Future Generation Computer Systems 25 (2009) 337–345

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

AUGUSTUS at MediGRID: Adaption of a bioinformatics application to grid computing for efficient genome analysis

Dietmar Sommerfeld[a,*], Thomas Lingner[b], Mario Stanke[b], Burkhard Morgenstern[b], Harald Richter[c]

[a] *Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen, Am Fassberg, 37077 Göttingen, Germany*

[b] *Department of Bioinformatics, University of Göttingen, 37077 Göttingen, Germany*

[c] *Department of Informatics, Clausthal University of Technology, 38678 Clausthal-Zellerfeld, Germany*

ARTICLE INFO

Article history:
Received 30 November 2007
Received in revised form
16 April 2008
Accepted 14 May 2008
Available online 30 June 2008

Keywords:
Grid computing
Bioinformatics
Gridification
Workflow
Grid portal

ABSTRACT

In past years, researchers from many domains have discovered Grid technology which opens up new possibilities in solving problems that are difficult to handle with traditional cluster computing. With the rapidly increasing number of partially or completely sequenced genomes, computational genome annotation is a particularly challenging task in computational biology. In this paper, we describe how we adapted the gene-finding tool AUGUSTUS to Grid computing in the context of the German MediGRID project. The gridification process starts with providing security requirements and running the application manually using Grid middleware. Afterwards, the application is described as a workflow of successive program executions, which are automatically distributed to appropriate Grid resources by a workflow engine. Finally, we show how a convenient graphical user interface for end users is created by means of a portal framework.

1. Introduction

Grid computing has become a key technology to solve large-scale computational problems by using distributed heterogeneous resources. It is a special kind of distributed computing where computing and data resources are shared across existing administrative domains. Grids make it possible to utilize resources at many different sites from different organizations. Within a Grid, all members form a new administrative domain called Virtual Organization (VO).

If applications need more resources than a single computing center can provide, they can be gridified, i.e. distributed to several sites. Gridification is the process of enabling an application to be executed on a Grid. It allows the execution of applications that are too large for one site and reduces their execution time by acquiring resources from outside that were not accessible before. Usually, resources in computing centers are not fully utilized. Such unused resources can be made available to the Grid. On the other hand, computational loads are never constant, and with Grid computing an overloaded site can migrate jobs to sites with less load. The amount of resources made available via Grid technology will increase because many public research institutes will no longer have the means to set up large local clusters. Instead, computational projects are increasingly encouraged and receive funding to use computing power from a Grid.

The work described in this paper was carried out within the German MediGRID project. MediGRID is part of the German e-Science initiative D-Grid. The aim of the project is to provide a community Grid for researchers in the fields of medicine, biomedical informatics, and life sciences. Four types of pilot application were selected for the first phase of MediGRID: bioinformatics, image processing, biomedical ontology, and clinical research applications. The users of these applications often have limited computer use skills. Therefore, it is crucial to make Grid solutions as user friendly as possible.

Resources used for scientific computing are very heterogeneous. To handle the heterogeneity, Grid middleware is employed to provide a common basis on which the development of distributed applications can be settled. MediGRID uses the Globus Toolkit 4 (GT4) [1] as its Grid middleware which is based on stateful web services. The services implemented by GT4 are compliant to the Open Grid Services Architecture [2] and are in particular responsible for security requirements, execution and data management, monitoring and resource discovery.

On top of the basic GT4 middleware, MediGRID employs further specialized middleware services. The most important one is an advanced workflow system for orchestrating the distributed execution of applications on Grid resources. An application

* Corresponding author. Tel.: +49 551 201 1841; fax: +49 551 201 2150.
E-mail addresses: dsommer@gwdg.de (D. Sommerfeld), thomas@gobics.de (T. Lingner), mario@gobics.de (M. Stanke), burkhard@gobics.de (B. Morgenstern), harald.richter@tu-clausthal.de (H. Richter).

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.05.010

workflow [3] consists of several successive program executions and intermediate data transfers. This allows to model sequences of programs and the dependencies between these programs.[1] At the presentation layer, the MediGRID portal provides a graphical interface which is suitable for end users.

This paper describes the gridification of applications for MediGRID which has to be done by experienced developers. The gridification process involves GT4, the workflow system and the portal framework as required components. It does not require changes to the program source code. We examine the process with AUGUSTUS [4] as an example from the field of bioinformatics. AUGUSTUS is a successful and widely used software program to identify gene structures in eukaryotic genomes, a fundamental task in computational genome annotation.

There are many life science applications from various research fields which have the same software architecture properties as AUGUSTUS. For example, analysis of time series like electroencephalograms, processing of image data as in gene-expression micro-array experiments and simulation of complex biological or chemical systems can be considered similar gridification candidates. We expect that Grid technology improves running time and accessibility of such applications.

The paper is organized as follows: in Section 2, we describe an AUGUSTUS application run and how the computation is distributed on clusters without using Grid middleware. Section 3 explains the execution of applications on a Grid using the services of GT4. Section 4 introduces the workflow system and shows what descriptions are necessary to enable automatic distribution on Grid resources. Section 5 describes how graphical user interfaces for Grid applications are created using portlets, and in Sections 6 and 7, the paper concludes with first results, the lessons learned, and a summary.

2. Augustus

AUGUSTUS [4] is a DNA-sequence-based gene prediction program for eukaryotes, i.e. organisms with a cell nucleus. The purpose of the program is to identify the location and structure of all protein-coding genes in a given genome. Gene prediction is the first and most important step in the analysis of newly sequenced organisms. During the last years, AUGUSTUS has been used in many genome projects, e.g. [6,7]. A web interface for AUGUSTUS has been available at the website of the Bioinformatics group in Göttingen for several years [8]. Because the computing power of the group is limited, the web interface only allows the input of relatively short sequences, up to 3 million base pairs.

AUGUSTUS is typical for a whole class of bioinformatic applications: the program uses sophisticated statistical modeling and prediction algorithms which are computationally demanding. It does not require user interaction and can be run unattended. Input and output data is accessed from a central data storage repository. An application run consists of several phases which are separated by i/o operations that address files. Via these files, data exchange between phases is done. The computing phases are loosely coupled, i.e. they have little communication in relation to the computation. Furthermore, the software is distributed as open source software and has no data-related security requirements.

AUGUSTUS can be used as an ab initio program, which means that the prediction is based solely on the input DNA sequence. As a second possibility, the software may also incorporate hints on the gene structure coming from extrinsic sources such as BLAST [9] and DIALIGN [10,11] search results. With external hints, the prediction performance of AUGUSTUS can be increased significantly [12]. On the other hand, the computational costs to find external hints can exceed the costs of the program itself, in particular when large databases are used for BLAST search. Therefore, the search for external hints is not integrated into the AUGUSTUS web interface. In the following, we will focus on ab initio use of AUGUSTUS. In this case the command line program has two mandatory arguments: the query file and the species. The query file contains the input DNA sequences in uncompressed (multiple) FASTA [13] format. The species parameter denotes where the DNA sequences originate from, e.g. human, fruit fly, baker's yeast, etc. There are several optional parameters which influence the way the genes are predicted. The output of AUGUSTUS is written to the command line output or into user-specified files. The output is compatible with the General Feature Format which can be visually interpreted by means of a genome browser [14]. Alternatively, the results can be transformed into gene maps in graphical representation via gff2ps [15]. A typical command line looks like:

```
augustus --species=human example.fa > outputfile
```

Running AUGUSTUS on a single computer is relatively easy. Nevertheless, depending on the number and length of sequences and the parameter values, gene prediction can be computationally very intensive. For example, the application to the human genome with its 3 billion base pairs can take up to several weeks. A possibility to speed up the computation without modifying the program source code is to parallelize the computation. This can be done by splitting the input file into smaller files which contain fewer sequences, and to run the program on these input files separately. Long sequences can be split into several smaller sequences if a suitable overlap according to maximum gene size is used. Data parallelization provides the possibility of distributing the computation across several machines, for example the nodes of a local cluster. Typically, these nodes have a shared file system, so the handling of input and output data is easy. However, additional effort is necessary for handling the batch system, and for checking if all submitted jobs are executed properly.

The organizational effort increases if local cluster resources are not sufficient for the demands of the computation, e.g. when an external hint search is incorporated. In this case, further suitable resource providers have to be found to support the computation. For the AUGUSTUS application, this means splitting the input data and distributing them to multiple sites where the program will run. Without Grid technology, this necessitates dealing with several potentially different authentication and resource management systems. In the end, the output files have to be transferred back to the user for result analysis.

Doing all these steps manually is tedious and time-consuming. The benefit of the parallelization is the overall decrease in turnaround time, allowing larger problem sizes to be tackled, which cannot be accomplished with limited local resources. Some of the organizational problems, such as handling of user accounts and job submissions on multiple sites, can be solved by the middleware layer of the Grid.

3. Executing AUGUSTUS on the Grid

The first step before we can execute AUGUSTUS on the Grid is to provide for authentication and authorization. The Grid security infrastructure (GSI) of GT4 is based on a public key infrastructure. This means, every user and every server in the Grid is authenticated by X.509 [16] identity certificates. Thus, the first step to use Grid resources, is to get a user certificate from a certificate authority that is recognized by the Grid's security policy. To get an authorization for the machines in the Grid the user has to become

[1] Throughout the paper we use the term application for complex usage scenarios involving several executables/executions, while program stands for a single executable/execution.