Enabling GeneHunter as a Grid Service

A Case Study for Implementing Analytical Services in Biomedical Grids

S. Krikov1; R. C. Price2; S. A. Matney1; K. Allen-Brady1; J. C. Facelli1,2

1Department of Biomedical Informatics, The University of Utah, Salt Lake City, Utah, USA; 2Center for High Performance Computing, The University of Utah, Salt Lake City, Utah, USA

Keywords
Grid computing, legacy applications, data modeling, caBIG, semantic integration, linkage analysis

Summary
Background: A cursory analysis of the biomedical grid literature shows that most projects emphasize data sharing and the development of new applications for the grid environment. Much less is known about the best practices for the migration of existing analytical tools into the grid environment. Objectives: To make GeneHunter available as a grid service and to evaluate the effort and best practices needed to enable a legacy application as a grid service when addressing semantic integration and using the caBIG tools.

Methods: We used the tools available in the caBIG environment because these tools are quite general and they may be used to deploy services in similar biomedical grids that are OGSA-compliant.

Results: We achieved semantic integration of GeneHunter within the caBIG by creating a new UML model, LinkageX, for the LINKAGE data format. The LinkageX UML model has been published in the caDSR and it is publically available for usage with GeneHunter or any other program using this data format.

Conclusions: While achieving semantic interoperability is still a time-consuming task, the tools available in caBIG can greatly enhance productivity and decrease errors.

1. Introduction

The substantial growth of biomedical data due to the advances in high throughput experimental techniques is an important challenge for the research community. There is a great need for developing efficient infrastructures in which researchers from different biomedical domains can easily collaborate and access computational tools and data regardless of geographical location and organizational boundaries [1, 2]. Grid computing, defined as “a form of distributed computing whereby a “super virtual computer” is composed of many networked loosely coupled computers acting in concert to perform a very large task” (http://en.wikipedia.org/wiki/Grid_computing) has emerged as a promising technology to provide data and analytical services for biomedical research [3]. Grid computing can be distinguished from other more conventional distributed computing architectures by its focus on large-scale resource sharing, innovative applications of distributed workflows, access to high-performance computing resources and a strong distributed security model [4]. The computer science community has devoted considerable resources to develop grid middleware, and now grid engineers have at their disposal commercially developed middleware products as well as open source products, such as the Globus Toolkit [1, 2, 4–6], to develop new grid applications.

Grid technology is now being used to deploy large infrastructures outside the computer science domain. In the biomedical field there are numerous examples of grid infrastructures [7–18].

The current literature shows that the efficient use of distributed resources and available services within a particular grid strongly depends on the metadata [10] used to describe these resources, but unfortunately there is a lack of standardized tools and methods to deal with such metadata. Often metadata for a particular grid implementation is generated and used in an ad hoc fashion, much of it buried in middleware libraries and database schemas. This ad hoc expression and use of metadata causes chronic dependencies on human intervention during the operation of the grid, leading to systems unable to handle frequent syntactic changes in resource coordination and sharing protocols. The layering over the grid infrastructure of an explicit semantic infrastructure could lead to increased interoperability and greater flexibility. The most recent attempt to define a core set of capabilities and behaviors for semantic grid systems is the Semantic Open Grid Service Architecture (S-OGSA), which makes use of so-called reference architecture and extends standard OGSA to support the explicit handling of semantics, and defines the associated knowledge services. We believe that any serious attempt...
to grid-enable legacy applications should therefore consider the semantic implications of developing such service and the effort required to achieve it.

Biomedical researchers have developed a large collection of useful and well tested tools that need to be enabled in a grid environment so this new research modality can take a foothold in the research community. But researchers have little information to assess the effort and best practices required to grid-enable a legacy application into a grid environment.

This paper reports our experience when porting to a grid environment a very important genetic epidemiology tool, GeneHunter [19], which is widely used in the biomedical research community. While we have used caBIG tools, which are freely available, as development tools and deployed services in caBIG, the tools and methodologies described here are applicable with little modifications to other OGSA-compliant biomedical grids.

2. Methods

GeneHunter [19] is a very important tool for linkage analysis that is widely used in biomedical research. Currently its use has been limited to local available data; a grid-enabled version of GeneHunter can allow important analysis using distributed data sources and its incorporation into complex workflow analysis. GeneHunter uses genotype data to identify the genetic risk of different diseases or phenotypes based on the maximum likelihood estimation of recombination rates. GeneHunter provides a variety of computational methods of which the most commonly used are the parametric calculation of LOD scores and the non-parametric (NPL statistic) analysis, but for the purpose of this project we decided to limit the use case of the tool to the following scenario: Using the data available in databases accessible by grid-enabled data services, the user sends the data to the GeneHunter grid service, which performs the computation of the LOD scores and NPL statistics at multiple biomarkers for a population sample of related individuals in a moderate size pedigree (approximately 10 individuals) and delivers the results to the researcher.

GeneHunter uses data in the LINKAGE format. This format has been evolving and there are multiple “flavors” of its implementation. The most common reference for the format description is the Laboratory of Statistical Genetics at Rockefeller University (http://linkage.rockefeller.edu/soft/linkage/), which is the one used in this work. In the LINKAGE format the data is arranged into two text files: PEDFILE and DATAFILE. Information in PEDFILE reflects pedigree structure and genotype for each individual in the population. The DATAFILE provides locus description, gene order, recombination rates and a variety of run-time parameters for the analysis. Both PEDFILE and DATAFILE are ASCII text files with values separated by the space character. The meaning of a particular value is determined by its relative position in the line and the preceding parameters within the same file.

For this study we used the caBIG development tools and test environment, because our previous work suggested that these tools can greatly enhance productivity when deploying grid services to any OGSA-compliant grid.

There are two main types of caBIG services: data services for sharing data in the grid environment and analytical services that provide access to analytical tools over the grid. Because GeneHunter is an analytical tool, our first approach was to create a simple analytical service with the functionality of GeneHunter exposed through a set of attributes and functions. This approach allows users and applications to use the grid implementation of GeneHunter with the same type of data (ASCII files) and the same functionality as a traditional implementation of GeneHunter. This approach is quite simple and using caBIG tools for its implementation required only a few weeks of work. This represents a baseline effort to expose an existing application into a grid environment with minimal changes. While it is still beneficial to have GeneHunter as an analytical service available over the grid, as depicted in Figure 1, the use of this service is limited to those applications or users capable of providing the data in the traditional LINKAGE format. Because preparing the data in the LINKAGE format has been recognized as a significant barrier to the use of GeneHunter by non specialists, we realized that moving a legacy application, like GeneHunter, into grid environment without addressing data availability issues for the application may not represent a significant improvement in its usability by non-experts. This simple grid implementation removes the need for installing a local copy of GeneHunter, a relative easy task, but does not relieve researchers of the cumbersome task of assembling the input files using data from disparate sources. A much more advantageous scenario would be for a researcher to obtain the necessary data through querying grid data services and submitting the data in its native format to the grid-analytical tool without having to perform cumbersome reformattting tasks. This mode of operation also leads naturally to incorporating GeneHunter into a complex workflow, an option that is not easily available in our simple implementation without addressing the data issues. To be able to execute this scenario it is necessary to decompose the data elements present in the LINKAGE format into independent objects and to represent them in semantic terms using the Data Standard Registry (caDSR, http://ncihb.nci.nih.gov/infrastructure/cacore_overview/cadsr). By connecting data elements from the LINKAGE format with corresponding elements within caDSR we enable the link between the LINKAGE format and semantically discoverable data sources on a grid.

cabIG has developed semantic integration tools around the unified modeling language (UML) method [20], in which objects are defined as abstract concepts. Through a series of steps it is possible to produce a detailed description of the data elements to be utilized to describe these abstract concepts. In this work we developed the UML models using the Enterprise Architect software, as recommended in the caBIG documentation [21]. UML models suitable for caBIG implementation require two major components: the Logical Model and the Data Model. The Logical Model represents the expert knowledge about the domain of interest. Here we represented essential concepts from the LINKAGE format, such as “Individual,” “Allele” and “Pedigree,” as classes with attributes and
defined relationships. The Data Model is a database schema where all classes (concepts) of the Logical Model are represented by tables and attributes. Once the Data Model is completed, its elements have to be mapped to the corresponding elements of the Logical Model; often inconsistencies in the Logical Model become obvious only during the Data Model design or at the mapping stage. In our case we refined the Logical Model at least a dozen times to obtain the final version that is depicted in Figure 2.

Achieving caBIG compatibility requires a proper annotation of the relationship between the Logical and Data Models, such that the annotation significantly reduces the barrier for using the resources by a remote party who was not involved in the development of the resource. A complete definition of caBIG compatibility can be found at https://cabig.nci.nih.gov/guidelines_documentation/Silver_Review/#silver. In practice the developers of a model can accomplish this by annotating every object from the logical model with a predefined set of tags, which provides the association of that object with elements of the data model. Defining these associations manually is an error prone and a very time-consuming process. To automate this process caBIG provides the caAdapter tool [21] with a model mapping service (MMS) component. Using a drag-and-drop interface we were able to create proper annotations between concepts in the LINKAGE format and the actual database schema used to store the concepts. The annotations are then stored within the XML Metadata Interchange (XMI) representation of our UML model. The visual representation of the process helped us to find inconsistencies in the UML model. Another helpful feature of the caAdapter is the validation of structure and mapping specifications (Fig. 3). After the mapping was completed, the UML model of the LINKAGE format was ready for the semantic integration into the caDSR. The caBIG project uses a model-driven architecture approach to enable syntactic and semantic interoperability between resources. Using UML tools and the NCI data modeling infrastructure, the domain models were converted into common data elements based on ISO/IEC 11179 standard (http://www.iso.org) and registered in the caDSR [22]. These data elements were annotated by terms and concepts drawn from the vocabulary registered in the Enterprise Vocabulary Services (EVS) [22], such that the concepts of data elements and their relationships become semantically described. In this work we used the caCORE Software Development Kit (SDK) [21] for semantic integration. The model, named LinkageX, was submitted to caBIG requesting its publication on the caBIG grid. An experienced analyst worked closely with us so we could publish our model in timely manner on the caBIG grid. LinkageX was published on May 4, 2009 in the caDSR. Note that using caBIG tools automatically guarantees syntactic interoperability when semantic interoperability has been achieved.

To test and demonstrate the use of the model, we created a data service with a small database. The schema for the database was retrieved from the LinkageX UML
Fig. 2 UML model of LINKAGE format (Logical Model)
model and the tables in the database were filled with a small sample of synthetic data sufficient to perform the calculation of the LOD score. For this test we used the test data provided with distribution of GeneHunter v2.1, which is recommended by the GeneHunter documentation to check the functionality of installed software. Once the Data Service was implemented we were able to communicate with it via the client API or web interface depicted in Figure 4. For the Analytical Service implementation we utilized an extension of the Introduce tool, the grid Remote Application Virtualization Interface (gRAVI) that allows for quickly wrapping and deploying command line executable applications as caGrid services. Through internal methods gRAVI allows a user to submit input files to the service and receive output over the grid. Finally, we created a script which invokes GeneHunter in batch mode and outputs a LOD score. All components of our developed and deployed system are shown on Figure 5.

3. Results and Evaluation

All components of the system described in the previous section have been tested, screenshots of the output of different data elements of the LinkageX model when requested from the caCORE SDK from the test database developed here are given in the supplementary materiala. All execu-

---
a Supplementary material is published on our website www.methods-online.com.
tions of the grid-enabled components of GeneHunter gave exactly the same results as those executed in the stand-alone version and the difference between execution times was negligible. Taking into consideration that this is a very small test set we expect that the overhead added by the GRID implementation will be even less significant for larger pedigrees in which the CPU needed for the calculations dominates GeneHunter execution.

We did not perform a rigorous analysis of the time and effort needed to accomplish the semantic integration of GeneHunter into the caBIG environment, but an approximation of the effort required can be inferred from the following observations. The work reported here was performed by a graduate student with average programming expertise and no experience in grid application development over the period of nine months. The student worked part-time (approximately 15 h/week), and he was able to learn the principles of grid technology, became familiar with the concepts and tools available in caBIG, and completed the project described here. The rest of the team listed as authors of the paper provided guidance and advice, but did not implement any part of the system. Clearly the effort needed for semantic integration is substantially larger than the one needed for our first baseline implementation, but still manageable for a small team or even single investigator team.

4. Discussion
Our experience shows that when porting legacy applications to a grid environment it is important to decide at what granularity
the application needs to be exposed to the grid to be useful to the potential user community. In the case presented here for GeneHunter, our a priori analysis suggested that semantic integration with extensive data modeling was a preferred option; however, this approach proved to be more time-consuming and labor-intensive. The most challenging and time-consuming part of our project was the creation of the UML model for the LINKAGE format. This crucial step is to transform the specific domain knowledge into a form suitable for representation on the grid. Since other legacy tools use the LINKAGE data format for linkage analysis, the UML model that we produced and published in the caDSR can be used when porting other similar tools to the grid. While mapping each element of the UML model to the concepts from the caDSR metadata repository we found that for a given element it is possible to have several matching concepts from which to choose. For example, “Age” can represent concepts like “Age of the patient”, “Age at diagnosis”, or “Age in months”. The developer is responsible for choosing the concept within the context of the UML model. When there are no concepts available in a metadata repository one can submit a new concept that will be added to the repository after revision. We found that concepts for most of the classes of our proposed model LinkageX model already existed in the caDSR. The only problematic class encountered in our work was the Liability class, which is typically used to define penetrance values for each genotype group associated with the trait of interest. Liability classes can be used to modify penetrance parameter estimates based on covariate information such as age and sex. Eventually we mapped the Liability class variable to the very generic concept of “Class” with the understanding that at a later time we should probably initiate the process of new concept submission. We encountered a different situation with the class BioMarker. In our test data set, we used SNP (single nucleotide polymorphism) data; therefore, we could have used the existing and well-defined concept of “SNP” already available in caDSR. But we decided to use the more inclusive concept of BioMarker, which would allow future extensions of the model.

Based on our experience we recommend the following steps to researchers attempting the migration of analytical tools into a grid:

1. Define the scope of the tool functionality and at what granularity it will be implemented on the grid.
2. Design an UML representation of the legacy data types.
3. Publish the UML Model for legacy data on the selected grid.
4. Create a source data with corresponding data service on the grid based on published UML Model.
5. Create an Analytical Service based on published UML Model.
6. Deploy all components into the grid caGrid environment or any other OSGA-compatible grid (production stage).

5. Future Work

The work described in this paper is a summary of our experience porting a legacy application to a grid environment and it may be representative of the challenges that other research teams may encounter when doing the same. In the process we were also to develop and implement in the caCSR the LinkageX data model. In order to fully take advantage of this work, in future work we will develop software to assemble the data from distributed sources. We intend to fully implement these tools and deploy GeneHunter in caBIG; deployment will be followed by a usability assessment. Using ontologies to integrate these services with other grid infrastructure will also be addressed in the future.
6. Conclusions

We succeeded in creating metadata for a grid-enabled service for GeneHunter using the tools developed by the caBIG community. The time and resources required to transform a legacy tool into a grid-enabled application, when considering its semantic integration, are considerable, but not beyond those obtainable by small teams and individuals. This may be a very beneficial exercise for students and professionals exploring grid environments for biomedical research that can create valuable services for the research community with limited resources. The tools developed under the caBIG project are capable of enabling processes for harmonization of the legacy application and corresponding data within the federative architecture of caBIG or other OGSA-compliant grids. At the end of the project we were able to identify the necessary steps for the porting legacy tool such as GeneHunter into grid environment. While our experience was based only on this tool, we believe that many of the lessons learned can be generalized to other applications and that the steps recommended at the end of the discussion section are of general applicability. More work is needed to improve the usability and documentation of caBIG tools, but a variety of products in caBIG pipeline promise a successful future for the grid technology in the biomedical domain [22].

Acknowledgments

This work has been partially funded by the National Library of Medicine, training grant LM007124 and NCRR translational science award 1KL2RR025763-01.

References