Michał ZOCHNIAK, Bartosz SAWICKI

Warsaw University of Technology

Reliable peer-to-peer computing system

Abstract. *Paper discuss a heterogeneous, distributed computing system commonly referred as a volunteer computing. Mechanism of determining correctness of results, based on replication and trust modeling, is proposed. In presented solution owners of computational power can flexibly join the network and sell spare machine time. Prototype implementation is based on light virtual machines distributed using BitTorrent protocol.*

Streszczenie. *Artykuł rozwa˙za niejednorodny, rozproszony system obliczeniowy (ang. volunteer computing). Zaproponowano mechanizmy określania poprawności wyników, które zostały oparte na modelowaniu zaufania i wiarygodności w˛ezłów obliczeniowych. W prezentowanym rozwi ˛azaniu właściciele mocy obliczeniowej mog ˛a w dowolnej chwili doł ˛aczac do sieci i sprzedawác czas pracy swoich maszyn. Prototypowa implementacja została oparta na technologii lekkich maszyn wirtualnych przesyłanych za pomoc ˛a protokołu BitTorrent. (**Wiarygodny system obliczeniowy typu P2P**)*

Keywords: cloud computing, desktop grid, volunteer computing, reliability model, virtual machines
Słowa kluczowe: chmura obliczeniowa, grid desktopowy, wiarygodność obliczeń, maszyny wirtualne

doi:10.12915/pe.2014.05.12

Introduction

Since chip manufacturing is reaching frequency limits related with technology, the parallel processing is definitely the future of computing. However there is not one approach to the parallelization. Well established systems with structured architecture, such as clusters and grids are dominating data centers all around the world. Their outstanding performance and efficiency can't be overestimated. On the other side we have unstructured systems based on extremely flexible, self-scaling, virtually indestructible peer-to-peer models.

Distributed computing is a paradigm in which computations are performed on a number of processors. Often, federation of computers is the only feasible way to perform more challenging tasks. This technology is used not only for academic purposes, but it is also widely adapted for business needs.

One way of acquiring computing power to do such computations is to buy or rent a specialized computer cluster.

Buying and setting up a set of machines can be a significant investment. Such cluster would then have to be maintained and will generate costs even when running idle.

"Could computing" was introduced to make parallel processing more economic and affordable. Companies such as Amazon offer „Pay as you go computing", because one only pays for what they use and there are no minimum or flat fees.

User is only paying for server time (in hours), disk space and bandwidth. When one user has finished his work on cloud server, cloud service providers can allocate his resources for other users. So in addition to lower costs, this solution may be also considered as more environment friendly, as we don't

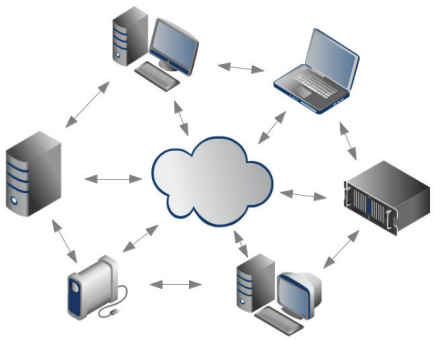

Fig. 1. Idea of decentralized, heterogeneous computing system. Resources are spread over randomly located machines which implements peer-to-peer model for direct communication.

The second direction in parallelization is „volunteer computing". This term is used to describe a heterogeneous distributed computing system where computing power is provided by, often unaffiliated, computer owners. Home users choose to share their computing resources when their computer is not in use. For example, program in a form of screensaver can perform computing tasks when user is away from the PC. An corporate networks can also benefit from such system. One could envision that idle time of many office computers produce significant income.

Prior works

There are many widely known projects making benefits from unstructured, distributed computing model. SETI@home, Folding@home, Climateprediciton.net were able to attract thousands of volunteers and successfully build system with petaflop power. Above systems where designed to solve specific problems, but BOINC is a general platform which allows to take advantages from decentralized volunteer computing [1]. It has been proved that wide range of satisfiability problems (SAT) could be efficiently solved using this architecture [2].

Contrary to homogeneous grid systems, decentralized computing networks suffer from problems related with its reliability. Establishing collaboration systems and maintaining trust within such system has been addressed in many research works over past decade. Liu and Shi [5] discuss trust and reputation management not only in computing systems, but also in fields like e-commerce. Domingues et al. [6] revise concepts of measuring trust in peer to peer system such as replication, sampling techniques, checkpoint-based verification, or reputation systems. Silaghi et al. [7] go through different models of malicious nodes sabotaging desktop grid computing platforms. Off-line processing algorithm is proposed to spot malicious nodes before accepting any computation results. Yurkewych et al. [8] present model of cost effective auditing of results using game theory. Optimal strategies for replication and auditing by using trusted nodes are discussed. Game theory is applied to create a model for optimal budgeting, under assumption that clients are solely motivated by the desire to maximize profits. Bendahmane et al. [9] examine replication-based voting algorithm for MapReduce computation using cloud computing. The system also provides dynamic blacklisting using weight threshold and maximum error index, improving reliability of whole mechanism.

Question raised by the authors of this paper is whether ordinary people could monetize their computing power using decentralized computing schema. Success of Bitcoin currency [3], where at the beginning the incentive to join net-

work was the ability to earn money in the process of „mining", suggests that it is worth to consider. Aspects related with trade-off between efficiency and reliability of the computational market will be discussed.

System architecture

Proposed system is based on idea of a free market for computing resources. Owners of computers that want to offer their resources would become Sellers. Entities that want to perform computation would become Buyers. One can imagine that there are many competing market systems, so it is essential point to design market which is more efficient than others.

Sellers that are willing to offer their resources are matched with Buyers that want to perform (or outsource) a computation work. Buyers are charged per work unit and Sellers are compensated for their work according to market rules. Buyers may choose how much are they going to pay for the work. The price then affects priority of how Sellers are choosing what to work on next, and is directly related to how quickly Buyer is going to receive results. Ideally, the price should regulate itself with the rules of free market, depending on the demand and total computing power put to sell.

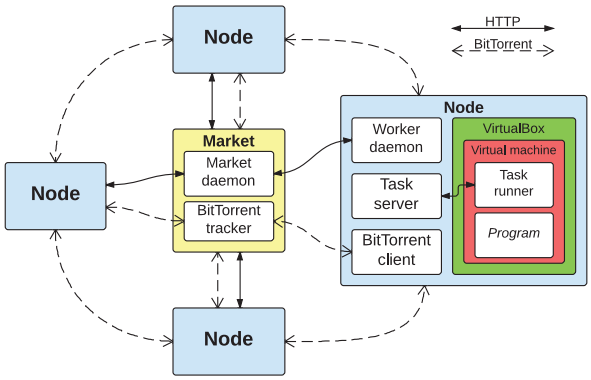

Fig. 2. Prototype architecture of computation market system

Technical system design is presented on Figure 2. It is based on a central, authoritative, entity called *market*. Rationale behind centralized market approach is that currently there is no way to securely and confidently handle transferring work and payment between buyers and sellers. Market servers is a escrow service which verifies results and payments. Each buyer does not have to trust every seller it „contracts", it is enough that both buyers and sellers trust the market. This model is widely adopted in many communication protocols, and known as a "Trusted third party".

To provide maximum flexibility of defining computing problems, virtualization technology is used. Similar approach was presented by designers of CernVM technology [4]. Buyers provide virtual machines with programs implemented with their technologies of choice. Seller runs VirtualBox system in headless (background) mode. Machines provided by buyers are expected to be set so they run and exchange data without user interaction. HTTP API is provided for virtual guest systems to do that, so the architecture completely platform and technology independent.

The machine images are then distributed to all peers willing to participate in computation. BitTorrent protocol is used in order to minimize bandwidth needed. Market server is running a BitTorrent client to initially send the project machine („seed" in BitTorrent language) - before enough clients get it so it can propagate on its own.

Prototype also consist of a server daemon that is ran by Market operator. Server uses HTTP to exchange information with the computational nodes. So it could be summarized that sellers fetch new projects (in form of virtual machines) by the BitTorrent system and run it inside VirtualBox. Then work units are sent by the HTTP protocol (in form of arbitrary data blobs).

Reliability model

To deal with reliability of the sold results, our system estimates trust of each node and each working unit. Jobs are recomputed by another nodes, and when the results match, submitted job is considered *confirmed*. Trust of particular node depends on how many jobs are confirmed and by whom. The more trusted a node is, the more valuable its confirmation is. As node has more of its results confirmed, its trust grows. Jobs are recomputed to ensure correctness of results. Hardware or software malfunction may render the results incorrect and not only useless for this particular job, but for the whole project, as it might be difficult to find out which (and if any) job has incorrect result. Redundant computing is also a method of protecting the system from dishonest nodes, which send incorrect results on purpose. Jobs that serve as verification are indistinguishable from others, which makes it even harder to cheat the system. This method is also called *replication*.

As presented in equation (1) mathematical model of reliability R of each work unit result is based on two factors: normalized historical experience E of the submitting node and its normalized correctness C within current project. Coefficient α decides which one is dominating, our initial setup is $\alpha = 0.5$. R is a function of $n$ode, $p$roject, $t$ime; and its value is between 0 and 1.

$$R(n,p,t) = \alpha E_{norm}(n,t) + (1-\alpha)C_{norm}(n,p) \tag{1}$$

Experience E is sum of all collected credits from historical contributions in the previous projects. Experience is a subject of time decay process.

$$E(n,t) = \sum_{i=0}^{k} (\frac{1}{2})^{t-t_i} \text{Credits}_i, \tag{2}$$

where k is number of finished projects, Credits_i is number of credits earned on the i'th project.

Equation (3) describes correctness C. It is a sum of all confirmations within current project. Let node_A be a node that submits results for particular job. Then node_B gets the same job, as part of redundant computing property of the system, and submit its results back. When the results match, node_A is awarded by the correctness points equal to the current R of node_B and node_B is awarded by the R of node_A.

$$C(n,p) = \sum_{i=0}^{l} \text{confirming} R_i, \tag{3}$$

where l is total number of completed task by the node for the project.

Maintaining reliability shall not depend only on measuring trust. Volunteer computing systems face problem of not being able to distinguish whether two nodes (presenting two different identities) are not controlled by the same entity. Sybil attack [10], where one entity can control significant number

of nodes and take majority of the network (since sending false results take little to none computational power), is a real threat for such systems. Additional means of ensuring correctness have to be employed. One of them is choosing job distribution algorithm that would distribute jobs in a way minimizing effect of malicious nodes colluding. However, only reliability model assuming that malicious nodes do not collude is discussed in this paper.

Reliability simulations

To verify behavior of the trust evaluation, simulations using 50 virtual nodes and a project consisting of 1000 tasks were made. Virtual nodes differed from one another to resemble real volunteers with different machines connected to the system. Nodes were different in time needed to complete a job, to simulate different computer speeds of nodes. Also nodes were idle for periods of time in between tasks, to simulate network latency in downloading new jobs and submitting results.

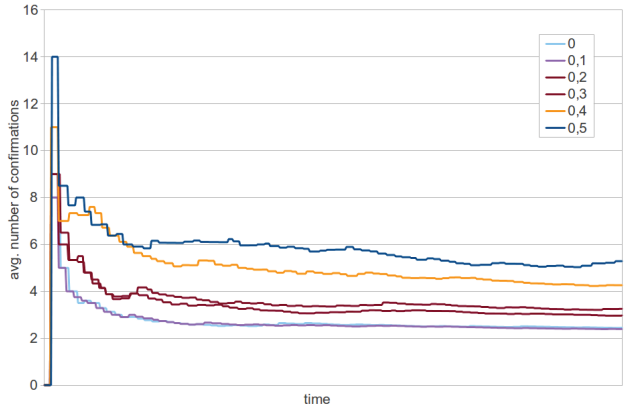

Fig. 3. Average number of confirmations over time for different ratios of incorrectness

First simulation (see Fig. 3) was conducted to verify behavior of algorithm in relation to the number of confirmations needed for each job. Simulation was ran 6 times, each time with different ratio of incorrect results. Figure 3 shows average number of confirmations over time. As the network stabilizes (nodes start with having reliability R), average number of confirmations gets to around 2 for simulations with 0 incorrect results factor (meaning that every result sent is correct). With factor 0.5 (50% of results are wrong), average number of confirmations stabilizes at around 6. This values should be compared with BOINC [1] fixed policy, where each work is repeated 3 times.

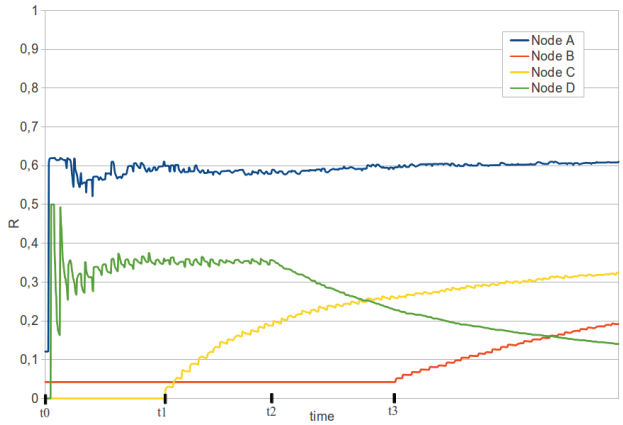

Fig. 4. Nodes' reliability factor R as a function of time

Second simulation (see Fig. 4) was performed to check how the reliability model reacts for changes in the system. Four time moments are noticeable:

time t_0: Nodes A and D start to work, however A has more computational power than D.

time t_1: Node C starts to work.

time t_2: Node D stops working; reliability decay is observed.

time t_3: Node B with non-zero historical experience joins the system and starts working.

Few things can be observed. First of all, we can see the stabilization period. Remember, that reliability factor is based on factors which are normalized - that is, reliability of a node depends on reliability of other nodes. To help the network establish reliability factors faster, random factor is added. At the beginning, it is mostly the hazard which decides on node's reliability. Secondly, when node is switched off, its reliability slowly declines. Network does not additionally penalize nodes which do not send results. Decline is only caused by normalization - because reliability of rest of the network increased, normalized reliability of idle nodes gets lower.

Summary

In this paper we discussed problems with designing a peer-to-peer computing cloud. Performed simulations show that our reliability model ensures correct results, assuming that malicious nodes do not collude with each other. Dynamic replication system depending on trust provides more efficient computation when network is generally correct and more safety when network is found to be incorrect.

As future work, we would like to introduce additional mechanism of ensuring correctness, that would take colluding nodes into account. Proper job distribution algorithm has to be devised and also additional heuristics to spot malicious nodes should be implemented.

BIBLIOGRAPHY

[1] David P. Anderson, *BOINC: A System for Public-Resource Computing and Storage*. Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 2004

[2] Mikhail Posypkin, Alexander Semenov, Oleg Zaikin:, *Using BOINC Desktop grid to Solve Large Scale SAT Problems*. Computer Science, 13 (1), 2012

[3] Satoshi Nakamoto: *Bitcoin: A Peer-to-Peer Electronic Cash System*, www.bitcoin.org, 2008

[4] Ben Segal, Predrag Buncic at al.: *LHC Cloud Computing with CernVM*, 13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research ACAT2010 Jaipur, India, February 22-27, 2010

[5] Liu, Ling, and Weisong Shi. "Trust and reputation management." Internet Computing, IEEE 14.5 (2010): 10-13.

[6] Domingues, Patricio, Bruno Sousa, and Luis Moura Silva. "Sabotage-tolerance and trust management in desktop grid computing." Future Generation Computer Systems 23.7 (2007): 904-912.

[7] Silaghi, Gheorghe Cosmin, et al. "Defeating colluding nodes in desktop grid computing platforms." Journal of Grid Computing 7.4 (2009): 555-573.

[8] Yurkewych, Matthew, Brian N. Levine, and Arnold L. Rosenberg. "On the cost-ineffectiveness of redundancy in commercial P2P computing." Proceedings of the 12th ACM conference on Computer and communications security. ACM, 2005.

[9] Bendahmane, Ahmed, et al. "Result verification mechanism for MapReduce computation integrity in cloud computing. Complex Systems (ICCS), 2012 International Conference on. IEEE, 2012.

[10] Douceur, John R. "The sybil attack." Peer-to-peer Systems. Springer Berlin Heidelberg, 2002. 251-260.

Authors: Michał Zochniak and Bartosz Sawicki, Faculty of Electrical Engineering, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warszawa, Poland, email: zochniam,sawickib@ee.pw.edu.pl.