Deploying a fault-tolerant computing middleware over Grid'5000: performance analysis of CONFIIT and its integration with a quantum molecular docking application

Luiz Angelo Steffenel, Jean-Charles Boisson, Chantal Barberot, Stéphane Gérard, Eric Hénon, Christophe Jaillet, Olivier Flauzac, Michaël Krajecki

Deploying a fault-tolerant computing middleware over Grid'5000: performance analysis of CONFIIT and its integration with a quantum molecular docking application

Luiz-Angelo Steffenel[1], Jean-Charles Boisson[1], Chantal Barberot[2] Stéphane Gérard[2], Eric Hénon[2], Christophe Jaillet[1], Olivier Flauzac[1], and Michael Krajecki[1]

[1] CReSTIC-SysCom - EA 3804
[2] Institut de Chimie Moléculaire - UMR CNRS 6229
Université de Reims Champagne-Ardenne
BP 1039, F-51687 Reims Cedex 2, France
firstname.lastname@univ-reims.fr

Abstract. P2P computing middlewares are interesting options for grid computing applications that require scalability and resiliency. Nevertheless, most P2P computation systems rely on partially centralized or hybrid decentralized architectures to distribute tasks and collect the results, raising fault tolerance and bottleneck issues. CONFIIT (Computation Over Network with Finite number of Independent and Irregular Tasks) is a purely decentralized middleware for grid computing, relying on a virtual ring for topology management and for task scheduling. Firstly this paper evaluates the impact of node placement and task granularity on the performance of CONFIIT while solving the well-known Langford permutation problem. Secondly, an application of CONFIIT on a complex real life problem is presented.

1 Introduction

CONFIIT is a purely decentralized peer-to-peer middleware. It was designed to cope with joining the computation resources to be used on a local area network or on the Internet, all while providing fault tolerance and autonomy to the peers. This paper presents recent results on the deployment of CONFIIT over Grid'5000. In the first experiment we evaluate the impact of nodes placement on the communication performance when using the **distributed mode** of CONFIIT. Indeed, CONFIIT relies on a virtual ring for the node management and tasks distribution, so it is essential to evaluate how the token passing mechanism is dependent on the network speed and the ring circumference. Using the well-known Langford's combinatorial problem as a test benchmark, we try to verify the impact of nodes placement in the overall system performance when deploying CONFIIT over a countrywide grid environment. In the second experiment we describe how existing applications can be easily integrated with CONFIIT and deployed on a grid. We selected a quantum molecular docking application that is quite resource consuming but has serious applications on the field of drugs research and enzyme-interaction analysis. Preliminary results with the PDE4-Zardaverine protein demonstrate the feasibility of this approach, obtaining fine results in a reasonable amount of time.

The rest of this paper is organized as follows: Section 2 describes the CONFIIT framework. We detail how the topology management is used to improve robustness, and how tasks are locally and globally scheduled on processors. Section 3 observe the overall performance of CONFIIT as we explore different node placements over Grid'5000. Section 4 details how CONFIIT can be interfaced with an existing algorithm for solving the quantum molecular docking. Finally, Section 5 presents our conclusions.

2 The CONFIIT Middleware

2.1 CONFIIT outline

The notion of FIIT applications was defined in [1], and they are composed by a Finite number of Independent and Irregular Tasks (FIIT). The well-known Mandelbrot's set [2] can be seen as a FIIT application: each pixel (or set of pixels) can be computed independently from the others, and its computation time is unpredictable since it depends on the corresponding region. In such a framework, the programmer needs to decide on how to divide the problem into a finite number of independent tasks, and how to compute each individual task. CONFIIT (Computation Over Network for FIIT) [3], is a middleware for peer-to-peer computing. Each computer collaborating in a CONFIIT computation is called a *node*. A node is set up with three main basic thread components: a topology and communication manager, a tasks manager and one or several task solvers. The nodes are connected according to a logical oriented ring, set up and maintained by the topological layer of the system. Basically, each node knows its predecessor and successor. Communication between nodes is achieved using a token, which carries the state of computation around the ring and ensures load balancing between nodes. Task status are exchanged to broadcast local knowledge on all nodes, and thus, to compute an accurate global view of the calculus. At the end of the computation, the ring spreads termination on nodes.

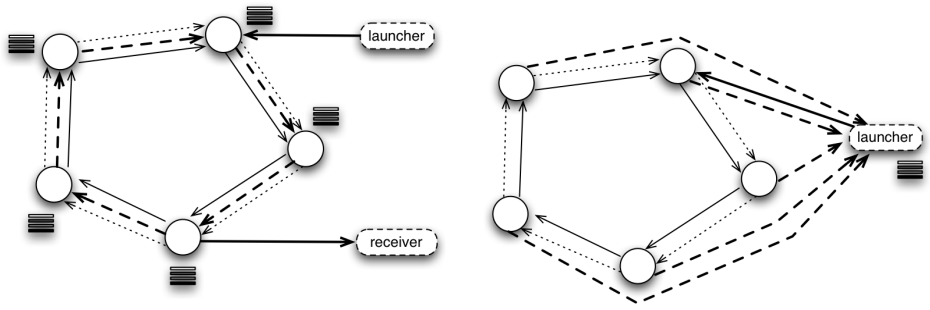
Typically, one *tasks solver* should be launched for each processor on the node. A *task manager* can cope with several *tasks solvers*. Over and above the main thread elements, each node owns a set of data representing the local

known state of the global computation. A node owns the different parameters of the current computations (a list of tasks and associated results). It is able to set up its local parameters of tasks to be computed. At the beginning of a computation, all tasks are marked as *uncomputed*. When a task is completed (locally), its result is stored and will be propagated to the whole ring with the token. To prevent loss of time, the task manager schedules a new task without waiting for the token. But the task could have already been scheduled by another node, and will be *replicated* in the ring. So, the local list of tasks owned by a node is randomly rearranged. Thus, when the task manager picks up the n^{th} task, it *probably* will be different from the n^{th} task on another node [1]. This strategy also guarantees the termination of the computation regardless node faults.

2.2 Programming models

Since constraints of a given application could be different and sometimes in contradiction (fault tolerance, efficiency...), CONFIIT offers two main programming models: distributed and centralized mode.

The **distributed mode** allows an accurate fault tolerance in the computation since task results are locally stored on each node in the community. Thus, a broken computation can be re-launched using already computed tasks. Fig. 1(a) shows information exchanges in the community for a distributed application. At first, the launcher sends the computing request to a node. The request is propagated along the community by the token (dotted arrows). During computation, results of individual tasks are propagated across the community (thick dashed arrows) such that each node could locally store all individual results (data blocks). Concurrently to the computations, information on the global computation is exchanged according to the thin arrows. Another interesting point from this mode is that the launcher only needs to be connected during the initiation phase. At the end of the computation, the global result can be retrieved from any node in the community.



(a) Distributed mode (b) Centralized mode

Fig. 1. Programming modes

The **centralized mode** reduces the global load of storage space and network communication, with the drawback of reducing fault tolerance. This mode is especially suited for the integration of external applications, which manage themselves the data storage. Fig. 1(b) shows information exchanges in the community for a centralized application. At first, the launcher sends the computing request to a node. The request is propagated along the community by a specific token (dotted arrows) as in the distributed mode, but the launcher must remain connected. During computation, results of individual tasks are sent to the initial launcher (thick dashed arrows), which has the storage in charge (data blocks). As in the distributed mode, information on the global computation evolution is updated through the token (thin arrows). A crash on a computing node is not a problem for the computation since the main community ring is fault tolerant, but a crash on the launcher will stop the computation because the gathering of information cannot be achieved.

2.3 Deployment of CONFIIT over Grid'5000

CONFIIT is available as a Java .jar package to simplify the launching of the daemon, the application and the administration tools. To deploy a CONFIIT community over Grid'5000 nodes, we must initially launch the daemons on each site: one "first" daemon that expects the initial connections (launched with the `-wait` option) and the subsequent nodes that join the community through one existing daemon (using the `-node` option). Please note that when one daemon contacts the community, it will be inserted in the virtual ring just after that daemon. This way we can control the initial node placement used in the next section. A daemon will launch as many computing threads as cores/processors a machine can hold, unless we explicitly limit this number (with option `-thread`).

```
Launch the first daemon on machine-01
user@machine-01$$ java -cp Confiit-0.6.jar CDaemon -init -clean &
Launch the daemon on the other nodes (they try to join the community via machine-01)
user@machine-02$ java -cp Confiit-0.6.jar CDaemon -clean -node machine-01
...
```

Back to machine-01 (or any other machine holding the application classes), we can launch the application. CONFIIT will copy the application classes in all nodes, launch the application and manage the tasks completion.

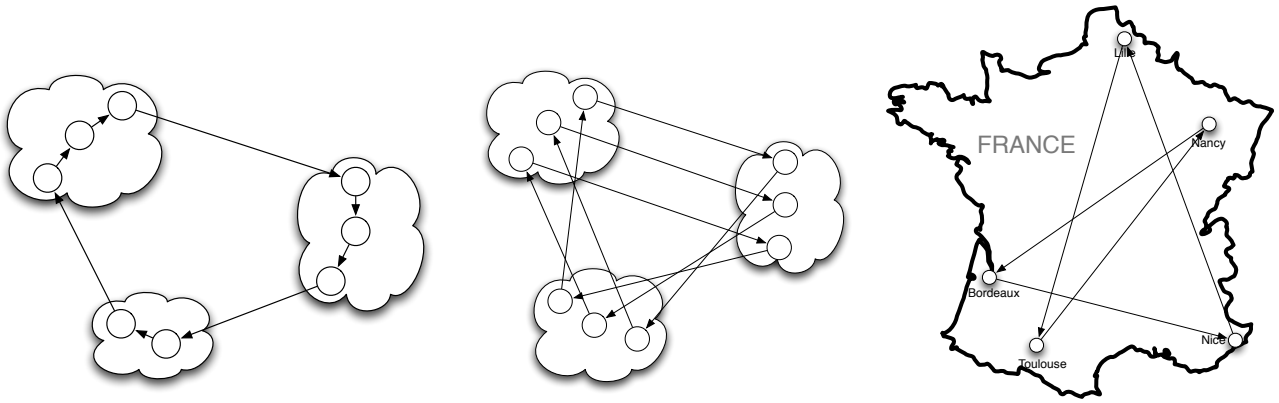
```
user@machine-01$ cd Applications/Langford
user@machine-01$ java -cp ../../Confiit-0.6.jar:. Langford 19 6
```

The CONFIIT community is fault tolerant so it supports both node joining, leaving and even the complete stop of all daemons (in the Distributed mode). Indeed, partial results and tasks completion list are saved in each node (at the `~/.confiit` directory) so that an application instance can be continued later (it is enough to restart a `CDaemon` without the option `-clean`). This behaviour is especially interesting for the *best-effort* submission mode in Grid'5000 and to conduct experiences that exceed the maximum reservation limits imposed by the usage agreement.

3 Scalability on a grid: impact of node placement

This first experiment analyses the impact of nodes placement on the communication performance when using the **distributed mode** of CONFIIT. In the case of the distributed mode, the virtual ring not only distributes tasks and informs nodes about their completion, but also replicates the results among the nodes. Hence, the token must also ensure that all nodes receive the results from different tasks. According to the amount of data exchanged, the underlying network performance and the ring round-trip speed may impact the computing performance.

From these aspects, the placement of nodes becomes an important factor to be evaluated. Contrarily to traditional grid applications, which follow a structured hierarchy, dynamic P2P platforms are subjected to node volatility that may reorder the topology in a small amount of time. In the aspects that concern a virtual ring based topology, this means that nodes can be structured according to a geographic distribution (Fig. 2(a)) but also may face worst scenarios, like when the token hops from cluster to cluster (Fig. 2(b)). This diversity of scenarios makes the token round trip time vary by several orders of magnitude, potentially compromising the performance of a platform that is not prepared to face such variations.



(a) Geographical distribution (b) Worst case distribution (c) Grid'5000

Fig. 2. Node distribution scenarios (a) and (b), and Grid'5000 sites used in the first experiment (c)

3.1 Platform Description

In order to conduct our experiments, we used 60 machines and up to 5 clusters from the Grid'5000 experimental platform[3], located in Bordeaux, Nancy, Nice (Sophia Antipolis), Lille and Toulouse (see Fig 2(c)). For a matter of uniformity, all clusters are composed of bi-processor, dual-core machines (AMD Opteron 2218 2.66 GHz, AMD Opteron 285 2.66 GHz or Intel Xeon 5110 1.6 GHz). This way, each machine runs 4 CONFIIT threads, one by core. Machines inside the same cluster are interconnected by a Gigabit Ethernet network while the clusters are connected by a private backbone of 10 Gbps. All nodes run Debian Linux, with kernel version 2.6.26.

3.2 Experiment scenarios

In order to evaluate the impact of nodes placement on the performance of CONFIIT, we designed a progressive scenario where we scale-up the number of clusters from 1 to 5, while keeping the same number of computing threads. Our objective is to evaluate how CONFIIT performance behaves with the increment on the ring circumference, on both best case (geographically distributed or **geo**, as in Fig. 2(a)) and **worst** case (Fig.2(b)). Indeed, if we consider the

[3] https://www.grid5000.fr

average network latency measured with the *ping* command, a ring with 60 machines in 5 clusters would take around 46.805ms to do a roundtrip in the best case (**geo**: Nancy → Bordeaux → Nice → Lille → Toulouse) and 535.788ms to do the same roundtrip in the **worst** case. Please note that these estimations can be under-evaluated as they do not take into account the communication throughput that is much more dependent on the application communication stack (for instance the Java network stack, in the case of CONFIIT). From this evidence, it is clear that if communication and computation do not overlap efficiently, this roundtrip time may represent a serious performance concern.

In our experiments, we also evaluate the performance of CONFIIT with different task sizes when solving the well-known Langford combinatorial problem, which consists in a permutation of the sequence of *2n* numbers 1, 1, 2, 2, ..., n, n in which the two ones are one unit apart, the two twos are two units apart, and more generally the two copies of each number k are k units apart. For example, a Langford pairing for n = 3 is given by the sequence 2,3,1,2,1,3. Langford's problem is the task of finding Langford pairings for a given value of n. This problem can be modelized as a tree search problem, and a parallel version can be derived by splitting the search tree at depth level d, generating a set of 2^d tasks. In the tests, we varied d from 9 (512 tasks) to 13 (8184 tasks).

3.3 Analysis

Table 1 represents the average of 3 runs for each combination of clusters, topology and number of tasks. The ring topology is constructed in the order Nancy → Bordeaux → Nice → Lille → Toulouse for both **geo** and **worst** scenarios (the worst scenario alternates machines from each cluster, in the same order). For the matter of comparison, we also indicate the execution time in a **single machine**.

Table 1. Computation times when varying the number of clusters and tasks

	512 tasks	**1024 tasks**	**2048 tasks**	**4092 tasks**	**8184 tasks**
Single machine	16425.32s	17242.13s	16675.48s	17545.15s	17361.09s
1 cluster	676.263s	524.347s	405.81s	381.576s	386,414s
2 clusters (geo)	663.646s	567.54s	447.152s	418,517s	417,009s
2 clusters (worst)	677.29s	587.003s	533.274s	422.769s	420.739s
3 clusters (geo)	697.262s	492.119s	422.628s	423.568s	415.886s
3 clusters (worst)	717.504s	538.026s	434.973s	412.001s	413.61s
4 clusters (geo)	673.082s	528.628s	456.385s	421.053s	404.126s
4 clusters (worst)	609.69s	534.328s	439.901s	405.943s	411.397s
5 clusters (geo)	654.827s	478.253s	412.802s	389.592s	413.817s
5 clusters (worst)	656.894s	499.64s	408.056s	402.154s	397.587s

From the analysis of these data, we obtain the following insights:

Impact of nodes placement In spite of the difference between the placement policies **geo** and **worst**, the computation times are barely affected by the ring circumference. While the **geo** strategy generally benefits from a small advantage in comparison with the **worst** strategy, this difference seems to vanish (or event to invert) when the problem is subdivided in smaller tasks. This can be explained by the task selection mechanism from CONFIIT. Because CONFIIT nodes randomly select unsolved tasks, nodes are allowed to start new tasks even if the token has not yet returned. With more small tasks, computation of tasks can continue with a small probability of "double work", while waiting for the token update. These results are valid for the 10Gbit/s interconnection on Grid'5000, and we cannot exclude performance problems if the inter-cluster networks become too slow in comparison with the local-area network (like the traversing of a PSTN or ADSL link).

Number of clusters In spite of the increment in the number of clusters, no slowdown could be observed in the experiments. Also, the ratios between the performances of Langford's algorithm in one or several clusters remain stable, which is an interesting fact. Although these results are encouraging, especially when considering grid environments, we believe that further scalability tests should be necessary to determine the limits of the platform if we plan to deploy experiments in open networks.

Impact of task size We finish the analysis of the experiments by looking on the impact of task sizes on the overall computation time. Table 1 shows that a small number of tasks does not parallelizes as well as when a larger number of tasks is deployed. Actually it is not easy to determine if a task will have a long or small duration, as FIIT tasks are independent and their computation time depends on the evaluation algorithms. Nevertheless, we can estimate that in a combinatorial tree search the more we subdivide the tree, the shorter will be the average computational time of each branch, up to the point where computation/communication overlap reaches its limits.

4 Integrating CONFIIT with existing applications

While CONFIIT proposes a rich API[4] for the development of FIIT applications directly in Java, it also supports the integration of existing external applications, acting this way as a deployment middleware. This section presents the preliminary results obtained in a collaboration with chemistry researchers.

4.1 Case Study: quantum molecular docking

Molecular modeling tools are widely used to search for new drugs in medicinal and pharmaceutical fields. Drugs are usually (relatively) small organic molecules called "ligand" and mostly produced their effect by interacting with a biological macromolecule such as protein. Among several molecular modeling techniques, molecular docking is employed to describe the best orientation of a ligand that binds to a particular protein of interest.

In the present work, several computational techniques are combined to carry out molecular docking of a known *PDE4* inhibitor (*zardaverine*) in view of tackling the design of new inhibitors derived from *pyridazinone*. *PDE4* inhibitors act by increasing intracellular concentrations of cyclic *AMP*, which has a broad range of anti-inflammatory effects on various key effector cells involved in asthma and chronic obstructive pulmonary disease (COPD). The very recent AlgoGen-DivCon[4] code combining a genetic algorithm for conformational search and the Divide and Conquer linear semiempirical quantum method for energy evaluation is applied to the *PDE4-zardaverine* complex.

Because genetic algorithms are inherently parallel and the evaluation of one chromosome does not affect the evaluation of another chromosome, several evaluations can be carried out simultaneously. Deploying the AlgoGen-DivCon code over a distributed grid environment with the help of CONFIIT allows an improvement of the results without increase the time needed to evaluate each generation. For instance, with a relatively small complex (1000 atoms), one solution can require more than 10 minutes to be evaluated. By distributing the evaluation of all a population of solutions on grid with different level of granularity, we allow a better exploration of the search space and avoid a premature convergence of the algorithm (if the number of processing cores is sufficient).

To assess the relative performance of the deployment of our problem over CONFIIT [3], we have chosen a test system consisting of the docking of the *PDE4-zardaverine* complex. The aim of this investigation is to obtain a reliable molecular dockings of PDE4-ligand complexes using a quantum description of the interaction energy obtained with AlgoGen-DivCon in a reasonable amount of time. Thus, we verify whether the program can correctly reproduce the binding mode of the *zardaverine* co-crystallized with *PDE4*.

4.2 Interfacing AlgoGen-DivCon and CONFIIT

In order to be deployed on grid, the original code combining a python genetic algorithm (GA) with a Fortran executable as evaluation function had to be interfaced with CONFIIT (in **Centralized mode**). In one hand, the GA does not need to know how the individuals are evaluated (sequentially, on a cluster or a grid). In the other hand, CONFIIT can be used with any algorithms and not only GAs. Therefore, to deploy the application, we use a script generator that relies on two other scripts: the first in Python for configuring the genetic algorithm and the second in shell script to prepare the system and launch CONFIIT. During the evolution of the genetic algorithm, each time an evaluation is required, a query is given to CONFIIT for the current population. CONFIIT distributes the evaluation of the different individuals on the available task solvers of the community. Finally, when the genetic algorithm terminates, an archive of all the results is made and put in the home directory of the user. The implementation of a script generator has been motived by several facts:

- the final user (chemists here) does not need (and does not want to know) how all the application works but wants only the results. Consequently, proposing an automation of the generation of the input files is essential.
- using an application to prepare the experiments allows to verify the parameters and optimize the configuration according to these parameters. For instance, if the user wants to launch the calculus on a simple multi-core machine, the script generator does not enable the use of CONFIIT and configures the genetic algorithm to make a multi-core experimentation.
- generally, using a grid generally means to use a submission server (OAR or PBS Pro, for example). So, the script generator helps the user to submit its experiment: the scripts prepare the data, send it to the grid, launch the experiment and gather the results from the grid to the user account.

DivconEval class according to the configuration, the genetic algorithm can make the evaluation of the population:

- sequentially on a mono-core processor: each individual is written on the file system and the command `divcon` is called for each respective individual.
- in parallel on a N-cores processor or N-processors node: N individuals are written on the file system and N instances of `divcon` are launched. The file system is regularly checked in order to know which individuals were already evaluated. If M individuals were already evaluated, $N - M$ evaluations are launched. The aim is to keep the cores/processors fully charged.

[4] `http://cosy.univ-reims.fr/~lsteffenel/CONFIIT`

– in parallel on a grid: the genetic algorithm writes all the individuals on a *NFS* location and an instance of the `DivconEval` class is provided.

In the last case, the CONFIIT community is contacted for evaluating a set of individuals. The `DivconEval` class contains all the information useful for the community in centralized mode, such as the size of the population and the location of the individuals on the file system. Each task in CONFIIT is identified by a number and, in our case, the task number corresponds to the rank of an individual in the population. The task 1 is the evaluation of the first individual of the population, the task 2 the second, etc. Each task solver of the community evaluates an individual by launching a script indicating the evaluation method (`DivCon`, in our case).

4.3 Preliminary results

As a result of this very preliminary study, the best conformation obtained by the docking is found to be relatively closed to the experimentally observed position indicated by an rmsd of 3,45 Å. As can be seen from Figure 3, the orientation of the *zardaverine* ligand is properly reproduced. However, it is very likely that the geometry optimization of the ligand we had performed before docking prevents the *zaradaverine* to be perfectly positioned in the active site of the *PDE4* structure.

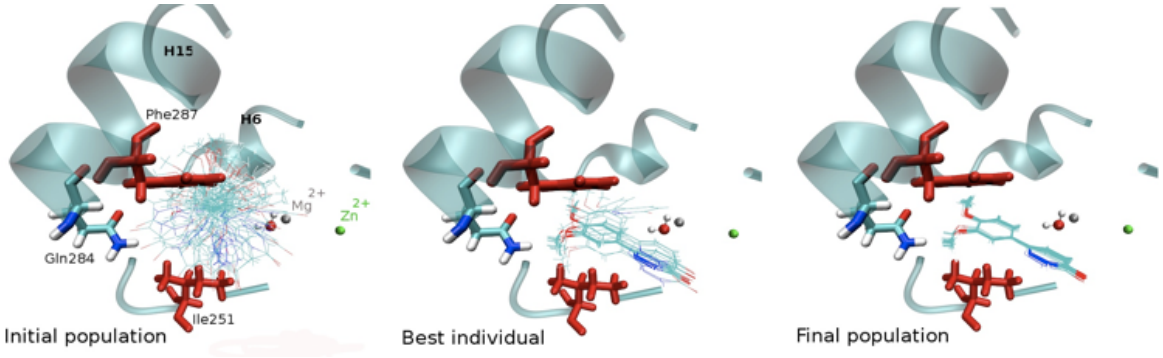


Fig. 3. Zardaverine conformations before docking; successive best solutions; individuals of the last generation.

The main interactions distances are reported in Table 2. The best solution emphasizes the key feature of the binding mode that is the interaction between the *catechol ether moieties* and the *Gln284* (see Fig 4). For the matter of comparison, the run time needed for this first calculation in a single processor (12 cores) is about 167 hours for a population of 32 individuals and 500 generations. A total number of 12 008 evaluations has been performed. It is to be noted that no adjustments of the parameters for the docking program has been required to achieve this result and that no other conformations cluster is obtained.

Table 2. Comparison of geometrical distances from the best docking solution, in Å.

	R1	R2	R3	R4
Crystal structure	3.36	3.05	4.32	4.01
AlgoGen-DivCon	4.80	3.38	3.60	4.71

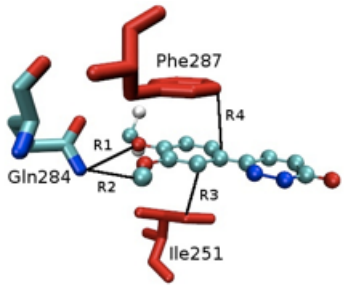


Fig. 4. Selected geometrical parameters from the best docking solution

This preparatory study reveals that the Algogen-Divcon molecular docking approach is able to sample the conformational space of the *zardaverine* inhibitor into the PDE4 protein in the presence of metal ions in a reasonable amount of time. Increasing the grid size should allow for exploring a new ligand in less than one day, as well as improving the population size. Of course, further calculations are needed to compare the influence of the parameters set during docking (population size, generations number, size of protein model, etc.). CONFIIT scheduling mechanism proved to

be adapted to such problems, which can also take advantage of CONFIIT's fault-tolerance aspects to deploy long-run experiments.

4.4 Discussions

This preparatory study reveals that the Algogen-Divcon molecular docking approach is able to sample the conformational space of the *zardaverine* inhibitor into the PDE4 protein in the presence of metal ions in a reasonable amount of time. Increasing the grid size will reduce the computing time as well as allowing the improvement of the population size. Of course, further calculations are needed to compare the influence of the parameters set during docking (population size, generations number, size of protein model, etc). CONFIIT scheduling mechanism proved to be adapted to such problems, which can also take advantage of CONFIIT's fault-tolerance aspects to deploy long-run experiments. In addition, attempt to include the free energy of solvation into the binding interaction model as well as the change in entropy ΔS between bound and unbound state should be considered for molecular docking investigations. Concerning the quantum method for energy evaluation, some other Hamiltonians should be used such as the PM3-PIF method[5], which has been carefully parametrized to improve the description of intermolecular interactions, more precisely, hydrogen bonds.

The next step in the grid deployment of the algorithm is to be able to evaluate the population at two levels: individual and sub-system. Currently the granularity of the deployment is limited to the individual evaluation i.e. an individual is fully evaluated by a single processor. But it can be possible to evaluate an individual in parallel. A MPI version of the evaluation function exists and shall be tested to speed-up the execution time. Nevertheless, in order to have an important population evaluated in parallel, the number of resources required is important. For instance, for evaluating 100 individuals with a 10-nodes individual paralleling evaluation, the ideal case (evaluation in one iteration) needs 1000 computing cores.

5 Conclusions

This paper aims to present CONFIIT and illustrate its integration on the Grid'5000 network through two recent experiments. The first experiment is a traditional scalability performance analysis where we observe how the geographical disposition of nodes in a grid environment impact on the performance of an application. Using a distributed solver for the Langford's problem as the evaluation subject, we were able to observe the main factors that influence the computation performance, resulting in the fact that CONFIIT is little sensitive to the performances of the network. As tasks computation is almost independent of the token mechanism, the nodes keep selecting uncomputed tasks even without new updates. The experiments show a trade-off between computation and communication overlap that limits the speed-up in some cases.

As CONFIIT proved to be efficient with well-known benchmarks, we tried to demonstrate how real life problems that need a lot of computational resources could also be adapted and deployed in a grid. For instance, we associate with chemistry researchers to develop a solution for a quantum molecular docking problem, which is a very resource expensive problem. CONFIIT allowed us to obtain an efficient grid version for the resolution of this problem, allowing the chemistry researchers to discovery new drugs and/or reach a better understanding of the secondary effects of existing drugs.

Acknowledgement

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see *https://www.grid5000.fr*). This work is partially supported by ANR (*Agence National de Recherche*), project reference ANR 08 SEGI 022.

References

1. Krajecki, M.: An object oriented environment to manage the parallelism of the FIIT applications. In Malyshkin, V., ed.: Parallel Computing Technologies, 5th International Conference, PaCT-99. Volume 1662 of Lecture Notes in Computer Science. Springer-Verlag, St. Petersburg, Russia (September 1999) 229–234
2. Dewdney, A.: A computer microscope zooms in for a close look at the most complicated object in mathematics. Scientific American (1985) 16–24
3. Flauzac, O., Krajecki, M., Steffenel, L.: Confiit: a middleware for peer-to-peer computing. Journal of Supercomputing **53**(1) (July 2010) 86–102
4. Thiriot, E., Monard, G.: Combining a genetic algorithm with a linear scaling semiempirical method for protein-ligand docking. Journal of Molecular Structure: THEOCHEM **891** (2009) 31–41
5. Bernal-Uruchurtu, M.I., Martins-Costa, M.T.C., Millot, C., Ruiz-López, M.F.: Improving description of hydrogen bonds at the semiempirical level: water–water interactions as test case. Journal of Computational Chemistry **21**(7) (2000) 572–581