109.8
One thread	23.3	71.8	181.2	98.14	55.8	-	433.2

When running the computational task on all devices, a balancing method was used to improve efficiency, considering the coefficient of the relative power of the node.

TABLE V. RESULTS OF THE SECOND EXPERIMENT.

HGRID	Network training time (sec)		
	PC	*HGRID*	*Acceleration*
PC-1 (balance)	4.2	3,13	1,34
PC-2 (balance)	48.1	10,3	4,67
PC-2	48.1	27,6	1,73

The figures show circular diagrams reflecting the contribution of each node to solving the task. The powerful personal computer PC-1 performed ¾ of the entire work, when PC-2 in the HGRID only ¼ (Fig. 3-4).

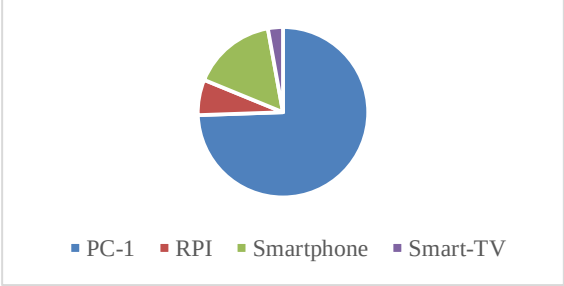

Fig. 3. Contribution of nodes to HGRID performance (first experiment).

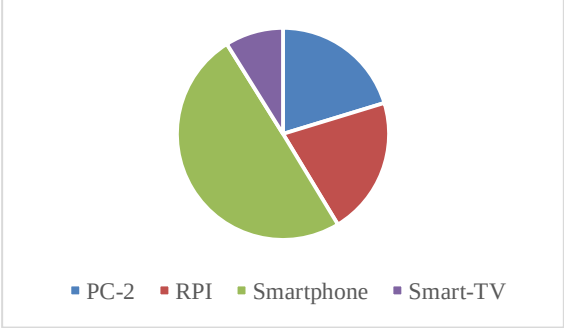

Fig. 4. Contribution of nodes to HGRID performance (second experiment).

The third row in Table 5 allows us to evaluate the efficiency of node load balancing. Without it, the network training time increased by 2.67 times.

V. Conclusion

The proposed approach to organizing a home distributed computing system has proven its functionality in practice during experiments. Even a small HGRID allows reducing the training time of a neural network, especially in the case of node load balancing. It has been shown that using the Termux application and an Ubuntu image installed on it makes the proposed solution universal. It should be noted that although a home network cannot compete with clusters of workstations, it can be used not only to debug distributed applications, but also to expand the functionality of IoT systems.

References

[1] P. K. D. Pramanik, S. Pal, P. "Choudhury, Mobile crowd computing: potential, architecture, requirements, challenges, and applications", The Journal of Supercomputing, vol. 80, i. 2. 2024, pp. 2223-2318. DOI: 10.1007/s11227-023-05545-0

[2] S. A. Balabaev, S. A. Lupin, A. M. Taik, "Monitoring system for load balancing nodes of a distributed computing system based on smartphones", International Journal of Open Information Technologies. – 2024. – vol. 12. – No. 10. – P. 78-85.

[3] M. Khaing, S. A. Lupin, A. Thu, "Evaluating the effectiveness of load balancing methods in distributed computing systems", International Journal of Open Information Technologies. – 2021. – vol. 9. – No. 11. – P. 30-36.

[4] H. C. Takawale, A. Thakur, "Talos app: on-device machine learning using tensorflow to detect android malware", 2018 fifth international conference on Internet of Things: systems, management and security. – IEEE, 2018. – P. 250-255. URL: DOI:10.1109/IoTSMS.2018.8554572

[5] H. Salem, "Distributed computing system on a smartphones-based network", Software Technology: Methods and Tools: 51st International Conference, TOOLS 2019, Innopolis, Russia, October 15–17, 2019, Proceedings 51. – Springer International Publishing 201 9 DOI:/10.1007/978-3-030-29852-4_26

[6] P. Kaushik, P. K. Yadav, "A novel approach for detecting malware in android applications using deep learning" , 2018 Eleventh International Conference on Contemporary Computing (IC3). – IEEE, 2018. – pp. 1-4. DOI: 10.1109/IC3.2018.8530668

[7] W. Fang et al. "Comprehensive android malware detection based on federated learning architecture", IEEE Transactions on Information Forensics and Security. – 2023. – T. 18. – P. 3977-3990. DOI: 10.1109/TIFS.2023.3287395

[8] J. Tang et al. "PE-FedAvg: A Privacy-Enhanced Federated Learning for Distributed Android Malware Detection", 2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom). – IEEE, 2023. – pp. 474-481 DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom59178.2023.00094

[9] I. Kurochkin et al. Using Mobile Devices in a Voluntary Distributed Computing Project to Solve Combinatorial Problems //Supercomputing: 7th Russian Supercomputing Days, RuSCDays 2021, Moscow, Russia, September 27–28, 2021, Revised Selected Papers 7. – Springer International Publishing, 2021. – pp. 525-537. DOI: 10.1007/978-3-030-92864-3_40

[10] A. A. Dolgov, "Deployment of a Grid System from Mobile Devices on the BOINC Platform", Cloud and Distributed Computing Systems in Electronic Management of ORVSEU-2022 within the Framework of the National Supercomputer Forum (NSCF-2022), 2022 pp. 24-29

[11] Official website of the BOINC project [Electronic resource] / URL: https://boinc.berkeley.edu/russia.php (Accessed: 11/30/24)

[12] V. Gurusamy, K. Nandhini, "International journal of engineering sciences & research technology IBIS: The new era for distributed computing", DOI: 10.5281/zenodo.1135392

[13] N. Palmer et al. "Ibis for mobility: solving challenges of mobile computing using grid techniques", Proceedings of the 10th workshop on Mobile Computing Systems and Applications. – 2009. – P. 1-6. DOI: 10.1145/1514411.1514426

[14] T. U. Kumar, R. Senthilkumar, "CWC* - Secured distributed computing using Android devices" //2016 International Conference on Recent Trends in Information Technology (ICRTIT). – IEEE, 2016. – pp. 1-7 DOI: 10.1109/ICRTIT.2016.7569590

[15] M. Y. Arslan et al., "Computing while charging: Building a distributed computing infrastructure using smartphones", Proceedings of the 8th International conference on Emerging networking experiments and technologies. – 2012. – P. 193-204. DOI: 10.1145/2413176.2413199

[16] S. A. Balabaev,S. A. Lupin, R. N. Shakirov, "Computing cluster based on Android smartphones and Raspberry Pi microcomputers", International Journal of Open Information Technologies. - 2022. - Vol. 10. - No. 7. - P. 86-93.

[17] A. Komninos, I. Simou, N. Gkorgkolis, and J. Garofalakis, "Performance of Raspberry Pi Microclusters for Edge Machine Learning in Tourism," in Proceedings of the 2019 European Conference on Ambient Intelligence (AmI 2019), Nov. 2019, vol 2492, pp. 1–10

[18] Z. Xu, "Teaching heterogeneous and parallel computing with google colab and raspberry pi clusters", Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis. – 2023. – P. 308-313 DOI: 10.1145/3624062.3624095

[19] V. Govindaraj, ''Parallel programming in Raspberry Pi cluster. A design project report,'' M.S. thesis, Dept. Elect. Comput. Eng., School Elect. Comput. Eng., Cornell Univ., Ithaca, NY, USA, Tech. Rep., 2016.

Paper received 16 April 2026.

Sergey Lupin, Ph.D., Professor, National Research University of Electronic Technology «MIET», (e-mail: lupin@miee.ru);
Sergey A. Balabaev, Ph.D. Student, National Research University of Electronic Technology «MIET», (e-mail: sergei.balabaev@mail.ru).