Parasite: Distributing Processing Using Java Applets[1]

Remo Suppi, Marc Solsona, and Emilio Luque

Dept. of Computer Science, University Autonoma of Barcelona,
08193, Bellaterra, Spain
Remo.Suppi@uab.es, Marc@ptv.es, Emilio.Luque@uab.es

Abstract. There is wasted and idle computing potential not only when applications are executed, but also when a user navigates by Internet. To take advantage of this, an architecture named Parasite has been designed in order to use distributed and networked resources without disturbing the local computation. The project is based on developing software technologies and infrastructures to facilitate Web-based distributed computing. This paper outlines the most recent advances in the project, as well as discussing the developed architecture and an experimental framework in order to validate this infrastructure.

1 Introduction

In the last five years, a growing interest in distributed computation has been observed. Projects such as Seti@home and Distributed.net are examples of metacomputing popularity and extension [5,6]. These two projects are clear examples of the trends in using particular user equipment for distributed computing. Simply stated, metacomputing is a set of computers (whose geographical distribution is of no relevance) that are interconnected and that together act as a supercomputer. The metacomputing concept is a very generic definition that has undergone specialization through several proposals. [1-7]

Our proposal, referred to as *Parasite* (**Para**llel **Site**), results from the need for computing power and from the fact that it is possible to extract available resources with idle time and without disturbing the local user workload. This is the principle underpinning several metacomputing projects; however, our project introduces new ideas with respect to user intervention, available resources, net interconnection, the distributed programming paradigm or the resident software on each user's computer.

2 Our Proposal: Parasite (Parallel Site)

The main idea of Parasite is the utilization of personal computers as computing nodes and the interconnection network without carrying out modifications in the hardware

[1] This work has been supported by the CICYT under contract TIC98-0433 and TIC 2001-2592.

B. Monien and R. Feldmann (Eds.): Euro-Par 2002, LNCS 2400, pp. 598–602.

interconnection equipment and without the need to install software in the user's computers -UC- (machines that will integrate the Parasite distributed architecture). To this end, previously installed software in the UC connected to Internet is used: the Internet browsers (navigators). These applications, together with the possibility of executing Java Applets, open the possibility of creating computing nodes.

Our proposal is based on creating a hardware-software infrastructure that supports the embarrassingly parallel computation model and that the local user does not have to modify installed software (or install new software) in the local machine. This infrastructure must provide to the distributed applications programmer with the benefits of massive parallelism (using the CPU free time of the UC) and without the typically attendant costs (topology, tuning application-architecture, mapping & routing policies, communication protocols, etc.). Furthermore, all this must be transparent to the local user (only initial assent will be necessary). From the programmer's point of view (the user of the Parasite infrastructure), the distributed application code will be executed in the maximum number of available resources at each moment, without changes in the application code.

Figure 1 shows the Parasite architecture (clients & server), the information fluxes and Internet traffic in two different operation modes. The concept of Parasite is based on two operation forms: **collaborative** (for users who wish to grant their resources to the distributed computing process in any place of Internet) and **transparent** (the local user does not make an explicit web petition to the Parasite server; the Java Applet is sent transparently by the Parasite host to the user computer during the user navigation). The first continues the collaborative line of work set out by the projects referred to above [5-7]. The second form of work (the transparent form) is proposed for local network environments. The Parasite server (fig. 1) is who coordinates, distributes and collects the data between the UCs. The UCs can be working, according to the location, in collaborative mode (UC in any place of Internet) or transparent mode (UC in a private or corporative net). These UCs will execute a Java applet sent by the server and each applet will form part of the distributed application code.

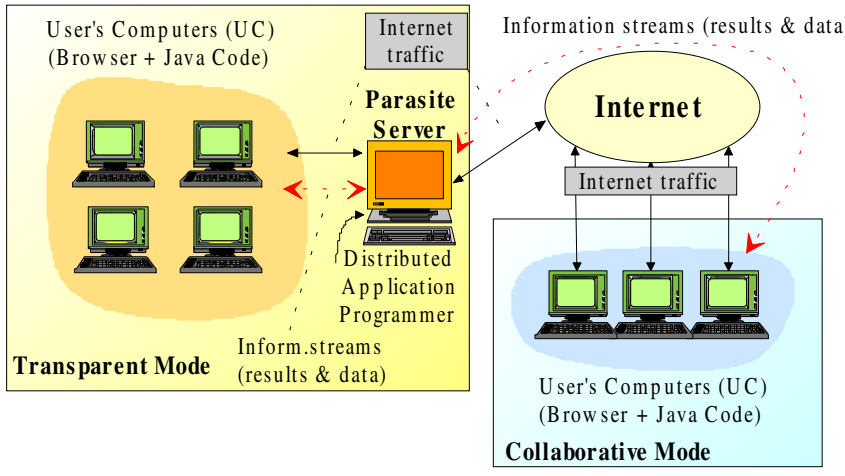

Fig. 1. Parasite architecture & working modes (collaborative & transparent)

The applet will be executed during the time that the user continues to use the navigator. It is therefore very important for the project objectives to analyze the users' navigation standards. This analysis will allow an estimation of the time (mean time) that the CPU remains free for the distributed computing (without affecting the user workload). With data obtained from [8,9], we can conclude that the average CPU time for distributed computing oscillates between 75% to 86% of the users' navigation time, according to user type and when considering the worse I/O case.

The Parasite architecture has been designed to sustain (but is not limited to) the "ideal" computation from the parallel computing point of view: a computation can be split into an independent number of tasks and each of these can be executed on a separate processor. This is known in the literature as *embarrassingly parallel computations* or *pleasantly parallel computations* [10]. There are a considerable number of applications appropriate for this model, such as the geometrical transformations of images, the Mandelbrot set, Monte Carlo methods, parallel random number generation, etc. The Parasite architecture also sustains variations of this model, such as the *nearly embarrassingly parallel computations,* where results need to be collected and processed, suggesting a master-worker organization.

3 Experimental Framework

In order to show the possibilities and performance of the Parasite architecture, real distributed computing experiments have been carried out. The program chosen for these experiments is based on the RSA laboratories proposal for 1997 to prove the robustness of the RC5 (RC5-32/12/8 –64 bits key-) encryption algorithm. [11,12]

Figure 2 shows the evolution of calculation (number of encrypted keys tested) versus the time without local workload whit Parasite working in collaborative mode. Figure 2.a shows the total number ($*10^6$) of key computed by the system. Figure 2.b shows average values: number of key ($*10^3$) per second and in the last 10 seconds. In figure 2.b, only the data for the first eight users are represented (in order to provide details). As can be observed in figure 2, the increase in computed key is practically linear. This fact is predictable, because the computing process satisfies the *truly embarrassingly parallel computation* model.

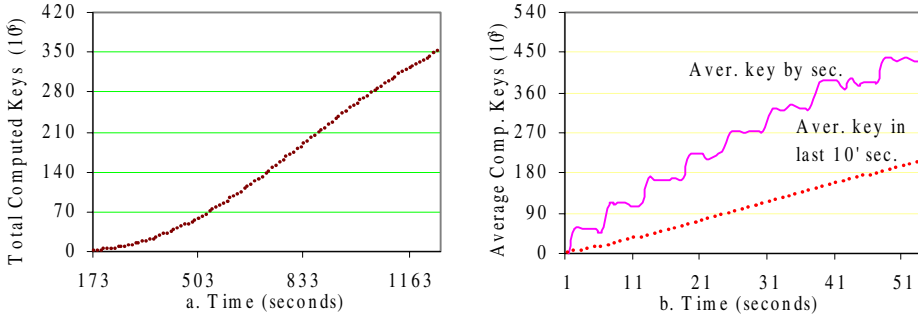

Fig. 2. Collaborative Mode: Evolution of Computed Keys.

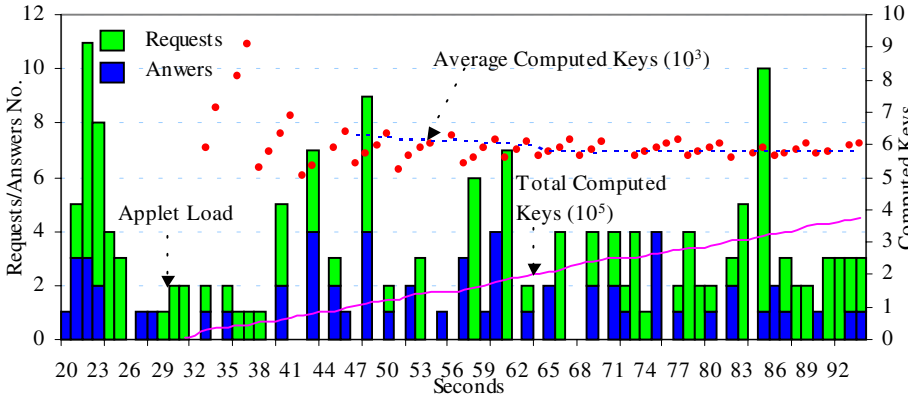

Fig. 3. Client behavior on google.com

In order to show client/server behavior in transparent mode, the www.google.com URL was selected. Figure 3 shows the navigator requests/answers number vs. the time and the behavior of the applet running in a representative node (computed keys/sec –dot line- and total keys –continuous line-). As can be observed in certain points of the computed keys/second (dots graph), there are some places where we do not find computed keys. This situation indicates a load increase in the local computer, and therefore the applet goes into a sleeping state. This situation generates a dispersion of the number of keys/sec, but if the tendency line (hyphens line) after the initial transient is observed, the system tends to be stabilized.

In order to compare system performance when the number of UCs is increased, a heterogeneous speedup has been defined. Figure 4 shows the speedup for a homogeneous system (continuous line) of 24 PC Pentium II 500 Mhz running in collaborative mode on a class C LAN. The dot line is the speedup for a heterogeneous system of Pentium III W9x, Pentium II Linux and Ultra 10 Sparc Solaris 2.x working in transparent mode in different LAN segments. As can be observed in figure 6, the results are excellent and the differences with respect to the linear speedup are due to the OS and net load. The results for the homogenous system and collaborative mode are better because the same LAN segment is used for the 24 machines and the server.

4 Conclusions

In a computer, there is wasted and idle computing potential not only when applications are executed, but also when a user navigates by Internet. To take advantage of this, an architecture named Parasite has been designed. This architecture allows jobs to be executed in the user computer without either affecting performance or modifying the user's work environment. The principal characteristic of the solution adopted is that the user does not have to install software in the local machine (only being required to give initial consent) and the Parasite system guarantees that it will not use more computing capacity than that which is idle. The system can work indistinctively in two modes: collaborative and transparent.

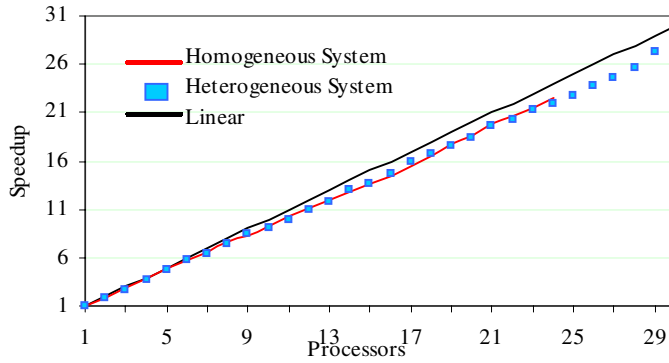

Fig. 4. Speedup

In order to show the capacities of the developed environment, a set of experiments based on the RSA laboratories proposal to prove the robustness of the RC5 encryption algorithm were undertaken. The conclusions for these experiments are that the environment is particularly suitable for applications based on the *(truly, nearly) embarrassingly parallel computations* model. The environment has been proven in homogenous-heterogeneous systems and the same or different LAN segments, the speedup obtained being close to the linear.

Future work will be guided towards: the need for a coordinated and hierarchical net of Parasite servers and the development of a set of applications based on *embarrassingly parallel computations* model in order to prove different granularity types and to determine their efficiency.

References

1. Anderson, T., Culler, D., Patterson, D. A case for NOW IEEE Micro (1995).
2. The Beowulf Project. (1998) http://www.beowulf.org
3. Litzkow, M., Livny, M., Mutka, W. Condor. A Hunter of Idle Workstations. Proc. 8th Int. Conf. Distributed Computing Systems. (1988) http://www.cs.wisc.edu/condor/
4. Foster, I., Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit. International Journal of Supercomputer Applications. (1997). http://www.globus.org/
5. Search for Extraterrestrial Intelligence Project. (2002) http://setiathome.ssl.berkeley.edu/
6. Distributed.net Project. (2002) http://distributed.net
7. Neary, M., Phipps, A., Richman, S., Cappello, P. Javelin 2.0: Java-Based Parallel Computing on the Internet. EuroPar 2000. LNCS 1900 (2000).
8. Nielsen Net Ratings. (2001) http://www.nielsen-netratings.com
9. Sizing the Internet. Cyveillance Corporate. (2000) http://www.cyveillance.com
10. Wilkinson, B., Allen, M. Parallel Programming. Techniques and Applications using networked workstations and parallel computers. Prentice Hall. ISBN 0-13-671710-1. (1999)
11. RSA Data Security Secret-Key Challenge. (1997) http://www.rsa.com/rsalabs/97challenge
12. Ronald Rivest. RC5 Encryption Algorithm. Dr. Dobbs Journal. 226 (1995)