Web Distributed Computing Systems Implementation and Modeling

Fabio Boldrin, University of Ferrara, Italy
Chiara Taddia, Lepida S.p.A., Italy
Gianluca Mazzini, University of Ferrara, Italy

ABSTRACT

This article proposes a new approach for distributed computing. The main novelty consists in the exploitation of Web browsers as clients, thanks to the availability of Javascript, AJAX and Flex. The described solution has two main advantages: it is client-free, so no additional programs have to be installed to perform the computation, and it requires low CPU usage, so client-side computation is no invasive for users. The solution is developed using both AJAX and Adobe®Flex®technologies embedding a pseudo-client into a Web page that hosts the computation. While users browse the hosting Web page, computation takes place resolving single sub-problems and sending the solution to the server-side part of the system. Our client-free solution is an example of high resilient and auto-administrated system that is able to organize the scheduling of the processes and the error management in an autonomic manner. A mathematical model has been developed over this solution. The main goals of the model are to describe and classify different categories of problems on the basis of the feasibility and to find the limits in the dimensioning of the scheduling systems to have convenience in the use of this approach. The new architecture has been tested through different performance metrics by implementing two examples of distributed computing, the cracking of an RSA cryptosystem through the factorization of the public key and the correlation index between samples in genetic data sets. Results have shown good feasibility of this approach both in a closed environment and also in an Internet environment, in a typical real situation.

Keywords: Adaptive Systems, Autonomic Systems, Distributed Computing, Resiliency, Web 2.0

INTRODUCTION

A well-known design pattern of distributed computing is the so called Master-Slave architecture, in which a problem is divided into a number of parts, called sub-problems; these sub-problems are separately solved, usually by independent computers/processors. Results of solved sub-problems are then reassembled in the correct manner to form the final solution of the original bigger problem.

There are many different hardware and software architectures that can be used for Master-Slave applications, such as for ex-
ample peer-to-peer or client-server ones. In peer-to-peer architecture there is no special machine that provides a service or manages the network resources but all the responsibilities are uniformly divided among all machines. A client-server architecture is based on a client that, through a specific code, contacts the server for data; the server side of the computation usually manages the distribution process giving a sub-problem as response of client query. Server can also assemble the subproblems to give the final solution of the original problem.

Literature offers a huge number of works devoted to the study of optimizing this class of distributed computing systems. Some works face the problem from the point of view of the network performances, in terms of connectivity (Massen & Bal, 2007) and bandwidth sharing mechanisms (Marchal, Primet, Robert, & Zeng, 2006). The paper (Ganeiredo, 2006; Agrawal, Boykin, Figu) describes a distributed system that combines virtual machine, overlay networking and peer-to-peer techniques to create scalable wide-area networks of virtual workstations for high-throughput computing.

A great variety of Master-Slave distributed computing projects have grown up in recent years, for example the Folding@home project of the Stanford University Chemistry Department, or the SETI@home of the Space Sciences Laboratory at the University of California, Berkeley and also the LHC@home, a project by the CERN for simulation on the new Large Hadron Collider.

A common aspect of these architectures and of the cited examples is the need of each peer or client to install a specific code to solve its sub-problem.

The work we present in this article develops a distributed computing system based on a client-server architecture, which uses the extremely large number of machines connected to the Internet as clients, in a non-invasive way. Internet has already been used as a network for distributed computing such as in the various @home projects cited before and in many others more or less publicized. The main innovation of our system is the idea of realizing a distributed computing in Internet without any additional software installation at the client side, but using only the direct capabilities of a Web browser. In fact, thanks to the now deeply diffused Web 2.0 technologies, such as Javascript, AJAX, FLEX, we can exploit the client side Web browser to execute tasks during user’s navigation: this is done by including directly on a Web page the code to be executed by the client. Every browser navigating into that particular Web site that hosts the pseudo-client of the system does a little amount of calculations using a small percentage of CPU, so users do not lose usability of their computer.

Another work, presented in (Milenkovic, Robinson, & Al, 2003), is oriented towards a Web browser based solution: authors consider what would be needed to make the Internet an application-hosting platform; a networked, distributed counterpart of the hosting environment traditional operating systems provide to applications within a single node. The foundation of their proposed approach is to disaggregate and virtualize individual system resources as services that can be described, discovered, and dynamically configured at runtime to execute an application.

Our work has the following purposes:

- Test the actual feasibility of this innovative approach and measure the capabilities and performances of our architecture in performing distributed computing. To this aim we have defined some performance metrics and we have implemented the distributed computing code for two specific problems: the factorization of a large integer (a basic step for the RSA cypher crack) and the calculus of the correlation between genetic samples. The first problem substantially aims at showing how to implement the base system while the second aims at testing the performance of the system in presence of large data packets and relatively limited amount of computation.
A Hybrid Kernel Extreme Learning Machine and Improved Cat Swarm Optimization for Microarray Medical Data Classification
www.igi-global.com/article/a-hybrid-kernel-extreme-learning-machine-and-improved-cat-swarm-optimization-for-microarray-medical-data-classification/165040?camid=4v1a