Future Generation Computer Systems 29 (2013) 242–249

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A P2P computing system for overlay networks

Grzegorz Chmaj, Krzysztof Walkowiak*

Department of Systems and Computer Networks, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland

ARTICLE INFO

Article history:
Received 30 May 2010
Received in revised form
26 August 2010
Accepted 8 November 2010
Available online 24 November 2010

Keywords:
Computing systems
P2P
Simulation
Overlay networks

ABSTRACT

A distributed computing system is able to perform data computation and distribution of results at the same time. The input task is divided into blocks, which are then sent to system participants that offer their resources in order to perform calculations. Next, a partial result is sent back by the participants to the task manager (usually one central node). In the case when system participants want to get the final result, the central node may become overloaded, especially if many nodes request the result at the same time. In this paper we propose a novel distributed computation system, which does not use the central node as the source of the final result, but assumes that partial results are sent between system participants. This way we avoid overloading the central node, as well as network congestion. There are two major types of distributed computing systems: grids and Peer-to-Peer (P2P) computing systems. In this work we focus on the latter case. Consequently, we assume that the computing system works on the top of an overlay network. We present a complete description of the P2P computing system, considering both computation and result distribution. To verify the proposed architecture we develop our own simulator. The obtained results show the system performance expressed by the operation cost for various types of network flows: unicast, anycast and Peer-to-Peer. Moreover, the simulations prove that our computing system provides about 66% lower cost compared to a centralized computing system.

1. Introduction

Distributed computing systems play a very significant role in today's academic and business world. These systems are applied to compute tasks requiring huge computation power, which is not available on a single machine (even on a super-computer). They are mainly divided into two categories: grid computing systems and Peer-to-Peer computing systems. Grids are constituted by organizations and institutions and contain a small number (usually up to hundreds) of machines connected by a computer network of high capacity [1,2]. Grids may share many kinds of resources: computing power, disk space, data, sensors, etc. [3]. Resources are centrally managed using a Resource Management System (RMS) that covers the following aspects: customizability, extensibility, scalability, etc. [4]. The scheduling is an important element of the grid that has a large influence on the system's efficiency [2,5,6]. The scheduling process should include such issues as: resource discovery, information gathering and task execution, concurrently with authorization, application management and monitoring [5]. Many previous papers assume simplifications of the scheduling model, correspondingly in this paper we focus on one aspect of scheduling, i.e., assignment of computational tasks to computing nodes. Constituting the grid system is a sophisticated task regarding both technical and financial aspects. Therefore, other distributed computing systems — like Peer-to-Peer (P2P) computing systems — have emerged in recent years [7]. These systems are built using many private machines, which are most often home computers (PC or Macintosh) or even gaming consoles. The user installs a special software on her/his machine and registers into a selected computing project. Then, she/he receives data chunks to compute and sends the results back to the central node, where partial results are combined into the final result. Network connections used in P2P computing systems are regular home access links such as DSL or cable. This approach is much simpler than grids, since the only requirement is to provide a suitable software. The P2P computing approach allows for unreliability of participants—they may freely join and leave the computing system, which is not used in grid systems. The most popular P2P computing project is SETI@home (started in 1999), which aims at looking for extra terrestrial intelligence [8]. It is based on a BOINC architecture [7,9]. Projects based on the BOINC aggregate almost 2 million users all over the world with over 5 million hosts having 5 TeraFLOPS of power (April 2010). Seti@home is the largest BOINC P2P computing project (over 1 million of users), other popular projects are: Einstein@home (250 thousand users—search for pulsar stars) and Climate Prediction (224 thousand users—climate change prediction). There are also other Peer-to-Peer computing frameworks, including systems dedicated to compute one project, e.g., [10,11].

Grid systems are mostly centrally managed, which means that there is one central node, which takes care of task preparation,

* Corresponding author. Tel.: +48 71 320 3539; fax: +48 71 320 2902.
E-mail address: krzysztof.walkowiak@pwr.wroc.pl (K. Walkowiak).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.11.009

scheduling and managing of results. P2P computing systems may also use this model, but as home users contribute their resources, they may also want to participate in the results. This entails the problem of distributing the complete result to each participant. In the case when the result is combined at one central node, a huge number of participants interested in the results and requesting it from one central machine may cause a server overload or even denial of service. For instance, the authors of the *Electric Sheep* project [12] propose a distributed computing system, which renders artificial forms of life—and allows participants to get complete animation. The animation is rendered by participants, but combined into the final result at the central node. The authors underline that their system struggles with the problem of downloading the final animation from the central node and plan to use a BitTorrent [13] protocol to solve this issue.

In this paper we propose a new idea of a distributed computing system. The main novelty is that the system is not centrally managed—partial results are not sent back to the central node, but transferred between nodes directly. Similar to the BitTorrent protocol [13], our system uses a special node called a *tracker*. The objective of the tracker in our system is twofold. First, the tracker performs the scheduling, i.e., it assigns individual tasks to computing nodes according to received requests. Second, the tracker maintains and offers the current database including information on location of already calculated results.

Distributed computing systems can be modeled like other network systems using a static approach, which assumes the creation of an optimization model (including decision variables, constraints and objective functions) [14,15]. Another popular approach to research on distributed systems is simulation, which applies a dedicated software and aims to act as close to a real (modeled) system as possible.

The proposed P2P computing system works in an overlay mode and uses the Internet as a transport layer. The overlay approach assumes that the network includes two layers: an upper application layer and a lower transport layer [16,17]. The transport layer provides direct connectivity between overlay nodes including routing. Moreover, some Quality of Service guarantees can be assured by the transport layer. Each node (participant of the computing system) is connected to the overlay network by an access link with specified download and upload bandwidth expressed in bps.

The major motivation for this paper is to propose a novel approach to distributed computing systems, as today's systems do not provide effective mechanisms to deliver results of computations to all participants. The main contributions of this paper are as follows. (1) A novel architecture for a P2P computing system considering both computation and result distribution. (2) Decision strategies developed for computing nodes participating in the system. (3) A simulator of the proposed distributed computing system implementing various types of network flows (unicast, anycast and P2P). (4) Simulation results showing the influence of the type of network flow and decision policies.

The remainder of the paper is as follows. In Section 2 we introduce the P2P computing system in detail. Section 3 includes the description of the simulator developed to examine the proposed system. In Section 4 we show results of the experiments. Section 5 includes related work. Finally, the last section concludes this work.

2. A P2P computing system architecture

In this section we present an architecture of a new P2P computing system. The main objective of the system is to minimize the OPEX cost of the system compromising both: computing and transfer costs. The former element refers to operating costs related to computation (e.g. energy, maintenance). The latter element is the delivery cost in the overlay network usually defined for each pair of nodes (e.g., lease cost of the access links). We assume that the system is collaborative and all participants want to receive the whole result. However, our architecture could also be easily used to model the situation when only some of participants download the result.

The system consists of many machines connected into one logical structure. It takes a computational task as the input, which is then computed by participants. As delivery of all results to each participant introduces significant network traffic, it is essential to provide effective distribution algorithms. Like in most distributed computation systems (e.g., Seti@Home and many others based on the BOINC [7] framework), the input data is divided into uniform blocks (we call them *source blocks*), which are sent to system participants in order to be computed. BOINC systems assume that the result of computation (a *result block*) is sent back to the central node, which collects all results for further processing and analysis. On the contrary, our research considers a situation, when all system participants are interested in the final results. Thus, the results are not delivered to the central node, but they are distributed over the network to all requesting users. As an example we can mention rendering distributed images or distributed rendering of 3d movies, where all participating users want to see the result of the rendering. In the case when the central node is used to combine partial results into the final result, downloads performed by many users cause high load and congestion problems at the central node. We investigate how to bypass the central node and use the advantages of various flows, to optimize distribution of results.

Let us now describe the details of our system architecture. It contains two types of elements:

- *nodes*—regular machines that do the computation and exchange results between each other;
- *tracker*—a central element, which assigns source blocks to nodes. It is also used as a database including the location of all results, what is similar to the idea of the tracker node in the BitTorrent protocol.

A node requests the source block from the tracker when it has free computation resources available (operation 1 in Fig. 1). The tracker responds with the source block if available (operation 2) or signals that no more source blocks are available for computation, i.e., all blocks included in the current computing project have been assigned to nodes for computation. When the node receives such information, it stops requesting new blocks from the tracker. The node has to compute at least one source block (operation 3), to become a participant of the project. The tracker stores information about all participants and does not provide information about result locations for nodes, which are not present on the participants' list. This way the system protects itself against unfairness—every node has to contribute to the system in order to obtain final output results. A node that wants to get the result, which was computed by another node, sends a location request to the tracker (operation 4). Then, the tracker responds with known locations (operation 5). The node selects one of them according to the selection policy (decision strategy). To make the tracker locations' list complete, the node sends an update to the tracker every time it acquires a new block available to send. This happens in two cases: the node has finished computing the source block or the node has finished downloading the result block from another node.

All elements of the system (nodes and tracker) are connected through the overlay network, which is the Internet in our case. This way we consider a network as one unified structure, that provides a direct connection between every two elements connected to this network. We do not consider how such a connection (in our