MapReduce calculating model on mobile devices

Vladyslav Pavlenko
Institute of telecommunication system
NTUU "Igor Sikorsky Kyiv Polytechnic Institute"
Kiev, Ukraine
e-mail: ghostor2011@gmail.com

***Abstract* — In this article is explained the use of the newest programming model which makes a large number of calculating operations in parallel on mobile devices, where the program runs in a sandbox without specific needs, and simulates mobile as powerful as PC. This approach helps to wipe borders between different OS and mobile manufacturers to make each node as close as possible to another and let it seem to be universal node to let system predict time and expense on such distributed calculations. All beauty of MapReduce method is tightly implemented in Client-Server system based on such "mobile" nodes. There is developed extra method of balancing micro tasks from one node to the next in case of losing connection or coming by node to execution time limit or in another difficult situation that needs fast reaction of system and providing delegation of that micro task to another node. In this paper is also described simplicity of realization, where system abstracts from transporting protocols and OS requirements, making it possible to use even new types of mobile nodes that haven`t been developed or presented yet. The following describes the steps of the method.**

Keywords — MapReduce; Cloud computing; Mobile technologies; Distributed calculations;

I. INTRODUCTION

The technology of distributed calculations has gained popularity over the last 5 years. It is caused by the global connection of a large number of computer devices to the Internet that, in turn, allows the nodes to be connected with each other.

The idea is based on the statement, that modern electronic machines do not use the whole processor power of their core (RAM, main processor) and spend energy irrationally [1].

The vocation of such technology is the system's loading of various simple tasks during the downtime of the operating system, not blocking application software of root user at the same time.

MapReduce is useful in a wide range of applications, including distributed pattern-based searching, distributed sorting, web link-graph reversal, Singular Value Decomposition, web access log stats, inverted index construction, document clustering, machine learning, and statistical machine translation [2]. Moreover, the MapReduce model has been adapted to several computing environments like multi-core and many-core systems, volunteer computing environments, dynamic cloud environments, mobile environments, and high performance computing environments [3].

MapReduce libraries have been written in many programming languages, with different levels of optimization, so it will help to easily implement this system to any mobile platform.

The reminder of this article is organized as follows: in Section III we will discuss the requirements to mobile devices. In Section IV, we discuss the method developed, which allows us to increase the efficiency of the calculations, and, finally, we conclude with a profit of mobile calculations in section IV.

II. PROBLEM FORMULATION

Today the question of cloud calculations is popular and solving problems connected to it make it possible to spread cloud computing more among common people. It brings to modern research more possibilities to find accurate results and makes it possible to calculate big data much faster.

The problem of PCs is that they are working only when user needs something to do, so that means that software, installed on those machines, is in the minority and cannot calculate something in the whole power, which is restricted for this program by user or task. Therefore, the total time of calculation on PCs is low and cannot be guaranteed by cloud system.

People use their mobile devices in different way than PCs. Most time mobile phones are working in a standby mode, so that means that they can produce some useful experience before user turns it up and starts to work with it.

Another problem that appears is that this whole network is built on unstable mobile nodes, those could leave network (disconnect from Internet) at any time. It is a pity, because if server sends a piece of task to working node – it should be calculated and result of that calculation should be returned [4]. There is no way to compel node to stay turned on until it is working on some task, but thanks to MapReduce, it is possible to implement middleware method, that will help to balance and reassign tasks, if they are lost.

Thanks to idea that nodes of such network are mobile devices that runs code (proceed tasks) in universal sandbox, it gives possibility to host server to know approximate time of task execution on node beforehand, and reassign task after execution timeout. Actually, the best way is to poll node after execution timeout to know if mobile device is still working on task or just disconnected from network [5].

978-1-5386-6611-1/18/$31.00 ©2018 IEEE

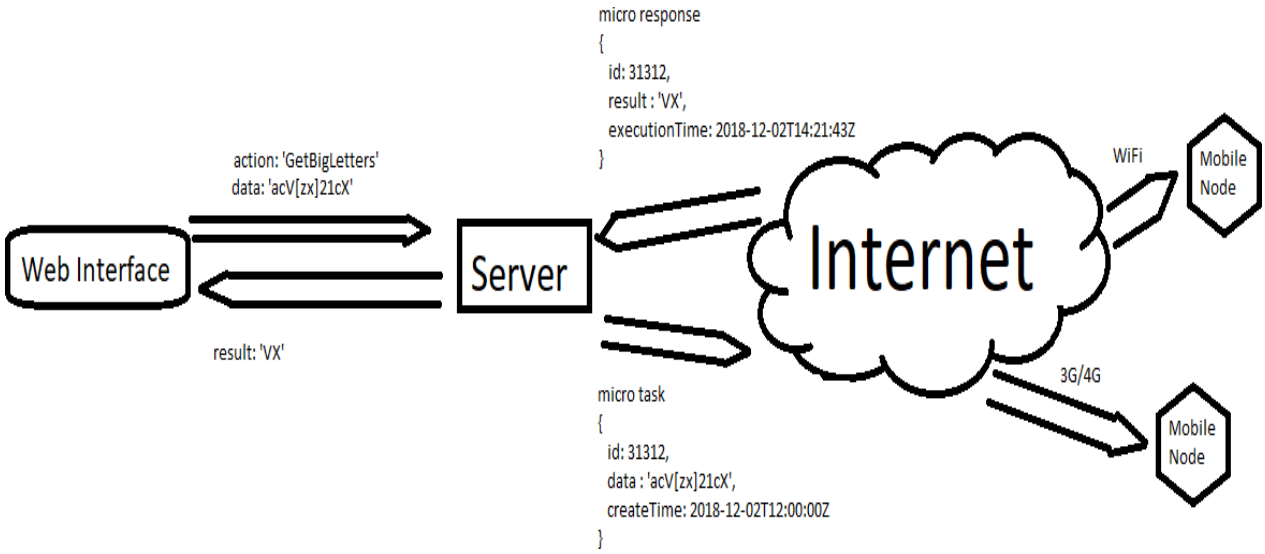

Fig.1 Distributed Calculating Network Scheme for mobile nodes

III. Requirements to Mobile Devices

Smartphones become very popular nowadays. It will be true to say that every modern man has it and even not one.

Such smartphones have powerful CPUs (even not one) and good efficiency in calculating small operations with non-big numbers/data. Thanks to this, it become possible to use them as small computers, so it let modern developers to create powerful and PC`s like applications and games [6]. So, it is possible to say, that mobile device for this system should have modern architecture of CPU and have multitasking OS on board such as Android or IOS.

Due to what is now on mobile devices installed, we have picture, when every mobile user should charge his phone every day or every 2 days and this charging period can longs up to 2 - 3 hours as minimum. So, mobile device should have long term battery, or user should manually setup application to work only when battery level is above a certain level or say application to work only when device is charging. This approach will help to safe battery life and give users a feel of freedom about what processes are running on their gadgets.

Of course, choosing a mobile platform (mobile phones, tablets, smart watches) as nodes of the distributed calculated system has several advantages:

- High performance for simple tasks
- Gadget Mobility
- Simplicity of application settings and creation
- Ability to use a large number of nodes
- Permanent online status due to widespread 3G connection
- Lots of downtime (in standby mode)
- Ability to uniquely identify the node by the gadget parameters (IMEI, GUID, Serial Number, mobile operator card number).

The other thing, is that mobile device should have enough storage space. It means, that all temporary and intermediate calculations will be uploaded to RAM, and if user want to run another application – it should be enough memory to store all needed data. And of course, it should be enough disk space on device, to store computed data before it will send to server.

IV. Method of Distributing Tasks

Schematically, the Distributed Calculation System is a network of linked nodes and a server in the Internet, using the popular HTTP protocol (figure 1). The system consists of a server, written and working on the Node.js technology, that simplifies the binding of nodes and work with the Internet protocols, and several nodes, connected via Wi-Fi and 3G with various mobile operating systems (iOS and Android). The server, responsible for dividing, mailing and bonding a task, must have a reliable access to the Internet, because it also has to monitor the state of the node and the progress of the calculation.

There is also a software snap-in on a mobile client, which carries out the acceptance of the task. Actually, for calculating purposes, such client should have a piece of computed algorithm, especially it is needed for Reduce phase, because it is important to build a logical chain of small pieces of calculated data to obtain whole correct result [7]. So, it is important to build flexible client, that can load calculating algorithms on a par with other information. This client is responsible for processing and formatting the response for further sending to the main node.

The nodes, working within one project, receive tasks from the main server, which initially divides the complex task into parts, then sends it to the nodes for execution, and after a set time t_{exec}, begins to collect the processed information by nodes in a single response.

The task is performed as a low-priority background process, taking up to 10% of productivity at parallel work with processes on behalf of the user, and up to 45% when the operating system goes into standby mode [8].

The server itself is a unique link in the work of this

technology and needs fine tuning. Unlike it, working units can be different configuration and even do not know about the existence of each other. This architecture feature allows using not only commonly accepted desktops, but also other gadgets, equipped with 32 or 64-bit processors [9].

After dispatching the tasks and processing them on the nodes, the gluing to the general answer by the method, reversed to division of a task is performed on the server.

The difference between these processes is that at the gluing stage the completeness and relevance of the data is also checked by the unique identifier of the device.

This stage includes an analysis of the node's operation (availability of the online status) and the need to re-send the lost part of the response repeatedly to another free working node in case of a sudden disconnection of the previous node, engaged in processing this part of the task.

For such a system there has been developed an application on popular mobile platforms - iOS and Android - which can be downloaded from the app store for each operating system, in accordance. It is to perform data receiving, processing, sending and formatting in the form of an answer for easy gluing by the server [10].

Thus, the use of mobile devices to carry out bulky and long-term calculations as small and medium-sized networks has several advantages over the use of stationary computers, namely: longer downtime of the operating system and standby time, an easy access to the Internet, easiness of platform configuration for tasks execution, centralized distribution system.

V. Implementation

For cloud computing system it is required to have main server connected to the Internet and web interface tool for creating and managing tasks as shown on figure 1. It this infrastructure for cross platform purposes is used HTTP protocol for transferring different types of data and for supporting different types of connection from main server to the working node.

Such system was built for testing purposes with using Node.js as main server for Android and IOS mobile phones. There are also many suitable libraries for Node.js that helps to implement all needed functionality and methods for MapReduce in this system. As for mobile phone application, it should be something like a sandbox that will help to unify all connected phones as nodes. Ionic framework is perfect for such purposes. This framework can build applications for both Android and IOS [11]. Also, server built on Node.js uses JavaScript as programming language, so Ionic is the best choice for mobile sandbox application, because it can emulate performable environment that can run JavaScript too. The same programming language on server and mobile node provides developer with much more flexibility and helps to write clearer MapReduce algorithms. It is very important, especially on Reduce phase, when server needs to collect all incoming data into completed result, and it will be more efficient if data from nodes will be in the same structure (thanks to the same programing language).

Thanks to MapReduce method, server can divide incoming tasks into uniform pieces, and knowing that all nodes have the same calculation power, it makes possible to forecast accurate calculation time of the whole task.

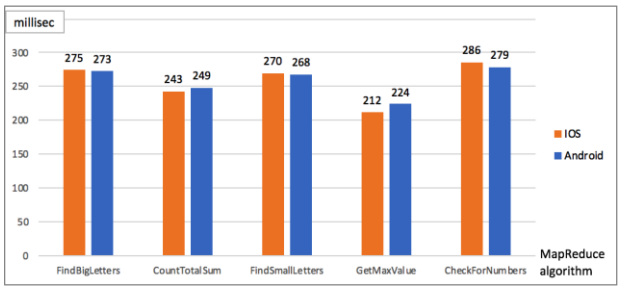

Fig. 2 Calculations on Android and IOS platforms

For testing purposes, system was loaded by different types of tasks, calculated on both Android and IOS platforms. Computing time was measured and the results of this test (figure 2) say, that Ionic framework really works and creates universal sandbox that gives possibility for the system to predict time of executions. All platforms was loaded with the same input data, so the little difference in millisecond could be ignored, because it could be caused by many small parameters, like temperature of CPU, amount of free RAM space, speed of the Internet connection [12], which cannot be amended by application or Ionic itself.

VI. Future Work

Today, the needs of modern society are growing fast and the amount of tasks` types, needed to be calculated, are growing too. Therefore, such cloud computing system is needed to be updated in time to follow new trends on the market.

At present time, this system can run and calculate only tasks connected to texts and simple numbers – more complicated logic is unbearable for now. Long arithmetic is not the strongest side of JavaScript programming language, but, thanks to Ionic sandbox, it is possible to make JavaScript engine improvements, which help to implement more complicated calculations and make this system reach the level of powerful neural networks.

Another part, that can be improved, is web interface. For the best experience, it should display more detailed information about task execution.

Monetization is a very important part too. To make mobile phones` users be more interested in this occupation, system should present something like crypto currency or another type of benefits to working nodes` owners.

There is a big problem of battery life saving. Modern mobile phones are starving from low accumulator capacity. It makes mobile phones working only for one or two days working without doing some heavy tasks. So, to solve this problem, before scientists invite more effective mobile phones accumulators, it is possible to make calculation only when phone is charging. It is about 2 – 4 hours per day (depends on mobile phone usage).

978-1-5386-6611-1/18/$31.00 ©2018 IEEE

VII. CONCLUSION

This article proposes a solution to the problem of calculating big data. In contrast to the methods for solving the problem presented earlier in other papers, the described approach has the following advantages:

1. The chosen algorithm is well suited for implementation of the distributed system because it solves the problem of parallelizing the task between the working nodes; the separation of the scope of devices, that is, the establishment of independent communications between devices; management of the fractional division of a task, to achieve the performance of a mobile device for easiest operations, to reduce the load on the device battery and reduce the risk of loss of a large part of the calculation due to the sudden disconnection of the node from the Internet.

2. Client for a mobile device will be a simple application like others and it would not be difficult for user to install it. Thanks to it cross-platform architecture, it is not needed to make extra configurations from user.

3. An original method is presented that allows obtaining the processing time of the HTTP request with minimized noise introduced by the network delay, which in practice often has a wide range of values.

The big data calculations based on nodes which are mobile devices is really and can be reached in an automated way. An exception is that computing power of each mobile node is not so big as PC and there exists maximum limit of pieces in which each task can be divided on map phase. In general, the proposed method for solving problem of making big data calculations depends on devices, that should be working as nodes.

REFERENCES

[1] J. Dean, "Simplified Data Processing on Large Clusters", vol. III, Eds. New York: Academic, pp. 271-350, Sept 2014.

[2] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm", Nature, vol. 324, no 4, 1986.

[3] A. Klein and San G., "A Self-Adaptive Network-Aware Approach to Service Composition", *IEEE Trans. Services Computing,* vol. 1, no. 99, 2013.

[4] S. Hachul and M. Jünger, "An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs" , *13th International Symposium*, vol. 3843, p. 235-250, Sept. 2013.

[5] M. Foley, "Hortonworks delivers beta of Hadoop big-data platform for Windows", *ZDNet*, February 2013.

[6] L. Guth, "The effect of wavelength of visual perception latency", *EEE Trans. Electron. Syst.,* vol. AES-4, pp. 567- 623, 1964.

[7] S. Gilmore and N. Koch, "Non-functional properties in the model-driven development of service-oriented systems", *Software Systems Modeling,* vol. 10, no. 3, pp. 287–311, 2011

[8] I. Zhukov, "Integral telecommunication environment for harmonized air traffic control with scalable display systems", *Aviation*, vol.16, 2010.

[9] M. Rabin, "Efficient dispersal of information for security, load balancing and fault tolerance", *Journal of the ACM*, vol. 36(2), pp. 335–348, 1989.

[10] A. Fox, S. Gribble, Y. Chawathe, E. Brewer, and Paul Gauthier, "Cluster-based scal- able network services", *Proceedings of the 16th ACM Symposium on Operating System Principles*, pages 78– 91, Saint-Malo, France, 1997

[11] H. Remzi, E. Anderson, N. Treuhaft and Kathy Yelick, "Cluster I/O with River: Making the fast case common", *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems (IOPADS '99)*, pages 10–22, Atlanta, Georgia, May 1999.

[12] E. Riedel, C. Faloutsos and A. Gibson, "Active disks for large-scale data process", *IEEE Computer*, pages 68–74, June 2001.