BIOINFORMATION
Discovery at the interface of physical and biological sciences

open access www.bioinformation.net Software

Volume 10(1)

GRIMD: distributed computing for chemists and biologists

Stefano Piotto[1]*, Luigi Di Biasi[2], Simona Concilio[3], Aniello Castiglione[2] & Giuseppe Cattaneo[2]

[1]Department of Pharmacy, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Salerno – Italy; [2]Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Salerno – Italy; [3]Department of Industrial Engineering, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Salerno – Italy; Stefano Piotto – Email: s.piotto@gmail.com; *Corresponding author

Received October 04, 2013; Revised November 29 2013; Accepted January 06, 2014; Published January 29, 2014

Abstract:
Motivation: Biologists and chemists are facing problems of high computational complexity that require the use of several computers organized in clusters or in specialized grids. Examples of such problems can be found in molecular dynamics (MD), *in silico* screening, and genome analysis. Grid Computing and Cloud Computing are becoming prevalent mainly because of their competitive performance/cost ratio. Regrettably, the diffusion of Grid Computing is strongly limited because two main limitations: it is confined to scientists with strong Computer Science background and the analyses of the large amount of data produced can be cumbersome it. We have developed a package named GRIMD to provide an easy and flexible implementation of distributed computing for the Bioinformatics community. GRIMD is very easy to install and maintain, and it does not require any specific Computer Science skill. Moreover, permits preliminary analysis on the distributed machines to reduce the amount of data to transfer. GRIMD is very flexible because it shields the typical computational biologist from the need to write specific code for tasks such as molecular dynamics or docking calculations. Furthermore, it permits an efficient use of GPU cards whenever is possible. GRIMD calculations scale almost linearly and, therefore, permits to exploit efficiently each machine in the network. Here, we provide few examples of grid computing in computational biology (MD and docking) and bioinformatics (proteome analysis).

Availability: GRIMD is available for free for noncommercial research at www.yadamp.unisa.it/grimd

Supplementary information: www.yadamp.unisa.it/grimd/howto.aspx.

Background:
The main ideas behind Distributed Computing are not new and several successful distributed systems have already been developed to harness idle computer time. Nowadays the most successful computational Grid model is the one named "Volunteer Computing"; this is a type of Distributed Computing where each computational device owner donates computing resources (usually processing power and storage) to a centralized "project". The distributed cooperation paradigm used in those cases is known as map-reduce, and is based on the replication of a common task on each computing node, properly designed to execute the desired algorithm on its chunks and followed by a phase in which all results are collected and processed accordingly. The first such application was SETI@home **[1]** to analyze the data from radio telescopes looking for signs of extraterrestrial life. Taking its inspiration from SETI, a protein folding project called Folding@Home has been in operation at Stanford University for several years. This is a pilot of the Berkeley Open Infrastructure for Network Computing (BOINC), a software platform for distributed computing using volunteer computer resources.

In the past many papers predicted that grid would have had a great impact both to industrial and academic users, but after

years we must admit that the use of grids remained limited. One factor that is often overlooked when it comes to evaluating grids is the set of technical skills required to setup and administrate the system. Often a grid of computers consists of a specialized hardware where a number of processors are connected via a dedicated internal bus to some shared piece of memory. These grids require quite advanced technical skills to setup, install, and use. GRIMD leverages a preprocessor that separates the work of the life scientist from network administration. The GRIMD user must simply modify an existing script to initiate a calculation of molecular dynamics, molecular docking, ab initio quantum mechanical calculation or proteome analysis. Currently, a user through the GRIMD website can run NAMD (http: //www.ks.uiuc.edu /Research/namd), Autodock (http: //autodock.scripps.edu), Vina (http: //vina.scripps.edu), YASARA (http: //www.yasara.org), Abalone (http://www.biomolecular-modeling.com/Abalone), or Mopac (http://openmopac.net) calculations. The implementation of other programs is straightforward and described in the online manual. The work of the grid is totally transparent to the user.

Main features:

GRIMD can divide a complex calculation in a number of smaller jobs. The jobs are sent to available PCs (slaves) and, after completion, the most relevant results are collected and made available via web interface.

Architecture

The architecture of GRIMD is a typical MASTER/SLAVE, based on the SPMD and MapReduce **[2]** paradigms. MapReduce is a parallel programming model that facilitates the processing of large distributed datasets **[2].** It was originally proposed by Google to index and annotate data on the Internet. In this paradigm, the programmer specifies two functions: map and reduce. The map function takes as input a key/value pair, then it performs the map function, and then outputs a list of intermediate key/value pairs that may be different from the input. The runtime system automatically groups all the values associated with the same key, and then it forms the input to the reduce function. The reduce function takes as input a key/value pair, then performs the reduce function, and finally outputs a list of values. Many real world tasks of life scientists are fit to this computation model; moreover, MapReduce is appealing to scientific problems because of its ease of programming, its automatic load balancing and failure recovery, and overall ease of scaling.

The Master, which is a dedicated machine, runs a program that takes care of the input data partitioning, the scheduling, the tasks execution across a set of machines (dynamically updated), the handling of machine failures, and the managing of the required inter-machine communication. The Master implements a basic authentication mechanism when a new slave subscribes to the "GRIMD Network", managing communication privacy through channel encryption (a sort of VPN) and client-side strong authentication through session key negotiation. This was necessary to prevent a malicious user from subscribing to the net to get access to sensitive data, or, even worse, to influence the final results with wrong data that could never be discovered. More sophisticated authentication mechanisms (such as Kerberos ticket granting) could be introduced in the next releases.

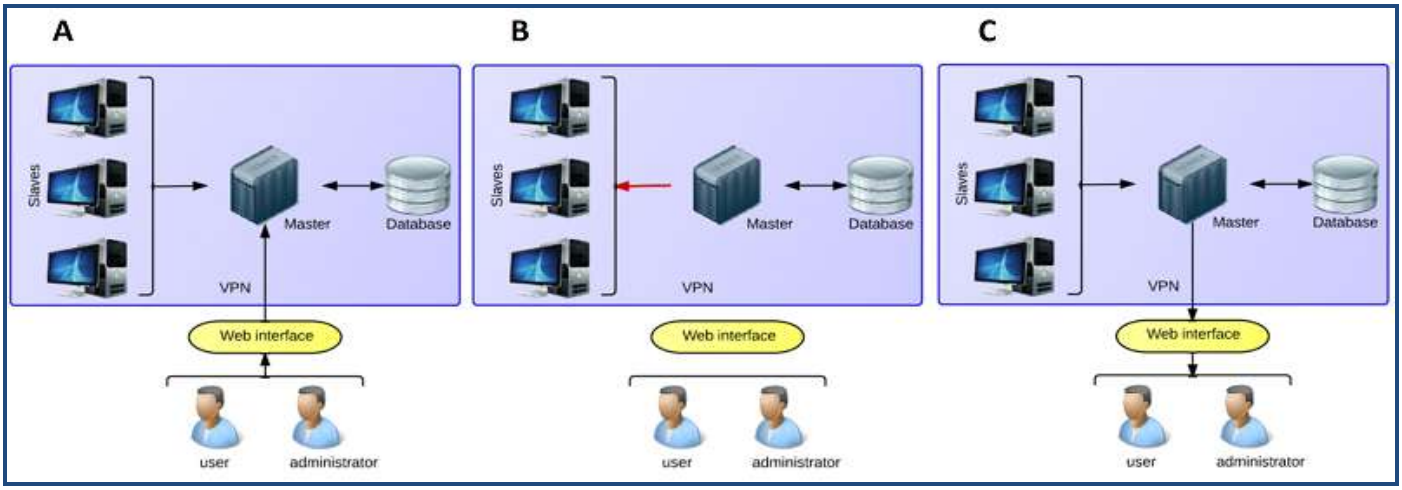

Figure 1: (a) The Slave is a Windows service running in background. The Slave connects to the Master VPN and sends its status information to the Master. The GRIMD user interacts with the grid through a web interface; **(b)** The Master deploys the calculation to all available Slaves. If necessary, the Slave downloads missing packets and updates; **(c)** Once the job is completed, the Slave uploads the results on the Master. The Master reduces the results from all Slaves and sends the Result.out file to the user.

The architecture of GRIMD is summarized in **(Figure 1).** The Slave is a Windows service running in background with SYSTEM rights. The service is set up to work in Idle Time with low priority. Once the Slave is on, it attempts to connect to the "Grimd Network", which is a VPN.

Once the slave is connected it sends to the Master the information about its state, the IP address, the slave version, and the modules that are already present. Once the Master received these data, it can choose among different scenarios: it can switch off the slave, when there is a version mismatch; in this case the Slave must be reinstalled. In case the Slave lacks a module that is required for a specific calculation, the Master can order the Slave to download the missed module. Finally, if the Slave is ready for the calculation, the Master sends one job instance in the queue.

The data are transferred as ZIP file through HTTP from Master to Slave. The Slave unzips the file to get all that is needed to perform the calculation (macro, structural files and the file execjob.grimd). During the job execution, the Slave notifies the Master about the state of the calculation. After job completion, the Slave uploads the results and updates the Master; this is important to certify that the Slave is up, otherwise the job is assigned to another Slave. All the traffic is on VPN tunnel connections using standard TCP/IP protocols to guarantee high security.

Explode Points

A peculiar aspect of GRIMD is the introduction of explode points (EPs). The EPs are used to automate mapping, allowing a significant reduction of user workload. An EP indicates to the preprocessor how to generate the distributed jobs.

Following the Map/Reduce paradigm, the mapping is achieved by the GRIMD preprocessor (prGRIMD) that parses the Main Macro (MM) checking for EPs. Once an EP is found, prGRIMD treats the different types of EPs to generate a number of daughter macros. Two EP types are supported: WRITE_RANGE [startingpoint,endpoint,step], and WRITE_ARRAY [first, second, third, ..., last].

When the GRIMD preprocessor finds WRITE_RANGE [startingpoint,endpoint,step], it creates (endpoint-startingpoint)/step different macros and assigns each value to the variable EXPLOSION.

A typical use is:
##EXPLOSION(1) = WRITE_RANGE[1,5,1] @ENDEXP
In this case the preprocessor assigns to the variable EXPLOSION(1) the values 1, 2, 3, 4 and 5. These values replace the string @EXPLOSION(1)@ anywhere in the Main Macro
A typical use of WRITE_ARRAY[first, second, third, ..., last] is:
##EXPLOSION(2) = WRITE_ARRAY[1aki, 2crn, 2aox] @ENDEXP
In this case the preprocessor assigns to the variable EXPLOSION(2) the values "1aki", "2crn", "2aox". These values replace the string @EXPLOSION(2)@ anywhere in the Main Macro. The jobs are distributed to the slaves according to the chosen scheduling policy.

Granularity

To find the best granularity in splitting the calculation, after the split step, a randomly chosen task is run directly in the master to get information about its complexity (the single task run time length). For instance, if the runtime is less than a fixed threshold, warning messages are sent to the user. The user can then choose a different map function in order to reduce the grid overhead.

Security issues

Security of donor machines becomes a major issue in a programmable distributed system, as these systems have the ability to download and execute arbitrary code. This was one area that several systems **[3, 4]** did not consider a priority.

In order to deal with the threats coming from volunteer computing that let everyone to join the computational project, it is important to ensure: (i) confidentiality, so that even if the network traffic is sniffed at the packet level an attacker would only see meaningless data; (ii) sender authentication to prevent unauthorized users from accessing the VPN; (iii) message integrity to detect any instance of tampering with transmitted messages.

Such technology is the VPN, which as a side-effect also creates a reliable routing path among all the clients taking parts to the project without considering from where a client is connecting. VPN also protects from the sabotage issue that undermines the volunteer computing paradigm. In fact, each client taking part in the computational task by using VPN technology is automatically identified, and it can also be banned if the administrator detects that sabotage is underway.

In a variegated environment, such as a university, there are PCs mainly dedicated to teaching, and PCs dedicated to research. The formers are exposed to several misuses like unplugged cables or forced shutdown. The latter are often secured with UPS devices and are accessible to a very limited number of users. In such environment, the loss of data due to hardware crashes and user misuses is a serious risk. In GRIMD, the Slaves backup the ongoing calculations on external drive. In the standard configuration GRIMD employs Google Drive for protecting the data.

Applications:

GRIMD was born to provide an easy and scalable method to set up calculations that cannot be executed on a single, albeit powerful, computer. GRIMD effortlessly distributes scripts working with Yasara, Namd, Autodock, Vina, Abalone and Mopac with the addition of few lines.

GRIMD for bioinformatics analysis

Antimicrobial peptides (AMPs) are a class of promising antibiotic compounds that attracted attention in virtue of their broad spectrum of use, and of the extremely reduced phenomenon of bacterial resistance **[5].** In spite of extensive research, still a valid method of their mechanism of action is missing. In the last years we searched for specific sequence pattern among all known AMPs extracted from the YADAMP database **[5],** and in proteomes. A new metric to define distances among peptides and proteomes was suggested and made available online (www.yadamp.unisa.it/protcomp). We defined the dictionary (D) of a proteome Si as the collection of all aminoacid substrings of length l with the relative number of occurrence **(see supplementary material for equation).**

max_l is the limit of sequences length. We have created the dictionaries for over than 200 organisms, mainly bacteria, but also eukarya and archea. We have then calculated the distances among organisms as in the equation: **(see supplementary material for equation).**

Whereas D_{ai} and D_{bi} represent the dictionaries of the two species *a* and *b* for the string of length *i*. The full description of the algorithm and the method will appear in a different work. The creation of the dictionaries is computationally expensive. We used GRIMD to divide the proteome in partially overlapping *k* blocks. Each slave performed the analysis of the block and all the partial results are combined in a single dictionary.

Presently, the site www.yadamp.unisa.it/protcomp contains more than 500 dictionaries describing the sequence pattern of proteomes, genomes and transmembrane proteomes (data unpublished).

GRIMD for molecular dynamics (Yasara and NAMD)

It is well known that the time required for a drug or a short peptide to cross a lipid membrane can be several microseconds. This time is much longer than the 10-100 ns of typical molecular simulations. As a consequence, the energy evaluation of trajectories lasting few tens of nanoseconds cannot guarantee that the observed system will reach equilibrium. GRIMD was used to perform properties calculations of a large database of peptides **[5]** and to calculate the free binding energy of a drug across a lipid membrane **[6]** generating, for each system drug/membrane, 160 initial conformations different in orientation and penetration of the drug inside the membrane. In order to avoid bumps and to avoid abnormal non-covalent interactions, we downsized the drugs to 20% of the original size and reduced the non-covalent interactions to 10% of the normal value. The drugs where then pulled through membrane till the desired initial positions were reached. The size and the energy constants were gradually brought back to normal values through cycles of steepest descent minimization, and a cycle of annealing until the speed of the fastest atom dropped below 500 m/s. The membranes were then heated to 310 K and an equilibration dynamics of 1 ns completed. After completion, free energies were calculated on the last 100 ps of the 160 trajectories and the conformation with higher binding energy was chosen as initial structure for a longer and more accurate MD run of 50 ns. More information and a working example (macro + input files) to perform a molecular dynamics study are available on the site www.yadamp.unisa.it/grimd.

GRIMD for molecular docking (AutoDock and VINA)

Conformational changes of biological macromolecules when binding with ligands remain a challenge for automated docking methods. GRIMD was used to perform a flexible ligand-flexible receptor docking encoding the conformational spaces of molecules through a protocol of molecular dynamics, followed by the generation of an ensemble of rotamers. These conformational subspaces can be built to span a range of conformations important for the biological activity of a protein. A variety of motions can be combined, ranging from domains moving as rigid bodies or backbone atoms undergoing normal mode-based deformations, to side chains assuming rotameric conformations. In addition, GRIMD can be easily used for the screening of several receptors against a large library of ligands. Recently we applied GRIMD for a massive docking analysis of potential ATP synthase ligands **[7, 8],** and in the screening of potential inhibitors of COX1 and COX2 **[9].** More information and a working example (macro + input files) to perform a flexible-flexible molecular docking on a library of ligands are available on the site www.yadamp.unisa.it/grimd.

Performance evaluation:

In order to evaluate the behavior of the grid with a growing number of PCs, we performed 3 types of calculations (M1, M2, M3). M1 is a very simple macro that is available on the site with the name "Hello World". M1 is performed on a single machine in about 30s. M2 and M3 are more complex macros which require 200 and 450 hours to be executed on a single machine. All the times are listed in Table 1 along with the bootstrap time and the comparison between efficiency of the grid and the theoretical speed up. The macros were run on a number of PCs between 1 and 50.

GRIMD shows linear scaling for macros that require more than one hour. For shorter macros, the dispatcher overhead makes inconvenient the distribution on the grid. Figure 2 represents the behavior of the grid.

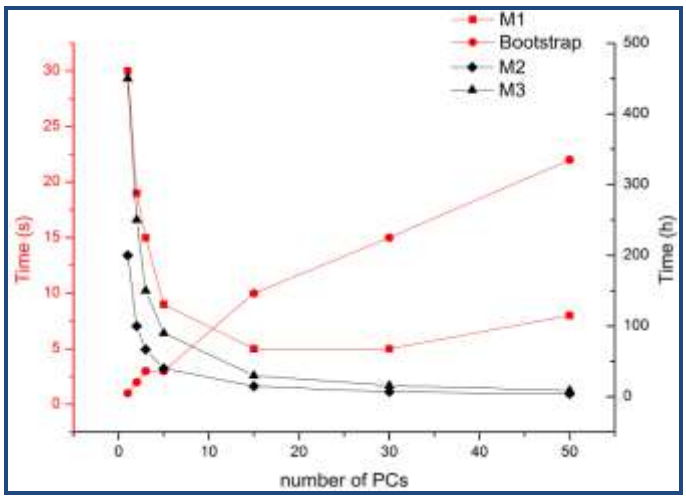

Figure 2: Graphical representation of the time employed by a different number of PCs to perform 3 macros: M1, M2 and M3. M2 (diamonds) and M3 (triangles) are macros which require several hours for completion. M1 (squares) is a macro that requires few seconds. The bootstrap time (circles) is the time in seconds that is required to assign a job to all slaves.

References:

[1] Anderson DP *et al.* *Communications Of The Acm*. 2002 **45**: 56
[2] Dean J & Ghemawat S, *Commun Acm.* 2008 **51:** 107
[3] Cappello P & Mourloukos D, *Sci Programming – Neth.* 2002 **10**: 159
[4] Krieger E & Vriend G, *Bioinformatics* 2002 **18**: 315 [PMID: 11847079]
[5] Piotto SP *et al. Int J Antimicrob Agents.* 2012 **39**: 346 [PMID: 22325123]
[6] Crul T *et al. Curr Pharm Des*. 2013 **19**: 309 [PMID: 22920902]
[7] Piotto S *et al. Biochimica Et Biophysica Acta* **In Press**.
[8] Piotto S *et al. Eur J Med Chem.* 2013 **68**: 178 [PMID: 23974017]
[9] Lopez DH *et al. Plos One* 2013 **8**: E72052 [PMID: 24015204]

Edited by P Kangueane
Citation: **Piotto *et al.*** Bioinformation 10(1): 043-047 (2014)

BIOINFORMATION open access

Supplementary material:

Applications:

GRIMD for bioinformatics analysis

A new metric to define distances among peptides and proteomes was suggested and made available online (www.yadamp.unisa.it/protcomp). We defined the dictionary (D) of a proteome S_i as the collection of all aminoacid substrings of length l with the relative number of occurrence.

$$D_{n,l} \supset S_i | l(S_i) \leq max\_l$$

max_l is the limit of sequences length. We have created the dictionaries for over than 200 organisms, mainly bacteria, but also eukarya and archea. We have then calculated the distances among organisms as in the equation:

$$distance = \frac{\sum_{i=1}^{max\_l} |D_a \cap D_b|}{(|D_a| + |D_b|) \cdot max\_l + \Delta(D_a \cap D_b)}$$

Whereas D_{ai} and D_{bi} represent the dictionaries of the two species a and b for the string of length i. The full description of the algorithm and the method will appear in a different work.

Table 1: Benchmark results of GRIMD

N	M1 (s)	M2 (h)	M3 (h)	Bootstrap (s)	Speed up M2	Speed up M3	Theor speed up
1	30	200	450	1	1	1	1
2	19	100	250	2	2	1.8	2
3	15	67	150	3	2.99	3	3
5	9	40	90	3	5	5	5
15	5	15	30	10	13.33	14.99	15
30	5	7	16	15	28.57	28.13	30
50	8	4	9	22	49.98	49.99	50

N is the number of PCs; M1, M2 and M3 are 3 distributed jobs. Bootstrap is the time in seconds that is required to assign a job to all slaves.