IGI PUBLISHING **ITJ 4537**

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

This paper appears in the publication, International Journal of Grid and High Performance Computing, Volume 1, Issue 1
edited by **Emmanuel Udoh © 2008, IGI Global**

Large-Scale Co-Phylogenetic Analysis on the Grid

Heinz Stockinger, Swiss Institute of Bioinformatics, Switzerland

Alexander F. Auch, University of Tübingen, Germany

Markus Göker, University of Tübingen, Germany

Jan Meier-Kolthoff, University of Tübingen, Germany

Alexandros Stamatakis, Ludwig-Maximilians-University Munich, Germany

ABSTRACT

Phylogenetic data analysis represents an extremely compute-intensive area of Bioinformatics and thus requires high-performance technologies. Another compute- and memory-intensive problem is that of host-parasite co-phylogenetic analysis: given two phylogenetic trees, one for the hosts (e.g., mammals) and one for their respective parasites (e.g., lice) the question arises whether host and parasite trees are more similar to each other than expected by chance alone. CopyCat is an easy-to-use tool that allows biologists to conduct such co-phylogenetic studies within an elaborate statistical framework based on the highly optimized sequential and parallel AxParafit program. We have developed enhanced versions of these tools that efficiently exploit a Grid environment and therefore facilitate large-scale data analyses. Furthermore, we developed a freely accessible client tool that provides co-phylogenetic analysis capabilities. Since the computational bulk of the problem is embarrassingly parallel, it fits well to a computational Grid and reduces the response time of large scale analyses.

Keywords: *bioinformatics; co-phylogenetic analysis; Grid computing; phylogeny*

INTRODUCTION

The generation of novel insights in many scientific domains such as biology, physics, or chemistry increasingly relies on compute-intensive applications that require high-performance or large-scale, distributed high-throughput computing technology and infrastructure. In the discipline of bioinformatics, biological insight is typically generated via data analysis pipelines that use a plethora of distinct and highly specialized tools. Most commonly, bioinformaticians and biologists collaborate to analyze data extracted from large databases containing DNA and/or protein data in order to study, e.g., the function of living beings, the effect and influence of diseases and defects, or their evolutionary history. Early "classic" bioinformatics tools, such as CLUSTALW (Thompson et al., 1994) or BLAST (Altschul et al., 1997) that have

been ported to Grid computing environments deal with biological sequence search, analysis, and comparison. Typically, these programs are embarrassingly parallel and therefore represent ideal candidate applications for Grid computing environments (Stockinger et al., 2006).

The study of the genome represents a way to obtain new insight and extract novel knowledge about living beings. In particular, stand-alone phylogenetic analyses have many important applications in biological and medical research. Applications range from predicting the development of emerging infectious diseases (Salzberg et al., 2007), over the study of Papillomavirus evolution that is associated with cervical cancer (Gottschling et al., 2007), to the determination of the common origin of Caribbean frogs (Heinicke et al., 2007).

Recent years have witnessed significant progress in the field of stand-alone phylogeny reconstruction algorithms, which represent an NP-complete optimization problem (Chor and Tuller, 2005), with the release of programs such as TNT (Goloboff, 1999), RAxML (Stamatakis, 2006), MrBayes (Ronquist and Huelsenbeck, 2003) or GARLI (Zwickl, 2006). Because of the continuous explosive accumulation and availability of molecular sequence data coupled with advances in phylogeny reconstruction methods, it has now become feasible to reconstruct and fully analyze large phylogenetic trees comprising hundreds or even thousands of sequences (organisms). However, current meta-analysis methods for phylogenetic trees such as programs that conduct co-phylogenetic tests can currently not handle such large datasets.

To alleviate this bottleneck in the meta-analysis pipeline, we recently parallelized, and released the highly optimized co-phylogenetic analysis program AxParafit (Axelerated Parafit - Stamatakis et al., 2007) that implements an elaborate statistical test of congruence between host and parasite trees (Legendre et al., 2002). AxParafit is a typical stand-alone Linux/Unix command line program. AxParafit has been integrated and can be invoked via a user-friendly graphical interface for co-phylogenetic analyses called CopyCat (Meier-Kolthoff et al., 2007). In this article, we present an enhanced version of this tool suite (henceforth denoted as CopyCat(AxParafit)) for co-phylogenetic analyses, that is packaged into a client tool which makes use of a world-wide Grid environment and thereby allows for large-scale data analysis. In the current version, the underlying Grid middleware is gLite (Laure et al., 2006) that is coupled with an efficient submission and execution model called Run Time Sensitive (RTS) scheduling and execution (Stockinger et al., 2006).

The remainder of this article is organized as follows: initially, we provide a brief introduction to the field of phylogenetic inference, co-phylogenetic analyses, and related software packages in Section 2. Next, we discuss the implementation and architecture of our new approach for efficient adaptation of the CopyCat(AxParafit) tool-suite to a Grid environment. Finally, we provide detailed performance results on the EGEE (Enabling Grids for E-SciencE, http://www.eu-egee.org) Grid infrastructure (where the gLite middleware is deployed in production mode) and demonstrate the performance as well as scalability of our proposed bioinformatics tool.

BACKGROUND

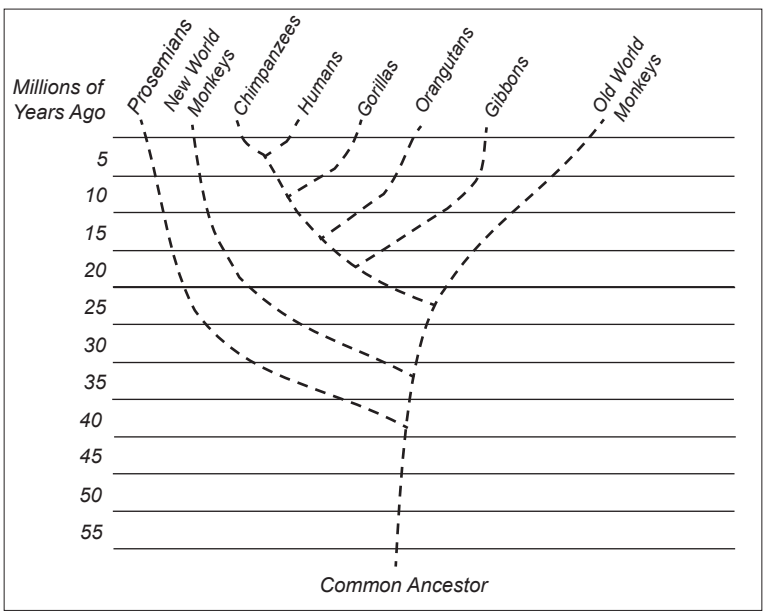
Phylogenetic (evolutionary) trees are used to represent the evolutionary history of a set of s currently living organisms, roughly comparable to a genealogical tree of species rather than individual organisms. Phylogenetic trees or simply phylogenies are typically unrooted binary trees. The s organisms, which are represented by their DNA or AA (Amino Acid/Protein) sequences that are used as input data for the computation, are located at the leave nodes (tips) of the tree while the inner nodes of the topology represent common extinct ancestors. There exist various methods and models to reconstruct such trees which differ in their computational complexity and also in the accuracy of the final results, i.e., there exists a "classic" trade-off between speed and accuracy. As already mentioned in

the introduction, phylogenetic analysis has many important applications in medical and biological research. In Figure 1, we provide a simple example for the phylogenetic tree of monkeys.

In the context of this article, however, we will not address stand-alone phylogenetic inference, but consider the problem of co-phylogenetic analysis. Given two phylogenetic trees that represent the evolutionary histories of hosts and their respective parasites, the "classic" example being mammals and lice, and given the extant associations between the former and the latter, we want to determine whether the parasite phylogeny is more similar to the phylogeny of the respective hosts than expected by chance alone. The main interpretation of such a congruence between the trees is that parasites have been associated with respect to their evolutionary history and mostly speciated in parallel (co-speciated) with their hosts (Page, 2002). Given a parasite tree with n organisms and a host tree with m organisms (sequences), their associations can be represented as a n times m binary matrix, that contains information of the type: does parasite x $(x=1...n)$ occur or live on host y $(y=1...m)$? In addition to the question of global congruence, one may also be interested in whether individual associations significantly increase the agreement between the phylogenies. Such associations can be interpreted as being caused mainly by co-speciation.

As previously mentioned, recent advances in stand-alone phylogenetic inference methods in combination with the increasing availability of appropriate sequence data, allow for large-scale phylogenetic analyses with several hundred or thousand sequences (Stamatakis, 2006). Thus, large-scale co-phylogenetic studies have, in principle, become feasible. However, most common co-phylogenetic tools or methods such as BPA, Component, TreeMap, TreeFitter (cf. review in Charleston, 2006) or Tarzan (Merkle, 2006) are not able to handle datasets with a large number of organisms or have not been tested in this regard with respect to their statistical properties and scalability. Faster methods based on topological distances between trees, like, e.g., I_{cong} (de Vienne, 2007) are even limited to the analysis of bijective associations only. In this

Figure 1. Phylogenetic tree of monkeys



context bijectivity means that each parasite can only be associated to one single host, and vice versa. Therefore, there is a performance and scalability gap between tools for phylogenetic analysis and meta-analysis. The capability to analyze large datasets is important to infer "deep co-phylogenetic" relationships which can otherwise not be assessed (Meier-Kolthoff et al., 2007; Stamatakis et al., 2007). Deep relationships are relationships that determine the extant associations between parasite and host organisms at a high taxonomic level, such as, e.g., families and orders.

Parafit (Legendre, 2002) and the analogous highly optimized AxParafit (Stamatakis et al., 2007) program implement a statistical test to assess hypotheses of global congruence between trees as well as the impact of individual associations. This test is based on the permutation of the entries in the association matrix. The null hypothesis is that the global similarity between the trees, or the respective impact of an individual local association on the similarity, is not larger than expected by pure chance. Extensive simulations have shown that the Parafit test is statistically well-behaved and yields acceptable error rates. The method has been successfully applied in a number of biological studies (Hansen et al., 2003; Ricklefs et al., 2004; Meinilä et al., 2004).

In addition, the type-II statistical error of Parafit decreases with the size of the dataset (see Legendre, 2002), i.e., this approach scales well on large phylogenies of hosts and parasites in terms of accuracy. The AxParafit program is a highly optimized version of Parafit which yields exactly the same results. The sequential version of AxParafit is up to 67 times faster than the original Parafit implementation, while the speedup increases with increasing input size, caused by higher cache efficiency. The speedup of AxParafit has been achieved via low-level optimizations in C, re-design of the algorithm, omission of redundant code, reduction of memory footprint, and integration of highly optimized BLAS (Basic Linear Algebra Subroutines, http://www.netlib.org/blas/) routines.

Earlier work describes these optimizations together with a respective performance study. Moreover, the program was used to conduct the largest co-phylogenetic analysis on real-world data to date. The underlying data were smut fungi and their respective host plants (Stamatakis et al., 2007). Smut fungi are parasitic mushrooms that cause plant diseases. For economically important hosts, such as barley and other cereals, smut fungi can for instance cause considerable yield losses (Thomas and Menzies, 1997).

Workflow of a Co-Phylogenetic Analysis with CopyCat and AxParafit

In this section, we provide an outline of the work-flow for a full co-phylogenetic analysis using CopyCat(AxParafit). The input for a co-phylogenetic analysis with CopyCat(AxParafit) are the host and parasite phylogenies, that might have branch lengths, depending on which method/model was used to calculate the trees. The aforementioned associations are represented as a plain text file containing a list of sequence (organism) name pairs of hosts and parasites, i.e., an adjacency list. This input data representation is henceforth also referred to as list of host-parasite associations. Initially, these files are parsed and transformed into the appropriate file format by CopyCat. In a first step, a principal coordinate analysis is conducted on the respective tree-based distance matrices induced by the host and parasite trees. This analysis is carried out by the AxPcoords (Axelerated Principal Coordinates) program (Stamatakis et al., 2007), which is an optimized version of the analogous DistPCoA program (Legendre and Anderson, 1998). The output of AxPcoords for the host and parasite trees is then parsed and appropriately prepared for the AxParafit analysis which takes the two principal coordinates matrices and the binary matrix with the associations as input. The output of this computation is a list of probabilities for the individual null hypotheses that a certain association does not improve the fit between host and associate phylogenies. In addition,

a probability for the global null hypothesis of the absence of congruence between host and parasite trees is computed. Upon termination of AxParafit the output files are read by the CopyCat tool and presented in a human-readable format. It is important to note that the computations with AxParafit represent the by far largest part (over 95%) of the computational effort required to conduct such a co-phylogenetic analysis. Therefore, the AxPcoords and CopyCat parts of the workflow can be handled sequentially and executed locally. We will, thus, mainly focus on the parallel and gridified versions of AxParafit in the next sections. The basic workflow is outlined in Figure 2 (at the end of the article).

Parallel AxParafit

The most compute-intensive operation (95% of execution time) conducted by AxParafit to compute the statistics is a dense matrix-matrix multiplication of double precision floating point numbers. This is the rationale for integration of highly optimized BLAS routines. In the remainder of this article, we thus always refer to the BLAS-based version of AxParafit.

Initially, the program will compute the statistics for the global congruence of the complete list of host-parasite associations. This part of the computation is significantly less expensive than the individual tests for each host-parasite association, which take *nz* times longer, where *nz* is the number of non-zero entries in the binary association matrix, i.e., number of entries in the original host-parasite association list. For large datasets that require parallel and distributed computing resources as well as a sufficient amount of memory typically $nz >> 1$. The statistics computed during the global test of congruence are required as input data for the individual tests of host-parasite associations, hence there is a sequential dependency: global test → *nz* local tests. Thus, in the MPI-based parallel implementation we only parallelized these *nz* local tests which can be computed independently of each other via a straightforward master-worker scheme. The master simply distributes the *nz* individual host-parasite association tests to the worker processes.

The potential bottleneck induced by the sequential part of the computations can be alleviated by using, e.g., the respective shared-memory implementations of BLAS. With respect to a gridification, this sequential dependency actually has advantages. Since the inference time as well as memory footprint of the global test of congruence are nearly identical (same type of operation, identical matrix sizes, permuted input data) to each of the individual *nz* tests, the information on run-times and memory requirements collected during the global tests can be used for scheduling decisions, as well as to determine an optimal level of granularity and to assess respective resource requirements.

FIT FOR THE GRID

In the following section, we describe how CopyCat(AxParafit) has been adapted and modified for use in a Grid environment. The overall architecture of the client tool will be explained as well as the integration with an existing middleware toolkit.

An important design goal of the Grid-based system for co-phylogenetic analyses was to re-use the current graphical user interface of CopyCat such that the deployment of Grid resources is hidden from the end-user. One fundamental difference between the standard and Grid-enabled versions of CopyCat (AxParafit) is that specific Grid credentials are required (an X.509 user certificate) since Grid jobs can only be submitted by authenticated and authorized users.

Overall Architecture

The basic workflow of a co-phylogenetic study using CopyCat and the AxPcoords/AxParafit programs has already been outlined in the above section. Here, we will describe the architecture and workflow for a gridified analysis in greater detail. The input data consists of three files: a host tree file, a parasite tree file, and a host-

parasite association list. In the reminder of this article, the following terminology is used:

- **Individual test (job):** an individual test is the minimal "work unit" or processing entity that has to be conducted by AxParafit to calculate a single host-parasite association. In the context of AxParafit this is also referred to as *job*. In total, *nz* individual tests have to be computed to achieve the final result.
- **Task:** a task consists of a fraction (subset) of the *nz* individual tests that have to be conducted by AxParafit.
- **Grid job:** a Grid job is an executable that is scheduled by the Grid middleware to be executed on a Worker Node of a Grid computing resource (also referred to as Computing Element). In our model, a single Grid job can execute one or several such tasks.

The overall workflow is depicted in Figure 2. The most important Grid-enhancement is the interface to the Grid (represented by the Perl program AxParafit.pl in Figure 2). Once the input files are validated, CopyCat uses AxParafit.pl to determine a specific set of tasks (to be registered with a Task Server as indicated in Figure 3) and Grid jobs which are then submitted to Grid computing resources using the gLite middleware. Each individual Grid job then requests tasks from the Task Server, processes them, and stores the result on a Grid Storage Element.

AxParafit.pl will constantly monitor the overall Grid job status and presents intermediate results in a CopyCat control window. Once all results are obtained and merged, CopyCat indicates where the final result can be obtained. Further details about AxParafit.pl, AxWorker.pl etc. will be given in the other section.

Implementation Details

In the following section we describe the necessary modifications and adaptations of the existing CopyCat and AxParafit tools as well as additional components that were necessary to implement the system outlined in Section 3.1.

CopyCat

Previous versions of CopyCat already provided straight-forward GUI-based functionality for the preparation and analysis of co-phylogenetic datasets. The CopyCat GUI is implemented in Java using the Standard Widget Toolkit (SWT). Upon startup, the user can load the host and parasite trees (represented in the standard Newick tree format: http://evolution.genetics.washington.edu/phylip/newicktree.html), together with a host-parasite association list in a simple plain-text format that contains one host-parasite association per line.

When starting an analysis, the user can now utilize a new Grid interface that connects CopyCat to the gridified program AxParafit. Instead of directly calling the AxParafit executable, the interface invokes a Perl script (AxParafit.pl) which hides the Grid-related parts from the user and CopyCat. By delegating the invocation process to a script, dependencies between the user front-end and the Grid software are minimized. Thus, future modifications like the development of a Web interface for job submissions (see Conclusion) or the usage of a different middleware system are possible.

The AxParafit.pl script entirely manages the execution of AxParafit on the Grid and provides status updates to the standard output stream at the same time. As CopyCat is listening to the output stream of the external programs it invokes, it also receives the status updates generated by the aforementioned Perl-script and writes them to the CopyCat log-message window, thus keeping the user informed about the progress of Grid jobs. Upon termination of the script, the output of the Grid jobs (individual tests of host-parasite associations), as well as the global significance test results, are read by CopyCat. The results can then be displayed and further analyzed via the CopyCat evaluation window.

Figure 2. Detailed work- and dataflow for co-phylogenetic analysis on the Grid

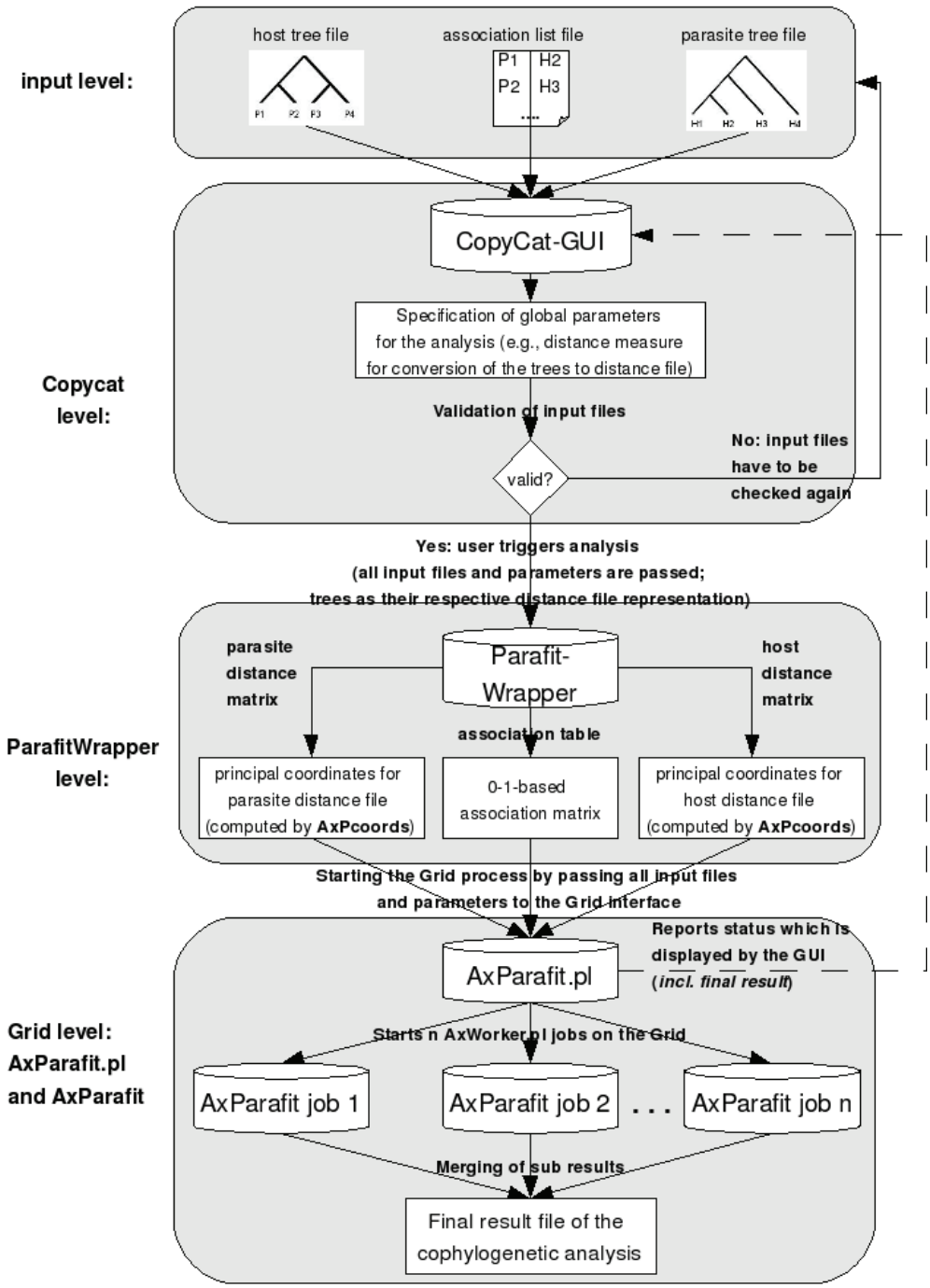
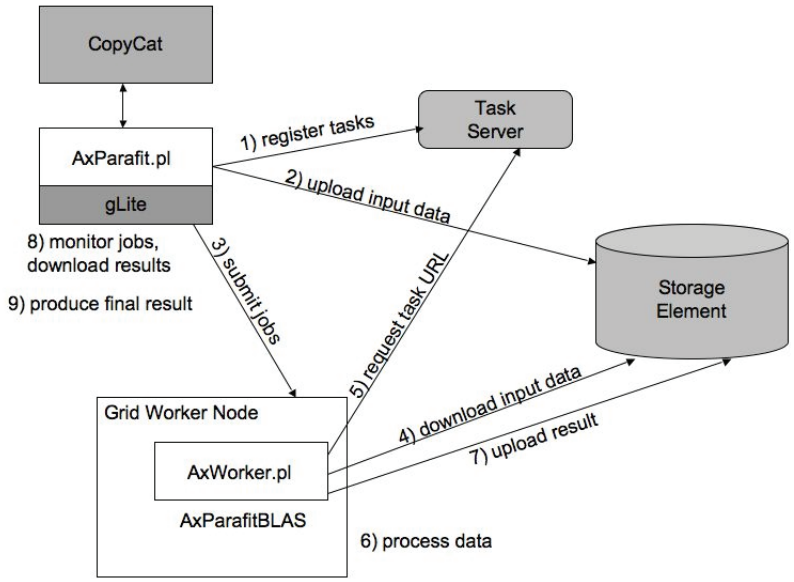


Figure 3. Interaction of AxParafit.pl with the gLite Grid middleware, a Task Server and a Storage Element. Each submitted Grid job will execute on a Grid Worker Node.



Within the context of an automated Grid-driven simultaneous analysis of several distinct datasets (and other potential script-based applications, based on CopyCat), the program has been extended by a command-line interface. As a side-effect, this enables CopyCat users to speed-up certain analyses by simply executing a specific command-line call with a defined set of parameters (please refer to the CopyCat manual for detailed information on the command-line options).

Application-Side Modifications of AxParafit

As outlined in other section, the parallel MPI implementation of AxParafit uses a simple master-worker scheme. In order to devise a distributed version of AxParafit we modified the code as follows: initially, we appropriately modified the global test of congruence in AxParafit to write an additional file called "gridData.RUN-ID" where RUN-ID is the output file name appendix for a specific analysis that is passed to AxParafit via a command line parameter (for details see the AxParafit Manual at http://icwww.epfl.ch/~stamatak/). This file contains the necessary data to make scheduling decisions for the distributed computation of the *n* individual tests of host-parasite associations, i.e., the number of jobs *nz,* e.g. Jobs=2000, and the approximate execution time per job in seconds, e.g., ComputeTime=10 . This data can then be used to determine the level of granularity for individual Grid tasks since in the current example the scheduling overhead induced by distributing 2,000 jobs of 10 seconds each, along with the comparatively large input datasets on the Grid, would be immense. We have, thus, extended the implementation of the individual host-parasite association tests in AxParafit by two additional command line parameters -l (lower limit) and -u (upper limit). These parameters allow for computation of several host-parasite associations in one single program run. The lower and upper limits just

refer to the order of the *nz* non-zero entries in the binary association matrix. Thus, in the present example, we can schedule larger, in terms of execution times, Grid jobs by only distributing two Grid jobs with -l 0 -u 1000 and -l 1000 -u 2000 that would require approximately 10,000 seconds of execution time each, i.e., Grid job 0 would compute statistics for the first 1,000 host-parasite associations and Grid job 1 for the remaining 1,000 associations. The result files of these distributed Grid jobs only need to be recovered and concatenated in the order of the associations they computed, and the respective result file can then be read and visualized by CopyCat.

Grid-Side Adaptation

Parafit.pl provides the actual link between CopyCat and the gridified version of AxParafit. First, it reads the file "gridData.RUN-ID" to determine the number of tasks to be created (registered) for execution on the Grid (Step 1 in Figure 3). As outlined in Section 3.2.1, the basic idea consists of combining appropriate fractions (subsets) of the *nz* individual tests into a single *task*, i.e., a set of individual tests $k < nz$ are executed by a Grid job. In order to make efficient use of the Grid and to reduce scheduling overhead, a task contains a minimum of k individual tests, such that the respective job requires at least 30 seconds on an average CPU. After the number of tasks has been determined, a certain number of Grid jobs (approximately nz/k) needs to be submitted (Step 3 in Figure 3) which then ask for tasks to be executed, i.e., issue work requests. An individual Grid job can request and execute several tasks, as long as the Task Server can provide more work (Steps 5 and 6 in Figure 3). The protocol used for the Task Server is HTTP which allows for fast communication between the client and the server. For additional background and fault tolerance features of this processing model with a Task Server please refer to Stockinger et al. (2006).

Before Grid jobs can be submitted, AxParafit.pl creates the Grid job specification, i.e. the job description file to decide which files (data and/or executables) to send to Grid computing resources. A typical job description file looks as follows:(See Box 1).

The wrapper code (identified as "Executable" in the JDL file above) is AxWorker.pl using the command line arguments specified by "Arguments". Once AxParafit.pl is running on a Grid Worker Node, it is responsible for requesting tasks from a Task Server and executing AxParafitBLAS. The two programs (AxWorker.pl and AxParafitBLAS) are transferred to the Grid Worker Node as specified in the InputSandbox in the example above, i.e. gLite provides the means to transfer data from the client machine to the actual computing resource.

In parallel to the execution of Grid jobs, the script AxParafit.pl monitors the status and

Box 1.

```
Executable = "AxWorker.pl";
Arguments = "-j ax-May1319-41-28 -p 100 -1 2048 -2 2048 -3 2025 -4 2031 \
–A gsiftp://example.org/dpm/home/biomed/heinz/selection_2048.mat-ax-May1319-41-28 \
-B gsiftp://example.org/dpm/home/biomed/heinz/selection_2048_P.pco-ax-May1319-41-28 \
-C  gsiftp://example.org/dpm/home/biomed/heinz/selection_2048_H.tra-ax-May1319-41-28  -i
1";
Stdoutput = "output.txt";
InputSandbox = {"/home/stockinger/AxWorker.pl", "/home/stockinger/AxParafitBLAS"};
OutputSandbox = {"output.txt"}
```

is responsible for providing and assembling the final result (Steps 8 and 9 in Figure 3).

In particular, when tasks have been processed successfully, they are downloaded from the Storage Element and transferred to the client. Note that an alternative implementation option is to transfer the output of individual tasks via the gLite middleware (using the OutputSandbox). However, because of performance and reliability considerations, it has turned out to be more efficient to store files at an external Storage Element and retrieve them from there: one reason is that the actual job output can only be retrieved if gLite indicates that a job has been finished. However, because of update latencies in the Grid-wide information and motoring system, jobs might have finished already several dozens of seconds or even a few minutes ago while the job status is still indicated as pending or running.

As a final remark: since the gLite services can only be accessed by authorized users, the execution of the AxParafit.pl script requires the usage of a valid X.509 proxy certificate.

EXPERIMENTAL RESULTS

The main goal of the gridified version of CopyCat(AxParafit) is to accelerate and facilitate large-scale analyses. We present two experiments with large computational demands and study their performance on the Grid. The performance improvement is outlined with respect to running the application sequentially on a single machine. Moreover, we conduct a performance comparison between a dedicated compute cluster and the Grid.

Test Environment

The Grid platform that is supported by our application is gLite 3. Tests are conducted using gLite on the EGEE production infrastructure. In particular, we use the Virtual Organization (VO) that is dedicated to biomedical applications: "biomed". Members of this VO have access to about 50 Computing Elements (acting as front-ends to computing clusters), each having between 2 and a few hundred processing cores. The exact number of processing cores available to a single user at a given time cannot be easily obtained since it depends on the current system load as well as the general availability of a Computing Element at a certain point in time. Currently, gLite does not support resource reservation nor job priorities, which means that experimental results can not be fully reproduced. However, once one is correctly registered with the Virtual Organization, one can use it any time of the day.

On the client side, we used gLite on GNU/Linux on an AMD Opteron machine (2 GB RAM, 2.2 GHz CPU) located in Lyon, France – previous tests (in particular with the installation of CopyCat and the Grid interface have been conducted on a machine located in Lausanne, Switzerland). The gLite components used are the workload management system (for job submission and status monitoring) as well as data management clients for file transfer. Additionally, we deployed and used a Task Server that is located in Lausanne, using resources provided by the Vital-IT group of the Swiss Institute of Bioinformatics. In the second experiment, we used a dedicated compute cluster with 128 CPUs. In contrast to the Grid, the cluster had to be reserved in advance.

Experiment with Real-world Data

In the first experiment we are interested in the raw performance (response time) of AxParafit.pl, i.e., how long does it take to fully process a set of tasks on the Grid. In this experiment, we do not include CopyCat but directly invoke AxParafit.pl as follows:(See Box 2).

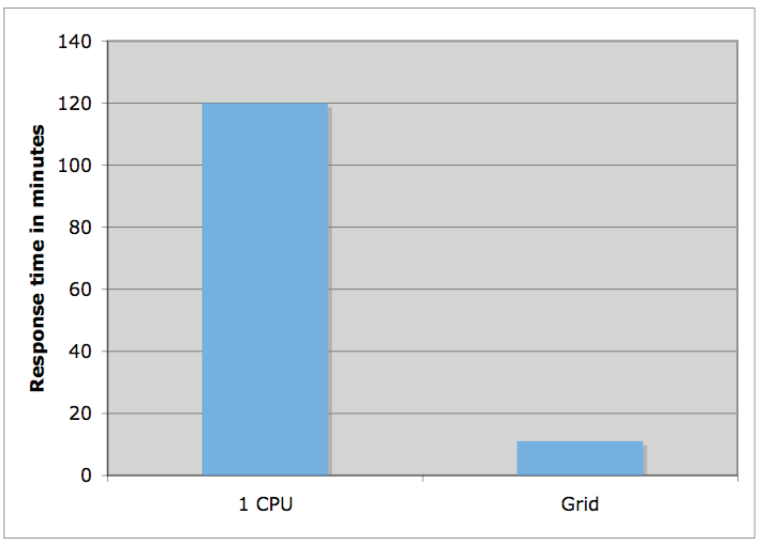
The parameters -1, -2, -3 and -4 specify the number of rows and columns in the association matrix as well as the number of rows and columns in the parasite and host matrices; -p represents the number of permutations conducted by the statistical test; -A, -B, and -C are used to read the plain-text input files; -n specifies a run ID that is appended to all output files (for details on the AxParafit program parameters please

Box 2.

```
AxParafit.pl -p 10 -1 413 -2 1400 -3 1390 -4
411 \
        -A smuts010907.mat -B smuts010907_P.pco
-C smuts010907_H.tra -n RUN_1
```

refer to the AxParafit manual at http://icwww.epfl.ch/~stamatak/). The dataset we used is the aforementioned (Section 2) dataset for the study of smut-fungi, that was used to demonstrate performance of the stand-alone AxParafit code by Stamatakis et al. (2007). As already mentioned, this dataset represents the largest *real-world* co-phylogenetic study conducted to date. While the sequential execution time for this dataset still appears to be acceptable, such studies were previously not feasible with Parafit which is between 1-2 orders of magnitude slower than AxParafit. Since the host-parasite association list contains *nz*=2,362 entries, 2,362 individual tests need to be performed. The execution of AxParafit to compute global congruence of the trees returned an estimated run time of 3 seconds per job, i.e., an overall expected run time of almost two hours (2,362 x 3 seconds). The main goal of the first test is therefore to minimize the expected response time. We also executed the full test, as specified above, on a single machine and observed that the estimated run time of about 2 hours (7,000 seconds) is

Figure 4. Comparison of smut-fungi dataset on a single CPU and on a Grid using 124 Grid jobs and 150 tasks. Note that there is a rather high redundancy in Grid jobs and not all 124 jobs really participate in the overall calculation because of start-up latencies. In fact, a few Grid jobs (AxWorker.pl) started, requested tasks and found out that there were no more tasks available and gracefully finished.



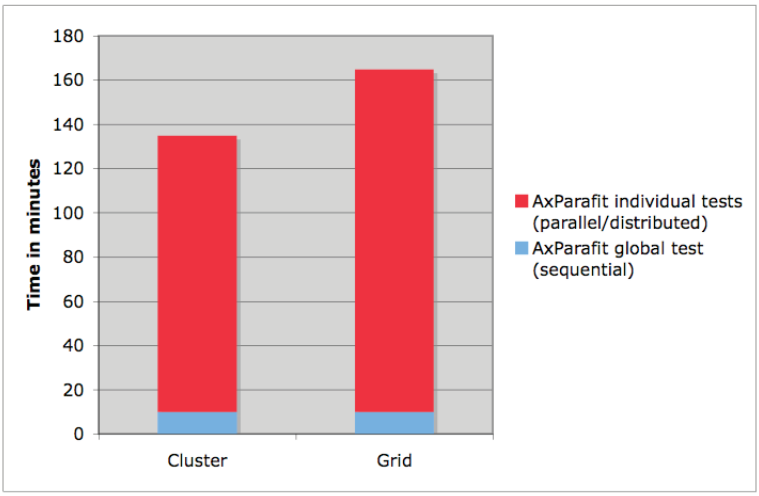
almost identical to the measured run time (7,200 seconds). Therefore, we deduce that the run time prediction mechanism is sufficiently accurate for our application. In our experiments, we varied the number of tasks (in the range between 60 and 162) as well as the number of parallel Grid jobs (in the range between 24 to 124) to experimentally determine the minimal response time. However, because of varying response times of the Grid (i.e. the various Computing Elements and their job queues etc.) it was not possible to determine an optimal number of Grid jobs and tasks. Finally, in the experiment we used 124 Grid jobs and 150 tasks which have been proposed by the work distribution algorithm outlined in aforementioned section. The overall response time to produce the final output was 11 minutes and 15 seconds (cf. Figure 4). Consequently, we observe a clear runtime improvement with respect to a single, sequential run. Note that the AxParafit.pl program had to be adapted to allow for parallel downloads of the individual results: originally, results were downloaded sequentially, which increased the overall response time by several minutes. By overlapping communication with computation, this problem was resolved.

Experiment with Synthetic Data

In another experiment, we used a larger (*synthetic*) test dataset that had been extracted from a larger empirical dataset to test scalability of AxParafit and compared the runtime of the Grid with the infiniband cluster at the Technical University of Munich equipped with 128 AMD Opteron 2.4 GHz CPUs. In the association list, there were nz=2,048 non-zero entries (equivalent to 2,048 tasks) and we used 100 permutations. The expected runtime of a single task was 568 seconds, i.e., about 10 minutes. As a result, the expected sequential response time to finish all 2,048 tasks is about 13.4 days. We used the wrapper as follows: (See Box 3).

Note that the input files are bigger than in the previous experiment: they cannot be directly submitted with the Grid job but they are uploaded to a Storage Element and then dynamically downloaded by Grid jobs when needed.

Figure 5. Performance comparison of a 128 CPU cluster with a Grid using between 90 and 175 parallel jobs. Note that the number of Grid job varied and was never constant.



Box 3.

```
AxParafit.pl -p 100 -1 2048 -2 2048 -3 2025 -4
2031 -A selection_2048.mat \
-B selection_2048_P.pco -C selection_2048_H.tr
a -n RUN_2
```

A direct performance comparison between the cluster and a Grid is not feasible since the cluster we used had several favorable features that a multi-institutional Grid does not have: a shared file system between all processing nodes which minimizes the data transfer time; homogeneous hardware infrastructure; pre-defined number of CPUs that are available which does not require an automatic task assignment, no overhead for job submission etc. However, the cluster needed to be reserved in advance (larger slots can only be obtained overnight) which means that it was only available at a specific time, whereas Grid resources are available on demand at any time. Intuitively, one expects a cluster to provide a better response time to a large size application but it has a considerable "reservation latency", a fact that should not be underestimated.

The final performance results of the experiments are depicted in Figure 5. For the Grid execution, we used between 90 and 175 parallel jobs (the number varied during the overall execution time). Given the number of parallel jobs used in the Grid, the cluster performed better. However, if the number of jobs is increased on the Grid, the cluster can actually be out-performed. Note that, the Grid response time comprises the sequential run time that is necessary to determine the number of tasks and jobs and compute the test for global congruence that is then used as input data for the *nz* individual tests. This initial part of the analysis also needs to be executed sequentially on the cluster. In addition, the Grid response time also includes the job submission overhead that is imposed by the gLite workload management systems. In order to avoid congestion problems at the submission server, only a certain number of jobs are submitted at a given time by AxParafit.pl. The actual processing time of the 2,048 AxParafit tests can then be better compared to the cluster performance. Another interesting observation is the average processing time of 13.3 min per single task on the Grid compared to the local execution time of 11 min on the Grid client machines. This indicates that distinct Computing Elements have CPUs with rather different CPU speeds and latencies.

Overall, our Grid-based approach requires computing times that are in the same order of magnitude as those of a dedicated cluster. Consequently, the gridified version provides an easier to use alternative to a compute cluster with comparable performance.

CONCLUSION

We have demonstrated how a compute-intensive application for a statistical test of congruence between host and parasite phylogenies can efficiently be distributed on the Grid. The proposed Grid-based implementation can greatly contribute to the reduction of response times for large-scale analyses and to the computation of a larger number of test permutations, which in turn improve upon accuracy. Moreover, we have integrated the access to Grid resources into an easy-to-use Graphical User Interface (CopyCat) which entirely hides the technical details related to the exploitation of Grid resources from the user. Note that in particular for non-expert

users, easy accessibility and usability of HPC resources represents a major criterion for the selection of software and systems. We thus believe that the proposed architecture will greatly facilitate access to HPC resources for real-world biological studies on host-parasite evolution. Nonetheless, the requirement to obtain access and accreditation to use Grid resources (valid X.509 proxy certificate) will possibly hinder a large amount of potential users to exploit these new possibilities offered by the Grid. Based on previous experience with the development of the freely accessible RAxML Web servers for phylogenetic reconstruction (Stamatakis et al., 2008, over 8,000 job submissions in the first 8 months of operation) that are however scheduling jobs to dedicated clusters instead of the Grid, we believe that a freely accessible Web server for this Grid-enabled system for co-phylogenetic analyses can contribute to the generation of biological insights, by further simplifying the access to HPC resources. Thus, future work will concentrate on the development of such a Web server, as well as the integration with the aforementioned RAxML servers such as to provide a comprehensive phylogenetic and co-phylogenetic analysis pipeline.

ACKNOWLEDGMENT

This work was funded in part by the EU project EMBRACE Grid which is funded by the European Commission within its FP6 Program, under the thematic area "Life sciences, genomics and biotechnology for health", contract number LUNG-CT-2004-512092. The Exelixis lab (AS) is funded under the auspices of the Emmy-Noether program by the German Science Foundation (DFG).

REFERENCES

Altschul, S.F., Madden, T.L., Schaffer, A.A., et al. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389-3402.

Chor, B., & Tuller, T. (2005). Maximum likelihood of evolutionary trees: hardness and approximation. *Bioinformatics,* 21(1):97-106.

Goloboff, P. (1999). Analyzing Large Data Sets in Reasonable Times: Solutions for Composite Optima. *Cladistics* 15(4): 415-428.

Charleston, M.A., & Perkins L. (2006). Traversing the tangle: Algorithms and applications for cophylogenetic studies. *Journal of Biomedical Informatics,* 39 (2006):62-71.

Gottschling, M., Stamatakis, A., Nindl, I., et al. (2007). Multiple Evolutionary Mechanisms Drive Papillomavirus Diversification. *Molecular Biology and Evolution,* 24(5):1242-1258.

Hansen, H., Bachmann, L., & Bakke, T.A. (2003). Mitochondrial DNA variation of *Gyrodactylus* spp. *Monogenea, Gyrodactylidae* populations infecting Atlantic salmon, grayling, and rainbow trout in Norway and Sweden. *International Journal of Parasitology,* 33(13): 1471-1478.

Heinicke, M.P., Duellman, W.E., & Hedges, S.B. (2007). From the Cover: Major Caribbean and Central American frog faunas originated by ancient oceanic dispersal. *Proceedings of the National Academy of Sci*

Laure, E., Fisher, S., Frohner, A., et al. (2006). Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12(1):33-45.

Legendre, P., Anderson, M.J. (1998). DistPCOA program description, source code, executables, and documentation: http://www.bio.umontreal.ca/Casgrain/en/labo/distpcoa.html

Legendre, P., Desdevises, Y., & Bazin, E. (2002). A Statistical Test for Host-Parasite Coevolution. *Systematic Biology,* 51(2):217-234.

Meier-Kolthoff, J.P, Auch, A.F., Huson, D.H., & Göker, M.(2007). COPYCAT: Co-phylogenetic Analysis tool. *Bioinformatics*, 23(7):898-900.

Meinilä, M., Kuusela, J., Zietara, M.S., & Lumme, J. (2004). Initial steps of speciation by geographic isolation and host switch in salmonid pathogen *Gyrodactylus salaris (Monogenea: Gyrodactylidae)*. *International Journal of Parasitology,* 34(4):515-526

Merkle, D., & Middendorf. M. (2005). Reconstruction of the cophylogenetic history of related phylogenetic trees with divergence timing information. *Theory in Biosciences,* 123(4):277-299.

Ricklefs, R.E., Fallon, S.M., & Birmingham, E. (2004). Evolutionary relationships, cospeciation, and host switching in avian malaria parasites. *Systematic Biology,* 53(1):111-119.

Ronquist, F., & Huelsenbeck, J. (2003). MrBayes 3: Bayesian phylogenetic inference under mixed models. *Bioinformatics,* 19(12): 1572-1574.

Salzberg, S.L., Kingsford, C., Cattoli, G., et al. (2007). Genome analysis linking recent European and African influenza (H5N1) viruses. *Emerg Infect Dis,.* 13(5):713-8.

Stamatakis, A. (2006). RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics,* 22(21): 2688-2690.

Stamatakis, A., Hoover, P., & Rougemont, J. (2008). A Rapid Bootstrapping Algorithm for the RAxML Web Servers. *Systematic Biology*, in press.

Stamatakis, A., Auch, A.F., Meier-Kolthoff, J., & Göker, M. (2007). *AxPcoords & parallel AxParafit: statistical co-phylogenetic analyses on thousands of taxa. BMC Bioinformatics* 2007, 8:405.

Stockinger, H., Pagni, M., Cerutti, L., & Falquet, L. (2006). Grid Approach to Embarrassingly Parallel CPU-Intensive Bioinformatics Problems. *2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*, IEEE Computer Society Press, Amsterdam, The Netherlands.

Thomas, P.L., & Menzies, J.G. (1997). Cereal smuts in Manitoba and Saskatchewan, 1989-95. *Canadian Journal of Plant Pathology,* 19(2):161-165.

Thompson, J.D., Higgins, D.G.,, & Gibson, T.J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673-4680.

de Vienne, D.M., Giraud, T., & Martin, O.C. (2007). A congruence index for testing topological similarity between trees. *Bioinformatics*, 23(23):3119-3124.

Zwickl, D. (2006). Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. *PhD Thesis*, The University of Texas at Austin.

Heinz Stockinger has been working in Grid projects in Europe (CERN, etc.) and in the USA (Stanford Linear Accelerator Center) in various technical, scientific and management functions. Heinz is affiliated with the Swiss Institute of Bioinformatics where he works on diverse Grid subjects. He has been appointed "Privatdozent" at the University of Vienna - leading the Research Lab for Computational Technologies and Applications in 2005. Currently, he is also a lecturer at the Swiss Federal Institute of Technology in Lausanne (EPFL). Heinz holds a PhD degree in computer science and business administration from the University of Vienna, Austria.

Alexander Auch works as freelance software developer and consultant for industry as well as academia. He has received a master's degree in bioinformatics from the University of Tübingen in 2005 and is currently working on his doctoral thesis.

Markus Göker received his diploma in biology in July 1999 from the University of Heidelberg. In December 2003 he received his PhD for research on "Molecular and light microscopical investigations into the phylogeny of the obligate biotrophic Peronosporales" from the University of Tübingen. Since then he has been working as a postdoctoral researcher at the Institute of Organismic Botany/Mycology in Tübingen. His research interests include evolution, taxonomy and co-phylogenetic analyses of plant-parasitic fungi with a focus on downy mildews and smut fungi, and phylogenetic inference with alignment-free approaches

and from sequences with intra-individual variability. He has been particularly interested in compiling very large host-parasite datasets to conduct co-phylogenetic tests.

Jan Meier-Kolthoff is employed as a software developer in bioinformatics at a medium-sized biotech company in Bavaria, Germany. In March 2007 Jan received a master's degree in bioinformatics from Eberhard Karls Universität Tübingen. Despite his job-related occupation he is still highly interested and involved in scientific challenges.

Alexandros Stamatakis received his diploma in computer science in March 2001 from the Technical University of Munich. In October 2004 he received his PhD for research on "Distributed and Parallel Algorithms and Systems for Inference of Huge Phylogenetic Trees based on the Maximum Likelihood Method" also from the Technical University of Munich. From January 2005 to June 2006 he worked as postdoctoral researcher at the Institute of Computer Science in Heraklion, Greece. In July 2006 he joined Bernard Moret's group at the Swiss Federal Institute of Technology at Lausanne as a PostDoc. In January 2008 he moved back to Munich to set up a junior research group, that is funded under the auspices of the Emmy-Noether program by the German Science Foundation (DFG), at the bioinformatics department of the Ludwig-Maximilians University of Munich. His main research interest are: technical and algorithmic solutions for inference of huge phylogenetic trees, applications of high performance computing techniques in bioinformatics, and challenging phylogenetic analyses of real-world datasets in collaboration with biologists.