Exploring Large Scale Receptor-Ligand Pairs in Molecular Docking Workflows in HPC Clouds

Kary Ocaña[1], Silvia Benza[1], Daniel de Oliveira[2], Jonas Dias[1], Marta Mattoso[1]
[1]COPPE/Federal University of Rio de Janeiro - UFRJ, Rio de Janeiro - Brazil
[2]Fluminense Federal University - UFF, Niterói - Brazil
{silviabenza, kary, jonasdias, marta}@cos.ufrj.br; danielcmo@ic.uff.br

Abstract— **Computer-aided drug design techniques are important assets in pharmaceutical industry because of their support for research and development of new drugs. Molecular docking (MD) predicts specific compound's binding modes within the active site of target proteins. Since MD is a time-consuming process, existing approaches reduce the number of receptors or ligands in docking by evaluating only small sets of compounds. This restriction in the search space reduces the chances to uniformly cover the diverse space of compounds and misses opportunities to recognize whether new drugs can be identified. Another difficulty with large-scale is analyzing the results, e.g. browsing all directories manually to find which pairs were docked successfully. To address these issues we explored the potential of data provenance analysis and parallel processing of SciCumulus, a cloud Scientific Workflow Management System. We present SciDock, a molecular docking-based virtual screening workflow and evaluate its execution using 10,000 receptor-ligand pairs related to proteases enzymes of protozoan genomes. The overall performance of SciDock using 32 cores, in cloud virtual machines, reaches improvements up to 95.4% when running SciDock with AutoDock and 96.1% when running SciDock with Vina. We show how data provenance improved the result analysis and how it may indicate potential proteases drug targets for protozoan treatments.**

Keywords-component; workflow; cloud; drug discovery

I. INTRODUCTION

The process of drug discovery and development is considered to be time-consuming, risky and costly; *e.g.* a typical lifecycle for this process takes approximately 14 years [1] with a cost of approximately one billion USD [2]. Computer-assisted drug design [3] (CADD) is one of the existing strategies that have been developed to shorten the research cycle and reduce the expense risk of failure for drug discovery. CADD covers many aspect of drug discovery, including computer programs for designing compounds and tools for systematically assessing potential lead candidates. Among the most used computational approaches to drug discovery is structure-based drug design [4] (SBDD). One of the most prominent SBDD methods is molecular docking-based virtual screening that relies on the knowledge of the target structure, which is obtained from crystal structures (X-ray crystallography), nuclear magnetic resonance (NMR) spectroscopy data or homology models [5].

The complexity of designing a molecular docking-based virtual screening is choosing the set of programs and scoring functions that are adequate to the large-scale of receptor-ligand pairs. The complexity of its implementation is related to the management of thousands of combinations of receptor-ligand pairs and the consequent high number of heterogeneous results. For composing, executing and analyzing scientific experiments in large-scale, scientists can use Scientific Workflow Management Systems [6] (SWfMS) designed for High Performance Computing (HPC) such as Swift/Turbine [7], Pegasus [8], Tavaxy [9], and SciCumulus [10]. Scientific workflows help to mitigate the inherent complexity of scientific workflows. A scientific workflow is an abstraction that models the docking experiment as a set of activities connected by a data flow. These engines distribute several concurrent activity executions in an HPC environment while capturing provenance data [11], *i.e.* the historical information of the workflow execution. Scientific workflows for molecular docking are seen as a new exploring ground in bioinformatics and computational medicinal chemistry [12].

SWfMS can be configured to control input ligands combination automatically as a parameter sweep for each parallel workflow execution. Some of these workflows may execute for weeks or even months thus requiring HPC [13] support, such as cloud computing [56]. Clouds are becoming a viable HPC alternative to traditional cluster or grid environments, since clouds have already demonstrated applicability to a wide-range of problems in several scientific domains [9,22–28]. In addition, clouds like Amazon AWS [21] facilitate the deployment of experiments and data as services following a Software as a Service [14] (SaaS) deployment model, which facilitates the effective use of an HPC computing infrastructure by bioinformaticians.

Using parallel processing is only part of the solution to exploring large receptor-ligand datasets. Analyzing the resulting files also in large scale is an open issue. There are molecular docking workflows that explore parallel processing with different approaches, but all of them limit the number of receptors or ligands in each execution [22–29] and none of them varies the receptor for each ligand. Tracking these variations along the results is very complex in large-scale. To the best of the authors' knowledge, there is no report of large-scale molecular docking experiments that processes such large datasets of receptor-ligand pairs while varying the number of receptors for each ligand.

The goal of this work is to address the problems of designing a molecular docking workflow, executing it and analyzing its results in large-scale. We present the design of SciDock, a molecular docking-based virtual screening workflow, which contains high performance docking programs. SciDock adapts its definition to address the heterogeneous size of the compounds. When compounds are

978-1-4799-4116-2/14 $31.00 © 2014 IEEE
DOI 10.1109/IPDPSW.2014.65

large and flexible SciDock invokes AutoDock Vina (Vina), otherwise the workflow invokes AutoDock (AD4). For executing SciDock in parallel, we use cloud HPC parallel execution from SciCumulus SWfMS. A key feature in SciCumulus, not found in other cloud SWfMS, is its adaptive execution that takes advantage of cloud elasticity power. SciCumulus can adapt the number execution resources (i.e. virtual machines -VM) according to the current load. In SciDock, it acquires and releases VMs according to the profile of workflow programs. In addition, SciCumulus has a re-execution mechanism, which supports long running workflows, when some activity executions fail and need to be re-submitted. Finally, the results of SciDock are analyzed with the help of SciCumulus' provenance data generation by submitting high level database analytical queries.

We evaluate SciDock with the targeting of cysteine proteases [30] (CP) in protozoan neglected tropical diseases [31] (NTD). Although this is a representative example in the NTD domain [32], this is one example of thousands that could be explored. We present a performance analysis of SciDock parallel executions using SciCumulus up to 128 cores in Amazon AWS. The results show that SciDock is capable of processing up to 10,000 receptor-ligand pairs with significant performance improvements. The highest performance gain in SciDock reduces execution time from 12.5 days (2 cores) to 11.9 hours (128 cores) with AD4 and from 9 days (2 cores) to 7.7 hours (128 cores) with Vina. Due to time constraints, molecular docking typically evaluates small sets of compounds, which drastically limits the search space, thus reducing the chance to identify new drugs. With larger sets, *e.g.* analyzing 1,000 compound pairs, SciDock detects 287 and 355 favorable receptor-ligand interactions using AD4 and Vina, respectively. This discovery would have been impossible with a reduced number of receptor-ligand pairs or if they were in the complementary space with no favorable interactions.

This paper is organized as follows. Section II discusses related work. Section III presents background on molecular docking. Section IV describes the specification of the SciDock workflow and presents its implementation using SciCumulus cloud workflow engine. Section V shows the experimental results and Section VI concludes the paper.

II.RELATED WORK

The complexity of molecular docking workflows is addressed by several approaches [21-28] but they are all limited by the use of a reduced number of receptors or ligands. DockFlow [22], is a workflow for virtual screening that integrates different docking tools (FlexX, AutoDock, DOCK and GAsDock) on a grid execution while FReDoWS [23] is a workflow for molecular docking experiments that executes AutoDock in clouds with Hadoop, consuming a unique receptor (multiples conformations) and ligand. Both of them are not able to adapt the workflow according to the type of input data neither to control activity failures.

AutoDockCloud workflow [24] is a Hadoop-based workflow and it is executed on a private cloud platform. It only handles the docking procedure in the screening task (AutoDock), pre-docking and post-docking procedures were not included in the workflow performance analyses. Although AutoDockCloud is easy to be deployed since it is based on the Hadoop framework, it has the same weaknesses of Hadoop such as static scheduling, unability to execute adaptive workflows and lack of provenance support.

CometCloud [25] is an autonomic computing engine for running applications on hybrid computing environments, with a MapReduce programming layer and executed on Amazon EC2. CometCloud is “similar” to SciCumulus in the sense that both execute scientific applications with parallelism in MapReduce paradigm. However, it is not connected to the concept of scientific workflows and it does not provide workflow-specific features such as provenance analysis, re-execution of failed activities or optimizations based on domain-specific data. Although CometCloud team has implemented an interesting Protein Data Bank mining application, docking experiments were not addressed yet.

E-novo [26] is a workflow for virtual screening, with a fast validation and scoring methods that uses predefined optimization libraries for grids. FLIPDock [27] is a cloud tool for docking of a single flexible ligand-receptor pair using an older version of AutoDock. VSDocker [29] docking tool runs AutoDock on Windows computer clusters for parallel high-throughput virtual screening. Although existing approaches represent a step forward, even the ones that execute in parallel reduce the search space by fixing a small number of ligands, which may limit the biological inference.

III. MOLECULAR DOCKING

Molecular docking [33] refers to the prediction of the binding modes of small molecule ligands within the active site of the target protein models [34], available in structural databases *e.g.* RCSB-PDB [35]. A receptor’s (protein) 3D structure is compared to a ligand (small molecule) 3D structure in order to find the best binding energy between that receptor-ligand pair. This process is based on the model propose by Fischer [36], known as “key-lock” model, where the protein (lock) has a cavity in which the ligand (key) docks perfectly [37].

Molecular docking attempts to mimic the process of bringing together a protein and a ligand to form a non-covalent complex and to reveal the electrostatic and steric complementarity between the protein and ligand. Thus, a docking algorithm has three main tasks: determinate the binding site (receptor’s active site), place the ligand in the binding site and evaluate the binding energy between that receptor-ligand pair [12]. Another crucial aspect, during the docking process, is the scoring function applied to rank docking. More than 60 small-molecule docking programs and 30 scoring functions have been proposed as stated by Morris and M. Lim-Wilby [33], Taylor *et al.* [34] and Kitchen *et al.* [38]. The most popular docking tools are AutoDock [39], AutoDock Vina [40], FlexX [41], Glide [42] and GOLD [43].

Furthermore, molecular docking experiments represent a *big data* [44] scenario, since they need to manage high volumes of data (*e.g.* 600 GB for each execution of small-scale experiments). Thousands or millions of potential receptors and entire ligand databases need to be screened.

Yet, scientists manually test each docking process, although this process is tedious and error-prone due the high number of sequences in databases (*e.g.* NCBI [45]) and also due to the complexity of implementing molecular docking experiments to manage thousands of combinations of heterogeneous resulting receptor-ligand pairs.

IV. Designing SciDock Scientific Workflow

Designing a molecular docking-based virtual screening workflow is complex due to choosing small-molecule docking programs and scoring functions among a large number of options and combinations. Selecting the best alternative implies analyzing thousands of combinations of receptor-ligand pairs for each chosen program-scoring function combination. This Section presents the specification of SciDock and how SciCumulus helped in the configuration and result analysis by its provenance database support.

A. *SciDock Conceptual Specification*

Usually, molecular docking experiments are divided in four main macro-activities, labeled from A to D, and each macro-activity are further decomposed into one or more activities, labeled from 1 to 8 (Figure 1), as follows.

- Macro-Activity (A): Input Preparation, activities 1 to 3;
- Macro-Activity (B): Coordinates Generation, activity 4, 5;
- Macro-Activity (C): Docking Preparation, activity 6, 7;
- Macro-Activity (D): Molecular Docking, activity 8.

The SciDock activities are: (1) ligand transformation, (2) ligand preparation, (3) receptor preparation, (4) AutoGrid's parameter preparation, (5) receptor's coordinates map generation, (6) docking filter, (7) docking parameter preparation, and (8) docking execution.

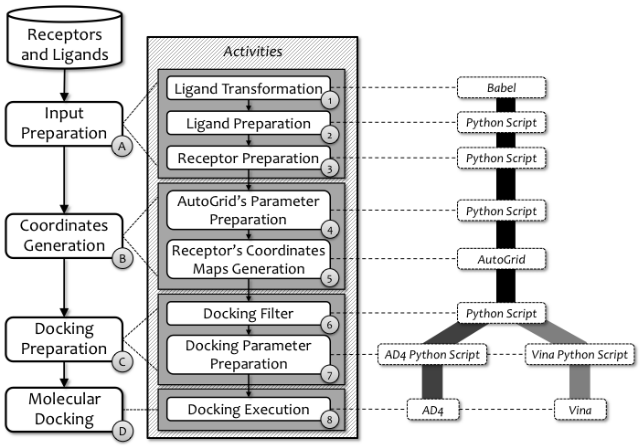


Figure 1. SciDock Workflow Conceptual View

The first activity executes Babel to convert the format of the ligand from SDF to Sybyl Mol2. Then, this ligand format is used as input by the second activity, which executes a python script from MGLTools (*prepare_ligand4.py*) and produces a new PDBQT ligand format file as output. The third activity executes a prepare-receptor script also from MGLTools and produces a PDBQT receptor file format from PDB containing the protein's structure. AutoDock can recognize both PDBQT files: from the ligand and receptor.

The fourth activity executes a python script from MGLTools that extracts parameters contained in the ligand and receptor PDBQT files and generates the Grid Parameter File (GPF). The fifth activity receives parameters defined on the GPF file (*e.g.* ligand and receptor atoms types extracted from PDBQT files) and executes AutoGrid to generate the coordinates maps based on that information. Files generated by AutoGrid are: one map file for each receptor atom type; two map files for the electrostatic and desolvation maps; the grid map field file; the dimension and coordinates box of the grid; and the execution log file.

The sixth activity executes an in-house python script to filter receptors by size, and generates two datasets to be executed using AD4 and/or Vina. The seventh activity has two possibilities: (7a) AD4 uses python scripts from MGLTools to extract parameters contained in the ligand and receptor PDBQT files generating the file Docking Parameter File (DPF); and (7b) Vina uses custom python scripts to extract the dimension and coordinates box of the grid generating the configuration file. The eighth activity: (8a) receives parameters defined on the DPF file (*e.g.* for the genetic algorithms used in docking) and executes AD4; and (8b) receives parameters defined on the configuration file and executes AutoDock Vina. AutoDock predicts the ligand-receptor binding process using coordinates maps. AutoDock generates an execution log file, which contains the binding process execution' information, a table of RMSD [46] (*i.e.* root-mean-square deviation) values, histograms, and the best conformation found by AutoDock for receptor-ligand pairs. AutoDock Vina predicts the ligand-receptor binding process using coordinates maps. AutoDock Vina generates the execution log file, which contains the information from the binding process execution and the best conformation found by the AutoDock Vina for receptor-ligand pairs. AutoDock Vina also generates a new version of the receptor PDBQT file, which contains the binding information.

B. *Using SciCumulus: Benefits and Challenges*

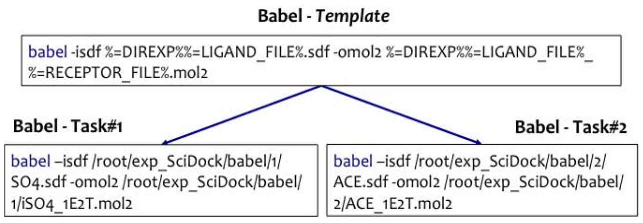
This conceptual specification of SciDock could be executed by any parallel SWfMS, scientists should choose the SWfMS that is more suitable for the type of experiment being executed. SciCumulus was our choice since its unique features have already demonstrated to be essential in several HPC bioinformatics workflows [17,47–49].

Before using SciCumulus, we instrumented SciDock activities using template files and extractor programs. This instrumentation allows for SciCumulus to capture all parameters involved in the workflow execution (with their associated values) to store them in the provenance repository to be further queried. In addition, using extractor components SciCumulus is able to open produced files, extract useful information (*e.g.* statistical binding energy values) and associating them to provenance records. This allows for domain-specific queries that can be used to analyze results and to improve SciCumulus scheduling as presented by Oliveira *et al.* [50]. SciDock is specified in an XML file, which is used by SciCumulus when the workflow is executed. Figure 2 shows an excerpt of the SciDock specification for Babel, its first activity.

```
<SciCumulus>
   <database name="scicumulus" server="ec2-50-17-107-164.compute-
   1.amazonaws.com " port="5432"/>
      <SciCumulusWorkflow tag="SciDock" description="Docking"
      exectag="scidock" expdir="/root/scidock/">
         <SciCumulusActivity tag="babel"
         templatedir="/root/scidock/template_babel/"
         activation="./experiment.cmd">
            <Relation reltype="Input" name="rel_in_1" filename="input_1.txt"/>
            <Relation reltype="Output" name="rel_out1"filename="output_1.txt"/>
            <File filename="experiment.cmd" instrumented="true"/>
</SciCumulusActivity>
</SciCumulusWorkflow>
</SciCumulus>
```

Figure 2. An excerpt of XML specification for the SciDock Babel activity

As observed in Figure 2, templates do not have the actual values of the parameters used, as they use tags. Tags are replaced by actual values dynamically during the execution, as executions are ready to be started as presented in Figure 3.



```
babel -isdf %=DIREXP%%=LIGAND_FILE%.sdf -omol2 %=DIREXP%%=LIGAND_FILE%_
%=RECEPTOR_FILE%.mol2
```

```
babel –isdf /root/exp_SciDock/babel/1/
SO4.sdf -omol2 /root/exp_SciDock/babel/
1/iSO4_1E2T.mol2
```

```
babel –isdf /root/exp_SciDock/babel/2/
ACE.sdf -omol2 /root/exp_SciDock/babel/
2/ACE_1E2T.mol2
```

Figure 3. Activity specification in SciCumulus for the first activity Babel

Once defined, workflows can be reused/modified as several images in Amazon AWS (ami-77f6381e, ami-4550c42c, ami-742bf91d), where workflows were deployed.

There are several advantages in using SciCumulus that are not available in other workflow engines. Scientists can query the provenance database to help in workflow configuration, reusing previous related workflows to model a new one. These queries can be as simple as "Obtain statistics related to SciDock executions" as presented in Query 1 (Figure 10) or "Retrieve names, sizes and locations of SciDock files with the extension '.dlg'" of Query 2 (Figure 11); or more complex by mixing these simple ones with *e.g.* the extraction of domain information *i.e.* contained in files. It is worth noticing that SciCumulus allows for runtime provenance query, which is a unique feature, yet it allows for user steering and anticipating results.

The second benefit is the scheduling cost model of SciCumulus. Since SciDock activities have heterogeneous execution time distribution, SciCumulus can schedule short-term activities to less powerful VMs and long-term activities to more powerful VMs. In addition, SciCumulus is able to scale the amount of VMs up and down according to performance behavior. For example, Vina is computing intensive and demands more computing power. By monitoring or querying Vina's execution history in the provenance database, SciCumulus scales up the amount of VMs to improve the performance.

The third benefit is related to fault tolerance. Each execution of SciDock contains about 10% of activity execution failures. These faulty executions have to be aborted and SciCumulus has to restart each activity. Since it has all information stored in the provenance repository it does not need to restart the entire workflow. It is easy to find and re-execute only the failed activities.

V. Executing SciDock and Result Analysis

In this Section we present an evaluation of the parallel execution of SciDock using SciCumulus cloud workflow engine. Section A presents the environment setup whereas Section B presents the experiment setup and SciDock execution. Section C presents a performance evaluation and Section D briefly discusses biological issues.

A. Environment Setup

SciCumulus engine is based on an algebraic approach [51] where each activity receives a relation as input and processes each tuple independently. By storing the workflows definition related to its provenance data execution, powerful domain queries can be defined. For example, for each parameter, SciCumulus records all steps and files associated to the executed activities with this parameter in the provenance database. These records can be queried, which allows for a systematic analysis of the experiment in partial, or as a whole, after its completion.

Amazon EC2 was chosen for our case study since it is very reliable and one of the most popular cloud computing environments. Therefore, some of the implementation decisions presented in this Section are specific for Amazon EC2. Current version of SciCumulus was developed using Java version 6.15. The components of the distribution and execution layers were implemented using MPJ (MPI for Java) [52]. Provenance data is stored using PostgreSQL relational database version 8.4.6. SciCumulus uses a shared file system, FUSE-based file system backed by Amazon S3 (s3fs[1]), to manipulate input and output files. To setup a virtual cluster, we used Amazon's Application Programming Interface to create and scale VMs in the cloud. A custom image (AMI) for the execution instances was built (AMI ID: ami-596f4d30).

There are several types of VMs in Amazon EC2, such as micro, large, extra-large, high CPU extra-large instance, and Quadruple Extra Large Instance. We used m3.xlarge and m3.2xlarge, as in Table 1. Each VM instance uses Linux Cent OS 5.5 (64-bit) and it was configured with libraries like MPJ and the bioinformatics applications. All instances had the same image, which was used to execute SciCumulus. We chose US East-N. Virginia location to instantiate all VMs.

Table 1. Characteristics of used VMs

Instance Type	*# cores*	*Physical Processor*
m3.xlarge	***4***	Intel Xeon E5-2670
m3.2xlarge	***8***	Intel Xeon E5-2670

[1] https://code.google.com/p/s3fs/

B. *Experiment Setup*

Our experiments use a dataset of 238 receptors (PDB format) of the CP clan named Peptidase_CA (CL0125) as input, with 42 CP-specific ligands (SDF format) (Table 2), all-out 10,000 receptor-ligands. Receptors and ligands were extracted from RCSB-PDB [35] (Table 2).

TABLE 2. RECEPTORS AND LIGANDS OF CLAN PEPTIDASE_CA (CL0125)

238 Receptor in PDB format
1AEC 1AIM 1ATK 1AU0 1AU2 1AU3 1AU4 1AYU 1AYV 1AYW 1BGO 1BP4 1BQI 1BY8
1CJL 1CPJ 1CQD 1CS8 1CSB 1CTE 1CVZ 1DEU 1EF7 1EWL 1EWM 1EWO 1EWP 1F29
1F2A 1F2B 1F2C 1FH0 1GEC 1GLO 1GMY 1HUC 1ICF 1ITO 1IWD 1JQP 1K3B 1KHP
1KHQ 1M6D 1ME3 1ME4 1MEG 1MEM 1MHW 1MIR 1MS6 1NB3 1NB5 1NL6 1NLJ 1NPZ
1NQC 1O0E 1PAD 1PBH 1PCI 1PE6 1PIP 1POP 1PPD 1PPN 1PPO 1PPP 1Q6K 1QDQ
1S4V 1SNK 1SP4 1STF 1THE 1TU6 1U9Q 1U9V 1U9W 1U9X 1VSN 1XKG 1YAL 1YK7
1YK8 1YT7 1YVB 2ACT 2AIM 2AS8 2ATO 2AUX 2AUZ 2B1M 2B1N 2BDL 2BDZ 2C0Y
2CIO 2DC6 2DC7 2DC8 2DC9 2DCA 2DCB 2DCC 2DCD 2DJF 2DJG 2F1G 2F7D 2FO5
2FQ9 2FRA 2FRQ 2FT2 2FTD 2FUD 2FYE 2G6D 2G7Y 2GHU 2H7J 2HH5 2HHN 2HXZ
2IPP 2NQD 2O6X 2OP3 2OUL 2OZ2 2P7U 2P86 2PAD 2PBH 2PNS 2PRE 2R6N 2R9M
2R9N 2R9O 2VHS 2WBF 2XU1 2XU3 2XU4 2XU5 2YJ2 2YJ8 2YJ9 2YJB 2YJC 3AI8
3BC3 3BCN 3BPF 3BPM 3BWK 3C9E 3CBJ 3CBK 3CH2 3CH3 3D6S 3E1Z 3F5V 3F75
3H6S 3H7D 3H89 3H8B 3H8C 3HD3 3HHA 3HHI 3HWN 3I06 3IEJ 3IMA 3IOQ 3IUT
3IV2 3K24 3K9M 3KFQ 3KKU 3KSE 3KW9 3KWB 3KWN 3KWZ 3KX1 3LFY 3LXS 3MOR
3MPE 3MPF 3N3G 3N4C 3O0U 3O1G 3OF8 3OF9 3OIS 3OVX 3OVZ 3P5U 3P5V 3P5W
3P5X 3PBH 3PDF 3PNR 3QJ3 3QSD 3QT4 3RVV 3RVW 3RVX 3S3Q 3S3R 3TNX 3U8E
3USV 4AXL 4AXM 4DMX 4DMY 4HWY 4K7C 4KLB 4PAD 5PAD 6PAD 7PCK 8PCH 9PAP
42 Ligand in SDF format
042 074 0D6 0E6 0I5 0IW 0LB 0LC 0PC 0QE 186 1RV 1ZB 23Z 25B 2CA 2HP 3FC
424 4MC 4PR 599 59A 73V 74M 75V 76V 77B 78A 935 93N ACE ACT ACY AEM ALD
APD

This input dataset is composed by different sizes of receptors. To evidence the advantages of adaptation with respect to compounds size, we fixed the docking program, *i.e.* independently of the compound size we processed the entire set with AutoDock with and without Vina (Figure 4).

- **Scenario I -** Executes AD4 for docking analyses using the dataset categorized as small receptors 3D structures.
- **Scenario II** - Executes Vina for docking analyses using the dataset categorized as large receptors 3D structures.

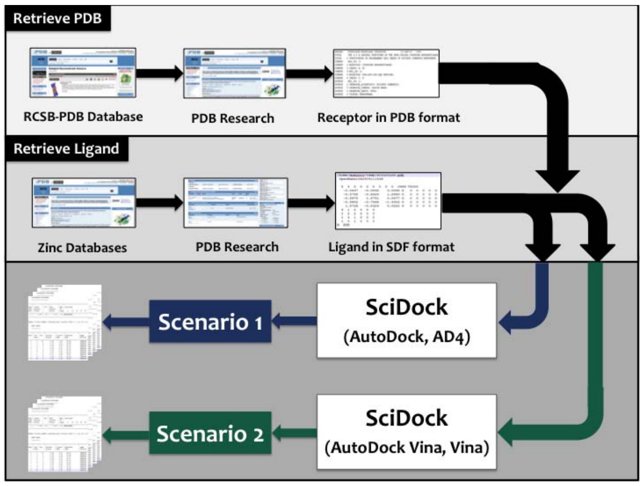


Figure 4. Molecular Docking Experiment Scenarios

For the molecular docking analysis, each input PDB-SDF pair is processed using the following versions of programs: Babel 1.6 [52]; AutoDock 4.2.5.1 [39] MGLTools 1.5.6 python scripts; a custom python script (for activity 6b), AutoGrid 4.2.5 [39]; AutoDock 4.2.5.1 [39]; and AutoDock Vina 1.1.2 [40]. All programs were configured with default parameters.

According to Chang *et al.* [53], there is a clear association between molecular docking predictions of AutoDock (AD4) and AutoDock Vina (Vina). In terms of the reported scoring values (*i.e.* FEB, RMSD), it is expected that conformations assigned by both programs would tend to be similar. Also, it seems that Vina is more scalable in addressing more difficult docking problems (*i.e.* larger, more flexible compounds) than AD4. Moreover Vina's other strengths include streamlined parameters and much faster docking performance.

C. *Performance Evaluation of SciDock*

Before discussing the overall performance of SciDock we analyze the time to execute each activity of the workflow. Analyzing the provenance repository of SciCumulus we show, in Figure 5, the histogram of the execution time for all executions of SciDock activities. This histogram can be generated using a simple SQL query such as:

```
SELECT extract ('epoch' from (t.endtime-t.starttime))
FROM hworkflow w, hactivity a, hactivation t
WHERE w.wkfid = a.wkfid
AND a.actid = t.actid
AND w.wkfid = % ID OF THE WORKFLOW %
ORDER BY t.endtime
```

Based on this query result, it is possible to calculate the average (1,703.5 seconds) and standard deviation (108.3 seconds) for the activities execution time. The main advantage of having such distribution of execution times is that the workflow engine (*i.e.* SciCumulus) is able to distribute compute intensive executions (*i.e.* long term executions) to more powerful VMs. On the other hand, SciCumulus dispatches less intensive executions (*i.e.* short term executions) to less power VMs.

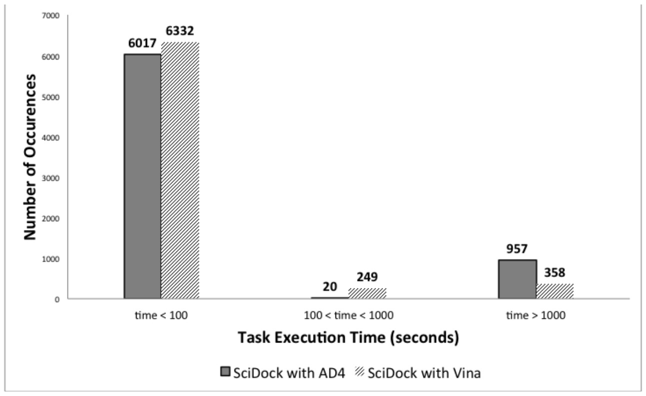


Figure 5. Number of ocurrences of SciDock

Such heterogeneous time distribution leads to a heterogeneous execution time of SciDock activities as shown in Figure 6 where we present the execution time distribution per activity considering 16 cores execution. We note that the last activity of the workflow is the most computing intensive. SciCumulus adapts the execution accordingly. Following, we present performance results of the entire execution of

SciDock with AD4 and Vina for each one of the aforementioned scenarios.

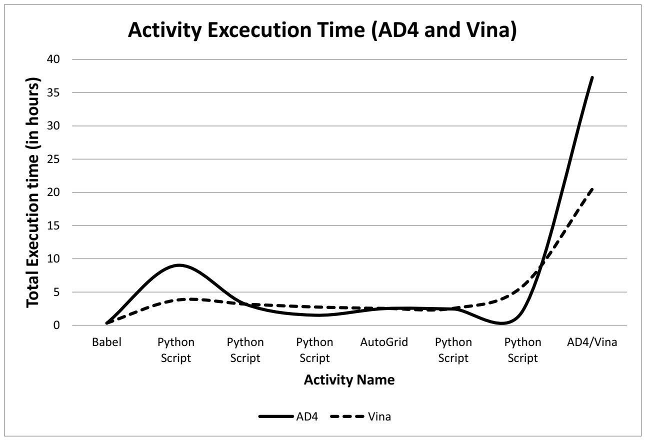


Figure 6. Execution time per Activity

For each activity of SciDock, presented at Section III, we first measure the performance of all programs on a single VM to analyze the local optimization before adding more VMs. We measured the scalability of SciDock using a combination of m3.xlarge and m3.2xlarge VMs up to 32, totalizing 128 virtual cores. As the number of VMs increases (and consequently the number of virtual cores), the Total Execution Time (*i.e.* TET) of SciDock with AD4 and Vina executions decreases (Figure 7).

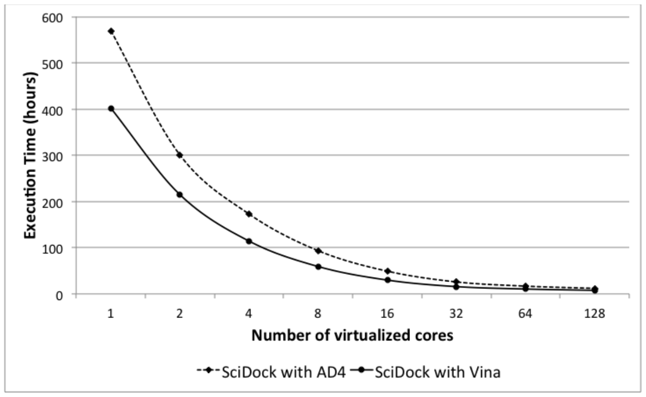


Figure 7. Total execution time of SciDock

The highest performance efficiency was obtained using 2 cores and the gains are very encouraging. For example, when SciDock with AD4 processes 10,000 receptor-ligand pairs, the TET was reduced from 12.5 days (using 2 cores) to 11.9 hours (using 128 cores) and for SciDock with Vina the TET was reduced from approx. 9 days (using 2 cores) to 7.7 hours (using 128 cores).

To evaluate the behavior of performance gains according to the number of virtual cores, we used the speedup metric (Figure 8). In clusters and supercomputers, the speedup value is impacted by serial portions of the code and communication between processors, while in the cloud, besides these factors, we have to consider others such as heterogeneity of the environment, performance fluctuations due to the virtualization and high communication latency [54]. However, even with cloud performance fluctuations, When using 16 cores SciDock is approximately 13 times faster than the best-performing workflow execution on a single core.

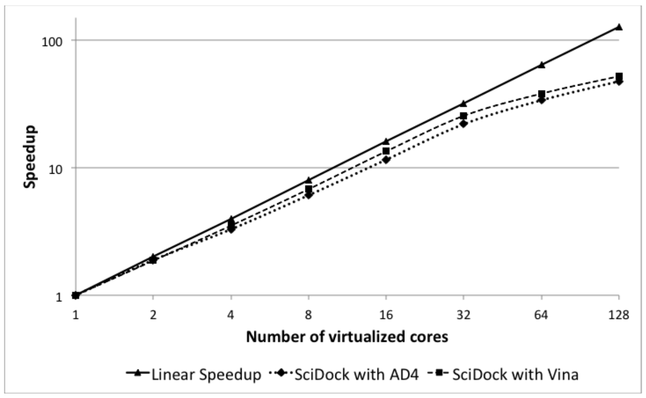


Figure 8. Speedup of SciDock

There is always a gain by adding more virtual cores, from 32 up to 128 cores for both SciDock with AD4 and Vina, but the speedup presents a small degradation in both executions since the VMs are heterogeneous and load balancing becomes more complex, thus introducing more overhead in the activity distribution by SciCumulus. However, from 2 to 32 cores, the speedup was near linear in both SciDock with AD4 and Vina. This result indicates that acquiring more than 32 VMs may not bring the expected benefit, particularly if financial costs are involved (since *m3* VMs in Amazon are expensive types).

We also observed that, when the number of activity executions becomes extremely large, SciCumulus introduces an overhead to manage the distribution of activities. Consequently, some VMs may remain idle. This happens because SciCumulus has a native weighted cost model associated with a greedy scheduling algorithm. When we increase the amount of activities to execute and the number of available VMs, the greedy algorithm tends to require more time to process the scheduling plan at runtime since the search spaces increases exponentially. This behavior can be seen in Figure **9** where the efficiency of SciDock decreases as the number of VMs increases from 32 to 128 cores.

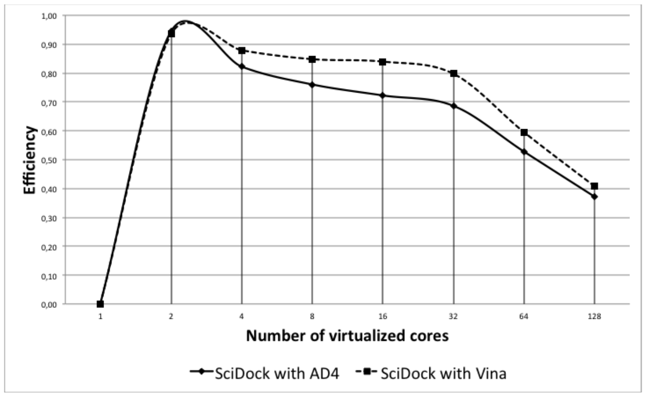


Figure 9. Efficiency of SciDock

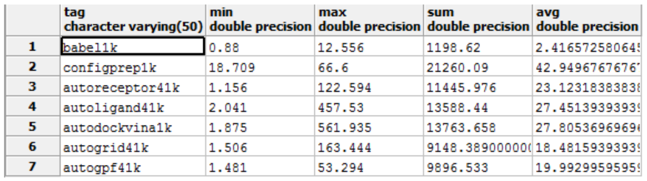
Overall, our performance analyses were facilitated by the information obtained by querying SciCumulus provenance repository after the workflow termination. The provenance repository of SciCumulus is based on the W3C PROV and PROV-Wf [55] models. Due to that, the following query was executed to extract the desired information:

Query 1: "Obtain the TET, statistical averages and biological information related to the SciDock executions".

By querying provenance repository, it is possible to extract the execution time for all executions of activities of SciDock. This information is very useful in special in large-scale experiments, as scientist have the possibility to know how experiments are running, or if any execution fails. Scientist can steer at runtime. It is possible that the error is related with the size of the receptor or ligand or by the presence of an atom (*e.g.* Hg) that cannot be recognized by the docking programs. Figure 10 shows the result of Query 1.

```
SELECT a.tag,
          min(extract ('epoch' from (t.endtime-t.starttime))),
          max(extract ('epoch' from (t.endtime-t.starttime))),
          sum(extract ('epoch' from (t.endtime-t.starttime))),
          avg(extract ('epoch' from (t.endtime-t.starttime)))
FROM hworkflow w, hactivity a, hactivation t
WHERE w.wkfid = a.wkfid
AND a.actid = t.actid
AND w.wkfid =432
GROUP BY a.tag
```

Based on this information scientists evaluate the time required for processing docking for each receptor-ligand pair and analyze the influence of the receptor or ligand sizes in docking execution time. Moreover TET (Figure 7), speedup (Figure 8) and efficiency (Figure 9) can be calculated by using Query 1's resulting data.



	tag character varying(50)	min double precision	max double precision	sum double precision	avg double precision
1	babel1k	0.88	12.556	1198.62	2.41657258064
2	configprep1k	18.709	66.6	21260.09	42.9496767676
3	autoreceptor41k	1.156	122.594	11445.976	23.1231838383
4	autoligand41k	2.041	457.53	13588.44	27.4513939393
5	autodockvina1k	1.875	561.935	13763.658	27.8053696969
6	autogrid41k	1.506	163.444	9148.38900000	18.4815939393
7	autogpf41k	1.481	53.294	9896.533	19.9929959595

Figure 10. Result of the Query 1

By using steering in HPC docking executions, scientists can explore at runtime their experiment executions. Following we present some benefits identified by running SciDock with SciCumulus.

In docking experiments, there is an extensive set of ligands and receptors, to form the dataset (receptor-ligand pairs) for the docking experiment. Both receptor and ligands belonging to CP enzymes were mapped and extracted from the PDB-RCSB database.

After the initial SciDock executions, a particular behavior was observed: there are several activities with abnormal execution time (they remain in looping state) when processing specific ligands. For those activity executions, no error messages were generated by the docking program, since they remain in looping state waiting for the intervention of the scientist. Using SciCumulus we dynamically detected these errors and aborted/adapted them at runtime. Simultaneously, we queried the SciCumulus provenance database by searching all "problematic" ligands that could present the same behavior, thus avoiding generating the same errors in future.

Another similar error captured by querying the SciCumulus provenance database was identified, but now related to receptors. The third activity (Receptor preparation) consumes approximately 10 seconds, but in some cases these executions remained, yet again, in looping state and did not stop until the scientist's interaction. It was observed that inside those receptors the "Hg" molecule is present. Then, after this discovering, it was added one routine in SciCumulus that recognizes the presence of these Hg in receptors. Then those activity executions were identified and aborted before their execution. Specific biological analyses also benefited from SciCumulus's provenance as follows.

D. *Biological Analysis*

NTDs specially affect populations around the world with low socioeconomic status, thus we need new inhibitors for those diseases. CP [30] enzymes are known to have important pathogenicity factors of protozoan parasites, which indicates that they are potential targets for rational antiparasitic drug design. Clan Peptidase_CA (CL0125) is the most representative CPs' clan with 46 members. CL0125 is the chosen CP clan in our experiments. Thus, 238 CP receptors with 42 CP-specific ligands, all-out 10,000 receptor-ligand pairs (

Table **2**) were used to execute the molecular docking and scoring processes. We evaluate the result details for the first 1,000 receptor-ligand pairs (238 CP receptors with 4 CP-specific ligands: 042, 074, 0D6, 0E6) as presented in Table 3.

Table 3 shows results for all molecular docking processes using SciDock with AD4 and Vina. Our evaluation (1) examines values of RMSD, (2) examines Free Energy of Binding (FEB), (3) rank the orientations-conformations according to their FEB [56] scores, and (4) rank the total number of negative FEB or FEB (-). The smaller (most negative) the FEB value, the better the binding of the ligand into the receptor-binding pocket. RMSD values usually with 1.5 or 2 Å (depending on ligand size) have performed successfully. Nevertheless, there is no consensus about what is the reasonable range for FEB and RMSD values [57].

TABLE 3. RESULTS OF MOLECULAR DOCKING PROCESSES FOR SCIDOCK

Ligand	***Total Number of FEB (-)***		***Average FEB (-) (kcal/mol)***		***Average RMSD (Å)***	
	SciDock AD4	***SciDock Vina***	***SciDock AD4***	***SciDock Vina***	***SciDock AD4***	***SciDock Vina***
042	79	91	-4.9	-4.5	55.4	10.3
074	76	83	-5.9	-4.7	57.3	9.1
0D6	65	70	-8.4	-5.7	53.5	9.7
0E6	67	111	-7.2	-5.2	53.1	9.5

Docking outputs (Table 3) show good average of FEB, which means that receptors conformation leads to a favorable ligand association. SciDock with AD4 resulted in FEB scores that range from -4.9 to -8.4 kcal/mol and SciDock with Vina from -4.5 to -5.7 kcal/mol. We also observe that RMSD scores for SciDock with Vina are the lowest. Regarding the total number of negative FEB, it represents favorable receptor-ligand interactions. From a total of 1,000 docking experiments, SciDock with Vina generates 355 FEB (-) and SciDock with AD4 287 FEB (-). The remaining ones (not negative FEB) are related to non-favorable ligand-receptor interactions or others docking experiments that do not converge to a favorable interaction.

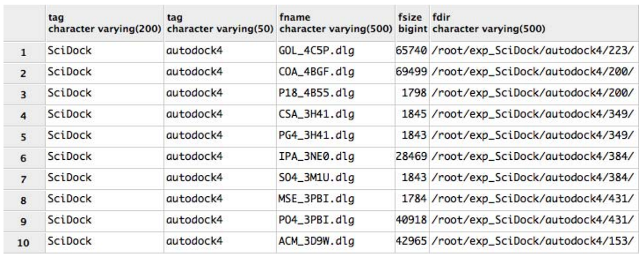
Our results reinforce a previous virtual screening study that compare the programs AutoDock 4 (AD4) and AutoDock Vina (Vina) [53]. According to Chang *et al.* [53], there was a clear association between the predictions from AD4 and Vina. In terms of FEB, it is expected that the conformations reported by both programs would also tend to be similar. Also, it seems that Vina is more scalable in addressing more difficult docking problems (*i.e.* larger, more flexible compounds) than AD4. Moreover, Vina has other strengths such as streamlined parameters and a much faster docking performance. Finally, Vina's authors state that, for their study, docking each library lasted approximately 10 times longer with AD4 when compared to Vina. However, this claim was not based on an experiment designed for HPC.

For all molecular docking processes (Table 3), the average RMSD is not well acceptable (*i.e.* RMSD > 4 Å). We analyzed the top ten best interactions. We observed acceptable FEB values for SciDock with AD4 and Vina. However RMSD for SciDock with AD4 continues to be too high. The best three interactions (receptor-ligands) are 2HHN-0E6, 1S4V-0D6 and 1HUC-0D6, which can be associated to potential drug target for protozoan CPs. However, these receptor-ligand associations should be refined and reinforced using alternative approaches, such as: (i) testing other receptor or ligand conformations; (ii) redocking, molecular dynamics or QSAR analyses; (iii) testing others algorithms, programs or parameters.

Finally, we presented a biological analysis overview focused mainly in the information obtained by querying SciCumulus provenance repository after the workflow ends. By querying the provenance database, it is possible to obtain the TET, statistical averages, and several docking parameters with resultant files as those reported in Query 1. In Query 2, we extract some biological results contained in molecular docking outputs *e.g.* '.dlg' files.

Query 2: "Retrieve the names, sizes and locations of files with the extension '.dlg' (containing docking parameters and results), which were produced for all SciDock workflow executions. Recovering also, which workflow and activities produced those files".

Query 2 is crucial for real-time monitoring of SciDock workflow execution. The result of Query 2 is presented in Figure 11, and allows scientists to find files that are being generated by SciDock. Query 2 helps biological analysis in real-time. For instance, scientists may use graphical tools to verify the resulting structure obtained after the docking (contained in '*.dlg*' files). These '*.dlg*' files contain the binding affinity values between ligands and target receptors and also which one is the best ligand (*e.g.* E06) for each receptor (*e.g.* 2HHN), based on a prediction method for positioning the ligand to the binding site. One example of the visualization of this '*.dlg'* file is presented on Figure 12.



	tag character varying(200)	tag character varying(50)	fname character varying(500)	fsize bigint	fdir character varying(500)
1	SciDock	autodock4	GOL_4C5P.dlg	65740	/root/exp_SciDock/autodock4/223/
2	SciDock	autodock4	COA_4BGF.dlg	69499	/root/exp_SciDock/autodock4/200/
3	SciDock	autodock4	P18_4B55.dlg	1798	/root/exp_SciDock/autodock4/200/
4	SciDock	autodock4	CSA_3H41.dlg	1845	/root/exp_SciDock/autodock4/349/
5	SciDock	autodock4	PG4_3H41.dlg	1843	/root/exp_SciDock/autodock4/349/
6	SciDock	autodock4	IPA_3NE0.dlg	28469	/root/exp_SciDock/autodock4/384/
7	SciDock	autodock4	SO4_3M1U.dlg	1843	/root/exp_SciDock/autodock4/384/
8	SciDock	autodock4	MSE_3PBI.dlg	1784	/root/exp_SciDock/autodock4/431/
9	SciDock	autodock4	PO4_3PBI.dlg	40918	/root/exp_SciDock/autodock4/431/
10	SciDock	autodock4	ACM_3D9W.dlg	42965	/root/exp_SciDock/autodock4/153/

Figure 11. Result of Query 2

Without querying the provenance database with Query 2, scientists would need to browse all directories manually and search which pairs were docked successfully. Then they would need to separate and open these files to extract the information of molecular docking process. The provenance database stores all this data and its relationships on a structured model. Thus it simplifies the querying process and allows for long-term analyses over experimental data.

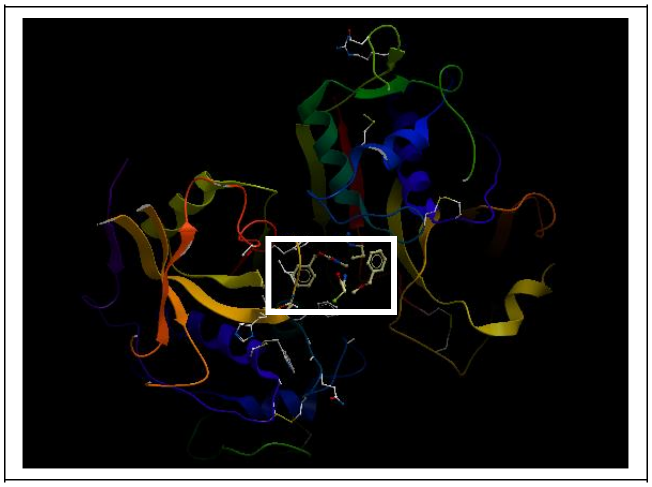


Figure 12. 3D structure of the complex 2HHN-0E6 obtained with SciDock. Receptor 2HHN and into the white box the best ligant 0E6

Figure 12 shows the structure of the receptor 2HHN with the ligand E06 with the highest affinity obtained by SciDock. The 2HHN receptor is the "Cathepsin S in complex with non-covalent arylaminoethyl amide", which have been implicated in a variety of important biological events and have also been validated as drug targets of high promise [58]. We demonstrate that, using SciDock, it is possible to test *in silico* a set of receptors and ligands of interest for drug target candidates. Furthermore, performance and biological data can be mapped and queried via SQL.

VI. Final Remarks and Future Work

Molecular docking workflows executed by parallel SWfMS and HPC environments can manage a large volume of receptors and ligands comparisons, reducing the long processing time for molecular docking analyses. In this paper, we proposed the SciDock workflow to execute and manage molecular docking data-intensive experiments aiming at discovering alternative drug target for NTD treatments. SciDock was executed with SciCumulus in Amazon EC2 using parallel processing.

Our experiment evaluated 10,000 receptor-ligand pairs related to proteases enzymes belonging to protozoan genomic data. In the first 1,000 receptor-ligands pairs, SciDock already detected 287 and 355 favorable receptor-ligand interactions using AD4 and Vina, respectively. These interactions represent potential proteases drug targets for protozoan treatments.

SciDock generated 140,000 workflow activity executions (10,000 executions of the 7 activities of 2 workflows) and data files, producing 600 gigabytes of data for each workflow execution. By analyzing the overall performance, through the provenance database, we state that SciDock obtained significant gains with AD4 and Vina. For example, executions with 32 cores reach performance improvements up to 95.4% for SciDock with AD4 and 96.1% for SciDock with Vina. Based on the TET, speedup, efficiency and molecular docking scoring values, we observe that SciDock with Vina performs better than SciDock with AD4.

Analyzing the results, we may conclude that as we increase the 'cover diversity space of compounds', this positively influences the chance of identifying new drugs. Thus, using 1,000 compound pairs involving an entire family of enzymes, we detected 287 and 355 favorable receptor-ligand interactions using SciDock with AD4 and Vina. This scenario would have been impossible with a reduced number of receptor-ligand pairs, especially if we use only one receptor/ligand or if the input sample belongs to the complementary space in which no favorable interaction was found (*i.e.* 1,000 less the 287 + 355 interactions we found).

Overall, the overhead imposed by the executions of SciDock with SciCumulus is compensated by the advantages of data parallelism without too much effort from scientists. SciDock results provide evidence that large computations involving MD experiments can benefit from SciCumulus in HPC clouds as verified in previous publications [17,47–49].

Finally, results presented in this paper can be extrapolated to the development of workflows in other areas that also require the exploration of large amounts of data. As future work, we plan to model other computing-intensive CADD workflows (*e.g.* molecular modeling, dynamics, ligand-based and structure-based virtual screening, 2D and 3D QSAR) to explore complete protozoan genomes of actual interest and to search new candidate drug target enzymes.

Acknowledgment

The work was partially funded by the Brazilian funding agencies CAPES, FAPERJ and CNPq.

References

[1] S. Myers and A. Baker, 2001, Drug discovery--an operating model for a new era, *Nature biotechnology*, v. 19, n. 8 (Aug.), p. 727–730.

[2] H. Moses 3rd, E.R. Dorsey, D.H.M. Matheson, and S.O. Thier, 2005, Financial anatomy of biomedical research, *JAMA: the journal of the American Medical Association*, v. 294, n. 11 (Sep.), p. 1333–1342.

[3] D. Loughney, B.L. Claus, and S.R. Johnson, 2011, To measure is to know: an approach to CADD performance metrics, *Drug Discovery Today*, v. 16 (Jul.), p. 548–554.

[4] V. Lounnas, T. Ritschel, J. Kelder, R. McGuire, R.P. Bywater, and N. Foloppe, 2013, Current progress in Structure-Based Rational Drug Design marks a new mindset in drug discovery, *Computational and Structural Biotechnology Journal*, v. 5, n. 6 (Feb.)

[5] L.O. Sillerud and R.S. Larson, 2012, Advances in nuclear magnetic resonance for drug discovery, *Methods in molecular biology (Clifton, N.J.)*, v. 910, p. 195–266.

[6] E. Deelman, D. Gannon, M. Shields, and I. Taylor, 2009, Workflows and e-Science: An overview of workflow system features and capabilities, *Future Generation Computer Systems*, v. 25, n. 5, p. 528–540.

[7] J. Wozniak, T. Armstrong, K. Maheshwari, E. Lusk, D. Katz, M. Wilde, and I. Foster, 2012, Turbine: A distributed-memory dataflow engine for extreme-scale many-task applications, In: *Proceeding of 1st International workshop on Scalable Workflow Enactment Engines and Technologies*

[8] E. Deelman, G. Mehta, G. Singh, M.-H. Su, and K. Vahi, 2007, "Pegasus: Mapping Large-Scale Workflows to Distributed Resources", *Workflows for e-Science*, Springer, p. 376–394.

[9] M. Abouelhoda, S. Issa, and M. Ghanem, 2012, Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support, *BMC Bioinformatics*, v. 13, p. 77.

[10] D. Oliveira, E. Ogasawara, F. Baião, and M. Mattoso, 2010, SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows, In: *3rd International Conference on Cloud Computing*, p. 378–385

[11] J. Freire, D. Koop, E. Santos, and C.T. Silva, 2008, Provenance for Computational Tasks: A Survey, *Computing in Science and Engineering, v.10*, n. 3, p. 11–21.

[12] C. Liao, M. Sitzmann, A. Pugliese, and M.C. Nicklaus, 2011, Software and resources for computational medicinal chemistry, *Future Medicinal Chemistry*, v. 3, n. 8 (Jun.), p. 1057–1085.

[13] M. Mattoso, K. Ocaña, F. Horta, J. Dias, E. Ogasawara, V. Silva, D. de Oliveira, F. Costa, and I. Araújo, 2013, User-steering of HPC workflows: state-of-the-art and future directions, In: *Proceedings of the 2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, p. 1–6

[14] D. Oliveira, F. Baião, and M. Mattoso, 2010, "Towards a Taxonomy for Cloud Computing from an e-Science Perspective", *Cloud Computing: Principles, Systems and Applications*, Nick Antonopoulos and Lee Gillam edHeidelberg: Springer-Verlag

[15] J.T. Dudley and A.J. Butte, 2010, In silico research in the era of cloud computing, *Nature Biotechnology*, v. 28, n. 11 (Nov.), p. 1181–1185.

[16] V.C. Emeakaroha, P. \Labaj, M. Maurer, I. Brandic, and D.P. Kreil, 2011, Optimizing bioinformatics workflows for data analysis using cloud management techniques, In: *Proceedings of the 6th workshop on Workflows in support of large-scale science*, p. 37–46

[17] D. Oliveira, K.A.C.S. Ocaña, E. Ogasawara, J. Dias, J. Gonçalves, F. Baião, and M. Mattoso, 2013, Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows, *Future Generation Computer Systems*, v. 29, n. 7 (Sep.), p. 1816–1825.

[18] S.V. Angiuoli, M. Matalka, A. Gussman, K. Galens, M. Vangala, D.R. Riley, C. Arze, J.R. White, O. White, et al., 2011, CloVR: a virtual machine for automated and portable sequence analysis from the desktop using cloud computing, *BMC Bioinformatics*, v. 12, p. 356.

[19] J.T. Dudley, Y. Pouliot, R. Chen, A.A. Morgan, and A.J. Butte, 2010, Translational bioinformatics in the cloud: an affordable alternative, *Genome Medicine*, v. 2, n. 8, p. 51.

[20] Google Developers, 2012. Android Cloud to Device Messaging Framework - Android — Google Developers. Available at: https://developers.google.com/android/c2dm/. Accessed: 27 Aug 2012.

[21] Amazon EC2, 2010, *Amazon Elastic Compute Cloud (Amazon EC2)*, http://aws.amazon.com/ec2/.

[22] A. Wolf, M. Hofmann-Apitius, M. Ghanem, N. Azam, D. Kalaitzopoulos, K. Yu, and V. Kasam, 2009, DockFlow - a prototypic PharmaGrid for virtual screening integrating four different docking tools, *Studies in health technology and informatics*, v. 147, p. 3–12.

[23] R. De Paris, F.A. Frantz, O.N. de Souza, and D.D.A. Ruiz, 2013, wFReDoW: a cloud-based web environment to handle molecular docking simulations of a fully flexible receptor model, *BioMed research international*, v. 2013, p. 469363.

[24] S.R. Ellingson and J. Baudry, 2014, High-throughput virtual molecular docking with AutoDockCloud: High-throughput virtual molecular docking with AutoDockCloud, *Concurrency and Computation: Practice and Experience*, v. 26, n. 4 (Mar.), p. 907–916.

[25] M. AbdelBaky, H. Kim, I. Rodero, and M. Parashar, 2012, Accelerating MapReduce Analytics Using CometCloud, p. 447–454

[26] B.C. Pearce, D.R. Langley, J. Kang, H. Huang, and A. Kulkarni, 2009, E-novo: an automated workflow for efficient structure-based lead optimization, *Journal of chemical information and modeling*, v. 49, n. 7 (Jul.), p. 1797–1809.

[27] Y. Zhao and M.F. Sanner, 2007, FLIPDock: docking flexible ligands into flexible receptors, *Proteins*, v. 68, n. 3 (Aug.), p. 726–737.

[28] R. Spitzer and A.N. Jain, 2012, Surflex-Dock: Docking benchmarks and real-world application, *Journal of Computer-Aided Molecular Design*, v. 26, n. 6 (May.), p. 687–699.

[29] N.D. Prakhov, A.L. Chernorudskiy, and M.R. Gainullin, 2010, VSDocker: a tool for parallel high-throughput virtual screening using AutoDock on Windows-based computer clusters, *Bioinformatics*, v. 26, n. 10 (Apr.), p. 1374–1375.

[30] I. Berdowska, 2004, Cysteine proteases as disease markers, *Clinica chimica acta; international journal of clinical chemistry*, v. 342, n. 1-2 (Apr.), p. 41–69.

[31] J.A.L. Lindoso and A.A.B.P. Lindoso, 2009, Neglected tropical diseases in Brazil, *Revista do Instituto de Medicina Tropical de São Paulo*, v. 51 (Oct.), p. 247–253.

[32] Dávila and Kary A. C. S. Ocaña, 2011, Phylogenomics-Based Reconstruction of Protozoan Species Tree, *Evolutionary Bioinformatics* (Jul.), p. 107.

[33] G.M. Morris and M. Lim-Wilby, 2008, Molecular docking, *Methods in Molecular Biology (Clifton, N.J.)*, v. 443, p. 365–382.

[34] R.D. Taylor, P.J. Jewsbury, and J.W. Essex, 2002, A review of protein-small molecule docking methods, *Journal of computer-aided molecular design*, v. 16, n. 3 (Mar.), p. 151–166.

[35] P.W. Rose, C. Bi, W.F. Bluhm, C.H. Christie, D. Dimitropoulos, S. Dutta, R.K. Green, D.S. Goodsell, A. Prlic, et al., 2013, The RCSB Protein Data Bank: new resources for research and education, *Nucleic acids research*, v. 41, n. Database issue (Jan.), p. D475–482.

[36] H. Kunz, 2002, Emil Fischer—Unequalled Classicist, Master of Organic Chemistry Research, and Inspired Trailblazer of Biological Chemistry, *Angewandte Chemie International Edition*, v. 41, n. 23 (Dec.), p. 4439–4451.

[37] W.L. Jorgensen, 1991, Rusting of the lock and key model for protein-ligand binding, *Science (New York, N.Y.)*, v. 254, n. 5034 (Nov.), p. 954–955.

[38] D.B. Kitchen, H. Decornez, J.R. Furr, and J. Bajorath, 2004, Docking and scoring in virtual screening for drug discovery: methods and applications, *Nature reviews. Drug discovery*, v. 3, n. 11 (Nov.), p. 935–949.

[39] G.M. Morris, R. Huey, W. Lindstrom, M.F. Sanner, R.K. Belew, D.S. Goodsell, and A.J. Olson, 2009, AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility, *Journal of Computational Chemistry*, v. 30, n. 16 (Dec.), p. 2785–2791.

[40] O. Trott and A.J. Olson, 2010, AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading, *Journal of computational chemistry*, v. 31, n. 2 (Jan.), p. 455–461.

[41] B. Kramer, M. Rarey, and T. Lengauer, 1997, CASP2 experiences with docking flexible ligands using FlexX, *Proteins*, v. Suppl 1, p. 221–225.

[42] R.A. Friesner, J.L. Banks, R.B. Murphy, T.A. Halgren, J.J. Klicic, D.T. Mainz, M.P. Repasky, E.H. Knoll, M. Shelley, et al., 2004, Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy, *Journal of medicinal chemistry*, v. 47, n. 7 (Mar.), p. 1739–1749.

[43] M.L. Verdonk, J.C. Cole, M.J. Hartshorn, C.W. Murray, and R.D. Taylor, 2003, Improved protein-ligand docking using GOLD, *Proteins*, v. 52, n. 4 (Sep.), p. 609–623.

[44] K. Michael and K.W. Miller, 2013, Big Data: New Opportunities and New Challenges [Guest editors' introduction], *Computer*, v. 46, n. 6 (Jun.), p. 22–24.

[45] K.D. Pruitt, T. Tatusova, W. Klimke, and D.R. Maglott, 2009, NCBI Reference Sequences: current status, policy and new initiatives, *Nucleic Acids Research*, v. 37, n. Database issue (Jan.), p. D32–D36.

[46] K. Ginalski, 2006, Comparative modeling for protein structure prediction, *Current opinion in structural biology*, v. 16, n. 2 (Apr.), p. 172–177.

[47] K.A.C.S. Ocaña, F. Oliveira, J. Dias, E. Ogasawara, and M. Mattoso, 2013, Designing a parallel cloud based comparative genomics workflow to improve phylogenetic analyses, *Future Generation Computer Systems*, v. 29, n. 8, p. 2205–2219.

[48] K.A.C.S. Ocaña, D. de Oliveira, F. Horta, J. Dias, E. Ogasawara, and M. Mattoso, 2012, "Exploring Molecular Evolution Reconstruction Using a Parallel Cloud-based Scientific Workflow", *Advances in Bioinformatics and Computational Biology*, , chapter 7409, Berlin, Heidelberg: Springer, p. 179–191.

[49] K.A.C.S. Ocaña, D. Oliveira, E. Ogasawara, A.M.R. Dávila, A.A.B. Lima, and M. Mattoso, 2011, "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes", In: O. Norberto de Souza, G.P. Telles, and M. Palakal, eds., *Advances in Bioinformatics and Computational Biology*, , chapter 6832, Berlin, Heidelberg: Springer, p. 66–70.

[50] D. Oliveira, K. Ocaña, F. Baião, and M. Mattoso, 2012, A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds, *Journal of Grid Computing*, v. 10, n. 3, p. 521–552.

[51] E. Ogasawara, J. Dias, D. Oliveira, F. Porto, P. Valduriez, and M. Mattoso, 2011, An Algebraic Approach for Data-Centric Scientific Workflows, *Proc. of VLDB Endowment*, v. 4, n. 12, p. 1328–1339.

[52] N.M. O'Boyle, M. Banck, C.A. James, C. Morley, T. Vandermeersch, and G.R. Hutchison, 2011, Open Babel: An open chemical toolbox, *Journal of Cheminformatics*, v. 3, n. 1, p. 33.

[53] M.W. Chang, C. Ayeni, S. Breuer, and B.E. Torbett, 2010, Virtual Screening for HIV Protease Inhibitors: A Comparison of AutoDock 4 and Vina, *PLoS ONE*, v. 5, n. 8 (Aug.), p. e11955.

[54] L.M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, 2009, A break in the clouds: towards a cloud definition, *SIGCOMM Comput. Commun. Rev.*, v. 39, n. 1, p. 50–55.

[55] P. Missier, K. Belhajjame, and J. Cheney, 2013, The W3C PROV family of specifications for modelling provenance metadata, In: *Proceedings of the 16th International Conference on Extending Database Technology*, p. 773–776

[56] G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, and A.J. Olson, 1998, Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function, *Journal of Computational Chemistry*, v. 19, n. 14 (Nov.), p. 1639–1662.

[57] R. Wang, Y. Lu, X. Fang, and S. Wang, 2004, An extensive test of 14 scoring functions using the PDBbind refined set of 800 protein-ligand complexes, *Journal of chemical information and computer sciences*, v. 44, n. 6 (Dec.), p. 2114–2125.

[58] J.C. Mottram, M.J. Helms, G.H. Coombs, and M. Sajid, 2003, Clan CD cysteine peptidases of parasitic protozoa, *Trends in Parasitology*, v. 19, n. 4 (Apr.), p. 182–187.