

Data Parallelism in Bioinformatics Workflows Using Hydra

Fábio Coutinho¹, Eduardo Ogasawara¹, Daniel de Oliveira¹,
Vanessa Braganholo¹, Alexandre A. B. Lima¹, Alberto M. R. Dávila², Marta Mattoso¹

¹Federal University of Rio de Janeiro - Rio de Janeiro – Brazil

²Oswaldo Cruz Institute – FIOCRUZ – Rio de Janeiro Brazil

{fabio,ogasawara,danielc,assis,marta}@cos.ufrj.br,

braganholo@dcc.ufrj.br, davila@fiocruz.br

ABSTRACT

Large scale bioinformatics experiments are usually composed by a set of data flows generated by a chain of activities (programs or services) that may be modeled as scientific workflows. Current Scientific Workflow Management Systems (SWfMS) are used to orchestrate these workflows to control and monitor the whole execution. It is very common in bioinformatics experiments to process very large datasets. In this way, data parallelism is a common approach used to increase performance and reduce overall execution time. However, most of current SWfMS still lack on supporting parallel executions in high performance computing (HPC) environments. Additionally keeping track of provenance data in distributed environments is still an open, yet important problem. Recently, Hydra middleware was proposed to bridge the gap between the SWfMS and the HPC environment, by providing a transparent way for scientists to parallelize workflow executions while capturing distributed provenance. This paper analyzes data parallelism scenarios in bioinformatics domain and presents an extension to Hydra middleware through a specific cartridge that promotes data parallelism in bioinformatics workflows. Experimental results using workflows with BLAST show performance gains with the additional benefits of distributed provenance support.

Categories and Subject Descriptors

H.4.1 [Information Systems Applications]: Workflow management. H.2.8 [Database Management]: Scientific. I.6.7 [Simulation and Modeling]: Simulation Support Systems Environments

General Terms

Performance, Measurement, Experimentation

Keywords

Scientific Workflows, Provenance, Bioinformatics.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HPDC'10, June 20–25, 2010, Chicago, Illinois, USA.

Copyright 2010 ACM 978-1-60558-942-8/10/06 ...\$10.00.

1. INTRODUCTION

Over the last years, bioinformatics scientists have dealt with large amounts of data in their research. Such research is based on several *in silico* experiments that make use of computational simulations [1] and demands large computing power. In most bioinformatics laboratories and genomics research centers, experiments are composed by programs or services based on solid methods and algorithms such as BLAST or MAFFT [2]. These scientific experiments are being modeled as scientific workflows [3-5]. Scientific workflows are usually managed by complex engines called Scientific Workflow Management Systems (SWfMS). There are a huge amount of SWfMS available [6-11] and some of them have tools devoted to bioinformatics such as Taverna [7].

Bioinformatics activities within a workflow often involve large scale data processing and many data conversion using shimming activities [12]. Each one of the workflow activities may receive huge amounts of data to be processed. This large scale processing may be unfeasible if scientists do not run their experiments in high performance environments and using parallelism techniques such as data fragmentation [13,14]. Data fragmentation occurs when a piece of input data is broken up into many parts (also named fragments) that are not inter-dependent. This way, each one of the fragments can be individually processed, producing its own output that must be merged with other partial outputs to produce the final result.

However, designing data fragmentation and controlling parallel execution of many tasks is far from trivial. In fact, it may represent a barrier to the execution of the entire experiment. Experiments that require data fragmentation may involve fragmentation of hundreds of input files and in many cases they are based on heterogeneous formats. Although they are natural candidates for parallel processing, the total number of (independent or dependent) tasks and available processors makes this computational model complex to be managed in an *ad-hoc* way. Recently, the Many Tasks Computing (MTC) paradigm was proposed [15] to address the problem of executing multiple parallel tasks in multiple processors.

In addition, in order to be accepted by scientific community as a valid scientific experiment, it must be able to be reproducible under the same conditions in different environments. To reproduce an experiment, scientists must analyze all information related to it. Provenance data [16] may be the source of the information related to the experiment, *i.e.* its definition and all data produced by different workflow

executions is fundamental for the experiment to be considered consistent and “scientific”.

Provenance information contains the complete history of the experiment (parameters values, produced data, intermediate data, execution times, etc.) or its lineage. Thus, it is fundamental that provenance data can be captured and stored when scientists are executing a scientific experiment, so they can analyze it *a posteriori*.

Hydra [17] is a middleware that provides a set of components to be included on the workflow specification of any SWfMS to control parallelism of activities following the MTC paradigm. Hydra is based on a homogeneous cluster environment and relies on a centralized scheduler (such as Torque [18]). Using Hydra, the MTC parallelism strategy can be registered, reused, and provenance may be uniformly gathered during the execution of workflows. In previous work [17,19], parameter sweep mechanisms were developed and explored using Hydra. However, Hydra still lacks on data parallelism mechanisms coupled to provenance facilities.

In this paper, we explore data parallelism in bioinformatics workflows using Hydra. Fragmentation techniques were encapsulated in Hydra cartridges to promote data parallelism. We have evaluated our approach by executing experimental studies parallelizing the BLAST activity within a scientific workflow.

This paper is organized in five sections besides this introduction. Section 2 presents a brief explanation about Hydra architecture and main components. It also details the data fragmentation cartridge implemented for the experimental studies. Section 3 presents the experiment used to evaluate data parallelism in Hydra while Section 4 brings experiment results and some performance analysis. Section 5 presents related work and finally Section 6 concludes the paper and points some future work.

2. HYDRA

This section briefly describes the middleware used as the basic computational infra-structure to implement data parallelism in bioinformatics workflows. Hydra [17] is a distributed middleware that aims at distributing, controlling and monitoring the execution of scientific workflow activities (or even entire scientific workflows) in cluster environments. Hydra controls the parallel execution of workflow activities in different nodes of a cluster. It is composed by a two-layer architecture: (i) Hydra Client Components that are in charge of starting the distributed and parallel execution using a local SWfMS such as Kepler [6], Taverna [7] and VisTrails [8], and (ii) Hydra MTC Layer that manages the execution of distributed activities. Hydra is based on the conception of cartridges [20], which means that each component can be changed as needed. This section gives an overview of both Hydra architecture and the formalism for data fragmentation, which is the main focus of this work.

2.1 Data Parallelism in Scientific Workflows

Our parallelism formalism is adapted from [13,14]. A given scientific workflow W is composed by a set of activities chained in a coherent flow to represent one scientific model. A workflow W is represented by a quadruple (A, Pt, I, O) where: $A = \{a_1, a_2, \dots, a_n\}$ is a set of activities that composes the

workflow W ; $I = \{i_1, i_2, \dots, i_m\}$ is a set of input data to be consumed; $Pt = \{pt_1, pt_2, \dots, pt_r\}$ is a set of parameters of W and $O = \{o_1, o_2, \dots, o_n\}$ is a set of output data.

Hydra architecture is prepared to support two types of parallelism: data parallelism and parameter sweep parallelism [21]. Each one of those types may represent a barrier for the scientist to control, gather and register workflow provenance, since they require a great effort and discipline to manage too much information when executed in an ad-hoc manner. The focus of this paper is on exploring data parallelism with Hydra. This type of parallelism is characterized by the simultaneous execution of one activity a_i of a given workflow W , where each execution of a_i processes a subset of the entire input data. This parallelism can be achieved by replicating the activity a_i in all nodes of the MTC environment (in our case, a cluster), with each node having a specific subset of the entire input data, as presented in Figure 1.

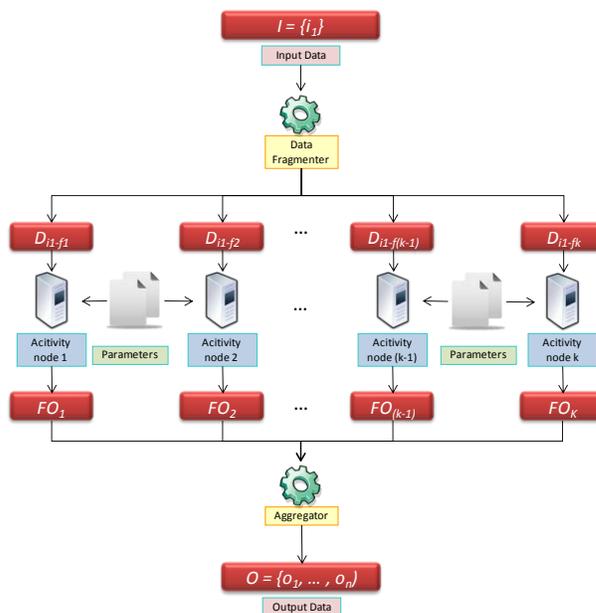


Figure 1 Data Parallelism in Hydra

2.2 Hydra Conceptual Architecture

Figure 2 presents the Hydra conceptual architecture on top of two main layers. The first layer (left-hand box in Figure 2), comprises local components, which run at the desktop of the scientist invoked by local SWfMS. The second layer (right-hand box) is composed by components that deal with activity distribution and parallelization, which are run on HPC. The numbers alongside the arrows in Figure 2 denote the execution sequence of the architecture components. Hydra client components aim at communicating with the Hydra MTC layer to promote data distribution and parallelization transparently. Hydra can be used by any local SWfMS [6-11,22]. Hydra client layer has five main components: Setup, Uploader, Dispatcher, Gatherer and Downloader.

The Setup is the component responsible for offering to the scientist a setup mode to configure the parallelism, explicating the model of parallelism. In the case of data parallelism, scientists have to inform metadata such as data files that need to be transferred to the MTC environment, local files that must

be uploaded to the template data directory, and management of distributed applications. These configurations are used by Hydra MTC layer to configure the MTC environment and to control the type of parallelism during workflow execution. It is worth noticing that the setup is usually done once for each type of experiment. Additional executions do not need to execute this step.

The Uploader is responsible for transferring data from the local desktop to the distributed environment while the Downloader is responsible for downloading data from the cluster environment. It is important to highlight that to improve the impact of transferring data, the Downloader and the Uploader components were extended to compress files during transferring. Each one of the input and output files may be compressed before being transferred over the network as presented in the pipeline of Figure 3. The Dispatcher is responsible for starting and monitoring the execution process of the distributed activity on the MTC environment, while Gatherer collects provenance data from the distributed environment.

Different from Hydra Client Components, the Hydra MTC layer is itself distributed and aims at controlling and managing the distributed execution of workflow activities. It is composed by five main components: MUX, Parameter

Sweeper, Data Fragmenter, Dispatcher/Monitor and Data Analyzer.

MUX is responsible for interfacing between the Hydra Client Components and the distributed layer. It verifies which type of parallelism was chosen and invokes either Parameter Sweeper or Data Fragmenter component. Parameter Sweeper handles the combinations of the parameters received by the client workflow. The Data Fragmenter, which is the focus of this paper, fragments the original data to be distributed for each instance of the activity/workflow being parallelized. It is important to highlight that Data Fragmenter is a problem-dependent component (like a Map function in Map-Reduce approach [23,24]) thus it needs to be customized to fragment different types of input data, as detailed in Section 2.3.

The Dispatcher/Monitor component is responsible for invoking external schedulers (such as Torque [18] or Falcon [25]) to execute the activity in the distributed environment. And finally, the Data Analyzer component is responsible for combining the final results performing an aggregation function (like a Reduce function in Map-Reduce [23,24]) in order to organize the final results to be transferred to the client layer (SWfMS). The next section details the proposal of the fragmentation cartridge to promote data parallelism in Hydra.

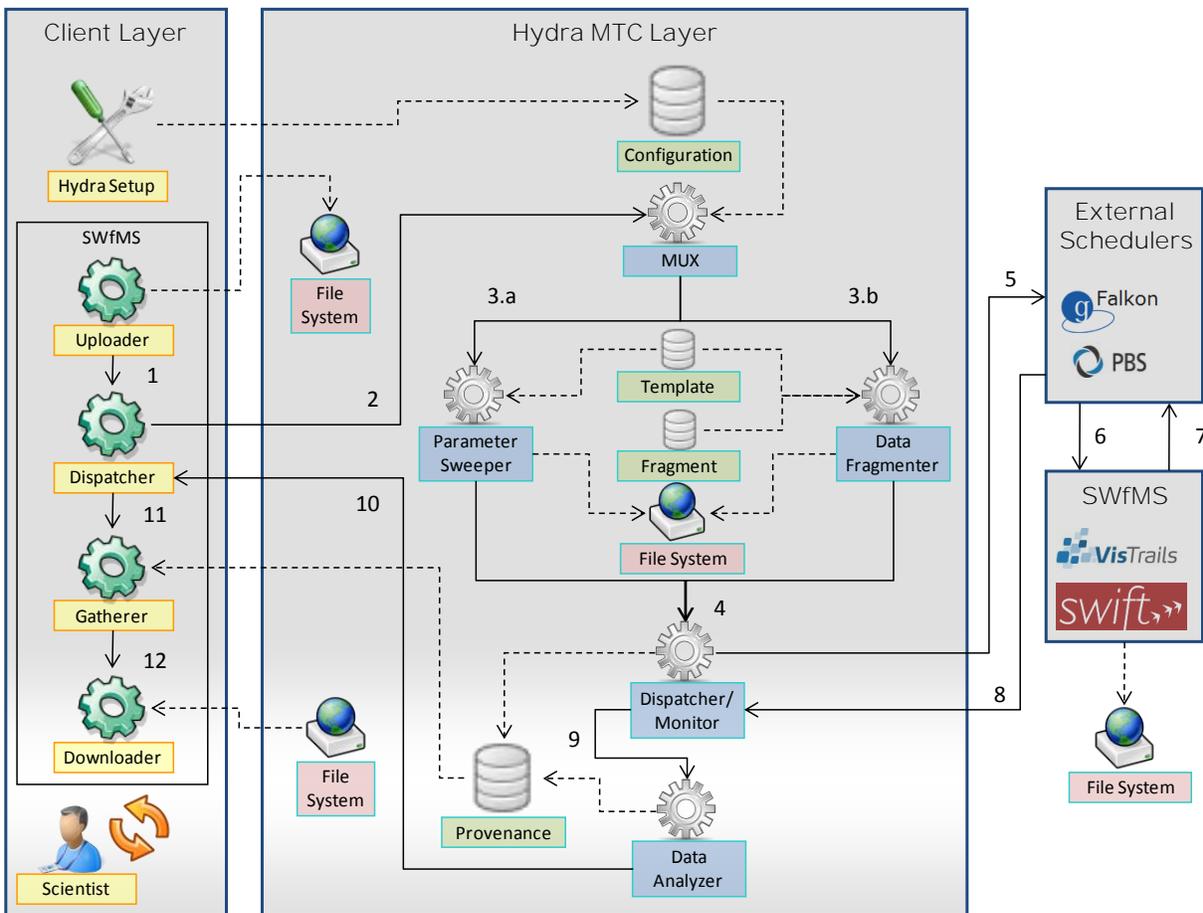


Figure 2 Hydra Conceptual Architecture

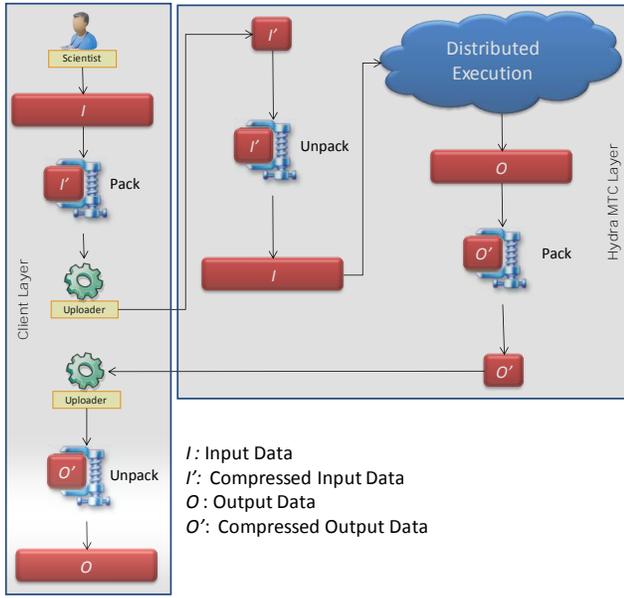


Figure 3 File compressing pipeline in Hydra

2.3 Data Fragmentation Cartridge for Bioinformatics Domain

In order to promote data parallelism in bioinformatics

workflows we present here a specific cartridge that we have developed to fragment FASTA files [26] that represents either DNA or protein sequences, in which nucleotides or amino acids are represented using single-letter codes. Each one of the fragments is generated following a specific fragmentation rule. This fragmentation rule is the one used to generate data fragments, thus separating the sequences to be analyzed. The data parallelism cartridge is based on the program FASTASplitter, and it was encapsulated in a problem-dependent cartridge.

The fragmentation algorithm is graphically presented in Figure 4. It is composed by two main phases: (a) Naïve fragmentation, and (b) Adjustment of data fragments. The Naïve fragmentation phase is responsible for fragmenting the input FASTA file in parts ($f_1, f_2, f_3, \dots, f_{n-1}, f_n$) with the same data size. The number of fragments is defined by the scientist during the desktop SWfMS invocation. Let us show an example of the file *aa-pfalciparum2.fasta* that has 246 MB to be fragmented in 256 fragments. The fragmentation cartridge fragments the FASTA file in 256 fragments ($f_1, f_2, f_3, \dots, f_{255}, f_{256}$) with 0.96MB each. However, these fragments may not have any biological meaning, since a specific DNA sequence may be split in more than one fragment resulting in incomplete or truncated sequences which is not acceptable.

To avoid this inconsistency, the second phase of the algorithm adjusts the fragments in order to assure that fragments contain

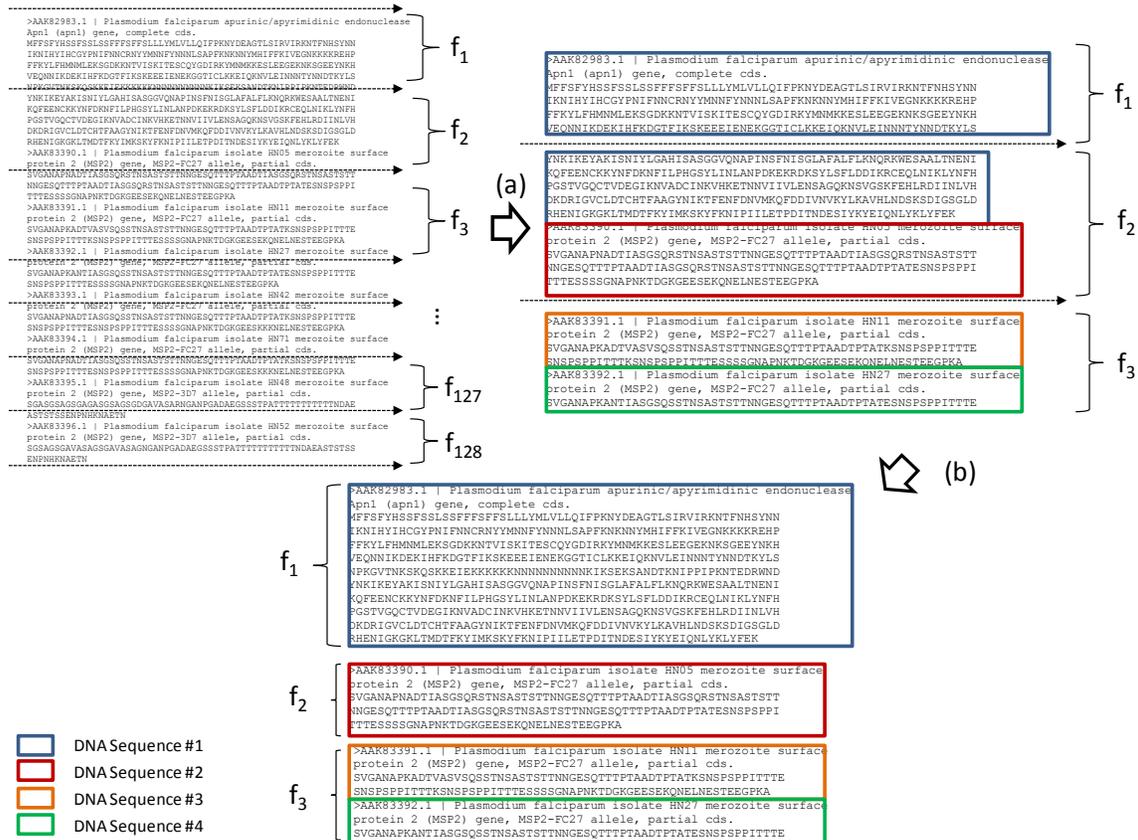


Figure 4 - (a) Naïve Fragmentation (b) Adjustment of Data Fragments

complete DNA sequences. To do so, it rearranges the data in the fragments, moving sequences from one fragment to another. In Figure 4(a) we may see that the DNA Sequence #1 was initially split in fragments *f1* and *f2*. After the adjustment phase in Figure 4 (b), part of the DNA sequence #1 that was in fragment *f2* was moved to fragment *f1*, which contains the complete DNA sequence #1. To identify the parts that should be rearranged, the algorithm considers that the first line of each new sequence starts with a “>” symbol, that is a convention of FASTA format.

3. CASE STUDY

This section describes an experiment in the bioinformatics domain. BLAST tools are widely used by scientists in order to find regions of similarity between biological sequences. In our case study, BLAST has been used to identify similar sequences between *Plasmodium falciparum* sequences and SWISS-PROT (<http://www.expasy.ch/sprot/>) database release 56.0 of 22-Jul-08 of UniProtKB/Swiss-Prot that contains 392,667 sequence entries, comprising 141,217,034 amino acids abstracted from 172,036 references. Genomics data from *Plasmodium falciparum* are analyzed and hosted by ProtozoaDB System [5]. ProtozoaDB integrates heterogeneous databases, (re)annotation systems, analyses tools based on distributed computing. It also provides analyses with emphasis on distant similarities (HMM-based) and phylogeny-based annotations, including orthology analysis [5].

Considering parallel execution, BLAST presents characteristics that reinforce the implementation of data parallelism as proposed by Hydra architecture. Therefore, the experiment described in this section explores data parallelism over BLAST application. In this case, some issues should be considered, such as a decision about whether to fragment input query file and/or database.

There are three possible fragmentation scenarios in parallel execution of BLAST. In the first one, only the input query file is fragmented. This way, each execution node invokes a replica of BLAST database and consumes a fragment containing a subset of input sequences. Finally, the results are combined into a single output file. In the second one, only database is fragmented and the entire input file is replicated in each execution node. Then, each node compares the entire

In this experiment, we have chosen to fragment the input file in order to use it in agreement with data parallelism proposed by Hydra. The fragmentation of input file is performed by the Data Fragmenter component and its behavior is detailed in section 2.3.

Figure 5 presents the instrumented file used in Hydra. It contains a BLAST command-line with additional special tags. In other words, the script file is instrumented by follow tags: `<%=DIREXP%>` and `<%=IDX%>`. These tags indicate the path to the experiment directory and the fragment index, respectively. NCBI BLAST package version 2.2.22 was used.

4. EXPERIMENTAL RESULTS

The bioinformatics workflow was developed in VisTrails using Hydra client components to distribute a BLAST activity. This activity was distributed using MTC paradigm in a cluster through Hydra MTC data fragmentation components. The execution of the bioinformatics workflow using Hydra was performed on a 64-node SGI Altix ICE 8200 cluster with Intel Xeon 8-core processors/node installed at the High Performance Computing Center, COPPE/ UFRJ. Hydra MTC infra-structure was setup to use Torque Cartridge as the job scheduler [18] to setup the job submission for the serial version of the BLAST activity.

Through this experiment, we could observe that Hydra middleware provides a systematic approach for supporting the parallelization of the workflow through data fragmentation. Initially, we used Hydra Setup to define the parallel configuration for the workflow. This configuration was stored at the Data Fragmentation Catalog, which represents the prospective provenance for the data fragmentation definition. The configuration files or commands line parameters for each program were then instrumented using tags that are easy to understand, as presented in Figure 5.

The workflow was executed on VisTrails, and then the input data for the BLAST execution was transferred to the MTC environment. Hydra Client Components invoked Hydra MTC to parallelize several BLAST activities. Hydra MTC automatically handles and creates an isolated workspace that includes all needed files for each distributed activity executed at the MTC environment. An individual workspace is created for each fragment of the input BLAST file, as presented in

```

ogasawara@service0:~/blast/template> cat experiment.sh
/home/users/ogasawara/blast-2.2.22/bin/blastall -p blastp -d /home/
users/ogasawara/blast/dbs/swissprot/swissprot -i <%=DIREXP%>/aa-pfa
lciparum2.fasta_<%=IDX%> -o <%=DIREXP%>/result_<%=IDX%>.txt
ogasawara@service0:~/blast/template> [2~

```

Figure 5 Hydra instrumented file

input file against a portion of the database. This scenario can generate some distortions related to statistical information. In order to correct this drawback, some parameters should be taken into account on BLAST invocation. One advantage of reducing database size is to execute less disk operations. In the third scenario, scientists may combine input file and database fragmentation.

section 3. Hydra retrospective provenance mechanism registers each workspace and its association to a specific fragment used in the data fragmentation. Once all workspaces are created, Hydra MTC dispatches these activities to the distributed scheduler execution. Each individual activity execution is monitored. The execution catalog represents the retrospective provenance for the parallel Hydra MTC execution. If any error

occurs, they are recorded by Hydra retrospective provenance mechanism. Also, execution information for each activity is recorded at the execution catalog. Yet, after all activities are completed, the execution control returns to VisTrails. The Gatherer sends the retrospective provenance from the MTC environment back to the SWfMS (VisTrails).

Table 1 presents a summary of Hydra MTC execution obtained from the Hydra retrospective provenance schema for an experiment executed using 4 nodes with 8 cores each, thus totalizing 32 cores. A FASTA file of 246 MB was compressed, uploaded, and decompressed in 0.8 min. The file was sliced into 256 FASTA fragments.

Table 1 - Summary of a Hydra MTC execution using 4 nodes with 8 cores obtained from the provenance repository

Number of activities (fragments explored)	256
Start date/time	2010-03-09 12:40
Uploader time (compression + data transfer + decompression)	0.8 min
HPC Execution Time (Dispatcher + Gatherer)	941 min
Downloader time	0.1 min
Total Execution Time	941.9 min
Speedup	31.5

After Hydra distribution of execution, the result was then sent

registered by the retrospective provenance is important to evaluate and reproduce the simulation. Important provenance data could be lost if a systematic controlled approach such as Hydra was not used.

Also, from the retrospective provenance data obtained from the execution catalog, it is possible to build the histogram of the execution time for all activities, as presented in Figure 6. Based on the information obtained from the execution catalog, it is possible to calculate the average (103.7 min) and standard deviation (8.13min) of each activity. It is also possible to estimate the equivalent sequential time for all activities, which was 29,664.4 minutes. Additionally, scientists can use the catalog to check which parameter generated the desired result or submit queries like “what is the average execution time of BLAST activities that executed in 8 cores”.

The execution time for Hydra MTC and the whole process of distributing the workflow (Uploader, Dispatcher, Gatherer and Downloader) is presented in the “Total Workflow” line of the Figure 7, where we can observe the overhead of transferring data. This way, the “Total Workflow Time” line represents the total execution time of the distributed workflow, thus considering the effective execution of the distributed activities and the upload and download time to and from the HPC environment. On the other hand, the “Hydra MTC” line represents only the execution time of the distributed activities (Dispatcher and Gatherer) without taking into account the transfer times (upload and download). As mentioned before, the upload and download have a fixed time, since the input and output data size is fixed.

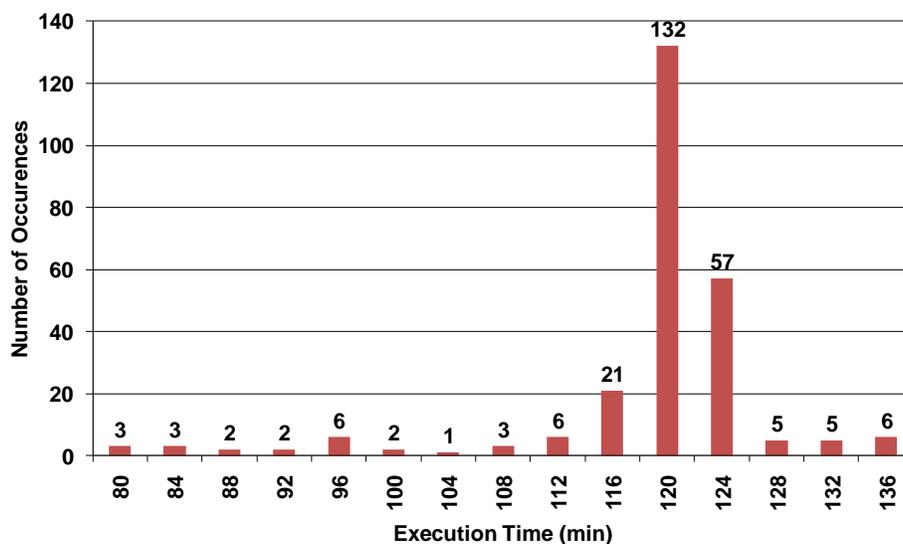


Figure 6 Activity execution time distribution

back, and the download output data transfer time was done in 0.1 min. These transfers (0.8 min + 0.1 min) have a fixed time and they represent all the data transfer between the SWfMS (client) and the HPC environment (server). The HPC execution time represents the total time needed to execute all distributed activities (941 min) including the dispatcher, makespan of all activities and provenance gatherer time. This led to a speedup of 31.5 (considering 32 cores available). The information

We may observe in Figure 7 that as the number of cores increases the total execution time of distributed activities decreases, as expected. However, the transfer time remains the same, thus increasing the impact of transfer time on total execution time, as the number of cores increases. It was observed that the benefits of parallelizing are clear, but depending on the input data transfer uploading time and the output data transfer downloading time, acquiring more nodes

for execution does not bring too much benefit, particularly if the number of activities becomes close to the number of nodes.

It is worth noticing that Hydra client components can be used by any SWfMS. They are not restricted to VisTrails. Each one of Hydra Client components is implemented as platform-independent JAR files, which can be easily invoked from any SWfMS. For example, in Kepler, through the command line actors, it is possible to invoke Hydra components. The Hydra setup component can also be installed as a Kepler Plugin.

Our first experiment was executed to evaluate Hydra data parallelism. Although the results were very positive, scaling tests for bioinformatics problems of larger sizes on larger machines is an ongoing work. Hydra MTC infra-structure is planned to be deployed in Galileo supercomputer (#76 in November 2009 Top-500 machines). This machine is a Sun Blade x6048, Xeon X5560 2.8 GHz, with Infiniband QDRmore and each processor is a Intel EM64T Xeon X55xx (Nehalem-EP) 2800 MHz (11.2 GFlops) installed at Federal University of Rio de Janeiro.

5. RELATED WORK

There are some approaches in the literature that propose mechanisms to provide MTC capabilities. However many of them are not focused on data parallelism using fragmentation. Nimrod/K [27] proposes a set of components and a run time machine for SWfMS Kepler. Nimrod/K is based on an architecture that uses tagged dataflow concepts for highly parallel machines. These concepts are implemented as a Kepler director that orchestrates the execution on clusters, grids and clouds using MTC. Although Nimrod/K is a step forward in defining parameter sweep, it does not concern about data fragmentation and provenance capture.

The approach proposed by Fei *et al.* [24] introduced a Map-Reduce based scientific workflow composition framework that is composed by a dataflow-based scientific workflow model and a set of dataflow constructs, including Map, Reduce, Loop, and Conditional constructs to enable a Map-Reduce

approach to scientific workflows. The framework was implemented in the VIEW system [28] and a case study was conducted for evaluation. However, differently from Hydra, this approach does not concern about capturing provenance data in distributed environments and integrating it with the SWfMS. Additionally, it introduces some complexity, since scientists have to program to achieve parallelism.

CloudBLAST [26] approach proposes and evaluates techniques for parallelization, deployment and management of bioinformatics applications that integrates several cloud environments [29]. It uses the Map-Reduce paradigm to parallelize bioinformatics tools and manage their execution. However, CloudBLAST approach is disconnected from the concept of scientific workflows and it is not concerned about capturing and analyzing provenance data.

G-BLAST [30] framework implements a solution based on grid services for supporting the submission of mpiBLAST [31] jobs to cluster systems. In spite of dealing with BLAST parallel execution, G-BLAST adopts a different fragmentation schema from that used in our work: it chooses to fragment the database rather than input file. To fragment the database, G-BLAST developed a Database Segmentation System instead of using *mpiformatdb*. The goal is to generate each fragment with size proportional to the grid site performance (cluster performance) where it will be processed. Although G-BLAST can store results into a specific database schema, provenance data related to the execution (like execution start time and end time) are not stored and provenance services are not provided with G-BLAST. Moreover, differently from our modular view (using cartridges), G-BLAST approach is very bounded, being strongly dependent of mpiBLAST and NCBI BLAST packages. Our cartridge may be used in any program that process FASTA files as input.

OpenWP [32] is a programming and runtime environment that aims at easing the adaptation and execution of applications onto grids [33]. It allows scientists to express the parallelism and distribution in existing programs or services by using

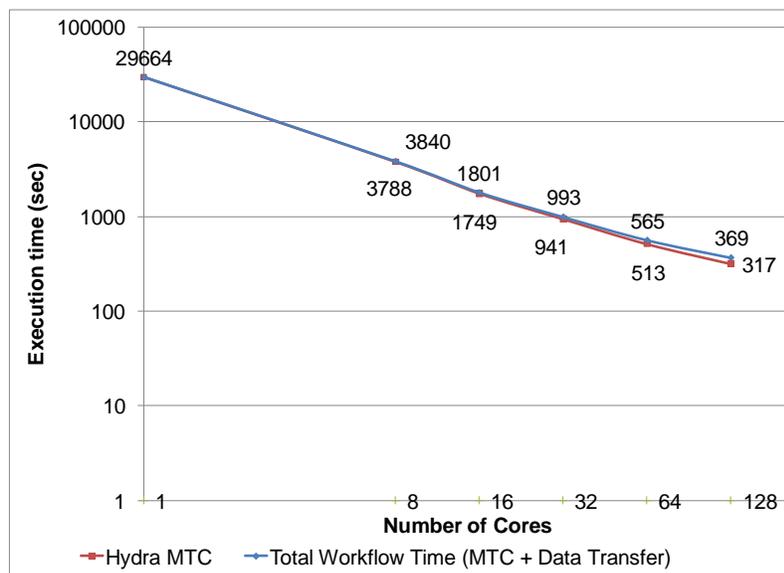


Figure 7 Workflow execution time

directives and to execute these applications on grids using existing SWfMS. Although it offers an interesting environment to promote parallelism, OpenWP is not concerned about fragmenting data and capturing provenance. Its main focus is on controlling the parallel tasks that were dispatched from the SWfMS.

MyCluster [21] is a system that provides an infrastructure for submitting parallel job proxies to the local scheduler (PBS, for example) of a cluster with support for transient faults. Also, MyCluster allows jobs to access files transparently from the submission host across remote sites. MyCluster can also be seen as complementary to Hydra, since MyCluster supports parallel job chained execution and could be used as one of Hydra's components. Unlike MyCluster, Hydra provides integration with generic SWfMS, which includes support for provenance.

6. CONCLUSION

Large scale bioinformatics experiments are usually computational intensive. Data parallelism is an important approach to increase performance in these experiments, but for most bioinformatics scientists, which are not familiar with computational clusters or supercomputers, using data parallelism in the experiments may become a hard task to accomplish, particularly with the need for provenance gathering.

Hydra is a middleware that is composed by a set of components to be included on the workflow specification of any SWfMS to promote parallelization of activities following MTC paradigm in a uniform and controlled way. By using Hydra components, it is possible to achieve data parallelism. Data fragmentation components are usually problem-dependent. We have designed cartridges, which are extensible components [20,34], to generalize data fragmentation behavior. It means that Hydra provides ways for the scientist to choose a suitable fragmenter for the input data. Although a cartridge is also problem-dependent like the Map function in Map-Reduce approach, its concept is flexible, since it can be dynamically changed and it just involves the setup of a program that is capable to fragment a data file in a number of parts.

Hydra data fragmentation cartridge splits an input file, and leaves to Hydra the responsibility to automatically associate tasks to each data fragment in the MTC environment. Hydra dispatches and monitors the execution of each of these tasks. Once all generated tasks are finished, Hydra collects the distributed provenance to promote provenance integration with the provenance repository of the SWfMS. Additionally, transferring data between the SWfMS and MTC environment may be time consuming. To cope with this task, Hydra may compress data prior to transfer and uncompress it after the transferring, saving transfer time.

In this paper we explored data parallelism in Hydra by modeling and parallel executing a bioinformatics workflow with BLAST activity. We have setup a fragmentation cartridge based on FASTA format using the FASTASplitter program that encapsulates the fragmentation rules. Our experiment with Hydra data fragmentation has been implemented on top of the VisTrails SWfMS although Hydra components may be deployed in any SWfMS. Hydra MTC layer is deployed in SGI Altix cluster of the High Performance Computer Center of

Federal University of Rio de Janeiro. By analyzing the performance of this architecture regarding data fragmentation, we have observed the benefits (performance and usability associated with integrated provenance gathering) of using Hydra middleware.

An important part of our future work is to evaluate different parallelism scenarios, such as those described in Section 3, using Hydra architecture and the implemented cartridges. Adaptive fragmentation is planned to be incorporated in Hydra middleware. This type of fragmentation dynamically tunes partition sizes, without requiring any additional knowledge, as discussed by Lima *et al.* [35] and Furtado *et al.*[36].

An important future work is to evaluate Hydra parallel executions comparing our solution to existing parallel approaches like Map-Reduce. Giving a systematic Map-Reduce support for SWfMS is not trivial and we are currently working on this.

7. ACKNOWLEDGMENTS

This work was partially sponsored by CNPq, CAPES, Faperj and Finep. The authors are also grateful to the High Performance Computer Center of Federal University of Rio de Janeiro (NACAD/UFRJ), where the parallelization experiments were performed, and Oswaldo Cruz Institute (FIOCRUZ).

8. REFERENCES

- [1] G.H. Travassos e M.O. Barros, 2003, Contributions of In Virtuo and In Silico Experiments for the Future of Empirical Studies in Software Engineering, In: *Proc. of 2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering, Roma*
- [2] M. Zvelebil e J. Baum, 2007, *Understanding Bioinformatics*. 1 ed. Garland Science.
- [3] R.D. Stevens, H.J. Tipney, C.J. Wroe, T.M. Oinn, M. Senger, P.W. Lord, C.A. Goble, A. Brass, e M. Tassabehji, 2004, Exploring Williams-Beuren syndrome using myGrid, *Bioinformatics*, v. 20, n. suppl_1 (Ago.), p. i303-310.
- [4] S.M.S.D. Cruz, V. Batista, A.M.R. Dávila, E. Silva, F. Tosta, C. Vilela, M.L.M. Campos, R. Cuadrat, D. Tschoeke, et al., 2008, OrthoSearch: a scientific workflow approach to detect distant homologies on protozoans, In: *Proceedings of the 2008 ACM symposium on Applied computing*, p. 1282-1286, Fortaleza, Ceara, Brazil.
- [5] A.M.R. Dávila, P.N. Mendes, G. Wagner, D.A. Tschoeke, R.R.C. Cuadrat, F. Liberman, L. Matos, T. Satake, K.A.C.S. Ocaña, et al., 2008, ProtozoaDB: dynamic visualization and exploration of protozoan genomes, *Nucleic Acids Research*, v. 36, n. Database issue (Jan.), p. D547-D552.
- [6] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, e S. Mock, 2004, Kepler: an extensible system for design and execution of scientific workflows, In: *SSDBM*, p. 423-424, Greece.
- [7] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M.R. Pocock, P. Li, e T. Oinn, 2006, Taverna: a tool for

- building and running workflows of services, *Nucleic Acids Research*, v. 34, n. Web Server issue, p. 729-732.
- [8] S.P. Callahan, J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, e H.T. Vo, 2006, VisTrails: visualization meets data management, In: *Proc. SIGMOD 2006*, p. 745-747, USA.
- [9] E. Deelman, G. Mehta, G. Singh, M. Su, e K. Vahi, 2007, "Pegasus: Mapping Large-Scale Workflows to Distributed Resources", *Workflows for e-Science*, Springer, p. 376-394.
- [10] I. Taylor, M. Shields, I. Wang, e A. Harrison, 2007, "The Triana Workflow Environment: Architecture and Applications", *Workflows for e-Science*, Springer, p. 320-339.
- [11] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, e M. Wilde, 2007, Swift: Fast, Reliable, Loosely Coupled Parallel Computation, In: *Services 2007*, p. 206, 199
- [12] C. Lin, S. Lu, X. Fei, D. Pai, e J. Hua, 2009, A Task Abstraction and Mapping Approach to the Shimming Problem in Scientific Workflows, In: *Proc. Services 2009*, p. 284-291
- [13] L.A. Meyer, S.C. Rössle, P.M. Bisch, e M. Mattoso, 2005, "Parallelism in Bioinformatics Workflows", *High Performance Computing for Computational Science - VECPAR 2004*, , p. 583-597.
- [14] L. Meyer, D. Scheftner, J. Vöckler, M. Mattoso, M. Wilde, e I. Foster, 2007, "An Opportunistic Algorithm for Scheduling Workflows on Grids", *High Performance Computing for Computational Science - VECPAR 2006*, , p. 1-12.
- [15] I. Raicu, I. Foster, e Yong Zhao, 2008, Many-task computing for grids and supercomputers, In: *Workshop on Many-Task Computing on Grids and Supercomputers*, p. 1-11
- [16] J. Freire, D. Koop, E. Santos, e C.T. Silva, 2008, Provenance for Computational Tasks: A Survey, *Computing in Science and Engineering*, v.10, n. 3, p. 11-21.
- [17] E. Ogasawara, D. Oliveira, F. Chirigati, C.E. Barbosa, R. Elias, V. Braganholo, A. Coutinho, e M. Mattoso, 2009, Exploring many task computing in scientific workflows, In: *MTAGS 09*, p. 1-10, Portland, Oregon.
- [18] A. Bayucan, R.L. Henderson, e J.P. Jones, 2000, Portable Batch System Administration Guide, *Veridian System*
- [19] G. Guerra, F. Rochinha, R. Elias, A. Coutinho, V. Braganholo, D.D. Oliveira, E. Ogasawara, F. Chirigati, e M. Mattoso, 2009, Scientific Workflow Management System Applied to Uncertainty Quantification in Large Eddy Simulation, In: *Congresso Ibero Americano de Métodos Computacionais em Engenharia*, Búzios, Rio de Janeiro, Brazil.
- [20] C. Szyperski, 1997, *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Professional.
- [21] E. Walker e C. Guiang, 2007, Challenges in executing large parameter sweep studies across widely distributed computing environments, In: *Workshop on Challenges of large applications in distributed environments*, p. 11-18, Monterey, California, USA.
- [22] M. Wiczorek, R. Prodan, e T. Fahringer, 2005, Scheduling of scientific workflows in the ASKALON grid environment, *SIGMOD Rec.*, v. 34, n. 3, p. 56-62.
- [23] J. Dean e S. Ghemawat, 2008, MapReduce: simplified data processing on large clusters, *Commun. ACM*, v. 51, n. 1, p. 107-113.
- [24] X. Fei, S. Lu, e C. Lin, 2009, A MapReduce-Enabled Scientific Workflow Composition Framework, In: *ICWS*, p. 663-670, Los Alamitos, CA, USA.
- [25] I. Raicu, Y. Zhao, C. Dumitrescu, I. Foster, e M. Wilde, 2007, Falcon: a Fast and Light-weight tasK executiON framework, In: *SC07*, p. 1-12, Reno, Nevada.
- [26] A. Matsunaga, M. Tsugawa, e J. Fortes, 2008, CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications, *IEEE eScience 2008*, v. 0, p. 229, 222.
- [27] D. Abramson, C. Enticott, e I. Altinas, 2008, Nimrod/K: towards massively parallel dynamic grid workflows, In: *SC 08*, p. 1-11, Austin, Texas.
- [28] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, e F. Fotouhi, 2008, Service-Oriented Architecture for VIEW: A Visual Scientific Workflow Management System, In: *Services*, p. 335-342
- [29] D. Oliveira, F. Baião, e M. Mattoso, 2010, "Towards a Taxonomy for Cloud Computing from an e-Science Perspective", *Cloud Computing: Principles, Systems and Applications (to be published)*, Heidelberg: Springer-Verlag
- [30] C. Yang, T. Han, e H. Kan, 2009, G-BLAST: a Grid-based solution for mpiBLAST on computational Grids, *Concurr. Comput. : Pract. Exper.*, v. 21, n. 2, p. 225-255.
- [31] P. Balaji, W. Feng, J. Archuleta, H. Lin, R. Kettimuthu, R. Thakur, e X. Ma, 2008, Semantics-based distributed I/O for mpiBLAST, In: *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, p. 293-294, Salt Lake City, UT, USA.
- [32] M. Cargnelli, G. Alleon, e F. Cappello, 2008, OpenWP: Combining annotation language and workflow environments for porting existing applications on grids, In: *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*, p. 176-183
- [33] I. Foster e C. Kesselman, 2004, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.
- [34] P. Fusaro, G. Visaggio, e M. Tortorella, 1998, REP - Characterizing and Exploiting Process Components: Results of Experimentation, In: *Proceedings of the Working Conference on Reverse Engineering (WCRE'98)*, p. 20-29
- [35] A.A.B. Lima, M. Mattoso, e P. Valduriez, 2004, Adaptive Virtual Partitioning for OLAP Query Processing in a Database Cluster, *Proc 19th SBBD, Brasilia, Brazil*, p. 92-105.
- [36] C. Furtado, A. Lima, E. Pacitti, P. Valduriez, e M. Mattoso, 2005, Physical and virtual partitioning in OLAP database clusters, In: *17th International Symposium on Computer Architecture and High Performance Computing, 2005. SBAC-PAD 2005.*, p. 143-150