BIOINFORMATICS

Vol. 18 no. 2 2002
Pages 315–318

Models@Home: distributed computing in bioinformatics using a screensaver based approach

Elmar Krieger and Gert Vriend*

CMBI, Center for Molecular and Biomolecular Informatics, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands

Received on June 14, 2001; revised on August 28, 2001; accepted on September 17, 2001

ABSTRACT

Motivation: Due to the steadily growing computational demands in bioinformatics and related scientific disciplines, one is forced to make optimal use of the available resources. A straightforward solution is to build a network of idle computers and let each of them work on a small piece of a scientific challenge, as done by Seti@Home (http://setiathome.berkeley.edu), the world's largest distributed computing project.

Results: We developed a generally applicable distributed computing solution that uses a screensaver system similar to Seti@Home. The software exploits the coarse-grained nature of typical bioinformatics projects. Three major considerations for the design were: (1) often, many different programs are needed, while the time is lacking to parallelize them. Models@Home can run any program in parallel without modifications to the source code; (2) in contrast to the Seti project, bioinformatics applications are normally more sensitive to lost jobs. Models@Home therefore includes stringent control over job scheduling; (3) to allow use in heterogeneous environments, Linux and Windows based workstations can be combined with dedicated PCs to build a homogeneous cluster.

We present three practical applications of Models@Home, running the modeling programs WHAT IF and YASARA on 30 PCs: force field parameterization, molecular dynamics docking, and database maintenance.

Availability: Models@Home is freely available including source code and detailed instructions from http://www.cmbi.nl/models.

Contact: elmar.krieger@cmbi.kun.nl

INTRODUCTION

No matter how smart the algorithm, it is always too slow to do the job—overnight on a desktop PC. And when PCs have finally become fast enough—the algorithm has become obsolete, replaced by a new approach, with fewer approximations. That is a well known experience in bioinformatics, almost comparable to Murphy's law. The usual approach is to split the problem into little jobs that can be executed independently. The execution time is then reduced by $1/n$, with n being the number of computers working in parallel. As noted by Amdahl (1967), this ideal speed-up cannot be reached in practice, leading to the more realistic formulation

$$S_n = \frac{1}{(1 - \mathrm{PF}) + \mathrm{PF}/A_n} \qquad (1)$$

in which S_n it the total speed-up when going from one to n processors, PF is the 'Parallelizable Fraction' of the program (i.e. the fraction of the total execution time that can be reduced by working in parallel), and A_n is the speed-up of the algorithm's PF on n processors.

Algorithms in bioinformatics tend to be applied to a large number of different targets, e.g. all ORFs in a genome, all sequences in CASP, or all potential drug candidates in a library. If each computer is assigned one target, these jobs are completely independent and parallelize perfectly, as no communication overhead is required: PF is close to one, A_n and S_n are close to n. The coarse-grained nature of typical bioinformatics projects is probably one of the reasons why most follow-ups to Seti@Home fall into this area, e.g. FightAids@Home (http://www.fightaidsathome.com) or Folding@Home (http://www.stanford.edu/group/pandegroup/Cosm). An overview of other approaches to distributed computing, clusters and computational grids is given in Table 1.

METHODS

Models@Home has been designed for use in typical departmental situations, in which a large number of Linux or Windows workstations is idle for about 16 h a day. The program consists of several functional units, as shown in Figure 1. Detailed installation instructions are available from http://www.cmbi.nl/models.

*To whom correspondence should be addressed.

Table 1. List of different approaches to clusters and distributed computing. S = source code available, O = run your own programs ($ requires payment), W = available for Windows, U = available for Unix/Linux

Program name	www address	S	O	W	U	Description
Beowulf	www.beowulf.org	+	+	−	+	Software for dedicated Linux clusters
Berkeley NOW	now.cs.berkeley.edu	+	+	−	+	In-house network of workstations
Condor	www.cs.wisc.edu/condor	+	+	+	+	Distributed computing library
Cosm	www.mithral.com	+	+	+	+	Distributed computing library
Distributed.net	www.distributed.net	−	−	+	+	Cracking encryption keys
Entropia	www.entropia.com	−	$	+	−	Distributed computing for Windows
FightAids@Home	www.fightaidsathome.com	−	−	+	−	Drug design using Entropia
Folding@Home	www.stanford.edu/group/pandegroup/Cosm	−	−	+	−	Protein folding simulations using Cosm
Globus project	www.globus.org	+	+	−	+	Computational grid for Unix
Legion	legion.virginia.edu	−	+	−	+	Worldwide computer for Unix
Models@Home	www.cmbi.nl/models	+	+	+	+	Screensaver cluster for any program
Mosix	www.mosix.cs.huji.ac.il	+	+	−	+	Software for dedicated Linux clusters
Seti@Home	setiathome.berkeley.edu	−	−	+	+	Search for extraterrestrial intelligence
United devices	www.ud.com	−	$	+	−	Distributed computing for Windows

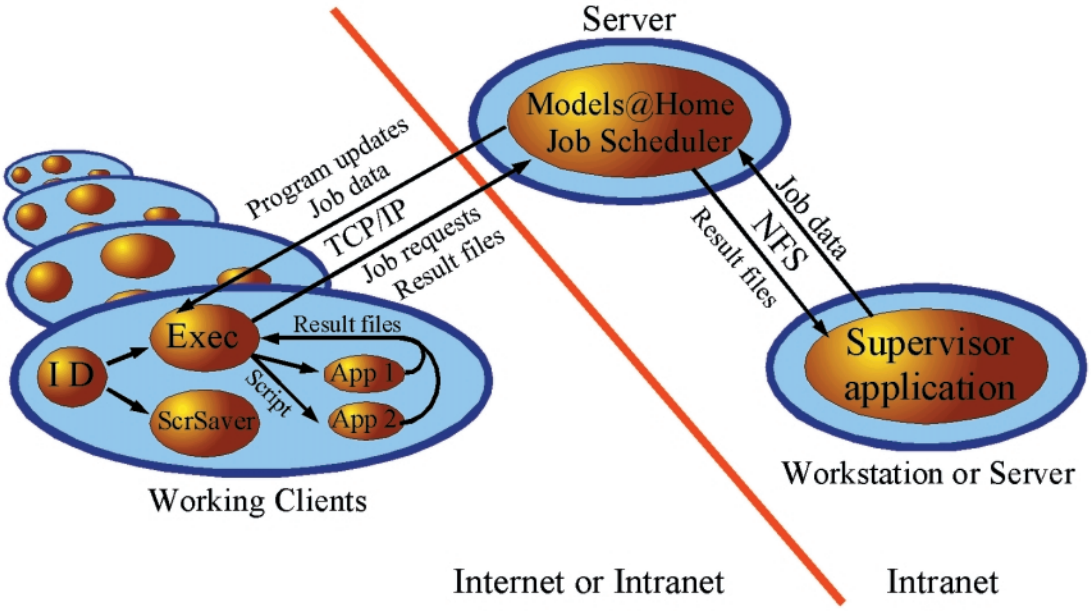

Fig. 1. Models@Home data flow. Refer to the methods section for details. Abbreviations: ID = Idle Detection module, Exec = Program execution module, ScrSaver = graphical screensaver module, App 1+2 = application 1+2, TCP/IP = Transmission Control Protocol/Internet Protocol, NFS = Network File System.

Supervisors

Users of the cluster run programs called 'supervisors'. These are typically applications that keep track of the work that has to be done, but do not do it themselves. Instead they cut it into pieces and submit the individual jobs to the Models@Home job scheduler (via an interface of C functions or a Python class). The supervisors thus form the part that has to be adapted specifically for a certain application. This is most easily done by adding a '*submit to Models@Home*'-command to the inner loop of an existing script. All other aspects of Models@Home are totally general.

Working clients

As the name suggests, they do the actual work. Initially, only the Idle Detection (ID) module is active on these computers. The ID module is highly operating system specific and has been derived from

existing open-source screensavers. The Linux version is based on Jamie Zawinski's XScreensaver (www.jwz.org/xscreensaver), with additional changes to Linux configuration files (/etc/X11/xdm/Xsetup_0, /etc/X11/xdm/Xsession), the Windows equivalent on Bill Buckel's work (www.escape.ca/~bbuckels). The modified sources are available from www.cmbi.nl/models.

As soon as the computer is idle (i.e. no mouse movements or key-strokes occur for 15 min), the ID module launches a graphical screensaver (ScrSaver) and the execution module (Exec). Both have been implemented in an operating-system independent way based on the SDL library, including SDL_net for the TCP/IP interface (www.libsdl.org). Users running batch jobs can configure the screensaver to become active only if they are not logged in.

The Exec module contacts the server. This contact message also contains the time stamps of files that should be kept up to date (these names are stored with other information, like the server's IP address, in the local configuration file 'cluster.cnf').

If there are jobs waiting in the queue, the Exec module receives a job description (the name of the application to run, command line parameters, scripts), file updates if needed, and the data files required to complete the job (e.g. specific PDB files). These are stored in the working directory /job. Exec runs the requested application (App 1 or App 2 in Figure 1) and waits until the job has finished. Result files are sent back to the server. Large applications consisting of hundreds of files should be permanently installed on the clients, small and compact programs can be transmitted as part of the job description.

If a user on a working client terminates the screensaver, Exec kills the running application and notifies the server that the job could not be completed. No attempt is made to put the application on hold and continue at a later time, as this would negatively affect the cluster's performance (it is not known if and when the client will be available again). Jobs taking a long time must therefore return checkpoint files in reasonable time intervals. Usually this does not require a modification of the source code, as most time-intensive programs already have these mechanisms built in (e.g. every molecular dynamics program can save a snapshot of the current simulation state).

Exec can also be run as a stand-alone module (without the screensaver) on the nodes of a dedicated cluster. It then allows a very efficient use of cluster resources: while many batch queueing systems generate the maximum overhead when the cluster is busy (by asking clients sequentially if they want to accept a job), Models@Home works the other way round: clients ask the server for a job, only reaching the maximum number of requests (and thus network load) when the cluster is not used at all.

Server

The server is the link between supervisors and working clients. It manages the job queue 'cluster.job' and distributes jobs according to their priority. It also handles the transfer of job data files from a job-specific directory on the supervisor (via NFS) to the working directory on the working client (via TCP/IP). Job results travel back in the reversed direction.

The latest file updates are stored in subdirectory /updates and transmitted to working clients to replace outdated versions. (In principle any file on the working client can belong to this update group, including the Models@Home software and the actual applications.)

The server also collects 'still alive messages' from working clients. If a client does not send such a message for a given time period, it is assumed to have 'disappeared without notice' (e.g. a power failure) and the job is retransmitted to another client.

IMPLEMENTATION

Models@Home is based on a straightforward client–server architecture using the TCP/IP protocol as shown in Figure 1. Every client has the Models@Home software and all applications that are too large for repeated transmission installed locally (which is most easily done by remote administration). The main component of Models@Home is a screensaver that becomes active when the client is not used for a certain time period. Beside displaying some graphical animations, the program contacts the central server and requests a new job. Jobs can be added via a Python or C interface. The screensaver detects when a program has finished, returns the specified result files and requests a new job. As all this is done by the Models@Home software, it is possible to run any program in parallel without modifications to the source code, as long as user input is not needed.

The above procedure applies to ideal conditions only. A number of additional features had to be implemented to cope with problems encountered in practice:

- Hard resets and power failures: computers occasionally stop working, making it impossible to notify the server about the job interruption. Clients therefore send a 'still alive message' in fixed time intervals. Especially short intervals require a permanent inter- or intranet connection and exclude any dial-up clients.

- Program development and bugs: if the program is under development, it is of course not reasonable to continuously reinstall it on every client. The Models@Home communication protocol therefore includes the time-stamps of selected files including executables. These are automatically updated if newer versions are available on the server.

- Security issues: in the majority of cases, Models@Home will be used in the intranet only (normally protected by a firewall). Either because this already provides enough computer power, or because the programs are not freely distributable, or because of the large amount of work required to support users outside the department. If the 'world wild web' is targeted, the update feature mentioned above must be deactivated, and it is up to the developer to ensure that the executed programs cannot do any damage (e.g. a simple SAVE command in a program script could already overwrite system files in a Windows environment).

DISCUSSION

The Models@Home environment has been installed and tested on 30 mixed Linux/Windows PCs at the CMBI (http://www.cmbi.nl). The following paragraphs describe some of the applications and concentrate on the aspects related to parallel execution, the very details will be described elsewhere.

Molecular dynamics docking

A protocol was developed where docking is performed with YASARA (http://www.yasara.com) during a molecular dynamics simulation (see Di Nola *et al.*, 1994 for an early description of a comparable method), which inherently considers both ligand and protein flexibility (Krieger *et al.*, 2001, submitted). The ligand is shot towards the protein, allowing side chain reorientations during complex formation, followed by a short MD simulation, an energy minimization (simulated annealing) and the evaluation of the final energy.

To sample conformational space reasonably well, thousands of molecular dynamics simulations with different initial orientations of ligand and protein are required. As these are completely independent, the supervisor could spawn all docking jobs at once, making molecular dynamics docking an ideally 'coarse-grained' application for distributed computing.

Force field parameterization

Selecting force field parameters that optimally fit a given force field equation is a lengthy procedure, usually requiring extensive validation studies. In addition it is very difficult to obtain an internally consistent parameter set. We therefore let a force field 'parameterize itself' while energy minimizing protein structures (Krieger *et al.*, 2001, submitted and http://www.yasara.com/nova). This was done with Monte Carlo moves in parameter space, that were accepted if the resulting force field did less damage to high resolution x-ray structures and at the same time improved models built by WHAT IF (Vriend, 1990).

Each parameter optimization cycle required an energy minimization of 50 protein structures. The supervisor therefore spawned 50 jobs. Each job included the PDB file of the structure, as well as a YASARA script to do the energy minimization and to finally calculate the RMSD from the initial structure. Only this RMSD value was sent back as a result. The 50 RMSDs were averaged and used as a progress indicator.

For this application of Models@Home, the rate limiting step was the time it took to minimize the largest protein in the set. All RMSDs had to be known before the quality of the current force field could be estimated and new jobs could be spawned. More computers would thus not have improved the performance. This is an example of a 'medium-grained' application.

Database maintenance

The CMBI hosts a large number of databases, many of which are not mirrored but generated in-house. Especially those related to protein structure are often time consuming to maintain, like the PDBREPORT database, that lists anomalies in protein structures (Hooft *et al.*, 1996 and www.cmbi.nl/gv/pdbreport). It contains one entry for every PDB file, and is mainly used by modelers to find optimally suited templates.

To keep this database up to date, the supervisor compares it with the PDB once a week, deletes obsolete PDB reports and spawns jobs to create new ones. All these jobs are independent and parallelize perfectly.

CONCLUSION

Models@Home provides a flexible environment for parallel execution of different applications without the need to modify any of these programs. It is therefore well suited for bioinformatics, where both the turnover of different software packages and the requirement for computer power are huge.

Models@Home can be reconfigured for use with different programs without making changes to the source code. It is freely available and can be downloaded from www.cmbi.nl/models.

ACKNOWLEDGEMENTS

We would like to thank all researchers at the CMBI for participating in the Models@Home screensaver project.

REFERENCES

Amdahl,G. (1967) The validity of the single processor approach to achieving large scale computing capabilities. *AFIPS Conf. Proc.*, **30**, 483–485.

Di Nola,A., Roccatano,D. and Berendsen,H.J. (1994) Molecular dynamics simulation of the docking of substrates to proteins. *Proteins Struct. Funct. Genet.*, **19**, 174–182.

Hooft,R.W.W., Vriend,G., Sander,C. and Abola,E.E. (1996) Errors in protein structures. *Nature*, **381**, 272.

Vriend,G. (1990) WHAT IF: a molecular modeling and drug design program. *J. Mol. Graph.*, **8**, 52–56.