Distributed computing as a virtual supercomputer: tools to run and manage large-scale BOINC simulations

Toni Giorgino[a], M. J. Harvey[b], Gianni de Fabritiis[a]

[a] *Computational Biochemistry and Biophysics Lab (GRIB-IMIM),*
Universitat Pompeu Fabra, Barcelona Biomedical Research Park (PRBB),
C/ Doctor Aiguader 88, 08003 Barcelona, Spain
[b] *High Performance Computing Service, Imperial College London,*
South Kensington, London, SW7 2AZ, UK

Abstract

Distributed computing (DC) projects tackle large computational problems by exploiting the donated processing power of thousands of volunteered computers, connected through the Internet. To efficiently employ the computational resources of one of world's largest DC efforts, GPUGRID, the projects' scientists require tools that handle hundreds of thousands of tasks which run asynchronously and generate gigabytes of data every day. We describe RBoinc, an interface that allows computational scientists to embed the DC methodology into the daily work-flow of high-throughput experiments. By extending the Berkeley Open Infrastructure for Network Computing (BOINC), the leading open-source middleware for current DC projects, with mechanisms to submit and manage large-scale distributed computations from individual workstations, RBoinc turns distributed grids into cost-effective virtual resources that can be employed by researchers in work-flows similar to conventional supercomputers. The GPUGRID project is currently using RBoinc for all of its *in-silico* experiments based on molecular dynamics methods, including the determination of binding free energies and free energy profiles in all-atom models of biomolecules.

Keywords: Distributed computing, BOINC, Grid computing, Molecular

Email addresses: `toni.giorgino@gmail.com` (Toni Giorgino), `m.j.harvey@imperial.ac.uk` (M. J. Harvey), `gianni.defabritiis@upf.edu` (Gianni de Fabritiis)

Dynamics, High Performance Computing

PACS 87.15.kp, 87.10.Tf, 87.15.ap

1. Introduction

Distributed computing (DC) is a recent computational model which allows scientists to gather the processing power donated by volunteers, dispersed worldwide, and connected through the Internet [1]. Hundreds of thousands of volunteers worldwide donate computer time and maintenance efforts to the solution of important scientific challenges [2, 3]. Researchers prepare computational experiments and arrange them in several "work units". When volunteers learn of project they wish to join, they install and run "client" software on their PC. The rest proceeds transparently for them: the client downloads the work, processes it, and uploads the results back to the server. On the other side of the grid, project scientists provide computation tasks in the form of work-units which are uploaded for computation, farmed out to volunteer computers and the results returned to them.

The DC model has become even more appealing with the advent of accelerated processors, like graphical processing units (GPUs). Once devoted to video display only, GPUs have now evolved into general-purpose computing processors that can be used to accelerate a variety of tasks [4]. GPUs are currently, in terms of raw computing power, an order of magnitude faster than standard processors. A moderate-sized DC project using accelerated processors can gather, in a cost-effective way, a raw computing power comparable to that of a current supercomputer [5], although with much less flexibility due to the slow interconnection links. Like a supercomputer, it needs a constant supply of tasks to be used efficiently. Server software such as the Berkeley Open Infrastructure for Network Computing (BOINC) [6, 7], provides all of the infrastructure required to manage the volunteer computers and the distribution of work and effectively addresses the problems of fault-tolerance, error-recovery and job scheduling associated with operating a grid of non-dedicated, unreliable compute resources.

Although numerous DC frameworks exist [8, 9, 10], BOINC has risen to preeminence in the field of volunteer computing. Thanks to the support for numerous computer platforms and its simple deployment, it currently counts on a base of approximately 350,000 *active* users and 600,000 active hosts, which provide computing power to dozens of scientific projects [11].

Despite the number of projects using BOINC, the tools it provides for the management of work-units and the corresponding results are rudimentary and, in each case, projects have either had to develop custom tools to integrate the grid with existing work-flows, or the project researchers have had to to deal directly with the complexities of job generation, operating directly on the BOINC server. In order to be generally useful, it would be desirable for a BOINC grid to instead resemble a "virtual supercomputer" that can be integrated in the design-execute-analyse cycle of computational experiments, in the same way that high performance computers (HPC) facilities currently are. HPC computational resources, in fact, are typically accessed through resource management and queueing systems, like the Sun Grid Engine [12], PBS Pro [13], SLURM [14], and Condor [15]. Resource-management systems typically offer simple command-line tools to submit and monitor tasks.

This paper describes a newly-developed interface for the submission and management of distributed computing simulations which extends BOINC in a generic manner. The proposed system constitutes a "researcher-side" component, called RBoinc, which allows scientists to start simulations from their individual workstations, to manage the corresponding runs (either individually or in batches), and to retrieve the results once computed. RBoinc provides BOINC-based computational science projects with supervised submission, file staging, and accounting facilities, all well-established concepts in grid computing.

RBoinc is currently in production use for *all* of the DC computations performed within the GPUGRID project [16], a large DC network that exploits the power of accelerated processors to perform all-atom simulations of biomolecular processes, which we shall discuss as an exemplar high-throughput computational application. GPUGRID was the first project to rely solely on accelerated processors for its computing power, initially supporting Playstation's Cell [17], and currently leveraging GPUs [18]; it is currently the eighth DC project worldwide in terms of floating points operations computed per day [11]. Although developed within the GPUGRID project, the RBoinc system is fully generic, and can be used to operate arbitrary DC projects based on the BOINC middleware.

2. System architecture

The RBoinc interface is structured according to the client-server model. The scientists use the *remote client*: it is a component that receives the requests for starting, stopping and monitoring jobs; these actions are relayed to the *server* component. In addition to dispatching the various actions, the client stages the files in and out of simulations: *input* files are uploaded from the researcher's workstation to the server, while *result* files are retrieved on request.

A sequence diagram of the steps throughout a DC computation managed via RBoinc is shown in Figure 1. First, a scientist designs an experiment and creates the corresponding input files on their machine. The files, as we shall see later, describe all the experimental setup, including parameters, initial conditions, protocols, and so on. He will then invoke the *remote submit* operation, during which the necessary input files will be transferred to the server through the Internet (1). The server will perform some consistency checks on the prospective computations and, if passed, the work will be passed to the BOINC distributed system for its actual execution (2). The volunteers' PCs will receive the packages of work through their Internet connection and will start to compute them as soon as their resource-sharing policies allow it (3). When each PC completes its share of work, the results will be uploaded back to the server (4), where they will be stored temporarily. In case of long computations, more steps will be generated and submitted in sequence. At any time, the researcher may retrieve the results computed so-far invoking a *retrieve* operation on their PCs (5).

2.1. Serial and parallel computation structures

The principal attraction of using a DC grid is that it grants access to a large amount of aggregate computing power, albeit spread over many independent

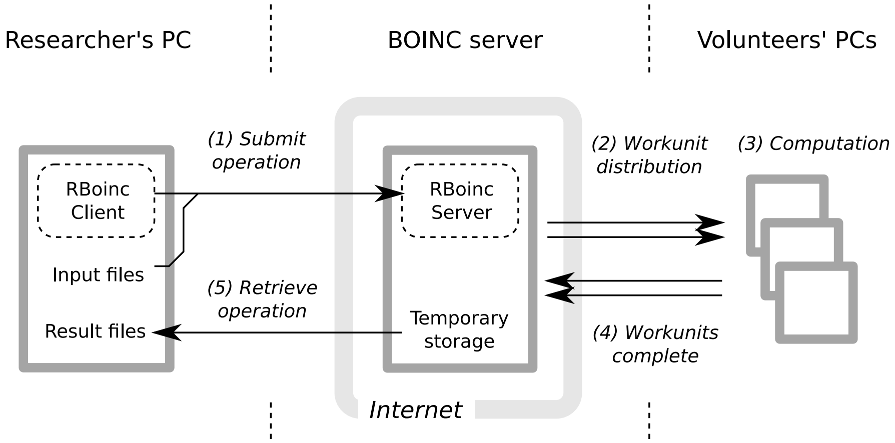


Figure 1: Sequence of the actions in the RBoinc architecture. (1) The researcher invokes the "submit" operation, which transfers the files to the server. (2) The server optimises the uploaded files (Section 3.1), prepares the work-unit, and distributes it to the BOINC clients. (3) The clients execute the assigned work, and return intermediate results (4), which are collected on the server. At any time, the scientist can recover the results computed so far by invoking the "retrieve" operation, which transfers them to his machine (5).

computers. To use it effectively, it is necessary to split the computation into as many pieces as possible in order to exploit the grid's parallel nature. The extent to which this can be achieve depends on the degree of parallelism intrinsic to the computation in question. A perfectly parallel problem could be decomposed into a large number of pieces that can be the computed independently of each other. In this case, the distribution of a large computation to a distributed computing grid is relatively straightforward, and each host can return the computed result for analysis as soon as it is ready. The central server would keep track of the failing hosts: a host could be turned off, disconnected from the network, or stop sharing resources at any time, and never return its assigned share of work [19].

However, complete parallelism is seldom achievable. It is quite common to have computations that take too long for a single client PC, even if equipped with GPUs, to finish in a practical length of time. This is commonly the case, for example, with molecular dynamics (MD) simulations, because a sufficient amount of simulated time must be sampled in order to correctly observe certain biological phenomena which occur on the scale of hundreds to thousands of

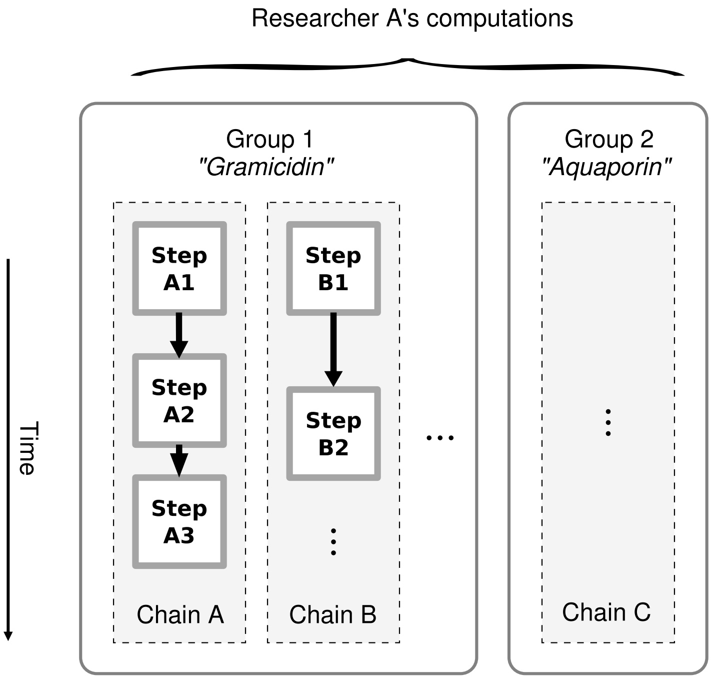


Figure 2: Each researcher may run multiple computations at the same time on the grid organising them as *groups*. Group are compound by one or more independent chains of computation. Users can manage the computations for which they have permissions, handling either whole groups or individual chains.

nanoseconds, whereas GPU-accelerated PCs can compute approximately ten nanoseconds per day of simulated time [18, 20].

To perform longer runs than can be accommodated by a work-unit of acceptable length, RBoinc provides facilities to concatenate several *steps* to be executed sequentially, thus extending the run length up to hundreds of times.

2.2. Computation groups and chains

Researchers can manage their computations with various degrees of granularity. Upon submission, users associate newly-submitted computations with a *group name*. Each group may contain one or more *chains* which, in turn, are composed of one or more *steps* of computation (Figure 2). Chains are executed in parallel whilst the various steps in a chain are executed sequentially which each step having a dependency on its antecedent.

A computation chain is defined supplying the *initial state*, to be used as the input of the first step, and the rest of the simulation *parameters*, which are re-used unchanged throughout all of the steps of the chain (Figure 3). Chaining is implemented by connecting the input and output files of consecutive steps: after each step, the outputs marked as “chained” will be used as input files for the next step. For example, in an atomistic simulation these files would be a “snapshot” of the current state of the system from which the simulation can be restarted exactly. Output files meant only for offline analysis, like log files, do not need to be chained. Computation chains are declared in “templates” that describe the inputs and outputs of a given application, as it will be shown in section 2.5.

At any time, the *retrieve* operation retrieves all of the files (both *Out* and *Result*) produced so far, allowing the researcher to inspect the progress of the simulation without waiting for the full chain to be completed. This is especially important for speeding up the setup-execute-analyse cycle: if a mistake is detected, the simulation can be interrupted immediately and resubmitted after amendment. The owner of a simulation can abort chains individually or whole groups.

2.3. Submission of computations

Researchers interact with RBoinc through command-line tools designed to be similar to those of batch systems found in HPC systems. The main tool to start a simulation is the *submit* command. After preparing the necessary input files on his workstation, the researcher invokes the submit operation with a command line similar to that shown in Figure 4 (top). Command line options for the submit operations specify the name of the group and chain, which can be used for later retrieval and management, and list all of the mandatory input files. If the submission is successful, the researcher receives a confirmation, and the corresponding work-units are queued on the DC grid.

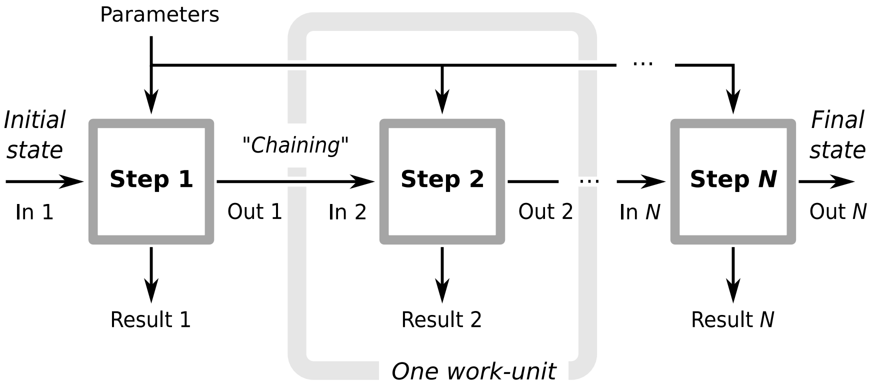


Figure 3: A computation chain is compound by N simulation steps. Each step starts from a given initial state and eventually produces an output configuration and other results. The output state is used as a starting point for the next step. In this way, long simulations, like MD trajectories, are split into work-units individually manageable by the hosts participating to the DC grid.

RBoinc's submission tool adapts to the type of application that is being started; the various applications, pre-defined by the system administrator, differ on the set of input files and parameters, and this is reflected by the command line options enabled for each. The `-help_parameters` option can be used to verify which input files are required by a given application. The command line options correspond to the input files defined in the template and are listed along with a text description of each file (Figure 4, bottom). In addition to the files required by the application, the user can attach an arbitrary *metadata* file to hold a description of the computation for future reference.

2.4. Monitoring

When work is submitted to a DC grid, it will impact a large number of users dispersed over the world. Each of the machines used will have different characteristics in terms of memory, compute resources, and availability. It is extremely important that each work-unit can run without disrupting the usability

```
shell:> rboinc_submit -app meta -num_steps 10
                      -group GRAMICIDIN -name ChainA
                      -pdb_file 1NEY.pdb
                      -psf_file GA.prmtop
                      -conf_file steered_md.conf ...
```

```
shell:> rboinc_submit -app meta
                      -help_parameters

Remote application: ‘meta’
Description: Standard ACEMD run with
             optional DCD and PLUMED

Options defined for this application:
    -conf_file        ACEMD input file
    -coor_file        Binary coordinates
    -metainp_file     (optional) PLUMED
                      metadynamics configuration
    -par_file         (optional) CHARMM parameters
    -pdb_file         PDB structure
    -psf_file         AMBER/PSF topology
    -vel_file         Binary velocities
```

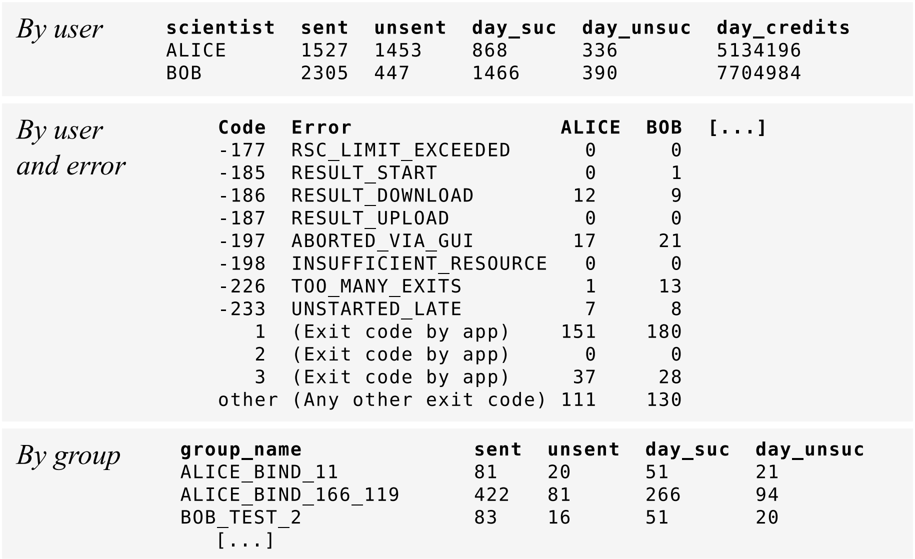
Figure 4: The RBoinc command line used for starting a distributed molecular dynamics simulation (top). The list of work-unit input files to be supplied on the command line depends on the application being invoked; the user can review it with the `-help_parameters` option (bottom).

of the volunteer's PC it is assigned to. The BOINC system takes care of matching hosts to work-units and limiting their impact on the normal activity of the machine (e.g., fixing a maximum amount of disk space that can be occupied). If a work-unit overruns these limits, it is interrupted before it can hamper the usability of the client computer.

Hitting resource limits is not the only potential cause for the interruption of a work-unit. The calculation may also fail because of computation errors, network connection problems, or if the volunteer explicitly decides to abort it. Computation errors should be closely monitored because they usually mean that the researcher has supplied parameters that are invalid. It is therefore important that the researchers have a way to receive feedback about the amount of simulations that they generate, and their outcomes.

The RBoinc interface provides two facilities to monitor the progress and occupancy of the DC grid, namely for the researchers and for the maintainers of the system. Firstly, users may invoke a "grid status" operation at any time, which produces a report on how much computation time was consumed by each simulation group (measured as total number of assigned credits), the number of work-units being processed and waiting to be dispatched, and detailed counts of the failed and successful work-units.

A second, more detailed status report is computed every night for the whole system, and e-mailed to the system administrators (Table 1). In this way, the presence of ill-designed work-units or other problems can be spotted timely. The nightly report also lists the amount of steps that are queued and ready to be sent, those being computed, and the number of errors and successful steps. Overall totals for the whole project, as well as aggregates by scientist, by computation group and by type of error, are provided. Finally, the report provides information on the turnaround time of work-units, which is directly related to the time required to complete a computation; late work-units can, if desired, be re-issued to faster hosts.



By user

```
scientist  sent  unsent  day_suc  day_unsuc  day_credits
ALICE      1527  1453    868      336        5134196
BOB        2305  447     1466     390        7704984
```

By user and error

```
Code  Error                  ALICE  BOB  [...]
-177  RSC_LIMIT_EXCEEDED       0      0
-185  RESULT_START             0      1
-186  RESULT_DOWNLOAD         12      9
-187  RESULT_UPLOAD            0      0
-197  ABORTED_VIA_GUI         17     21
-198  INSUFFICIENT_RESOURCE    0      0
-226  TOO_MANY_EXITS           1     13
-233  UNSTARTED_LATE           7      8
   1  (Exit code by app)     151    180
   2  (Exit code by app)       0      0
   3  (Exit code by app)      37     28
other (Any other exit code) 111    130
```

By group

```
group_name             sent  unsent  day_suc  day_unsuc
ALICE_BIND_11          81    20      51       21
ALICE_BIND_166_119     422   81      266      94
BOB_TEST_2             83    16      51       20
   [...]
```

Table 1: Example of the daily system monitoring report. Two hypothetical scientists, Alice and Bob, are running experiments. The upper portion of the report shows the amount of successful and unsuccessful computations performed by each scientist; the middle portion shows the most common causes of failures; the bottom part displays a detailed breakdown of the amount of computation performed in each simulation.

2.5. Application interface and templates

To compute a single step, the client computer must receive (a) the application, or executable, which will perform the analysis; and (b) the *input files*, containing the parameters, initial configuration, etc. that describe the computation. When the execution completes successfully, the client will report back a number of *result files.* The number and types of input and output files depends on the specific application; as an example, Figure 5 shows the interface involved in a typical MD simulation.

The types and names of input and output files for a work-unit constitute the *interface* of a given BOINC application. The RBoinc client adapts to the interface of the various applications. Interfaces are described in *template* files; they list all of the input and output files foreseen by the application, providing, for each, a symbolic name, a description (these are visible to the scientist as help for preparing the submission), and the handles used by the application to

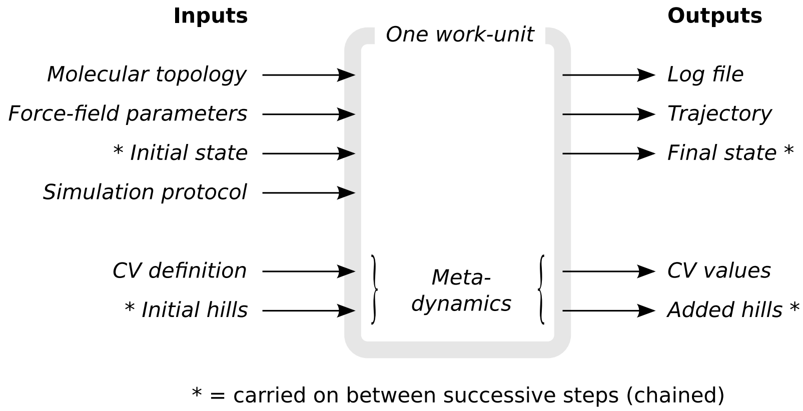


Figure 5: A work-unit is the amount of simulation assigned to a single PC connected to the project, with the corresponding input and output files. In GPUGRID, the input of a simulation are the molecular topology and force-fields, the initial point in the configuration space, and the simulation setup. When the simulation is finished, its outputs are reported back to the server. Depending on the problem, these may include detailed atom trajectories, logs of the energetics, and the history of chosen collective variables (CV).

reference them. Similarly the output template lists the files that are produced after the simulation, their expected sizes and types and whether they should be reused as starting points for the next step in the chain.

The server can have multiple applications installed, each with the corresponding executable and interface, and researchers choose which application they submit to. Depending on the chosen application, they will supply the appropriate set of input files, as explained in section 2.3.

3. Performance

3.1. File pooling

In high-throughput computational experiments, it is customary to have hundreds or thousands of individual simulations experiments set-up and executing at the same time, exploiting the distributed nature of the computation. Typically, the simulations will differ by a few parameters only, being otherwise equal. For example, in *in silico* biophysical experiments for the determination of free energy profiles, techniques such as *umbrella sampling* (US) are frequently used [21]. US requires one to compute a set of statistical distributions on several replicas of the same physical system; in each replica one atom is constrained to stay at a different position through a biasing force. When simulations are complete, the final free free energy profile can be recovered by merging the results of the various replicas through techniques such as the weighted histogram analysis method [22]. As a consequence, hundreds of simulations are started based on essentially the same input files.

Storing on the server one distinct copy of these common files for each replica would be extremely inefficient, because of the wasted disk space and much reduced ability of the system to cache them in disk buffers. The RBoinc system greatly alleviates this problem by indexing all simulation files and storing them in a *pool*. Upon submission, the content of each file is hashed and the corresponding unique ID is obtained via the MD5 algorithm [23]. Files with the same MD5 hash will be consolidated and stored just once in a pool directory. Whenever the specific file is required, e.g. for download by the volunteer's machines, it is accessed from the pool through a reference. Hashes computed by MD5 are 128 bit long, which correspond to approximately 10^{38} available identifiers; given that a typical group pool holds at most $\sim 10^4$ hashed files, the probability of an ID clash is remote.

3.2. Distributed load balancing

One typical case for aggregating several chains into a simulation group is when exploring various sets of parameters for a given system simultaneously. The chains are started at the same time, and run until adequate amounts of data (e.g., statistical sampling) have been collected. However, since they are composed of smaller discrete steps which get assigned to a variety of participants, some of these steps may take more time to be computed than others and so cause the entire chain to slow down. This may be caused by a variety of factors, including the heterogeneous compute power that the hosts participating to the grid can provide, and the fact that the hosts' owners may decide to stop sharing the PC at any time. Even though the BOINC system recovers "late results" reissuing them to more reliable computers, the computation chains will in general not proceed *on par*. This is especially disadvantageous for those problems in which the analysis depends on the complete availability of long chains: if even one chain is late, the whole analysis cannot proceed (Figure 6).

RBoinc incorporates two mechanisms for mitigating this problem. The first, a *distributed load balancer*, acts to assign the most reliable compute resources to lagging chains. If enabled, when a new work-unit in a chain is created, its step number is compared to the highest step number of all other chains the group. If the difference is larger than a configurable threshold, the work-unit is marked as "late", and will be assigned to one of the best-performing hosts of the computational grid. The reliability metric of a host is defined in terms of speed, result turnaround time and fraction of successful results and is tracked by the BOINC system [19].

A second, more aggressive technique implemented in RBoinc to ensure the timely availability of results is *late duplication*. A list of work-units assigned to a host, but not returned yet, is generated daily. The work-units which are holding back a computation chain, e.g. because the corresponding result has not been received within 48 hours, are cloned by the system, and the duplicate is dispatched to a different host. The first result that is computed, either the original or the duplicate, is then taken as canonical, and used to progress the computation chain (while the later result, should it ever arrive, is discarded). At any time, the fraction of late work-units is normally below 10% of the total amount of computation in progress.

3.3. Accounting and policies

The client-server communication protocol is opened by a handshaking and authentication stage. This step links the work-units with the identity of the researcher creating them; this information is used to enforce a number of policies, listed below.

Security and privileges. Only authenticated researchers are allowed to use the system; of course, each user is only authorised to control the computations

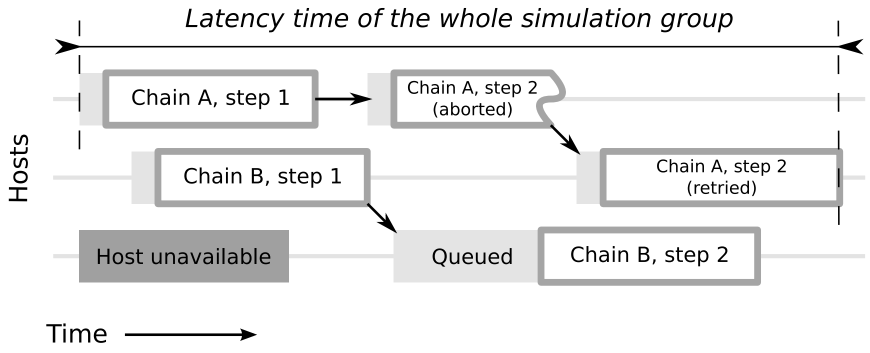


Figure 6: Scheduling of work-units (2 chains of 2 steps each) on three hosts. Hosts can abort the assigned computation at any time (top row). The scheduling system re-issues the step on a more reliable host (middle row). Full analysis of the simulation's results can take place only when all of the steps have been returned (latency time).

he submitted, and the protocols and files he has developed can not be inspected by other researchers.

Credit accounting. Ownership information is also used for credit accounting: when results are returned by a volunteer, he is awarded a certain amount of credit which reflects the amount of computation he has donated to compute it. At the same time, the same amount of credit is debited from the account of scientist that submitted the job. In this way, a project can implement arbitrary types of "credit economies", e.g. supplying researchers with a fixed quota of credit per time period, or even partially lending the system to "guest" groups.

Task safety. Sending a large batch of computations is a relatively risky operation for the project, because it has an impact on the computers of a large number of volunteers. If a work-unit is improperly designed, it could cause computers to hang, or volunteers not to receive credits due; the latter is especially undesirable as it might cause participants to opt out from the grid altogether, impacting all of the researchers. To avoid this risk, new work-units can be tested in a "sandboxed" grid of dedicated computers and, if successful, they automatically receive an approval for running on the volunteer grid.

Accountability and dissemination. Computing power is provided by volunteers, and one important aspect that keeps them attached to a scientific effort is that the directions of the research are, at least partially, understood and shared. When scientists submit jobs, they are requested to provide short description of the objectives of their research in order that volunteers receiving the corresponding work-units are aware of the purpose of their assigned work-units and efforts, and remain supportive of the project.

Authentication and communication between the researcher's workstations and the server takes place over the Internet through the hypertext transfer protocol (HTTP) protocol. The protocol guarantees that the system can operate in a wide variety of network environments, including corporate firewalls and proxies [24]. When operated outside of an intranet environment, security and confidentiality of the RBoinc communications can be assured with the standard HTTP Secure encryption and identification measures [25]. File transfers are carried out through the web distributed authoring and versioning (WebDAV) protocol [26], which ensures efficient file transfers and wide compatibility.

4. Results

The RBoinc system has been used for all of GPUGRID's all-atom MD simulations since April 2009. Approximately 10,000 new computation chains have been started every month. A total of approximately 1,100,000 results were distributed, computed and downloaded up to now, each consisting typically of 10 to 50 megabytes of data. The computation corresponding to one result occupies a high-end GPUs for approximately 8 hours (without interfering with the volunteer's ability to operate the machine) and provides approximately 2 nanoseconds of simulated time in all-atom systems of the order of 50,000 atoms [18].

Table 2 provides an idea of the orders of magnitudes involved in a number of high-throughput computational problems on bio-molecular systems that have been run in GPUGRID. The number of computational chains and work-units

	PDB	**Total**		**Daily**	
System	**code**	**chains**	**steps**	**results**	**MBytes**
SH2-pY	1LKK	73,672	696,000	805	12,100
hERG	—	6,171	55,000	351	5,300
HIV	1HHP	6,037	218,000	901	13,500
GPCR	—	2,921	72,000	169	2,500
GA	1JNO	2,295	31,000	39	600
TPI	1NEY	906	46,000	69	1,000
VILLIN	2F4K	101	4,000	45	700

Table 2: Summary of the results submitted on GPUGRID through the RBoinc submission system. "Chains" indicates the total number of chains which have been started for a specific system, while "Steps" counts the work-units computed and retrieved. "Daily results" is average number of results computed and returned in one day of operation, while "Daily MBytes" is the corresponding amount of output data generated. Key: SH2-pY, binding of Src homology 2 domain with various peptides containing a phosphorilated tyrosine; hERG, Kv11.1 potassium ion channel; HIV, human immunodeficiency virus; GPCR, G protein-coupled receptor D2; GA, gramicidin A ion channel; TPI, triosephosphate isomerase; VILLIN, chicken villin subdomain HP-35.

computed (columns "chains" and "steps") varies among the various systems due to the different priorities assigned and time that some classes of problems have been running.

The RBoinc system has proven a very convenient interface to the management of such extensive experiments; remarkably, each project researcher was able to independently manage one or more of the computation groups listed in Table 2 and the subsequent analysis tasks. In this context, the ability to structure computations in flexible groups and chains, and to monitor and retrieve them as logical units regardless of the actual amount of data involved, was especially useful.

4.1. Example application

To show how RBoinc fits into a realistic scientific work-flow we provide an account of a study conducted with GPUGRID to compute the free energy of binding of the Src Homology 2 domain (SH2) with a short peptide containing a phosphorilated tyrosine, two glutamate and one isoleucine residues (pYEEI). Free energies are a central quantity in computational biology, which describe how tightly two biological structures (e.g. a large biomolecule and a drug) interact with each other under realistic physiological conditions. The recognition between the SH2 domain and a phosphorilated tyrosine, in particular, is a prototypical problem in computational modelling [27, 28].

The general computational protocol employed in GPUGRID for determining free energies of binding is analogous to the one described by Roux et al. [29] and Doudou et al. [30]; for a more detailed description we refer the reader to the paper from Buch et al. [16]. The protocol includes a sequence of three steps: first, several independent replicas of *steered* MD simulations are started from the same initial configuration. In these runs, a biasing force is applied to the system that changes the value of one suitably-chosen parameter (reaction coordinate) [31]. The reaction coordinate is driven to progressively assume a range of values, which correspond to states to be sampled in the subsequent step. For the SH2-pYEEI experiment (Figure 7a), the reaction coordinate was taken as the distance between the center of mass of the pYEEI peptide and the corresponding binding pocket, projected along the z axis [16]. The peptide's center of mass has been driven along 10 independent paths from contact to 35 Å away from the SH2 binding pocket. A harmonic restraint of $k = 1$ kcal/mol/Å^2 was applied to every Cα atom residing in an α-helix or β-sheet of the protein further than 9 Å from the ligand; this prevented rotation and translation of the protein during ligand separation while preserving the flexibility of the binding pocket. The displacement of the ligand was carried out applying a linear force $F = -k_d(z - vt)$ to all carbon atoms of the ligand, where $k_d = 10$ kcal/mol/Å^2 and $v = 5$ Å/ns. Another biasing restraint of $k = 0.1$ kcal/mol/Å^2 was applied to the center of mass of the ligand to restrain it into the xy plane with respect to the initial bound position. Snapshots of the system coordinates were saved during the pulling trajectory at constant intervals corresponding to displacements of 0.1 Å.

Once this step is complete, various configurations were extracted from the ten trajectories, cyclically, at different values of the reaction coordinate. Each configuration was used as a starting state for a new replica of the system, thus building 381 umbrella sampling (US) *windows* [21]. The windows are independent replicas of the system, each subject to a different bias and run in parallel for 50 ns of simulated time apiece, corresponding to 20 remote submission steps (Figure 7b).

The US trajectories were retrieved through RBoinc and analysed with the weighted histogram method [22]. With 50 ns of simulated time per window, a depth of 10.8 kcal/mol is obtained for the potential of mean force profile, resulting in a ΔG of -8.5 kcal/mol [30], which compares well with the experimental value of -8.0 kcal/mol [32].

5. Discussion and conclusions

In this paper we presented an extension of the BOINC DC system that allows individual scientists to submit simulations from their workstations, manage them, and retrieve results automatically transferring files over the Internet. The RBoinc system, currently in operation within the GPUGRID project, can be configured for arbitrary BOINC projects, enabling researchers to approach and use DC in a way which is even closer to that of a “virtual supercomputer”. We discussed the architecture and use of the RBoinc interface with reference to our large-scale DC project dedicated to the in silico study of large bio-molecules, GPUGRID; however, the principles underlying the system apply equally well to most DC-based projects. BOINC’s template mechanism has been extended to describe the interface to remote applications, and to structure computations in groups that present the DC network as a coherent high-performance resource.

DC has become an established model for high performance computing. Unlike HPC facilities, however, DC grids are dependent on the contribution of volunteers. It is therefore important that the projects keep a high standard in the quality of work-units produced, communication of their objectives, and feedback. Also, even though a DC network may have a large aggregate power, its participating computers receive work packages individually, without a connection to the other clients. Volunteers can also suspend their computers at any time, or even elect to leave the grid, thus invalidating the currently assigned task. While these events are automatically handled to some extent by the BOINC middleware, the computation has to be carefully parallelised; the

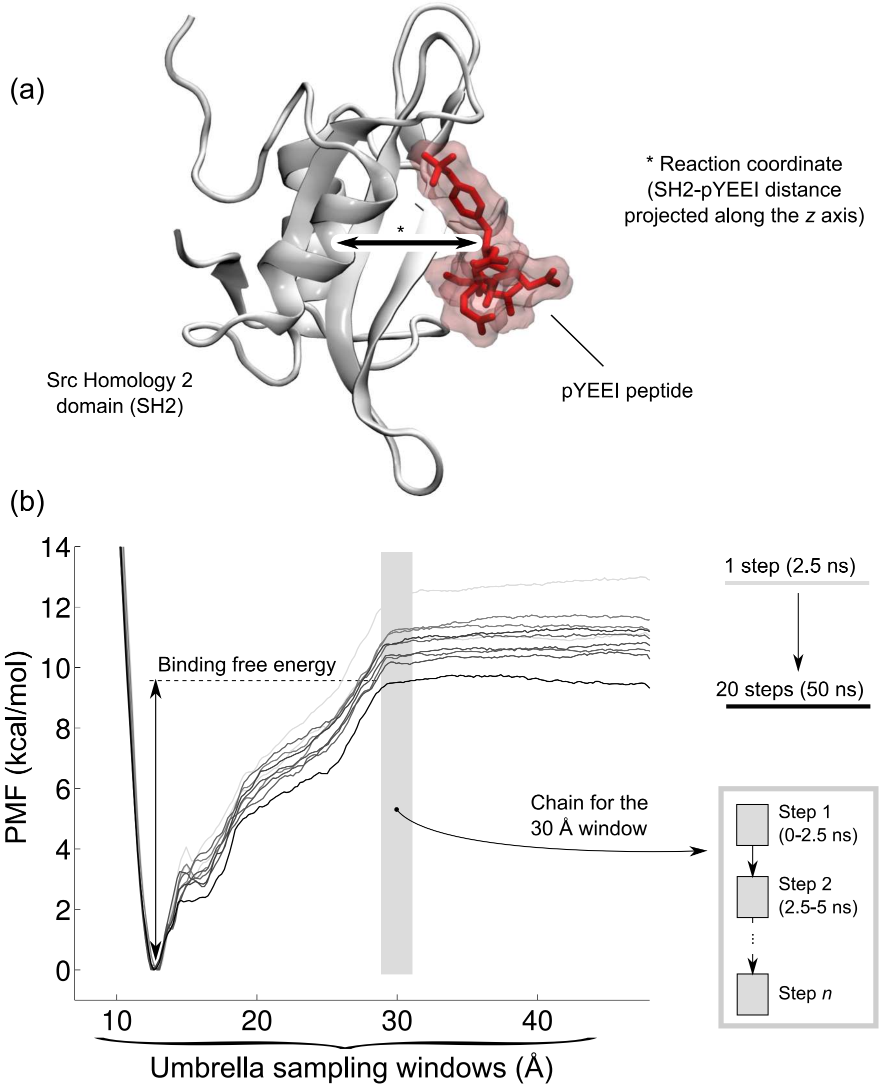


Figure 7: (a) The SH2-pYEEI complex and (b) the potential of mean force profiles obtained with the GPUGRID distributed computing network for the binding reaction [16]. A number of uncorrelated starting configurations are generated by pulling the pYEEI peptide out of the binding pocket, thus obtaining 381 different starting configurations (umbrella sampling windows). Each window is simulated with a different biasing potential for 50 ns, corresponding to a chain of 20 steps, managed by the remote submission system. Once all of the steps are computed and analysed through the weighted histogram methods, the depth of the PMF converges to a final value, which is then used to estimate the free energy of binding ΔG.

concatenation mechanism described serves the purpose of describing the packages of work in a logical structure that makes the work-units easy to define and manage.

Finally, computational scientists in many fields are accustomed to a "low latency" execution model: they prepare experiments, run them, examine interim results and decide whether to keep the computation running, or to stop and amend it. In most cases, following the hypothesis-run-analysis model, the results of one computation are preliminary to the creation of a different experiment. In this sense, it is necessary that not only to gather a large aggregate computation power, but also to minimise the latency for the whole experiment. HPC facilities achieve this allocating compute resources as contiguous chunks. In a DC grid, this can be achieved by limiting the time that one single result may stall the computation; RBoinc's load balancer is one such mechanism.

In the near future, RBoinc will be integrated with the official BOINC distribution; RBoinc is a fundamental component for integrating distributed computing into heterogeneous grids made of conventional HPC facilities, inter-operating transparently. This could be achieved making RBoinc a back-end for existing grid middleware, such as the Application Hosting Environment [33], UNICORE [34], and the Globus toolkit [35], to access DC resources.

6. Acknowledgments

This work was partially funded by the virtual physiological human EU network of excellence (VPH-NoE). MJH acknowledges support from the HPC-Europa2 project (project number 228398). GDF acknowledges support from the Ramon y Cajal scheme and the Spanish National Research Plan. A special appreciation goes to the volunteers of the GPUGRID.net project and their enthusiastic support.

[1] Anderson, D. P. and Kubiatowicz, J., Scientific American **286** (2002) 40.

[2] Rinaldi, A., EMBO Rep **10** (2009) 439.

[3] Science **321** (2008) 1425b.

[4] Giupponi, G., Harvey, M. J., and Fabritiis, G. D., Drug Discov Today **13** (2008) 1052.

[5] Loewe, L., Briefings in Bioinformatics **3** (2002) 377, PMID: 12511066.

[6] Anderson, D. P., Korpela, E., and Walton, R., High-Performance task distribution for volunteer computing, in *e-Science and Grid Computing, International Conference on*, pages 196–203, Los Alamitos, CA, USA, 2005, IEEE Computer Society.

[7] Anderson, D. P., Christensen, C., and Allen, B., Designing a runtime system for volunteer computing, in *Proc. ACM/IEEE SC 2006 Conference Supercomputing SC '06*, pages 33–33, 2006.

[8] Kondo, D., Taufer, M., Brooks, C., Casanova, H., and Chien, A., Characterizing and evaluating desktop grids: an empirical study, in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 26, 2004.

[9] Ren, X. and Eigenmann, R., iShare - open internet sharing built on Peer-to-Peer and web, in *Advances in Grid Computing - EGC 2005*, volume 3470/2005 of *Lecture Notes in Computer Science*, pages 1117–1127, Springer Berlin / Heidelberg, 2005.

[10] Cirne, W. et al., Journal of Grid Computing **4** (2006) 225.

[11] The BOINCstats website, http://www.boincstats.com, 2010, (Accessed 9 Jan 2010).

[12] Gentzsch, W., Sun grid engine: Towards creating a compute power grid, in *Cluster Computing and the Grid, IEEE International Symposium on*, page 35, Los Alamitos, CA, USA, 2001, IEEE Computer Society.

[13] Nitzberg, B., Schopf, J. M., and Jones, J. P., PBS pro: Grid computing and scheduling attributes, in *Grid resource management: state of the art*

and future trends, pages 183–190, Kluwer Academic Publishers Norwell, MA, USA, 2004.

[14] Yoo, A., Jette, M., and Grondona, M., SLURM: simple linux utility for resource management, in *Job Scheduling Strategies for Parallel Processing*, volume 2862/2003 of *Lecture Notes in Computer Science*, pages 44–60, Springer Berlin / Heidelberg, 2003.

[15] Litzkow, M., Livny, M., and Mutka, M., Condor-a hunter of idle workstations, in *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104–111, 1988.

[16] Buch, I., Harvey, M. J., Giorgino, T., Anderson, D. P., and Fabritiis, G. D., Journal of Chemical Information and Modeling **(In press)** (2010).

[17] Fabritiis, G. D., Computer Physics Communications **176** (2007) 660 .

[18] Harvey, M. J., Giupponi, G., and Fabritiis, G. D., Journal of Chemical Theory and Computation **5** (2009) 1632.

[19] Anderson, D. P. and McLeod, J., Local scheduling for volunteer computing, in *Parallel and Distributed Processing Symposium, International*, page 477, Los Alamitos, CA, USA, 2007, IEEE Computer Society.

[20] Harvey, M. J. and Fabritiis, G. D., Journal of Chemical Theory and Computation **5** (2009) 2371.

[21] Torrie, G. M. and Valleau, J. P., Journal of Computational Physics **23** (1977) 187 .

[22] Kumar, S., Rosenberg, J. M., Bouzida, D., Swendsen, R. H., and Kollman, P. A., Journal of Computational Chemistry **13** (1992) 1011.

[23] Rivest, R. L., The MD5 Message-Digest algorithm, RFC 1321, 1992, The Internet Engineering Task Force.

[24] Neerincx, P. B. T. and Leunissen, J. A. M., Briefings in Bioinformatics **6** (2005) 178, PMID: 15975226.

[25] Rescorla, E., HTTP over TLS, RFC 2818, 2000, The Internet Engineering Task Force.

[26] Dusseault, L., HTTP extensions for web distributed authoring and versioning (WebDAV), RFC 4918, 2007, The Internet Engineering Task Force.

[27] Fowler, P. W., Geroult, S., Jha, S., Waksman, G., and Coveney, P. V., Journal of Chemical Theory and Computation **3** (2007) 1193.

[28] Woo, H. and Roux, B., Proc Natl Acad Sci U S A **102** (2005) 6825.

[29] Roux, B., Computer Physics Communications **91** (1995) 275 .

[30] Doudou, S., Burton, N. A., and Henchman, R. H., Journal of Chemical Theory and Computation **5** (2009) 909.

[31] Isralewitz, B., Gao, M., and Schulten, K., Current Opinion in Structural Biology **11** (2001) 224.

[32] Lee, T. R. and Lawrence, D. S., Journal of Medicinal Chemistry **43** (2000) 1173.

[33] Coveney, P., Saksena, R., Zasada, S., McKeown, M., and Pickles, S., Computer physics communications **176** (2007) 406.

[34] Rambadt, M. et al., Experiences with using UNICORE in production grid infrastructures DEISA and D-Grid, in *Grid Computing*, pages 121–130, Springer US, 2009.

[35] Foster, I., Globus toolkit version 4: Software for Service-Oriented systems, in *Network and Parallel Computing*, volume 3779/2005 of *Lecture Notes in Computer Science*, pages 2–13, Springer Berlin / Heidelberg, 2005.