Mathematical model of a "tail" computation in a Desktop Grid

Evgeny Ivashko

Petrozavodsk State University
Institute of Applied Mathematical Research
Karelian Research Centre of RAS, Petrozavodsk, Russia
ivashko@krc.karelia.ru

Abstract. Task scheduling is an important problem of Desktop Grids. Among other special problems there is a "tail" computation problem. It is related to a final stage of computation in a Desktop Grid, when the number of tasks is less than the number of computing nodes. The excess of computing power could be used to implement redundant computing. Special mathematical models are needed to reduce duration of this final stage. In the paper a mathematical model of the "tail" computation is considered. It allows to estimate probability of reducing the time needed to finish a certain task by a certain computing node. So, a task with the maximal probability of time reducing effect could be replicated to a free node.

1 Introduction

Desktop Grids are of special interest as a cheap, easy to install and support, and potentially powerful high-throughput computing tool.

Desktop Grid is a computational paradigm based on a distributed computing system which uses idle time of non-dedicated geographically distributed general-purpose computing nodes (usually, personal computers) connected over Internet. Desktop Grids popularity is motivated by quick growth of the number of personal computers, as well as huge increase in their performance.

The first large volunteer computing project SETI@home was launched in 1999, providing the basis for development of the BOINC platform. There are a number of middleware systems for Desktop Grid computing. However, the open source BOINC platform [EFT+06] is nowadays considered as a de facto standard among them. Today there are more than 60 active BOINC projects utilizing more than 15 million computers worldwide [boi22]. So, Desktop Grids hold their place among other high-performance systems, such as Computing Grid systems, computing clusters, and supercomputers.

A Desktop Grid consists of a (large) number of computing nodes and a server which distributes tasks among the nodes. The workflow is as follows. A node asks

the server for work; the server replies sending one or more tasks to the node. The node performs calculations and finishing, sends the result (which is a solution of a task or an error report) back to the server.

Task scheduling in Desktop Grids is an important problem. Among other special problems there is the "tail" computation problem. It is related to a final stage of computation in a Desktop Grid, when the number of tasks is less than the number of computing nodes. Special mathematical models are needed to reduce duration of this final stage. In the paper a mathematical model of the "tail" computation is considered.

The structure of the paper is following. Section 2 describes motivation and related works. Section 3 is devoted to a mathematical model of the "tail" computation. Finally, Section 4 contains final remarks and conclusion.

2 Motivation and Related Works

As BOINC is the most popular Desktop Grid middleware, the paper is aimed at scheduling problem based on BOINC workflow.

BOINC is based on the client-server architecture. The client part is software, which is able to employ idle resources of a computer for computations within one or multiple BOINC projects. It is available for computers with various hardware and software characteristics.

The server part of BOINC consists of several subsystems responsible for tasks generation, distribution, results verification, assimilation, etc. It is based on Open Source software: Linux, Apache, mariadb, php, etc.

For each computational task BOINC holds multiple mostly independent instances or *replicas*. Replicas are computed separately with different nodes, and then their results are compared with the aims of verification. The *quorum* concept is used to define the necessary number of successful results to obtain for one computational task. One of the received results should be accepted as the correct via results "voting". This mechanism is useful to overcome processing errors and sabotage. Moreover, the BOINC settings allow to create and distribute more task replicas dynamically as needed.

BOINC employs PULL model to interact with computing nodes. It sometimes (in case of large number of computing nodes and low computational time of a single task) lead to high server load. To avoid it can be used PUSH model (a case of Enterprise Desktop Grid [Iva15]); another way is to reduce an overall server load forming effective-size parcels of tasks instead of sending to a client a single task [MNI15]. A *deadline* is set for each task instance to limit its completion time. If the server does not get a result before the deadline, the task instance is considered lost. This happened when a task processing fall into infinite loop or computing node itself become unavailable (abandon the Desktop Grid) and the server can not determine that.

The replication mechanism as a form of redundant computing serves a number of purposes. The main purpose is to increase reliability, by increasing the

chance to obtain the correct answer in time even if some nodes become unavailable without having finished the task. This helps, in its turn, to improve efficiency (in particular, throughput of successful results). Replication and voting are incredibly important in volunteer computing as a counter-sabotage defence measure. The reputational quorum is a method of voting with higher-reputation (for example, in terms of reliability) nodes' votes valued more: this approach employs both replication and reputation techniques the same time. In more details the architecture of BOINC is described in [And04]. The problem of search for optimal replication parameters is studied in practice in [Kur16]; there are also different mathematical models developed with the same objective, for example, [MNI15].

A valuable problem of task scheduling in Desktop Grids is optimization of the "tail" computation. A distributed computational experiment involving a batch of tasks inherently consists of two stages (see Fig. 1). At the first one the number of tasks is greater than the number of computing nodes (in the very beginning it can be much more). At this stage the computational power is limiting the performance, so it is reasonable to supply each node with a unique task (without replication; from the point of view of the makespan replication is useless as it was shown in [GL04]). With time, the number of unprocessed tasks decreases until it is equal to the number of computing nodes: then the second stage called the "tail" starts. At this stage there is excess of computational power which could be used to implement redundant computing to reduce overall computing time. The "tail" computation problem is studied in [KCC07,BYSS+12].

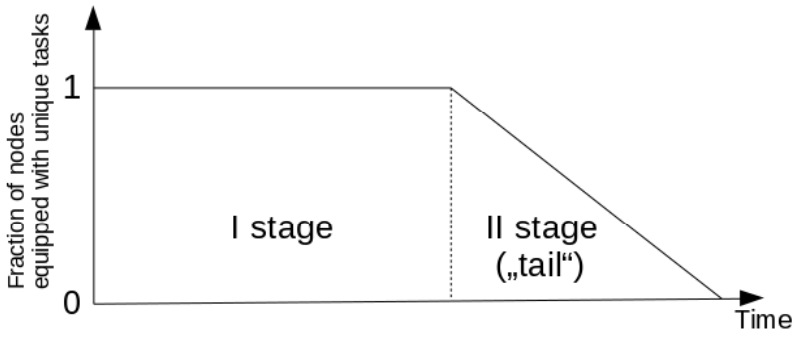

Fig. 1. Two stages of batch completion in a Desktop Grid.

A number of research problems related to specifics of Desktop Grids are connected to the two-staged batch completion. One of them is the fastest experiment or batch completion. In practice, the "tail" computation takes a long time (usually, about two or three times greater than deadline) because of missing deadlines. A computational network does not have information on the current status of tasks computation. So, the "tail" accumulates a lot of nodes that have abandoned the computing network. As a certain task assigned to such a node vi-

olates the deadline, it is assigned again to a different node, possibly also going to leave the network soon. So, this prolongs the "tail". The solution to the problem is in the redundant computing: currently processing tasks are assigned to vacant computing nodes. This strategy significantly increases the chances that at least one copy is solved in time. Employing this strategy, one should take into account characteristics of computational nodes (availability, reliability, computing power), processing the same task; accumulated task processing time; expected task completion time; and so on.

The fastest batch completion problem is more complex if a new tasks batch should be started after the current batch completion. In this case redundant computing reduces accessible computing power. One more complicated case is connected to inter-dependency between the tasks of the new batch and completion of certain tasks of the current batch.

3 Mathematical Model

To study the "tail" computation problem we present the following mathematical model.

There is a Desktop Grid consisting of n computing nodes.

Define a_k as computing performance of k-th node, $k = 1, .., n$.

Let computing complexity T is the same for all the tasks and known in advance. So,

$$T_k = \frac{T}{a_k},\ k = 1, .., n.$$

is working time needed for k-th computing node to finish a task. But a computing node of a Desktop Grid does not work continuously, so suppose that there is a probability distribution function $F_k(x)$ describing the probability of k-th node to finish a task in certain time. The form of the probability distribution function and its parameters are depend on computing node itself. In the simplest case it is availability rate.

Define d as deadline time of a task.

Having all thing considered, $\int\limits_{T_k}^{c} dF_k(x)$ is probability to finish a task in time c; $\int\limits_{T_k}^{d} dF_k(x)$ is a probability to successfully finish a task, i.e. not to miss the deadline; and $1 - \int\limits_{T_k}^{d} dF_k(x) = \int\limits_{d}^{\infty} dF_k(x)$ is probability to miss the deadline.

Redundant computing can be used to reduce time needed to finish a batch. At the beginning of the second stage there is a node appears which can not be equipped with a unique task. So, it can be used to reduce "tail" computation. The main question is following: What task should be replicated to reduce "tail" computing?

Let $n - 1$ nodes are equipped with unique tasks; each node is already have been computing its task for time $t_k,\ k = 1, .., n - 1$.

Probability of the node n to finish a task before the node k finishes it is

$$P(n,k) = \int\limits_{T_n}^{d} dF_n(x) \int\limits_{x+t_k}^{d} dF_k(y) + \left(1 - \int\limits_{T_k}^{d} dF_k(y)\right) \int\limits_{T_k}^{d} dF_n(x),\ k = 1, .., n-1. \tag{1}$$

Formula 1 can be used to make a decision which task should be replicated. A free node should take task of a node k^* maximizing probability to reduce task computation time:

$$k^* = \arg\max_k P(n,k).$$

4 Conclusion

The concept of Desktop Grid is a valuable part of high-performance computing industry. It allows to gather needed resources quick and easy to solve certain types of computational problems.

Task scheduling plays crucial role in providing computing performance of a Desktop Grid. This problem is complicated by intrinsic characteristics of Desktop Grids. So, big attention is payed to task scheduling.

One of the important problems in the domain of task scheduling is "tail" computation. It arises when the number of tasks become less than the number of computing nodes. Because of unreliable nature of nodes of Desktop Grids the "tail" stage can take much time despite of excess of computing power. The excess of computing power could be used to implement redundant computing. The special mathematical models should be developed to reduce time needed to complete this stage.

In the paper a mathematical model of "tail" computation is presented. It allows to estimate probability of reducing the time needed to a certain computing node to finish a certain task. So, a task with the maximal probability of time reducing effect could be replicated to a free node.

The presented mathematical model is based on probability to reduce computing time of a certain task. But there are different criteria could be used to optimize "tail" stage of computation. The examples of these criteria are mean computing time reduction, game theoretical multiobjective payoffs, or look-ahead criteria, etc.

Acknowledgements

This work is supported by the Russian Foundation for Basic Research (grant numbers 16-07-00622, 15-07-02354, and 15-29-07974).

References

[And04] David P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on*

Grid Computing, GRID '04, pages 4–10, Washington, DC, USA, 2004. IEEE Computer Society.

[boi22] BOINCstats. In *https://boincstats.com*, 2017-06-22.

[BYSS+12] Orna Agmon Ben-Yehuda, Assaf Schuster, Artyom Sharov, Mark Silberstein, and Alexandru Iosup. Expert: Pareto-efficient task replication on grids and a cloud. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 167–178. IEEE, 2012.

[EFT+06] T. Estrada, D.A. Flores, M. Taufer, P.J. Teller, A. Kerstens, and D.P. Anderson. The effectiveness of threshold-based scheduling policies in BOINC projects. In *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, pages 88–88. IEEE, 2006.

[GL04] Gaurav D Ghare and Scott T Leutenegger. Improving speedup and response times by replicating parallel programs on a SNOW. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 264–287. Springer, 2004.

[Iva15] E. Ivashko. Enterprise desktop grids. In *CEUR Workshop Proceedings*, volume 1502, pages 16–21, 2015.

[KCC07] D. Kondo, A.A. Chien, and H. Casanova. Scheduling task parallel applications for rapid turnaround on enterprise desktop grids. 5(4):379–405, oct 2007.

[Kur16] I. Kurochkin. Determination of replication parameters in the project of the voluntary distributed computing NetMax@home. In *International scientific conference "High technologies. Business. Society." 14-17.03.2016, Borovets, Bulgaria*, pages 10–12, 2016.

[MNI15] V.V. Mazalov, N.N. Nikitina, and E.E Ivashko. Task scheduling in a desktop grid to minimize the server load. In V. Malyshkin, editor, *Parallel Computing Technologies, International Conference on*, volume 9251, pages 273–278. Springer, 2015.