Implementation of a Parallel and Distributed Make on NFS with GATOS

Bertil FOLLIOT
CNRS / Laboratoire MASI, Université Pierre et Marie Curie
4, Place Jussieu 75252 PARIS CEDEX 05 FRANCE
e-mail: folliot@masi.ibp.fr

Introduction

GATOS is a distributed tasks manager operating on a set of workstations connected by a local network. Its main purpose is to balance the load between different hosts in order to optimize the use of processors. The present version of this project is implemented on SUN workstations connected by ETHERNET.

The use of this manager is carried out on two levels. First, external description of applications with the help of an execution graph. This graph indicates the dependencies among all programs of an application. Secondly, take in charge of the manager in the conception phase of new projects. GATOS provides operations, which bring up a total transparency of the program distribution.

GATOS has been built as a set of cooperative servers. Each one treats requests by local clients and by remote services coming from other servers. Such requests may be file migration, remote program execution and remote file access.

I. GATOS and Parallel Applications

To improve load balancing, GATOS manages not only individual programs, but also applications. An application is a set of programs connected by a precedence graph, this graph defines execution order. To easily describe such a graph, users have a formal syntax at their disposal.

Operations are also provided for the dynamic addition of programs to the execution graph, to stop the application and to transparently access remote files. If the used files are located on a NFS partition, our functions for file access are not needful.

The GATOREBEGIN command is used to launch an application. The work of GATOS is to take all the programs of the application and, according to the precedence graph, execute them on the less loaded hosts. When a program uses input and output, a special server is created on the user’s host: the interactivity server. This server prints program outputs on the user’s console and sends input to the programs. This mechanism hides differences between local and remote executions.

The main part of the manager is a remote procedure call server. Each server represents GATOS on one host. All the hosts running a GATOS server may be involved in the load distribution, unless this load is above a threshold. This limit prevents program executions by a remote user when the host is already too heavily loaded.

II. Implementation of Distributed Applications

A typical application needing a big amount of processing power is the UNIX command MAKE. This command is essentially used to build programs and applications. It takes commands in a file called a makefile. A program construction is usually done in two phases. First, the compilation phase: compilations of objects. Secondly, link phase: assembly of all the objects. As in most present compilers, compilation of objects is independent of other compilations, the classic MAKE only executes commands for sources that have been changed (partial compilation). This property also allows parallel compilation of the objects. Traditional MAKE does not include this advantage. Our distributed MAKE has been implemented by command conversion from usual makefile into a parallel graph executed by GATOS. This is totally transparent to users. Instead of utilizing the MAKE command they use the GATOMAKE command.

III. Speed up

The code of the GATOS server has been compiled by using the distributed GATOMAKE. Results are shown on the graphic below. Local execution has a speed up of 15% due to the parallelism. With three servers the gain is about 40%.

Conclusion and Perspectives

The advantages of GATOS are the automatic program distribution according to the host load, the total transparency of execution localization, the independence of distributed application vis-à-vis net configuration, the facility in distributing existing applications and the operations provided to build easily parallel applications.

New specifications have been established. This project will extend the use of GATOS to a heterogeneous environment, transparent communication facilities and new allocation policies, including user defined allocation. Some tools will be provided to automatically generate statistics about execution, communications and files access of programs and will allow to visualize and optimize allocation policies. A distributed debugger will also be included in order to facilitate the programmers.