ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

# A biomedical GRID infrastructure to enable high-performing data mining and knowledge extraction operations

**Παπουτσίδης Ευάγγελος**

**Μεταπτυχιακή εργασία**

Ηράκλειο, Φεβρουάριος 2006

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

# A biomedical GRID infrastructure to enable high-performing data mining and knowledge extraction operations

Εργασία που υποβλήθηκε από τον
Ευάγγελο Παπουτσίδη
ως μερική εκπλήρωση των απαιτήσεων
για την απόκτηση
ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΕΙΔΙΚΕΥΣΗΣ

Συγγραφέας:

_____
Ευάγγελος Παπουτσίδης
Τμήμα Επιστήμης Υπολογιστών

Εισηγητική Επιτροπή:

_____
Ευάγγελος Μαρκάτος
Αναπληρωτής Καθηγητής, Επόπτης

_____
Μανώλης Τσικνάκης
Ερευνητής Ι.Π. Ι.Τ.Ε., Επιβλέπων

_____
Ιωάννης Τόλλης
Καθηγητής, Μέλος

_____
Γεώργιος Ποταμιάς
Ερευνητής Ι.Π. Ι.Τ.Ε., Μέλος

Δεκτή:

_____
Δημήτρης Πλεξουσάκης
Αναπληρωτής Καθηγητής,
Πρόεδρος Επιτροπής Μεταπτυχιακών Σπουδών

Στους γονείς μου και τη Χαρά

# A biomedical GRID infrastructure to enable high-performing data mining and knowledge extraction operations

Papoutsidis Evangelos

Master Thesis

University of Crete
Computer Science Department

## Abstract

The breadth and depth of clinical and genetic information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality, improve therapies and meet the demanding needs of individualized healthcare.

The lack of a common infrastructure has prevented the clinical and research institutions from being able to mine and analyze disparate data sources. Research facilities have been working with islands of isolated data and informatics tools.

Knowledge discovery in large data repositories can find interesting hidden patterns and trends representing them in an understandable way. But data mining is both a data-intensive and compute-intensive task. In real world applications sequential algorithms of data mining and data exploration are often unsuitable for datasets with enormous size, high-dimensionality and complex data structure.

Grid computing promises unprecedented opportunities for unlimited computing and storage resources. The grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. In the case of knowledge discovery we whish to explore how grid technology is able to provide an infrastructure to access information from different data sources and the computational power to enable high-performing data mining techniques.

In such a context, this thesis represents an exploratory activity in studying various open issues related to the utilisation of Grid technologies and Service Oriented Computing for the resolution of critical biomedical informatics challenges.

As part of the work presented in this thesis, we have designed and developed a small grid environment; its users have the ability to perform the following actions: a) Access and retrieve data from clinical information systems, b) Query for (discover) the computational resources and data mining techniques (services) available in the grid. c) Execute a parallel implementation of Association Rules Mining (ARM) algorithm. The architecture of this environment was implemented on top of Globus Toolkit 4, using OGSA-DAI for access to data sources. The resources of the grid are governed by the TORQUE resource manager and mpiJava was used for the implementation of the parallel ARM algorithm.

**Master Thesis Supervisors:** Evangelos Markatos, Manolis Tsiknakis

# Σχεδίαση και ανάπτυξη μίας biomedical GRID υποδομής για την εφαρμογή αλγορίθμων εξόρυξης δεδομένων υψηλών επιδόσεων και λειτουργιών εξόρυξης γνώσης

Παπουτσίδης Ευάγγελος

Μεταπτυχιακή Εργασία

Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

## Περίληψη

Το μεγάλο φάσμα κλινικών και γενετικών πληροφοριών που είναι διαθέσιμες στην ερευνητική κοινότητα παρέχει μία τεράστια ευκαιρία για βελτίωση των ικανοτήτων μας στη μείωση της θνησιμότητας και τη βελτίωση των θεραπειών ενώ καθιστά ορατή την ανάγκη για εξατομικευμένη ιατρική περίθαλψη.

Η έλλειψη, όμως, μιας κοινής υποδομής δημιουργεί δυσκολίες στην ανάλυση διαφορετικών πηγών δεδομένων από τα κλινικά και ερευνητικά ιδρύματα με αποτέλεσμα ερευνητικές εγκαταστάσεις να χρησιμοποιούν απομονωμένα σύνολα δεδομένων και πληροφοριακά εργαλεία.

Η διαδικασία «ανακάλυψης γνώσης» σε μεγάλες βάσεις δεδομένων στοχεύει πρωτίστως στον εντοπισμό ενδιαφερόντων κρυμμένων προτύπων και τάσεων και ακολούθως στην ευνόητη παρουσίασή τους. Όμως, η εξόρυξη δεδομένων παρουσιάζει δυσκολίες εξαιτίας του όγκου των δεδομένων αλλά και της απαιτούμενης υπολογιστικής ισχύος. Σε εφαρμογές με πρακτική σημασία και όταν το σύνολο των δεδομένων είναι ιδιαίτερα μεγάλο και η δομής τους πολύπλοκη οι ακολουθιακοί αλγόριθμοι εξόρυξης δεδομένων συνήθως προκύπτουν ανεπαρκείς.

Η τεχνολογία «πλέγματος» (grid) διευρύνει σε σημαντικό βαθμό τη δυνατότητα για αύξηση της χρήσης των διαθέσιμων υπολογιστικών και αποθηκευτικών πόρων. Η φιλοσοφία της συγκεκριμένης τεχνολογίας είναι ο συντονισμένος διαμοιρασμός των πόρων με σκοπό τη λύση προβλημάτων σε δυναμικούς εικονικούς οργανισμούς (virtual organizations). Με τη διαδικασία «ανακάλυψης γνώσης» επιθυμούμε την εξερεύνηση του τρόπου με τον οποίο η τεχνολογία «πλέγματος» δύναται να παρέχει την κατάλληλη υποδομή για πρόσβαση σε διαφορετικές πηγές δεδομένων καθώς και την απαραίτητη υπολογιστική ισχύ για την εφαρμογή τεχνικών εξόρυξης δεδομένων υψηλών επιδόσεων.

Βάσει του πλαισίου αυτού, η συγκεκριμένη μεταπτυχιακή εργασία αποτελεί μια δραστηριότητα εξερεύνησης με απώτερο στόχο τη μελέτη ανοιχτών θεμάτων που αφορούν τη χρήση τεχνολογίας «πλέγματος» στο χώρο της βιοϊατρικής πληροφορικής.

Βασικός κορμός της εργασίας είναι η σχεδίαση και ανάπτυξη μιας πειραματικής grid υποδομής μέσω της οποίας ο χρήστης μπορεί να εκτελέσει τις εξής λειτουργίες: α) πρόσβαση και ανάκτηση δεδομένων από κλινικά πληροφοριακά συστήματα, β) ενημέρωση για τους

διαθέσιμους υπολογιστικούς πόρους και τις υπηρεσίες εξόρυξης δεδομένων στο grid, γ) εκτέλεση ενός παράλληλου αλγορίθμου για την εύρεση κανόνων συσχέτισης (Association Rule Mining) μεταξύ κλινικών γνωρισμάτων. Η αρχιτεκτονική της συγκεκριμένης υποδομής υλοποιήθηκε με τη βοήθεια του εργαλείου Globus Toolkit 4, ενώ η πρόσβαση στα κλινικά δεδομένα επετεύχθη μέσω της διεπαφής OGSA-DAI. Υπεύθυνος για τον έλεγχο των υπολογιστικών πόρων του grid ορίστηκε ο διαχειριστής TORQUE ενώ για την παράλληλη υλοποίηση του ARM αλγορίθμου χρησιμοποιήθηκε η βιβλιοθήκη mpiJava.

**Επόπτες Μεταπτυχιακής Εργασίας:** Ευάγγελος Μαρκάτος, Μανώλης Τσικνάκης.

## Ευχαριστίες

Μετά την ολοκλήρωση της παρούσας μεταπτυχιακής εργασίας αισθάνομαι την ανάγκη να ευχαριστήσω το Τμήμα Επιστήμης Υπολογιστών, όπως επίσης και το Ινστιτούτο Πληροφορικής του Ι.Τ.Ε. για την υλικοτεχνική υποδομή και την οικονομική υποστήριξη που μου προσέφεραν κατά τη διάρκεια των σπουδών μου.

Πρωτίστως, ένα μεγάλο ευχαριστώ οφείλω στον επιβλέποντα της εργασίας μου κ. Μανώλη Τσικνάκη, ο οποίος πρότεινε αρχικά το ενδιαφέρον αυτό θέμα, καθώς και στον κ. Γεώργιο Ποταμιά, ο οποίος με καθοδήγησε με χρήσιμες παρατηρήσεις σε κρίσιμα σημεία της εργασίας. Ιδιαίτερα σημαντική υπήρξε για μένα και η βοήθεια του επόπτη καθηγητή κ. Μαρκάτου, ο οποίος μαζί με τον κ. Τόλλη αποτέλεσαν ενεργά μέλη της τριμελούς επιτροπής για την αξιολόγηση της εργασίας μου.

Επίσης, ευχαριστώ το σύνολο των καθηγητών μου για τις γνώσεις που μου παρείχαν και το ενδιαφέρον που μου δημιούργησαν για τον τομέα της Ιατρικής Πληροφορικής από το προπτυχιακό ακόμα επίπεδο. Ιδιαίτερα ευχαριστώ τον κ. Ορφανουδάκη, ο οποίος με τον τρόπο της διδασκαλίας του ώθησε τα ερευνητικά μου ενδιαφέροντα στο συγκεκριμένο τομέα.

Φυσικά, οφείλω να ευχαριστήσω και όλους μου τους συνεργάτες από το εργαστήριο και κυρίως το Δημήτρη Κατεχάκη, το Στέλιο Σφακιανάκη και τον Λευτέρη Κουμάκη οι οποίοι υπήρξαν πάντα πρόθυμοι να με βοηθήσουν σε κάθε δυσκολία που παρουσιαζόταν.

Τέλος, ευχαριστώ τη Χαρά, τους γονείς μου και όλα εκείνα τα κοντινά και αγαπημένα μου πρόσωπα, που με βοηθούν σε κάθε εγχείρημα και στέκονται πάντα στο πλευρό μου.

# Contents

# 1 Introduction

## 1.1 Introduction to GRID

Grid computing aims at the provision of a global infrastructure that will enable a coordinated, flexible, and secure sharing of diverse resources, including computers, applications, data, storage, networks, and scientific instruments across dynamic and geographically dispersed organizations and communities (Virtual Organizations or VO) [1]. Grid technologies promise to change the way organizations tackle complex problems by offering unprecedented opportunities for resource sharing and collaboration. Just as the World Wide Web transformed the way we exchange information, the Grid concept takes parallel and distributed computing to the next level, providing a unified, resilient, and transparent infrastructure, available on demand, in order to solve increasingly complex problems.

Grids might be classified into Computational Grids, Data/Information/Knowledge Grids, and Collaborative Grids [2]. The goal of a Computational Grid is to create a virtual supercomputer, which dynamically aggregates the power of a large number of individual computers in order to provide a platform for advanced high-performance and/or high-throughput applications that could not be tackled by a single system. Data Grids, on the other hand, focus on the sharing of vast quantities of data. Information and Knowledge Grids extend the capabilities of Data Grids by providing support for data categorization, information discovery, ontologies, and knowledge sharing and reuse. Collaborative Grids establish a virtual environment, which enables geographically dispersed individuals or groups of people to cooperate, as they pursue common goals. Collaborative Grid technologies also enable the realization of virtual laboratories or the remote control and management of equipment, sensors, and instruments.

The most widely used middleware for developing Grid projects is the Globus Toolkit [6], [35]. Over the last years, Globus Toolkit, which has been originally designed for the needs of High Performance Computing (HPC) resource sharing in the academic community, has undergone a significant shift towards the adoption of a service-oriented paradigm, and the increasing support for and utilization of commercial Web Services technologies. The Open Grid Services Architecture (OGSA) is bringing Grid technologies and Web Services together. The implementation of OGSA on the Web Services Resource Framework (WSRF) [8], standardized by OASIS, is a significant step that will allow the utilization of standard Web Services technologies, which enjoy large scale industry support, for Grid computing.

## 1.2 The growing importance of IT in delivering efficient, high quality healthcare

The advent of HealthGrid applications, even at the research stage, coincides with a crucial period of investment and experimentation in IT for healthcare. The main drivers for this shift in the pace and levels of investment include:

- increased understanding of the impact of medical errors on patient safety and the resulting cost of care [40]. IT's basic value proposition includes the ability to regulate processes and scale information "written" once to many uses and contexts;
- demand for healthcare outstripping resources at all levels, driven by an ageing population in most countries, living longer but with access to an increasingly sophisticated armoury of tests, surgical interventions, medications etc. IT has the power to both add to the armoury of clinical tools and reduce costs through efficient

operation with reduced numbers of process steps, wasted activity (tests, prescribing unnecessarily etc.) and utilisation of disparate resources.

- the completion of the Human Genome Project sparked the development of many new tools for today's biomedical researcher to use in finding the mechanism behind disease. While the goal is clear, the path to such discoveries has been fraught with roadblocks in terms of technical, scientific, and sociological challenges. The breadth and depth of information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality, improve therapies and meet the demanding need of individualized care.

Capitalizing on the opportunities apparent to the clinical and research community requires the integration and exploitation of data and information generated at all levels, from molecular to organ and disease to the population.

## *1.3 Health-GRIDS*

It is widely accepted that there is a pressing need to move away from manual management of patient information to digital records. Technology to secure this information can also be complex and expensive to deploy. Moreover, access to many different sources of medical data, usually geographically distributed, and the availability of computer based tools that can extract the knowledge from that data are key requirements for providing a standard healthcare provision of high quality. Research projects in the integration of bio-medical knowledge, advances in imaging, development of new computational tools and the use of these technologies in diagnosis and treatment suggest that Grid-based systems can make a significant contribution to this goal.

A Health-Grid should be an environment where data of medical interest can be stored, processed and made easily available to the different healthcare participants: researchers, physicians, healthcare centres and administrations, and in the long term perspective citizens. If such an infrastructure were to offer all necessary guarantees in terms of security, respect for ethics and observance of regulations, it would allow the association of post-genomic information and medical data and open up the possibility for individualized healthcare. Health-Grids are Grid infrastructures comprising applications, services or middleware components that deal with the specific problems arising in the processing of biomedical data. Resources in Health-Grids are databases, computing power, medical expertise and even medical devices.

The future evolution of Grid technologies addresses most precisely problems that are very appropriate for Healthcare. Some of the issues that are identified as key features of Health-Grids are: Health-Grids are more closely related to data, but many hospitals are reluctant to let the information flow outside the hospital bounds. For a large-scale deployment of Health Grids it is important to leverage security up to a trustworthy level of confidence that could release a generalized access to data from the outside. Security in Grid infrastructures is sufficient for research, but it must be improved in the future to ensure privacy of data. Management of Distributed Databases and Data Mining capabilities are important tools for many biomedical applications in fields such as epidemiology, drug design or even diagnosis. Robustness and Fault Tolerance of Grids fits very well to the needs for 'always on' medical applications. Grid technologies can ease the access to replicated resources and information, just requiring the user to have a permanent Internet connection. Research communities in biocomputing or biomodelling and simulation have a strong need for resources that can be provided through the Grid. Finally, flexibility is needed for the control of VOs at a large level.

The management of resources should be more precise and dynamic, depending on many criteria such as urgency, users' authorization or other administrative policies.

## 1.4 Critical opportunities for distributed computing approaches

Of course, not all Healthcare informatics problems will be remotely suitable for a Grid solution. The characteristics of clinical problems that could have significant advantage from distributed computing-type solutions include:

- analyses that require dynamically assembled data-sets and investigation routines; for instance genetic-related investigations where the initial analysis may raise the need for further data sets to be added to give better, more representative results from analysis;
- processes of analysis and data assembly that cross organisational boundaries, where the ability to distribute both the data and analysis without recall to the normal "data process" flows is key. Again medical research, or in future patient-centric analyses, are probably two areas where the utility of the Grid will be highest;
- huge scale analysis, that requires a scalable infrastructure to deal with the potentially massive quantities of data both to be assembled and analysed. This leads us again to imaging and genetic analysis as potential opportunities;
- dynamic grouping of Healthcare professionals for review / analysis of diagnosis or research results, such that different "expert teams" can be assembled without a formal organisation structure (indeed across organisation structures). Feedback from clinicians on existing Grid health projects indicates a strong need to enhance collaboration on a daily basis between communities, removing their reliance on conferences to achieve this.
- Further benefits may be realised through the pooling of resources, whether it be the sharing of training cases to enable smaller clinics to benefit from the knowledge available in larger hospitals, or the sharing of compute resource to reduce the local investment on IT.

Therefore in summary there seems to be an advantage available from using Grid approaches where the clinical problem requires a scaleable, flexible infrastructure that can work across normal organisation and process boundaries.

*From the above list of potential areas where Grid technology is expected to provide cost-beneficial solutions for the healthcare domain, this master thesis focuses on the design and development of an experimental Grid infrastructure for access to heterogeneous, distributed clinical data and the implementation of services supporting efficient data mining operations, utilizing the computational power of the grid.*

# 2 Subject of the thesis and state of the art

## *2.1 State of the art*

Grid systems and applications aim to integrate, virtualise, and manage resources and services within distributed, heterogeneous, dynamic "virtual organizations". The realization of this goal requires the disintegration of the numerous barriers that normally separate different computing systems within and across organizations, so that computers, application services, data, and other resources can be accessed as and when required, regardless of physical location. Grid computing provides a novel approach to harnessing distributed resources, including applications, computing platforms or databases and file systems. Applying grid computing can drive significant benefits by improving information access and responsiveness, and adding flexibility, all crucial components of solving the data warehouse dilemma. These promises of grid computing are the reason why many biomedical projects have chosen to base their systems and application on grid technology.

The Open Grid Service Architecture (OGSA) defines mechanisms for creating, managing, and exchanging information among entities called Grid services. Succinctly, a Grid service is a Web service that conforms to a set of conventions (interfaces and behaviours) that define how a client interacts with a Grid service. These conventions, and other OGSA mechanisms associated with Grid service creation and discovery, provide for the controlled, fault-resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications.

Examples of biomedical grid-enabled project developed over the last years are the following:

### Cancer Biomedical Informatics Grid (caBIG)

The Cancer Biomedical Informatics Grid [17] is a project initiated by the United States National Cancer Institute to create a computational network connecting scientists and institutions to enable the sharing of data and the use of common analytical tools. The emergence of genomics and proteomics high-throughput technologies are creating a paradigm shift in biomedical research from small independent labs to large teams of researchers exploring entire genomes and proteomes and how they relate to disease. CaBIG is developing new software and modifying existing software within Clinical Trials Management Systems, Tissue Banks and Pathology Tools and Integrated Cancer Research tools to manage the huge volume of data being generated and to facilitate collaboration across the broad spectrum of cancer research.

CaBIG aims to create a virtual network (or grid) that will become the "World Wide Web of Cancer Research" that links individuals and institutions both nationally and internationally. NCI and over 50 of its designated Cancer Centers have formed the first group of developers and adopters for numerous projects currently underway. The project is managed by the NCI Center for Bioinformatics (NCICB). This scenario is depicted in the figure below:

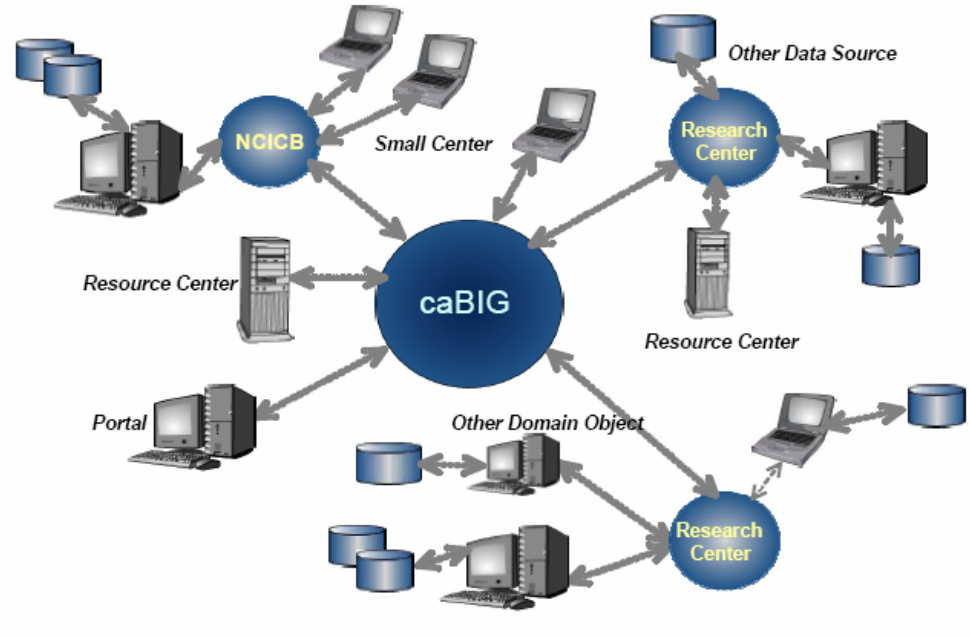



**Figure 1:** Schematic of a hypothetical grid demonstrating the connectivity the project will achieve

The caBIG project is developing a grid architecture, caGRID which is based on the Globus 4.0 grid framework. Other technologies that are included in caGRID are Globus security, OGSA-DAI as a grid data layer for exposing data services and GRAM (Grid Resource Allocation and Management) for execution of jobs (Fig. 2). A critical feature of caGRID is its attempts to create semantic interoperability between data resources. As part of the grid architecture, the caDSR (cancer Data Standards Repository) and the EVS (Enterprise Vocabulary System) are central parts in creating the semantic interconnection.

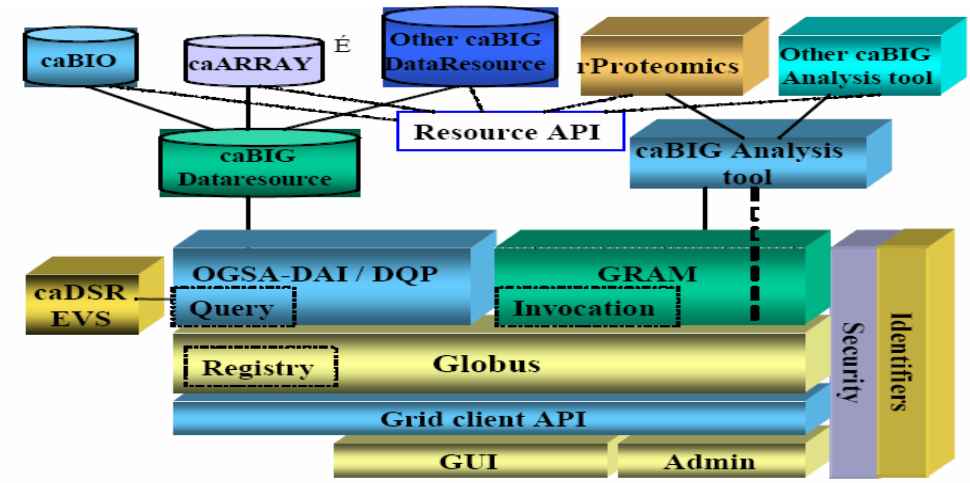



**Figure 2:** Diagram of the proposed grid architecture currently under development as part of the caBIG

## myGRID

The myGrid [18] project aims to exploit the growing interest in Grid technology, with an emphasis on the Information Grid, and provide middleware layers that make it appropriate for the immediate needs of bioinformatics.

Specifically, myGrid is building high level services for data and application resource integration such as resource discovery, workflow enactment and distributed query processing. However, these services merely enable experiments to be formed and executed. Additional services are needed to support the e-based scientific method and best practice found at the bench but often neglected at the workstation, notably provenance management, change notification and personalisation.

Bioinformatics already offers a huge selection of data and analytical resources for a biologist to perform in silico experiments. In such experiments services representing tools act upon data, producing more data until a goal is achieved or hypothesis revealed. With current tool it possible to reveal interesting biological insights computationally. A major barrier however in utilizing these resources is the time needed by skilled bioinformaticians to manually and repeatedly co-ordinate multiple tools to produce a result. Tasks that take minutes in computational time actually take days to run manually. myGRID middleware provides services to create and manage the information from running in silico bioinformatics experiments in a semantically enriched Grid aware environment.

## GridMiner

Knowledge discovery in data sources available on Computational Grids is a challenging research and development issue. The GridMiner project [32] at the University of Vienna aims, as the first Grid research effort, to cover all aspects of the knowledge discovery process and integrate them as advanced service-oriented Grid application. The innovative architecture provides a robust and reliable high performance data mining and OLAP environment and strengths the importance of Grid enabled applications in terms of business intelligence and detailed analysis of very large scientific data sets. The interactive cooperation of different services – data integration, data selection, data transformation, data mining, pattern evaluation and knowledge presentation - within the GridMiner architecture is the key to high performance knowledge discovery on large datasets.

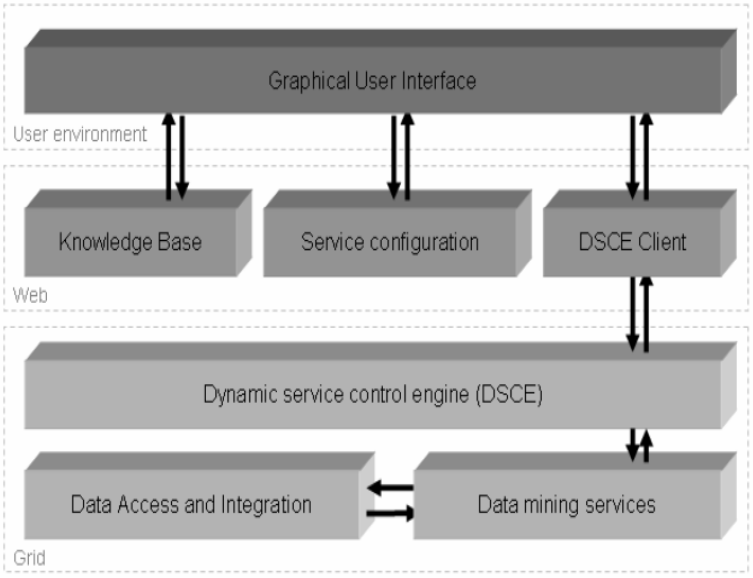



**Figure 3:** 3-layered architecture of Gridminer

It is not a simple matter to develop an integrative approach that exploits synergies between knowledge management and knowledge discovery in order to monitor and manage the full

lifecycle of knowledge and provides services quickly, reliably and securely. Several Grid research activities addressing some facets of this process have already been reported. The technology developed is being validated and tested on an advanced medical application addressing treatment of traumatic brain injury (TBI) victims. Medicine is just one application area where an environment is needed for continuous knowledge discovery and management.

The aim of the GridMiner application is to give to an expert (Dataminer) a tool which can ease the knowledge discovery process in the distributed Grid environment. So it is essential that the system provides a powerful, flexible and simple to use graphical user interface (GUI) which hides the complexity of the Grid but is still offering possibilities to interfere during the execution phase, control the task execution and visualize results.

Figure 3 shows a high-level and abstract view of the components in GridMiner's architecture and how they are connected, as it has been implemented in the current GridMiner prototype. The GridMiner is a service oriented application and has been implemented as a research prototype completely built on top of the Globus Toolkit Version 3 and standard web technologies.

## 2.2 Architecture of our experimental Grid infrastructure

Having studied the approaches of some of the most advanced efforts worldwide in applying Grid technology to the solution of selected biomedical problems, we have defined the architecture of our experimental Grid infrastructure and selected the services required to support the knowledge extraction scenarios at hand.

Fig. 4 shows a view of the components in our architecture and how they are connected.

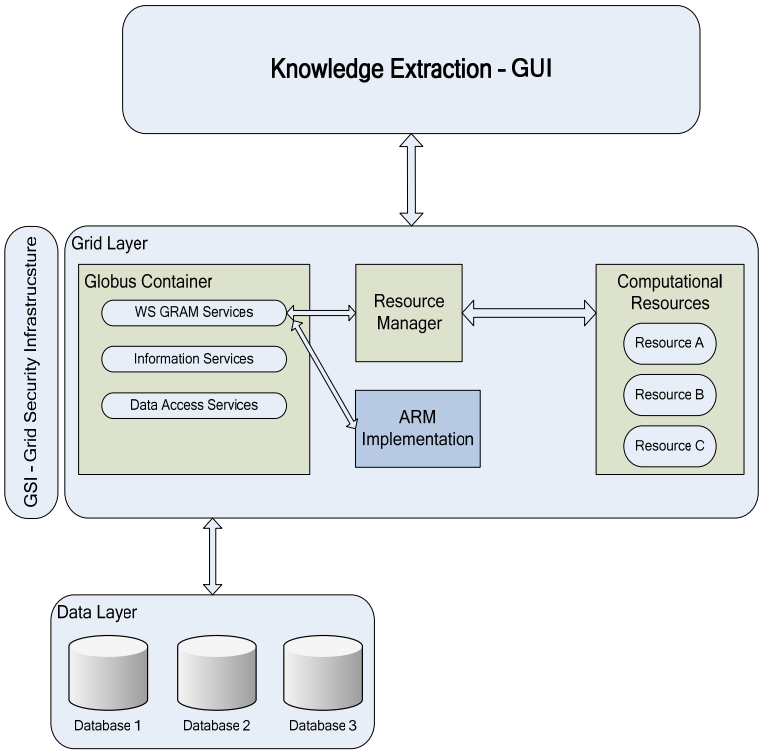



**Figure 4:** Architecture of the experimental grid environment

The Grid Layer provides the necessary computational resources to execute the parallel implementation of the Association Rules Mining (ARM) algorithm. WS-GRAM services are responsible for submitting, monitoring and cancelling jobs to computational resources. Access

to various clinical information systems is possible through data services. These services use the OGSA-DAI interface to communicate with the database of a specific information system. Information Services allow clients to discover the resources and services available in the Grid. Finally data mining tools, like the parallel ARM algorithm is made accessible to users through Data Mining Services.

Over the Grid Layer we have the Knowledge Extraction - GUI layer, from which the user has the ability to perform the following actions:

- Query for available computational resources
- Retrieve clinical data from information systems. This retrieval is made possible through the OGSA-DAI interface and the data received are in the form of XML document.
- Initialize the parameters for the ARM algorithm and submit the job to the grid layer.
- Visualization of results to the user.

Every component of the above architecture will be discussed in details in the following chapters.

## 2.3 Knowledge Extraction

Knowledge discovery from medical data can find interesting hidden pattern and trends that could be very useful in life sciences. Unfortunately the enormous size of medical data repositories makes this procedure extremely compute-intensive and slow. On this master thesis we focus in Association Rules Mining which is a well known data mining technique, able to find association rules between items. The data containing the medical information are in the form of XML documents. This type of mining is called XML-Content mining which requires a single DTD schema for the XML documents.

The drawback of ARM is that the number of possible association rules grows exponentially with the number of items considered. For 1000 items, for example, more than $2^{1000}$ have to be considered in a naive approach. So, it is clear why this algorithm has a great need for computational power.

## 2.4 Structure of this report

The rest of this report has the following structure:

- Chapter 3 describes with details the software components that comprise the Grid infrastructure. We explain various key concepts regarding Globus Toolkit 4 and Web Service Resource Framework (WSRF). We also present the functionality of the following GT4's components: Grid Resource & Allocation Management (GRAM), Monitoring and Discovery System (MDS4), Grid Security Infrastructure (GSI) and OGSA – Data Access & Integration (OGSA-DAI).
- Chapter 4 introduces association rules mining more formally and provides an overview of previous algorithms.
- Chapter 5 explains with details the experimental setup of the grid system. We also present our implementation the Association Rule Mining algorithm and the Graphical User Interface.
- Chapter 6 discusses issues for developing data mining tools as grid services in a wider context.
- Chapter 7 describes interesting extensions of our work and draws the final conclusions.

# 3 Grid Layer

## 3.1 Globus Toolkit 4 – Key Concepts (OGSA, WSRF & Web Services)

The Globus Toolkit [5] is an open source software toolkit developed by the Globus Alliance and many others. It is currently the most popular middleware for building open grids. The Globus Toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability. The purpose of the toolkit is to remove obstacles that prevent different (differently configured, using different software) sites to collaborate in the Grid. Its core services, interfaces and protocols allow users to access remote resources as if they were located within their own machine room while simultaneously preserving local control over who can use resources and when.

The toolkit first and foremost, includes quite a few high-level *services* that we can use to build Grid applications. These services, in fact, meet most of the abstract requirements of OGSA. The Open Grid Services Architecture (OGSA), developed by the Global Grid Forum (GGF), aims to define a common, standard, and open architecture for grid-based applications. The goal of OGSA is to standardize practically all the services one commonly finds in a grid application (job management services, resource management services, security services, etc.) by specifying a set of standard interfaces for these services. In order to create this new architecture the GGF needed some sort of distributed middleware on which to base the architecture. This base for the architecture could, in theory, be any distributed middleware (CORBA, RMI, or even traditional RPC) but for reasons that will be explained further on, Web Services were chosen as the underlying technology.

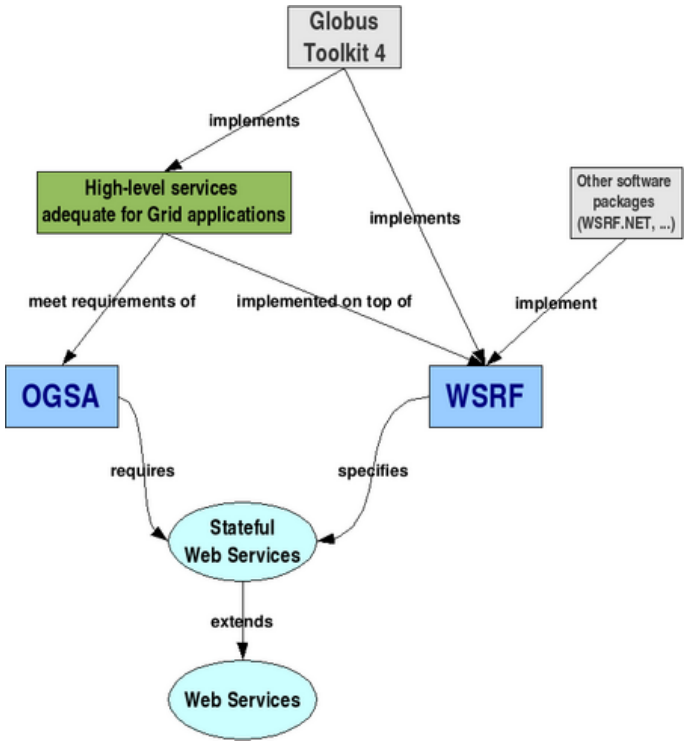



**Figure 5:** Relationship between OGSA, GT4, WSRF and Web Services

However, although the Web Services architecture was certainly the best option, it did not meet one of OGSA's most important requirements: the underlying middleware had to be stateful. Unfortunately, Web services are usually stateless and there is no standard way of

making them stateful. In order to overcome this problem OASIS developed the Web Services Resource Framework (WSRF) which specifies how Web Services can become stateful, along with adding a lot of other useful features. So WSRF provides the stateful services that OGSA needs.

So, Globus Toolkit provides grid services that meet the requirements of the Open Grid Service Architecture and are implemented on top of the Web Service Resource Framework. In fact Globus Toolkit 4 (GT4) includes a complete implementation of the WSRF specification. Figure 5 shows the relationship between OGSA, GT4, WSRF and Web Services.

### 3.1.1 Web Services

Web Services are a distributed computing technology that allows us to create client-server applications. Figure 6 shows a very simple Web Service that has the ability to distribute weather information given a zip code. The clients that want to access the weather information would contact the Web Service and send a service request. The server would return the forecast through a service response. Of course this is a very simplistic approach of a Web Service. In reality the whole procedure is more complicated.

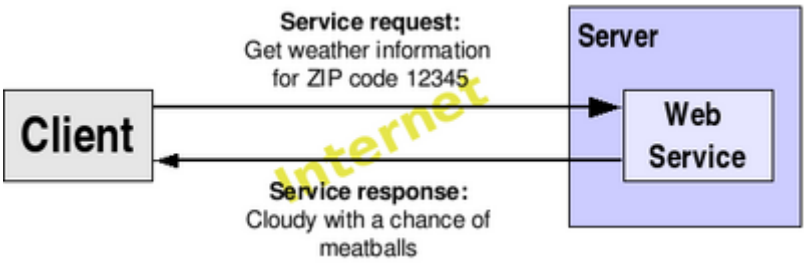



**Figure 6:** Web Services

This simple scenario can be implemented with other distributed technologies like CORBA, RMI or EJBs but Web Services has some very important advantages over these technologies. Web Services are platform-independent and language-independent, since they use standard XML languages. This means that the client program can be programmed in C++ and running under Windows, while the Web Service is programmed in Java and running under Linux. Most Web Services use http to transmit messages (such as service request and response). This is a major advantage if we want to build Internet-scale applications, since most of Internet's proxies and firewall want mess the http traffic.

The most important characteristic that distinguishes Web Services is that technologies such as CORBA and EJB are geared towards highly coupled distributed systems, where the client and the server are very dependent on each other. On the other hand, Web Services are more adequate for loosely coupled systems, where the client might have no prior knowledge of the Web Service until it actually invokes it. Highly coupled systems are ideal for intranet applications, but perform poorly on an Internet scale. So, Web Services are better suited to meet the demands of an Internet-wide application, such as grid-oriented applications.

### 3.1.2 WSRF: The Web Services Resource Framework

Web Services are the the technology of choice for Internet-based applications with loosely coupled clients and servers. That makes them the natural choice for developing grid-based applications. But plain Web Services have certain limitations. The goal of WSRF [8] is to overcome these limitations and provide the framework for building grid-applications.

The basic problem with Web Services is that they are stateless. This means that a Web Service can't "remember" information, or keep state, from one invocation to another. This fact

is not necessarily a bad thing. There are plenty of applications which have no need whatsoever for statefulness. However grid applications do generally require statefulness. So, we would ideally like our Web service to somehow keep state information. Of course a "statefull Web Service" is a bit of a contradiction in terms

Giving Web services the ability to keep state information while still keeping them stateless is a complex problem. The solution is simply to keep the Web service and the state information completely separate. Instead of putting the state in the Web service (thus making it stateful, which is generally regarded as a bad thing) keep it in a separate entity called a resource, which will store all the state information. Each resource will have a unique key, so whenever we want a stateful interaction with a Web service we simply have to instruct the Web service to use a particular resource. Figure 7 below, shows the resource approach to statefulness.

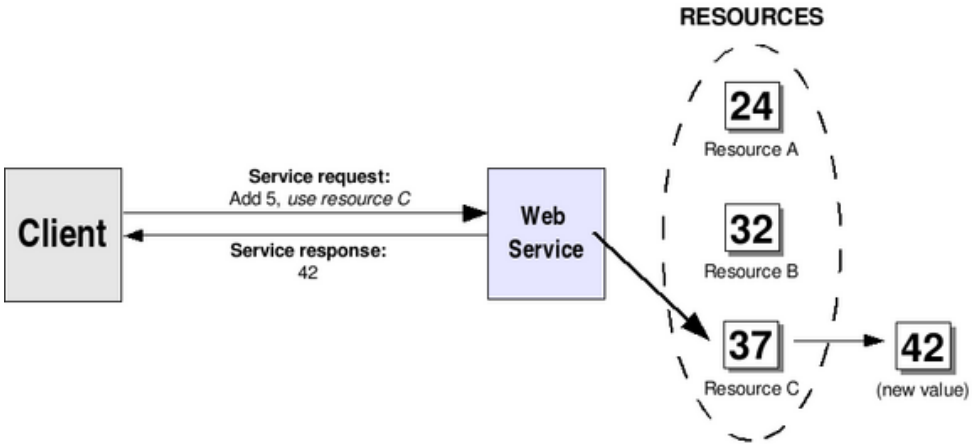



**Figure 7:** The resource approach to statefulness

Of course, resources can come in all different shapes and sizes. A resource can keep multiple values not just a simple integer value, as shown in the previous figure. A pairing of a Web Service with a Resource is called a WS-Resource. The address of a particular WS-Resource is called an endpoint reference.

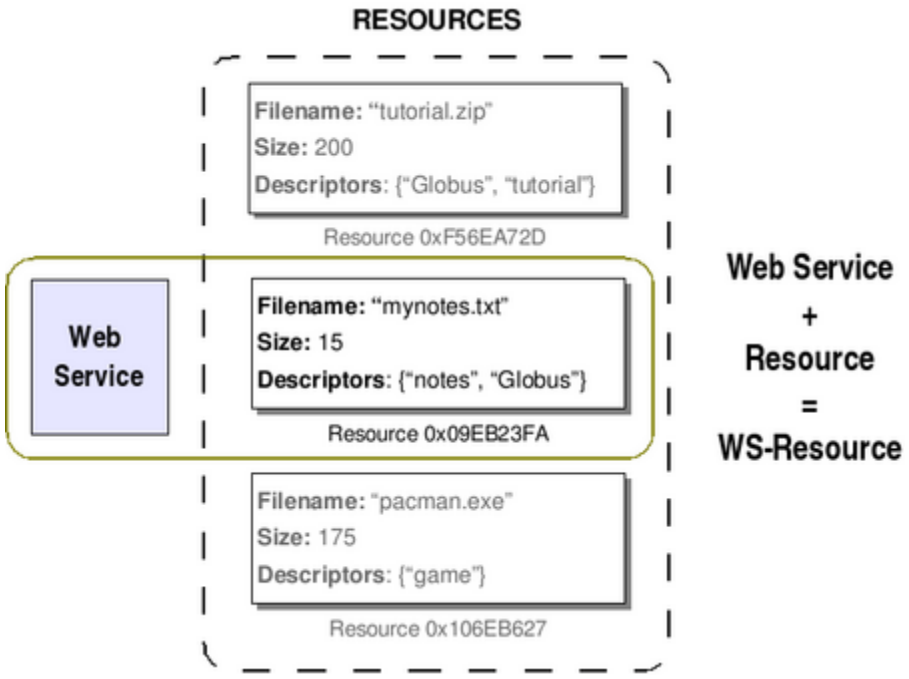



**Figure 8: WS-Resource**

### 3.1.3 Globus Toolkit 4 Architecture

WSRF is only a small, but important part of the whole GT4 Architecture. It is the infrastructure on top of which most of the toolkit is built. Besides the WSRF implementation, the toolkit includes a lot of components which can be used to program Grid applications. The Globus Toolkit 4 [5] is composed of several software components. These are divided in the following five categories:

- **Common Runtime:** The Common Runtime components provide a set of fundamental libraries and tools which are needed to build both WS and non-WS services.
- **Security:** Using the Security components, based on the Grid Security Infrastructure (GSI), we can make sure that our communications are secure.
- **Data Management:** These components will allow us to manage large sets of data in our virtual organization.
- **Information Service:** The Information Services, more commonly referred to as the Monitoring and Discovery Services (MDS), includes a set of components to discover and monitor resources in a virtual organization.
- **Execution Management:** Execution Management components deal with the initiation, monitoring, management, scheduling and coordination of executable programs, usually called jobs, in a Grid.

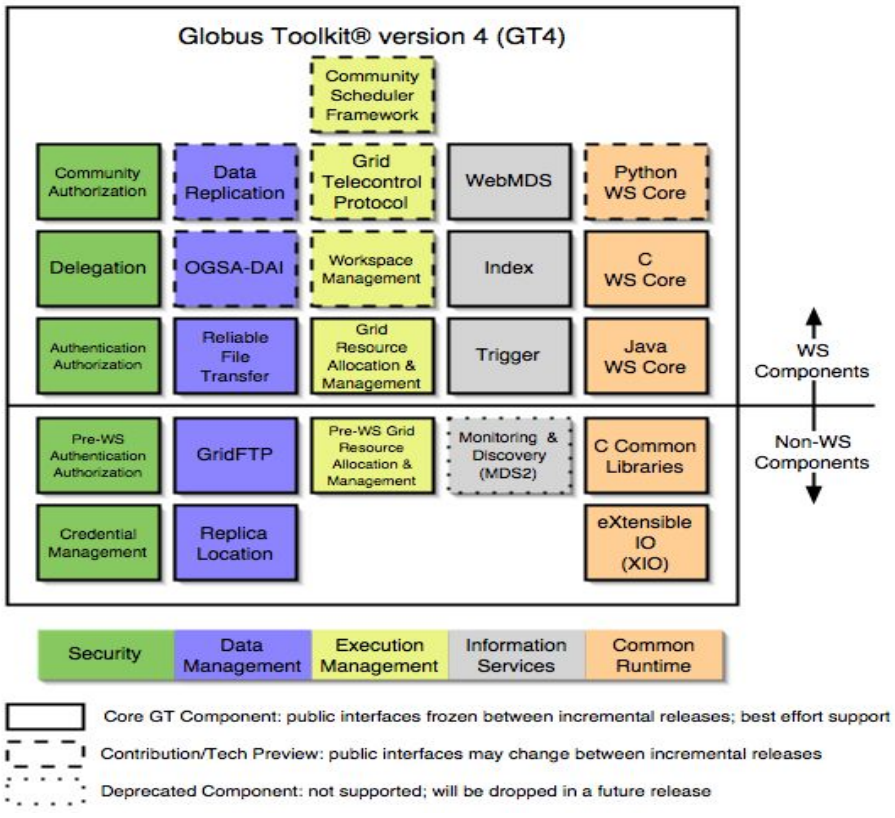



**Figure 9:** Globus Toolkit 4 Architecture

In this master thesis we are interested in utilizing the computational resources a Grid can provide. The component for resource management is WS-GRAM which will be described with details in the following section.

## 3.2 Web Services – Grid Resource Allocation & Management (WS-GRAM)

WS-GRAM is the component of Globus Toolkit 4 responsible for allocating jobs to resources that we have already set up and governed by local schedulers. The integration of WS-GRAM component of the Globus Toolkit with local schedulers allows using the resources governed by those schedulers in one, uniform way based on open protocols [34]. This way we are not interested in specifics of how the resources are governed locally and, even in heterogeneous environment, always use resources through the same interface.

More specifically, WS-GRAM provides just the interface that can be used to access underlying resources. The open, uniform interface is what GRAM is all about but that means WS-GRAM does not provide other important functionalities such as management of workload between many machines. The integration of WS-GRAM and local scheduler is very straightforward on condition that we already have the implementation of the WS-GRAM scheduler interface for that specific scheduler. So, in short, given an RSL (Resource Specification Language) job description WS-GRAM submits the job to a scheduling system such as PBS and monitors it until completion.

### 3.2.1 GRAM Overview

Execution management tools are concerned with the initiation, monitoring, management, scheduling, and/or coordination of remote computations. GT4 supports the Grid Resource Allocation and Management (GRAM) interface as a basic mechanism for these purposes. The GT4 GRAM server is typically deployed in conjunction with Delegation and RFT servers to address data staging, delegation of proxy credentials, and computation monitoring and management in an integrated manner.

Associated tools fall into three main classes. First, we have GRAM-enabled schedulers for clusters or other computers on a local area network (Condor , OpenPBS, Torque, PBSPro, SGE, LSF). Second, we have systems that provide different interfaces to remote computers (OpenSSH) or that implement various parallel programming models in Grid environments by using GRAM to dispatch tasks to remote computers (Condor-G, DAGman, MPICH-G2, GriPhyN VDS, Nimrod-G). Third, we have various “meta-schedulers” that map different tasks to different clusters (CSF, Maui).

The Globus Toolkit provides a suite of Web services with which clients can interact to submit, monitor, and cancel jobs on local or remote computing resources. This system is termed collectively “GRAM” or sometimes “WS_GRAM” to distinguish it from the pre-Web services system that offers similar functionality.

Imagine that you want to allow remote users to execute a program. One approach is to make this program accessible as a Web service. To do this, you define and implement a Web service operation that executes your program. Then, for example, a client might simply call “foo(a,b)” to request that “foo” be executed on a local or remote computer with “a” as input and returning “b.”

This simple approach to enabling remote access to programs can be quite appropriate in many settings. However, there are also requirements that this simple approach does not address:

- *State*. The computational task that is to be run (the “job”) may perform input/output operations while running that affect the state of the computational resource on which the job runs, and/or its associated filesystems. Thus, “exactly once” execution semantics are important: a user cannot simply resubmit a request if no

acknowledgement was received, as the job may have completed but the acknowledgement message was lost.

- *User executables*. We may wish to allow users to supply the programs to be executed.
- *Staging of input and output*. Executables, input data, and output data may be large, remote, and/or shared between different invocations. The ability to manage staging of this data can be important.
- *Streaming output*. Some users, particularly interactive ones, benefit from accessing output data files as the job is running.
- *Control*. A user may require the ability to terminate a job that consumes many resources.
- *Schedulers*. Larger computing resources are typically operated under the control of a scheduler that implements allocation and prioritization policies while optimizing the execution of all submitted jobs for efficiency and performance.
- *Monitoring*. Data staging and schedulers introduce the potential for more complex job state transitions: a job may not simply be running, completed, or failed, but may be pending, suspended, staging, and so forth. Some users require the ability to query and/or subscribe to determine job state.

The GRAM system and its client software have been developed to address these issues. *GRAM is meant to be used in situations where the ability to run arbitrary programs, achieve reliable operation, perform stateful monitoring, manage credentials, stage files, and interact with schedulers are important*. If these capabilities are not required, then GRAM may not be appropriate. If an application has only modest input and output data, operates in a stateless manner (i.e., all that matters is the result data or fault signal), and will be invoked many times, it may be a good candidate for exposure as an application-specific Web service.

The GRAM interface addresses this need for advance management functionality by creating for each successful job submission a stateful entity—a ManagedJob—on the compute host, with lifetime similar to that of the associated job. (In fact, the lifetime is typically somewhat longer, so that a client can determine a job's state even after it has terminated.) A successful GRAM "submit" operation returns a handle—specifically, a WS-Addressing endpoint reference, or EPR—for the new ManagedJob. A client can then use this handle to query the job's status, kill the job, and/or "attach" to the job to obtain notifications of changes in job status and output produced by the job. The client can also communicate this handle to other clients, who can perform the same operations if authorized. It is this ManagedJob construct that allows GRAM to support many of the advanced features mentioned in the preceding subsection.

A key notion here is that GRAM is not a resource scheduler, but rather a protocol engine for communicating with a range of different local resource schedulers using a standard message format. The GT4 GRAM implementation includes interfaces to Condor, LSF, SGE, and PBS schedulers, as well as a "fork scheduler" that simply forks a new Unix process for each request. The system administrator must perform appropriate local configuration operations for these local scheduler interfaces.

### 3.2.2 WS-GRAM Architecture

Figure 10 below shows the WS-GRAM Architecture [5].

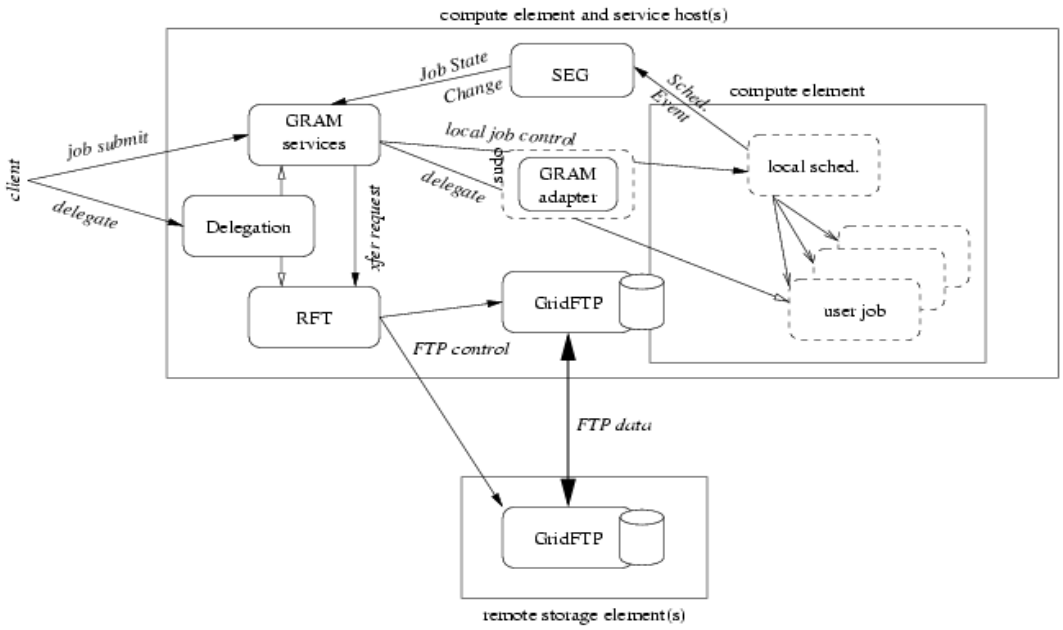



**Figure 10:** WS-GRAM Architecture

The heart of the WS GRAM service architecture is a set of Web Services designed to be hosted in the Globus Toolkit's WSRF core hosting environment. These services, as shown in figure 11 below, make use of platform-specific callouts to other software components of the Globus Toolkit.

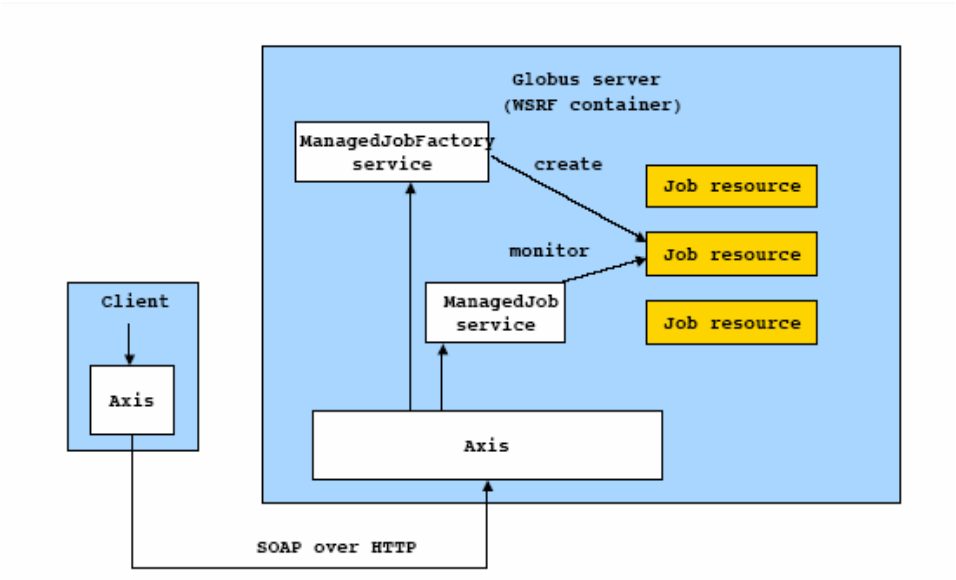



**Figure 11:** Web Services of WS-GRAM

- ***ManagedJob:*** Each submitted job is exposed as a distinct resource qualifying the generic ManagedJob service. The service provides an interface to monitor the status of the job or to terminate the job (by terminating the ManagedJob resource). The

behavior of the service, i.e. the local scheduler adapter implementation, is selected by the specialized type of the resource.

- ***ManagedJobFactory:*** Each compute element, as accessed through a local scheduler, is exposed as a distinct resource qualifying the generic ManagedJobFactory service. The service provides an interface to create ManagedJob resources of the appropriate type in order to perform a job in that local scheduler.

## Globus Toolkit Components used by WS-GRAM

### Reliable File Transfer Service

The ReliableFileTransfer (RFT) service of GT 4.0 is invoked by the WS-GRAM services to effect file staging before and after job computations. The integration with RFT provides a much more robust file staging manager than the ad-hoc solution present in previous versions of the GRAM job manager logic. RFT has better support for retry, restart, and fine-grained control of these capabilities. WS GRAM exposes the full flexibility of the RFT request language in the job staging clauses of the job submission language.

### GridFTP

GridFTP servers are required to access remote storage elements as well as file systems accessible to the local compute elements that may host the job. The ReliableFileTransfer Web service acts as a so-called third-party client to the GridFTP servers in order to manage transfers of data between remote storage elements and the compute element file systems. It is not necessary that GridFTP be deployed on the same host/node as the WS-GRAM services, but staging will only be possible to the subset of file systems that are shared by the GridFTP server that is registered with the WS-GRAM service. If no such server or shared file systems are registered, staging is disallowed to that WS-GRAM compute element. GridFTP is also used to monitor the contents of files written by the job during job execution. The standard GridFTP protocol is used by a slightly unusual client to efficiently and reliably check the status of files and incrementally fetch new content as the file grows. This method supports "streaming" of file content from any file accessible by GridFTP, rather than only the standard output and error files named in the job request. This approach also simplifies failover and restart of streaming to multiple clients. The integration with GridFTP replaces the legacy GASS (Globus Access to Secondary Storage) data transfer protocol. This changeover is advantageous both for greater performance and reliability of data staging as well as to remove redundant software from the GRAM codebase.

### Delegation Service

The Delegation service of GT 4.0 is used by clients to delegate credentials into the correct hosting environment for use by WS-GRAM or RFT services. The integration of the Delegation service replaces the implicit, binding-level delegation of previous GRAM solutions with explicit service operations. This change not only reduces the requirements on client tooling for interoperability purposes, but also allows a new separation of the life cycle of jobs and delegated credentials. Credentials can now be shared between multiple short-lived jobs or eliminated entirely on an application-by-application basis to make desired performance and security tradeoffs. Meanwhile, for unique situations WS GRAM retains the ability to refresh credentials for long-lived jobs and gains an ability to designate different delegated credentials for different parts of the job's life cycle.

### External Components used by WS-GRAM

**Local Job Scheduler**

An optional local job scheduler is required in order to manage the resources of the compute element. WS-GRAM has the ability to spawn simple time-sharing jobs using standard Unix fork() methods, but most large-scale compute elements will be under the control of a scheduler such as PBS, LSF, e.t.c.

**Sudo**

The de facto standard Unix sudo utility is used by WS-GRAM to gain access to target user accounts without requiring WS-GRAM to have general super-user privilege on the system. The sudo command is used to execute WS GRAM adapter tools in the user account context; these adapters perform the local system-specific operations needed to initialize and run user jobs. The sudo utility not only provides a simple way for WS-GRAM to run programs as other users without "root" privilege, but it provides fine-grained controls for the system administrator to restrict which user accounts are valid WS-GRAM targets as well as secure auditing of all operations requested by WS-GRAM. This mechanism replaces the root-privileged Gatekeeper component of the Pre-WS GRAM service in order to avoid running an entire WSRF hosting environment as root. This change provides enhanced security, at the expense of slightly more complicated deployment effort, and is motivated by the relative increase in the size of the WS GRAM and WSRF codebase as compared to the traditional Gatekeeper.

## *3.3 Information Services (MDS4)*

In a Grid environment, the set of resources available for use by a virtual organization can change frequently – new resources and services (compute servers, file servers) may be added; old ones may be removed; capacity may be increased or decreased; and basic properties of a resource or service may change. For example, a data server may be upgraded to one with larger capacity, different access rates, and different access protocols. Because these systems are so dynamic in nature, discovery – the process of finding suitable resources to perform a task – can be a significant undertaking. Similarly, monitoring – the process of observing resources or services to track their status for purposes such as fixing problems and tracking usage – can be more complicated in Grids because of the dynamic, distributed nature of these environments.

The Globus Toolkit's solution to these closely related problems is its Monitoring and Discovery System (MDS) [16]. MDS4 is a suite of web services to monitor and discover resources and services on Grids. This system allows users to discover what resources are considered part of a Virtual Organization (VO) and to monitor those resources. MDS4 includes two WSRF-based services: an Index Service, which collects data from various sources and provides a query/subscription interface to that data, and a Trigger Service which collects data from various sources and can be configured to take action based on that data.

Grid computing resources and services can advertise a large amount of data for many different use cases. MDS4 was specifically designed to address the needs of a Grid monitoring system – one that publishes data that is of use by multiple people across multiple administrative domains. As such, it is not an event handling system, as is NetLogger, or a cluster monitor in its own right, as is Ganglia, but can interface to more detailed monitoring systems and archives, and can publish summary data using standard interfaces.

**Web Services standards used by MDS4**

Grid computing resources and services can advertise a large amount of data for many different use cases. The best way to leverage a monitoring infrastructure is to have basic interfaces and monitoring functionality as part of every service in a standard way. In this way basic monitoring and discovery data would become part of the core of every service, not an exception to the rule. Several Web Services standards have emerged to address the interfaces for interacting with service data, including registration, querying, and naming:

- WS-ResourceProperties defines a mechanism by which Web Services can describe and publish resource properties, or sets of information about a resource. Resource property types are defined in the service's WSDL, and the resource properties themselves can be retrieved, in the form of XML documents, using WS-ResourceProperties query operations.
- WS-BaseNotification defines a subscription/notification interface for accessing resource property information.
- WS-ServiceGroup defines a mechanism for grouping related resources and/or services together as service groups.

The Index and Trigger services make extensive use of these standards and the mechanisms defined by them: both use service groups as part of their administrative interface, to keep track of what information they are to collect. The primary client interfaces to the Index Service are resource property queries and subscription/notification.

**MDS4 Services**

The central component in MDS is the Index Service, which collects information about Grid resources and makes this information available. It is similar to a UDDI registry, however it does not have the static limitations of that approach, and allows the last value for every data element to be cached in order to improve query performance. The Index Service interacts with data sources via standard WSRF resource property and subscription/notification interfaces (WS-ResourceProperties and WS-BaseNotification). Any WSRF-based service can make information available as resource properties, however the Index Service collects information from (potentially) many sources and publishes it in one place. Resource properties may be queried by name or via XPath queries.

Administration of the Index Service is done via service groups (WS-ServiceGroup); service group entries describe the mechanisms and associated parameters used to collect data and to hold the collected data itself.

There may be many Indexes available to a Grid user. Each GT4 container has a default Index Service that keeps track of resources that have been created within the container. In addition, sites and virtual organizations often keep track of all the containers, resources, and services that are available to the site or VO in an Index Service. Users need only know the location of a single suitable Index Service in order to discover and monitor all of the resources and services that it indexes.

MDS4 Index Services have a number of features that are sometimes surprising to new users, but are necessary due to Grid scalability and policy issues:

- Index Services can be arranged in hierarchies, but there is no single global Index that provides information about every resource on the Grid. This is deliberate, as each virtual organization will have different policies on who can access its resources. No person in the world is part of every virtual organization!

- The presence of a resource in an Index makes no guarantee about the availability of the resource for users of that Index. An index provides an indication that certain resources are likely to be useful, but the ultimate decision about whether the resources can be used is left to direct negotiation between user and resource. A user who has decided on a particular service to access based on MDS information might still find they are not authorized when they submit a job. This means that MDS does not need to keep track of policy information (something that is hard to do concisely) and that resources do not need to reveal their policies publicly.
- MDS has a soft consistency model. Published information is recent, but not guaranteed to be the absolute latest. This allows load caused by information updates to be reduced at the expense of having slightly older information. This delay is not a problem in practice – for example, it is generally acceptable to know the amount of free disk space on a system 5 minutes ago rather than 2 seconds ago.
- Each registration into an Index Service is subject to soft-state lifetime management. Registrations have expiry times and must be periodically renewed to indicate the continued existence of the resource. This allows each Index to be self-cleaning, with outdated entries disappearing automatically when they cease to renew their registrations.

The MDS-Trigger service, the other higher-level service distributed as part of MDS4, collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, an action takes place, such as emailing a system administrator when the disk space on a server reaches a threshold. This functionality, inspired by a similar capacity in Hawkeye [Haw], has proven useful in trouble shooting for projects such as the Earth Science Grid (ESG) [BBB+05], who used the GT3 version of this service and are in the process of transitioning to the new software

**Aggregator Framework**

The MDS-Index and MDS-Trigger service implementations are both specializations of a more general aggregator framework, a software framework for building services that collect and aggregate data. Services built on this framework are sometimes called aggregator services. Such services have three properties in common.

First, they collect information via aggregator sources. An aggregator source is a Java class that implements an interface (defined as part of the aggregator framework) to collect XML-formatted data. MDS4 supports three types of aggregator source. A Query source uses WS-ResourceProperty mechanisms to poll a WSRF service for resource property information. A Subscription source collects data from a service via WS-Notification subscription/notification. Finally, an Execution source executes an administrator-supplied program to collect information. Figure 12 summarizes how information flows through the aggregator framework.

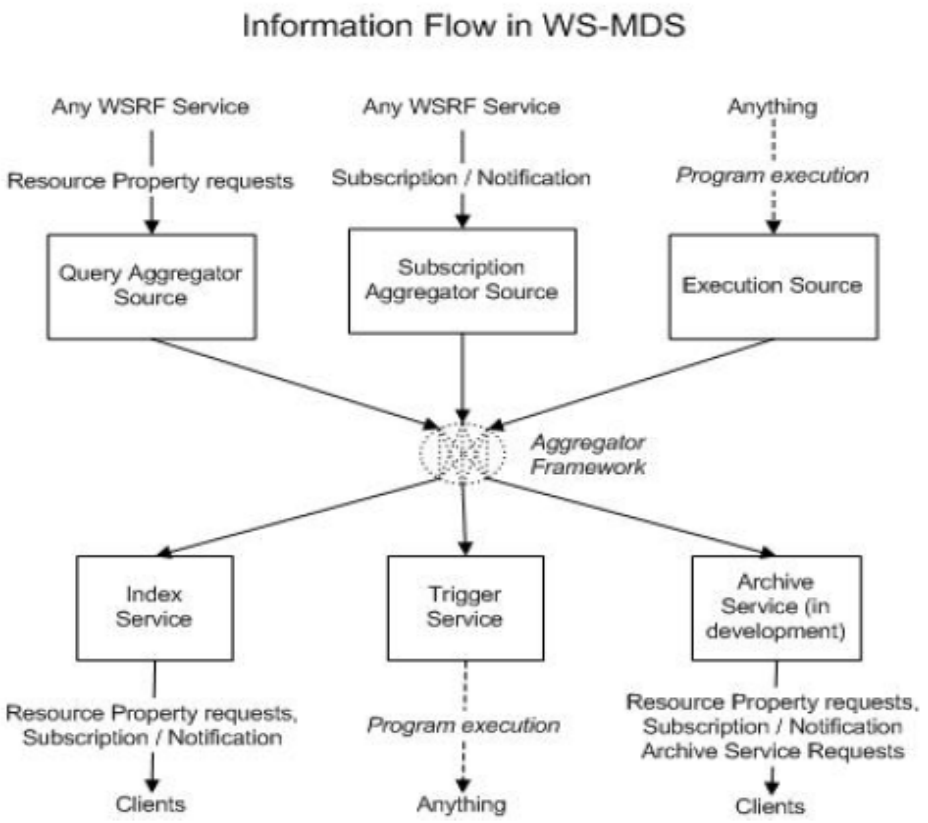



**Figure 12:** Information flow through the MDS4 aggregator framework

Second, aggregator services use a common configuration mechanism to maintain information about which aggregator sources to use and their associated parameters, which generally specify what data to get, and from where. The aggregator framework WSDL defines an aggregating WS-ServiceGroup entry type that holds both configuration information and data. Administrative client programs use standard WS-ServiceGroup registration mechanisms to register these service group entries to the Aggregator Service. Third, aggregator services are self-cleaning – each registration has a lifetime; if a registration expires without being refreshed, it and its associated data are removed from the server.

## Information Providers

The data that an MDS4 aggregator source publishes into its aggregator service is always obtained from an external component called an information provider. In the case of a Query or Subscription source, the information provider is a WSRF-compliant service from which data is obtained via WS-ResourceProperty or WS-Notification mechanisms, respectively. In the case of an execution source, the information provider is an executable program that obtains data via some domain-specific mechanism.

Seven such information providers have been implemented in GT4, as summarized in Table 1. For each provider it is given a name, the resource or service for which data is provided, the type of aggregator source, and the information made available. In all cases the schema is a standard XML document. For Hawkeye, Ganglia and GRAM, information is published using the XML mapping of the GLUE schema.

| Name | Info Source | Source Type | Information Provided |
|---|---|---|---|
| Hawkeye | Condor Pool | Execution | Basic host data (name, ID), processor information, memory size, OS name and version, file system data, processor load data, and other basic Condor host data. |
| Ganglia | Cluster | Execution | Basic host data (name, ID), memory size, OS name and version, file system data, processor load data, and other basic cluster data. |
| GRAM | GT4 grid resource allocation and management service | Query, Subscription | Processor information, memory size, queue information, number of CPUs available and free, job count information, and some memory statistics |
| RFT | GT4 reliable transfer service | Query, Subscription | RFT service status data, number of active transfers, transfer status, information about the resource running the service |
| CAS | GT4 community authorization service | Query, Subscription | Identifies the VO served by the service instance |
| RLS | GT4 replica location service | Execution | Location of replicas on physical storage systems (based on user registrations) for later queries |
| Basic | Every GT4 Web Service | Query, Subscription | ServiceMetaDataInfo element includes start time, version, and service type name |

**Table 1. Information providers included in GT4**

The most important information provider for our work is WS-GRAM which publishes information about the local job scheduler, including:

- Queue information
- Number of CPUs available and free
- Job count information
- Some memory statistics

## 3.4 OGSA – Data Access & Integration

OGSA-DAI [36] is a middleware product that allows data resources, such as relational or XML databases, to be accessed via web services. An OGSA-DAI web service allows data to be queried, updated, transformed and delivered. These services can be deployed within a Grid environment, thereby OGSA-DAI provides the means for users to Grid-enable their data resources.

Using the OGSA-DAI interfaces disparate, heterogeneous data resources may be accessed in a uniform manner. OGSA-DAI exposes data resources through services. Clients interact indirectly with data resources via these services. Each data service exposes zero or more data service resources. A data service resource is the component of OGSA-DAI that through a data resource accessor provides direct access to a single data source (database). This hierarchy is shown in the following figure.

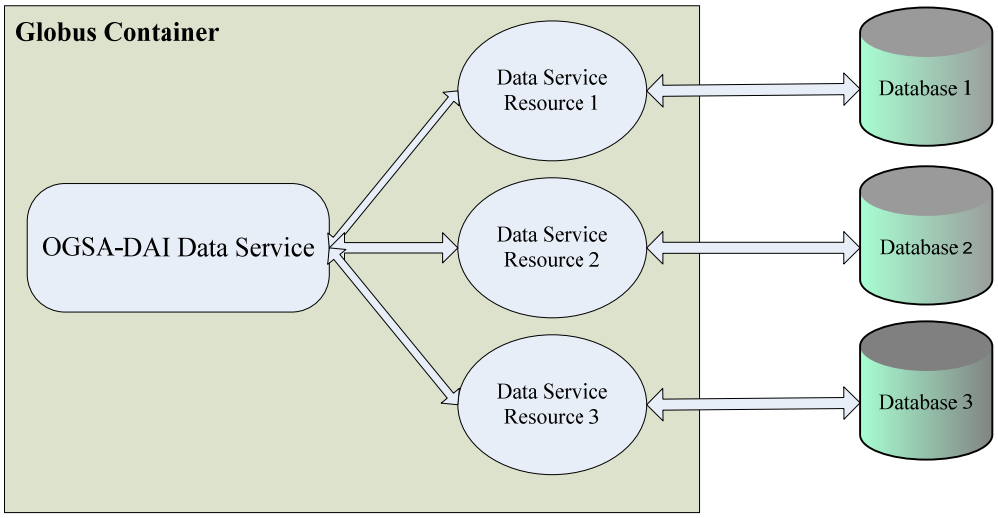



**Figure 13:** A data service resource provides access to an underlying data source

Clients who want to access a specific database communicate directly with the data service. The results of a client query are wrapped in an element which contains the SQL results in web row set format. A WebRowSet document is a XML representation of a SQL RowSet and consists of a properties section, a metadata section and a data section. The metadata gives information about the columns in the result set. The actual data is contained within the *data* element.

## 3.5 GSI – Grid Security Infrastructure

The Grid Security Infrastructure (GSI) [36] is the portion of the Globus Toolkit that provides the fundamental security services needed to support Grids. Security is one of the most important parts of a Grid application. Since a grid implies crossing organizational boundaries, resources are going to be accessed by a lot of different organizations. So access to resources must be allowed only by certain organizations and there must be a way to make sure that those organizations are who they claim to be. There is also a great need for secure communication between elements in a computational grid. This means that there is a need for authentication and in some cases data integrity and privacy.

GSI is composed of a set of command-line tools to manage certificates, and a set of Java classes to easily integrate security into web services. The features provided by GSI are the following:

- Transport-level and message-level security
- Authentication through X.509 digital certificates
- Several authorization schemes
- Credential delegation and single sign-on
- Different levels of security: container , service and resource

**Transport-level and message-level security**

GSI can enable security at two levels: the *transport* level or the *message* level. By "transport-level security" we mean authentication via TLS with support for X.509 proxy certificates. In transport level security all information exchanged between the client and the server is encrypted. By "message-level security" we mean support for the WS-Security standard and the WS Secure Conversation specification to provide message protection for SOAP messages. So in message-level security only the content of the SOAP message is encrypted while the rest of the SOAP message is left unencrypted. Figures 14 and 15 depict this difference.

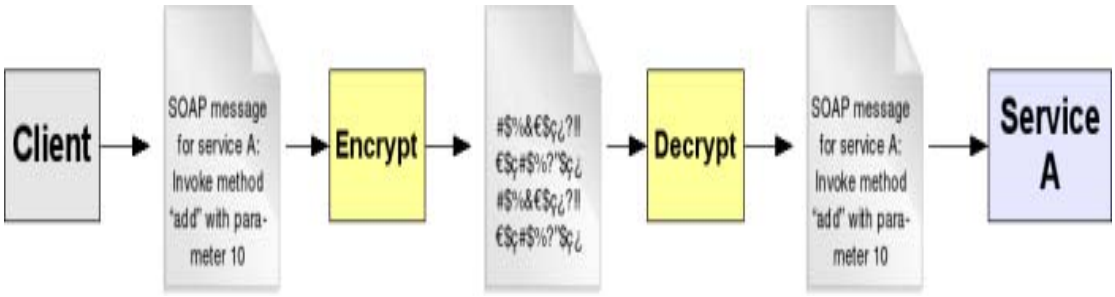



**Figure 14:** Transport-level security

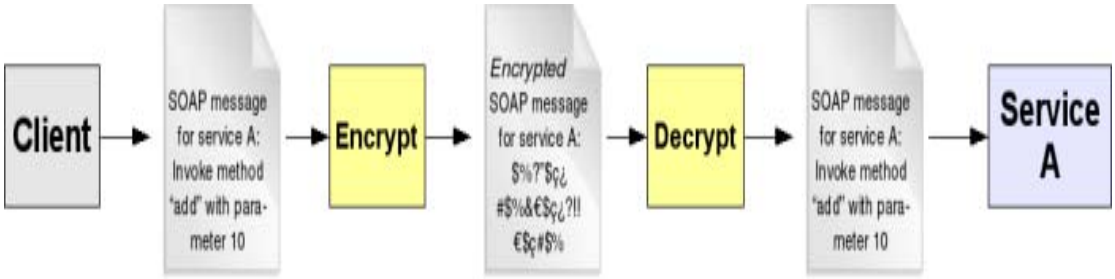



**Figure 15:** Message-level security

Both transport-level and message-level security in GSI are based on public-key cryptography and, therefore, can guarantee privacy, integrity, and authentication. In general, a GSI secure conversation must *at least* be authenticated. Integrity is usually desirable, but can be disabled. Encryption can also be activated to ensure privacy. GT4.0's support for message-level security is important as it allows us to comply with the WS-Interoperability Basic Security Profile. However, because current message-level security implementations have relatively poor performance, GT4.0 services use transport-level security by default. GSI offers two message-level protection schemes, and one transport-level scheme.

- **GSI Secure Message**: Provides message-level security and is based on the proposed WS-Security standard.

- **GSI Secure Conversation**: Provides message-level security and is based on the WS-SecureConversation specification. When this method is chosen, a *secure context* is first established between the client and the server. After an initial exchange of messages to establish the context, all the following messages can reuse that context, resulting in a better performance than GSI Secure Message (if the overhead of setting up the context is acceptable). Furthermore, GSI Secure Conversation is the only scheme that supports credential delegation
- **GSI Transport**: Provides transport-level security by using TLS (formerly known as SSL). It provides the best performance and is used by default in GT4.

**Authentication**

GSI supports authentication through X.509 certificates. All three protection schemes seen above can be used along with X.509 certificated to provide strong authentication. Another form of authentication supported by GSI is usernames and passwords. However with this method features like privacy, integrity and delegation will not be available.

**Authorization**

Authorization refers to who is authorized to perform a certain task. In a Web Services context it is required to know who is authorized to use a web service. GSI supports authorization in both the server-side and the client-side. The server has five possible authorization modes. Depending on the chosen authorization mode, the server will decide if it accepts or declines an incoming request.

- **None**: This is the simplest type of authorization. No authorization will be performed.
- **Self**: A client will be allowed to use a service if the client's identity is the same as the service's identity.
- **Gridmap**: A gridmap is a list of 'authorized users' akin to an ACL (Access Control List). . When this type of authorization is used, only the users that are listed in the service's gridmap may invoke it.
- **Identity authorization**: A client will be allowed to access a service if the client's identity matches a specified identity. In a sense, this is like having a one-user gridmap (except that identity authorization is configured programmatically, whereas the gridmap is represented as a file in our system).
- **Host authorization**: A client will be allowed to access a service if it presents a host credential that matches a specified hostname. In other words, we will only allow requests coming from one particular host.

With client-side authorization the client is able to figure out when it will allow a service to be invoked. The authorization modes of a client are the following:

- **None**: No authorization will be performed.
- **Self**: The client will authorize an invocation if the service's identity is the same as the client.
- **Identity authorization**: As described above, the client will only allow requests to be sent to services with a specified identity.

- **Host**: The client will authorize an invocation if the service has a host credential. Furthermore, the client must be able to resolve the address of the host to the hostname specified in the host credential. Note that this is different from server-side host authorization, where we check if the hostname in the credential is equal to a host specified by us.

**Delegation and single sign-on**

Credential delegation and single sign-on are one of the most interesting features in GSI. Let's take a look at the following scenario where delegation is necessary. Suppose organization org-A submits a task X to organization org-B. Org-B trusts org-A so it accepts the task. Now let's suppose that org-B decides to submit a subtask Z of X to another organization org-C. Org-C trusts org-A but it does not trust org-B. What should organization C do in this case? Figure 16 depicts this scenario.

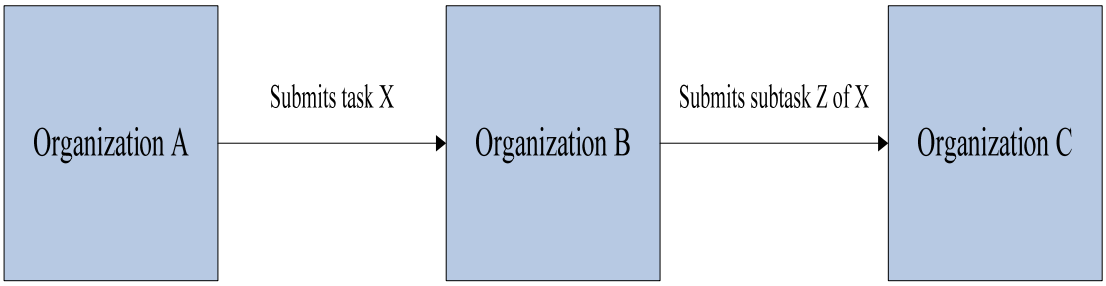



**Figure 16:** A scenario where delegation is necessary

Organization C has two options. Turn down the request because it does not trust org-B or because the original requester is org-A, accept the request. In this situation it is logical that organization C should accept org-B's request. However org-C has to know that org-B's request is performed on behalf of org-A, as shown in figure 17.

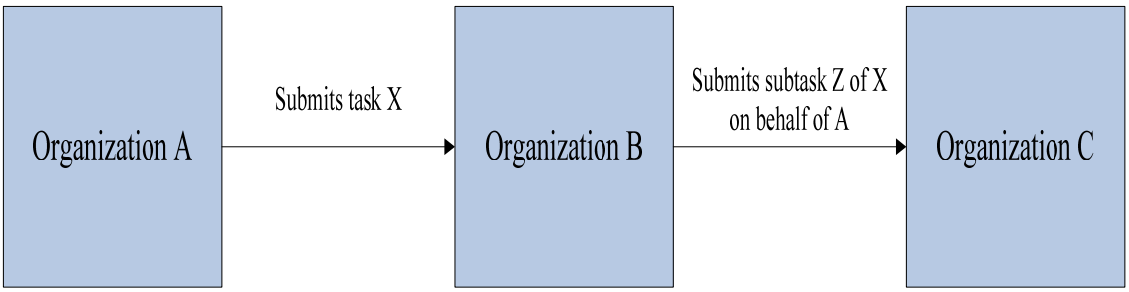



**Figure 17:** Delegation

Of course, this is not a very secure solution, since anyone could claim to be acting on org-A's behalf! One possible solution would be for org-C to contact org-A every time it receives a request on org-A's behalf. However, suppose that task X is composed of 20 different subtasks, and that each subtask is dispatched to a different organization by org-B. Organization A would be flooded with messages requisting confirmation that organization B asked me to perform a task on your behalf. In response, org-A would have to authenticate itself with all those organizations. A more elegant solution would be to somehow make organization C believe that org-B is organization A. This is achieved by a special kind of certificate called "**proxy certificate**".

A proxy certificate allows the holder of the certificate to act on someone else's behalf. It is very similar to X.509 digital certificates except that it is not signed by a Certificate Authority.

Instead it is signed by an end user.Confirmation that the certificate is authentic is achieved by checking its signature. The following figure shows a proxy certificate signed by organization A.

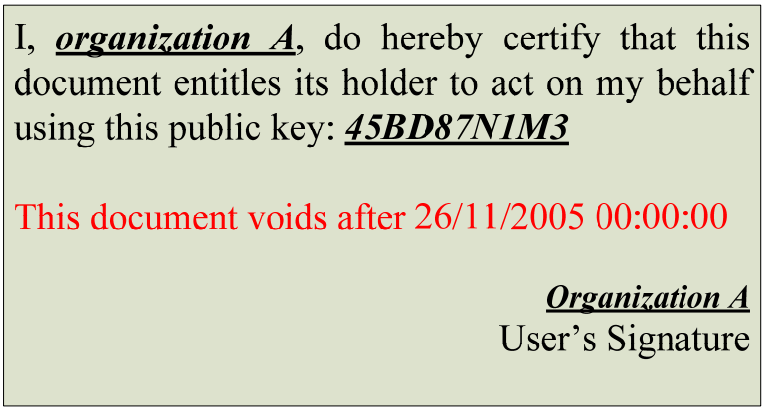



**Figure 18:** A proxy certificate with a limited life time

A proxy certificate has a private-public key pair generated specifically for the proxy certificate. This private-public key pair is mutually agreed upon by both parties (in this case, A and B), and Organization A will only allow the holder of that private-public key pair to act on its behalf (in this case, B). Allowing someone to act unconditionally on your behalf is risky. Therefore the lifetime of a proxy certificate is usually very limited (most of the times, up to 12 hours). So, even if the proxy certificate is compromised the attacker won't be able to make much use of it. So, a proxy certificate allows a user to act on another user's behalf. This is called credential delegation and it solves the problem described above.

Another desirable feature of proxy certificates is single sign-on. Without proxy certificates, Organization A would have to authenticate itself with all the organizations that receive requests 'on behalf of A'. In practice, this mean that the user in Organization A with permission to read the private key would have to access the key each time a mutual authentication is needed. Since private keys are usually protected by a password, this means that the user would have to *sign on* (provide the password) to access the key and perform authentication. Using proxy certificates, the user only has to sign in once to create the proxy certificate. The proxy certificate is then used for all subsequent authentications.

A proxy cerificate can be used to delegate a user's credential to another different user. This procedure is executed in a secure manner. For example, let's suppose that organization B need A's credential to make a request to C. B therefore needs a proxy certificate signed by A. The process of generating that cerificate is the following:

1. B generates a public/private key pair for the proxy certificate.
2. B uses the key pair to generate a certificate request, which will be sent to A using a secure channel. This certificate request includes the proxy's public key, but not the private key.
3. Supposing A agrees to delegate its credentials to B, Organization A will use its private key to digitally sign the certificate request.
4. A sends the signed certificate back to B using a secure channel.
5. B can now use the proxy certificate to act on A's behalf.

Organization B sends the proxy certificate to C in order to prove that it is acting on A's behalf. However, C must first validate the proxy certificate before it accepts the request. The process

of validating a proxy certificate is practically identical to the process of validating an ordinary certificate. The main difference is that the proxy certificate is not signed by a Certificate Authority, it is signed by a user. In the example above, the proxy certificate is signed by A, which means that A's public key is needed to test its authenticity. Since C is unlikely to have A's certificate, a request that uses a proxy certificate generally also sends the delegator's certificate, so the proxy certificate can be validated. Since the delegator's certificate will be signed by a Certificate Authority, the only step left is to validate the Certificate Authority's signature.

# 4 Knowledge Discovery

## 4.1 Introduction to Knowledge Discovery (KD)

Knowledge Discovery (KD) [38] is a nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns from large collections of data. Across a wide variety of fields data are being collected and accumulated at dramatic pace. There is an argent need for a new generation of computational theories and tools to assist humans in extracting useful information (knowledge) from the rapidly growing volumes of digital data. These theories and tools are the subject of the emerging field of knowledge discovery in databases (KDD). At an abstract level, the KDD field is concerned with the development of methods and techniques for making sense of data. The basic problem addressed by the KDD process is one of mapping low-level data (which are typically too voluminous to understand) into other forms that might be more compact, more abstract, or more useful. At the core of the process is the application of specific data-mining methods for pattern discovery and extraction.

The traditional method of turning data into knowledge relies on manual analysis and interpretation. For example, in the health-care industry, it is common for specialists to periodically analyze current trends and changes in health-care data, say, on a quarterly basis. The specialists then provide a report detailing the analysis to the sponsoring health-care organization; this report becomes the basis for future decision making and planning for health-care management.

For applications like the one described above, this form of manual probing of a data set is slow, expensive, and highly subjective. In fact, as data volumes grow dramatically, this type of manual data analysis is becoming completely impractical in many domains. Databases are increasing in size in two ways:

1. the number *N* of records or objects in the database
2. the number *d* of fields or attributes to an object.

Databases containing on the order of $N = 10^9$ objects are becoming increasingly common. Similarly, the number of fields d can easily be on the order of $10^2$ or even $10^3$, for example, in medical diagnostic applications. The job of handling millions of records, each having tens or hundreds on fields is certainly not one for humans. Therefore analysis work needs to be automated, at least partially.

The notion of finding useful patterns in data has been given a variety of names, including data mining, knowledge extraction, information discovery, information harvesting, data archaeology, and data pattern processing. The term data mining has mostly been used by statisticians, data analysts, and the management information systems (MIS) communities. It has also gained popularity in the database field. Knowledge discovery in databases (KDD) refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process. Data mining is the application of specific algorithms for extracting patterns from data.

The additional steps in the KDD process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, are essential to ensure that useful knowledge is derived from the data. Blind application of data-mining methods (rightly criticized as data dredging in the statistical literature) can be a dangerous activity, easily leading to the discovery of meaningless and invalid patterns.

The figure bellow shows an overview of the steps that compose the KDD process.

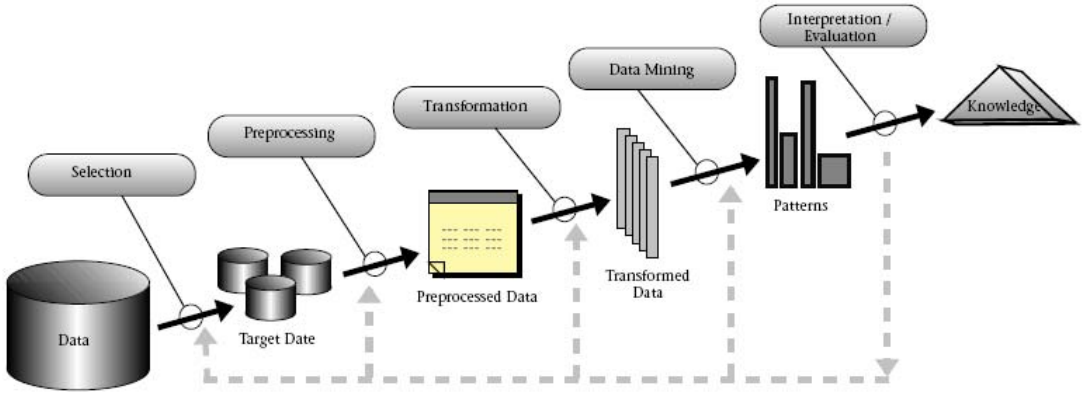



**Figure 19:** Overview of the KDD process

The KDD process involves using the database along with any required selection, preprocessing, subsampling, and transformations of it; applying data-mining methods (algorithms) to enumerate patterns from it; and evaluating the products of data mining to identify the subset of the enumerated patterns deemed knowledge. The data-mining component of the KDD process is concerned with the algorithmic means by which patterns are extracted and enumerated from data. The overall KDD process (figure x) includes the evaluation and possible interpretation of the mined patterns to determine which patterns can be considered new knowledge.

## *4.2 Types of Knowledge*

In the context of KDD knowledge is discovered information that is implicit, previously unknown and potentially useful. Being implicit, knowledge in the data mining sense goes beyond mere factual knowledge that has been stored and managed successfully by DBMS for years. This means that previously unknown need to be understood both from the system's perspective and with respect to the user's current level of knowledge. This type of knowledge can also be called meta-knowledge, which is patterns that characterize the data and hidden laws and structures that need not be strict functional dependencies, but may hold only with a certain probability. Also the statement of the discovered pattern must be simpler than the subset of data objects it describes.

There are three types of knowledge to be discovered in databases: classification, association and sequences. Classification tries to divide the given data set into disjoint classes using supervised or unsupervised learning, the latter also being referred to as clustering. The goal is to find a set of concept descriptions that characterize a class of data objects and can be applied to unknown objects to predict their class membership. An application example for classification is the case of a bank that wants to classify its customers to determine whether to give loans or not. In the case of sequences, time is given as an additional attribute and questions concentrate in dynamic patterns. An interesting problem concerning sequences is the discovery of episodes, frequent partially ordered sets of events that occur within a given time window. The last type of knowledge is association rules. Associations can be arbitrary rules of the form X→Y, X and Y being conjunctions of attribute value restrictions.

## 4.3 Association Rules Mining (ARM)

Association Rules Mining [24] is one problem in knowledge discovery that has received considerable attention during the past years. The purpose of this data mining technique is to discover associations between items in very large databases. There are a great number of publications concerning the ARM problem and many algorithms have been proposed so far. As with many other problems in knowledge discovery, a certain type of database and a special application has lead to specialized algorithms for this problem. In this case information in databases of large retail stores had to be searched for rules that characterize common consumer behavior. This type of mining may reveal for example, that 60% of customers that buy coke tend to buy pop corn as well. Although this example is very simple, other rules like this may not be so obvious and a means to find them is required. Furthermore mining tools enable users to quantify their assumptions and refute or confirm them through data mining queries. Although ARM algorithms were developed initially for commercial reasons, like the design, marketing and promotion of retail products, they can also find application in other areas. The medical area is also a field of application for ARM algorithms. Discovered rules from clinical data may by very useful in the design of drugs and general in healthcare decisions.

In the following section we give a formal definition of the association rules mining problem. First, however, we present frequent sets which form the basis for generating the actual association rules.

### 4.3.1 Definition of frequent sets

Given a set $I = \{I_1, I_2, \ldots, I_m\}$ of items a transaction T is defined as any subset of items in $I$ $(\subseteq I)$. Like sets, transactions do not contain duplicates, but we extend the pure notion of set and assume that the items in transactions and in all other itemsets we may consider are sorted.

Let the database D be a set of n transactions. A transaction T is said to support a set $X \subseteq I$ if it contains all items of X, i.e. $X \subseteq T$. We define the support of X, abbreviated supp(X), to be the fraction of all transactions in D that support X. If we have $supp(X) \geq s_{min}$ for a given minimum support value $s_{min}$, the set X is called frequent. The motivation behind minimum support is that we want to concern ourselves only with itemsets that occur often enough in D to be interesting. Infrequent itemsets, i.e. those that do not have minimum support, are not considered interesting. Finally, we will call an itemset X of cardinality k=|X| a k-itemset. The three properties of frequent sets shown bellow will be helpful later on; Properties 2 and 3 in particular form the foundation for all ARM algorithms.

1. **Support for subsets:** If $A \subseteq B$ for itemsets A, B, then $supp(A) \geq supp(B)$ because all transaction in D that support B, necessarily support A also.
2. **Suppersets of infrequent sets are infrequent:** If itemset A lacks minimum support in D, i.e. $supp(A) \leq s_{min}$, every superset B of A will not be frequent either because $supp(B) \leq supp(A) \leq s_{min}$ according to property 1.
3. **Subsets of frequent sets are frequent:** If itemset B is frequent in D, i.e. $supp(B) \geq s_{min}$, every subset A of B is also frequent in D because $supp(A) \geq supp(B) \geq s_{min}$ according to property 1. In particular if $A = \{i_1, i_2, \ldots, i_k\}$ is frequent all its k(k-1)-subsets are frequent. Note that the converse does not hold.

### 4.3.2 Definition of association rules

An association rule is an implication of the form R: X → Y, where X and Y are disjoint itemsets: $X, Y \subseteq I$ and $X \cap Y = \varnothing$. Furthermore, Y ≠ 0 is required. Such a rule can be understood as the prediction that if a transaction support X, it will also support Y with a certain probability, which is called the confidence (denoted conf(R)) of the rule.

The confidence of R is defined as the conditional probability that given that T support X, T will also support Y. More formally:

$$conf(R) = p(Y \subseteq T \mid X \subseteq T) = \frac{p(Y \subseteq T \wedge X \subseteq T)}{p(X \subseteq T)} = \frac{\sup(X \cup Y)}{\sup(X)}$$

The support for rule R in D is defined as $\sup(X \cup Y)$. The confidence of a rule reveals how often it can be expected to apply, while its support indicates how trustworthy the entire rule is. For a rule to be relevant it needs to have enough support and sufficient confidence. We will therefore say that rule R holds with respect to D, some fixed minimum confidence level $c_{min}$ a fixed minimum support $s_{min}$, if conf( R ) ≥ $c_{min}$ and supp( R ) ≥ $s_{min}$. Note that as a necessary condition for a rule to hold both, both anteceded and consequent of the rule have to be frequent.

To illustrate the importance of both requirements, minimum support and minimum confidence, assume first that we base a rule on just one data object. This rule will have maximum confidence of 100% but it does not describe a common pattern in the database and has to be discarded because it lacks minimum support. Let's assume now has enough support but low confidence for example it says that 2% of all customers that buy soap also buy tomatoes. Although this fact is well supported by the data in the database, it is not relevant because it does not express a strong correlation.

### 4.3.3 The basic algorithm for association rules mining

In this section we will describe the basic rule discovery algorithm that almost all ARM algorithms follow. To construct all association rules the support of every frequent itemset in the database must be computed. Therefore all algorithms are comprised of two stages. First all frequent itemsets are generated and in the second stage the actual rules and their confidence is generated from those frequent sets.

The number of potential frequent sets is equal to the size of the power set of all items, which grows exponentially with the number of items considered. A straightforward algorithm might perform an exhaustive search and test every set in the power set as to whether it is frequent or not. The basic method every algorithm follows is to create a set of itemsets called candidates that it believes could be frequent. How many such candidates are created, in which sequence and how often depends on the individual algorithm. To find out which of these candidate itemsets are actually frequent and what their exact support is, the support for each candidate set has to be counted in a pass over the database. Since counting the occurrences of a candidate set involves a considerable amount of processing time and memory, the obvious goal is to reduce the number of candidates generated by an algorithm.

When all support values are available, possible rules can be created and their confidence determined. For every frequent set X, each of its proper subset is chosen as anteceded of a rule and the remaining items constitute the consequent. Since X itself is frequent all of its

subsets have to be frequent as well and their support is known. So the confidence of the rule is computed and the rule is accepted or discarded, depending on the minimum confidence level.

### 4.3.4 Previous ARM algorithms

#### AIS

The AIS algorithm was the first published algorithm developed to generate all large itemsets in a transaction database [25]. It focused on the enhancement of databases with necessary functionality to process decision support queries. This algorithm was targeted to discover qualitative rules. This technique is limited to only one item in the consequent. That is, the association rules are in the form of $X \Rightarrow I_j \mid \alpha$, where X is a set of items and $I_j$ is a single item in the domain I, and $\alpha$ is the confidence of the rule.

The AIS algorithm makes multiple passes over the entire database. During each pass, it scans all transactions. In the first pass, it counts the support of individual items and determines which of them are large or frequent in the database. Large itemsets of each pass are extended to generate candidate itemsets. After scanning a transaction, the common itemsets between large itemsets of the previous pass and items of this transaction are determined. These common itemsets are extended with other items in the transaction to generate new candidate itemsets. A large itemset *l* is extended with only those items in the transaction that are large and occur in the lexicographic ordering of items later than any of the items in *l*. To perform this task efficiently, it uses an estimation tool and pruning technique. The estimation and pruning techniques determine candidate sets by omitting unnecessary itemsets from the candidate sets. Then, the support of each candidate set is computed. Candidate sets having supports greater than or equal to min support are chosen as large itemsets. These large itemsets are extended to generate candidate sets for the next pass. This process terminates when no more large itemsets are found.

It is believed that if an itemset is absent in the whole database, it can never become a candidate for measurement of large itemsets in the subsequent pass. To avoid replication of an itemset, items are kept in lexicographic order. An itemset A is tried for extension only by items B (i.e., $B=I_1, I_2, \ldots\ldots I_k$) that are later in the ordering than any of the members of A. For example, let I={p, q, r, s, t, u, v}, and {p, q} be a large itemset. For transaction T = {p, q, r, s}, the following candidate itemsets are generated:

| | |
|---|---|
| {p, q, r} | expected large: continue extending |
| {p, q, s} | expected large: cannot be extended further |
| {p, q, r, s} | expected small: cannot be extended further |

The expected support of A+B is the product of individual relative frequencies of items in B and the support for A, which is given as follows [10]:

$$s_{expected} = f(I_1) \times f(I_2) \times \ldots \times f(I_k) \times (x\text{-}c)/dbsize$$

where $f(I_i)$ represents the relative frequency of item $I_i$ in the database, and (x-c)/dbsize is the actual support for A in the remaining portion of the database (here x = number of transactions that contain itemset A, c = number of transactions containing A that have already been processed in the current pass, and dbsize = the total number of transactions in the database).

The main problem of the AIS algorithm is that it generates too many candidates that later turn out to be small [26]**.** Besides the single consequent in the rule, another drawback of the AIS algorithm is that the data structures required for maintaining large and candidate itemsets were not specified [25]**.** If there is a situation where a database has m items and all items

appear in every transaction, there will be $2^m$ potentially large itemsets. Therefore, this method exhibits complexity which is exponential in the order of m in the worst case.

## SETM

The SETM algorithm was proposed in [27] and was motivated by the desire to use SQL to calculate large itemsets. In this algorithm each member of the set large itemsets, $\overline{L_k}$, is in the form <TID, itemset> where TID is the unique identifier of a transaction. Similarly, each member of the set of candidate itemsets, $\overline{C_k}$, is in the form <TID, itemset>.

Similar to the AIS algorithm, the SETM algorithm makes multiple passes over the database. In the first pass, it counts the support of individual items and determines which of them are large or frequent in the database. Then, it generates the candidate itemsets by extending large itemsets of the previous pass. In addition, the SETM remembers the TIDs of the generating transactions with the candidate itemsets. The relational merge-join operation can be used to generate candidate itemsets. Generating candidate sets, the SETM algorithm saves a copy of the candidate itemsets together with TID of the generating transaction in a sequential manner. Afterwards, the candidate itemsets are sorted on itemsets, and small itemsets are deleted by using an aggregation function. If the database is in sorted order on the basis of TID, large itemsets contained in a transaction in the next pass are obtained by sorting $\overline{L_k}$ on TID. This way, several passes are made on the database. When no more large itemsets are found, the algorithm terminates.

The main disadvantage of this algorithm is due to the number of candidate sets $\overline{C_k}$ [26]. Since for each candidate itemset there is a TID associated with it, it requires more space to store a large number of TIDs. Furthermore, when the support of a candidate itemset is counted at the end of the pass, $\overline{C_k}$ is not in ordered fashion. Therefore, again sorting is needed on itemsets. Then, the candidate itemsets are pruned by discarding the candidate itemsets which do not satisfy the support constraint. Another sort on TID is necessary for the resulting set ($\overline{L_k}$). Afterwards, $\overline{L_k}$ can be used for generating candidate itemsets in the next pass. No buffer management technique was considered in the SETM algorithm [26]. It is assumed that $\overline{C_k}$ can fit in the main memory.

### 4.3.5 Apriori and Apriori-gen

All ARM algorithms proposed in the literature separate rule construction from finding frequent sets. The later is the most interesting part and the one solved differently from every algorithm. So far apriori [26] is considered the best algorithm proposed for the ARM problem[28]. Apriori uses apriori-gen for generating candidates and determining frequent sets. Most of the algorithms before apriori created a vast number of candidates. Apriori-gen which is described bellow has been so successful in reducing the number of candidates that it was used in every algorithm proposed after its publication.

**Candidate generation: Apriori-gen**

Apriori-gen generates only those candidates for which all subsets have been previously determined to be frequent. In particular, a (k+1)-candidates will be accepted only if its k-subsets are frequent.

Assume candidates of size (k+1) are to be created. The algorithm shown in the figure below, takes the set of frequent k-itemsets Lk as input and searches for pair of sets that have their (k-1) smallest items in common. Taking the k-1 common items and the two different items, these two sets are then joined to form a prospective (k+1)-candidate. Duplicates are avoided by demanding that the largest item of the second set be greater than the largest item of the first. So far the frequency of only two subsets has been asserted, because their mere presence in the input set allowed the candidate to be created in the first place.

```
insert into C_{k+1}
select a.item_1, a.item_2, …, a.item_k, b.item_k
from L_k a, L_k b
where a.item_1=b.item_1, …, a.item_{k-1}=b.item_{k-1}, a.item_k<b.item_k

for all itemset c ∈ C_{k+1} do
    for all k-subsets s of c do
        if (s ∉ L_k) then
            delete c from C_{k+1}
end
```

**Figure 20:** Pseudo-code for apriori-gen algorithm

If for example, {1, 3, 4, 6} and {1, 3, 4, 8} are frequent, those two are joined to create the candidate set {1, 3, 4, 6, 8}. The subsets that remain to be checked are {3, 4, 6, 8}, {1, 4, 6, 8} and {1, 3, 6, 8}, and if any of these are infrequent, the candidate is discarded. Note that it is enough to pair only those frequent sets that differ only in their largest set. Consider frequent sets {1, 3, 4, 6} and {1, 3, 5, 6} which differ only on their third item. The candidate that could be created from this pair is {1, 3, 4, 5, 6}. But if this set should be a candidate at all, its subsets {1, 3, 4, 5} and {1, 3, 4, 6} are in $L_4$ and will be used to generate it. Since the 1-candidates are simply all the sets that contain only one item, this procedure is not necessary to generate them. Apriori-gen is first used to generate $C_2$ from $L_1$.

In apriori-gen candidate generation is done prior to and separate from the counting step, and the algorithm is only called once to create the candidates of a given size. So with this algorithm fewer candidates are created and their not created repeatedly for every transaction.

## Apriori

Apriori was the first algorithm to use apriori-gen for candidate generation. Each pass of apriori consists of a call to apriori-gen to generate all candidates of a given size (size k in pass k) and a counting phase that determines the support of all the candidates. Each counting phase scans the entire database. Upon reading a transaction T in the counting phase of pass k, apriori has to determine all the k-candidates supported by T and increment the support counters associated with these candidates. The pseudo-code for this algorithm is shown below:

```
Initialize: L1 = 1-candidates    //all sets that contain 1 item
For (k=2; Lk+1 ≠ ∅; k++)
    Ck = apriori-gen(Lk-1)   // generate new candidates
     for all transactions t in database D do
         Ct = subset(Ck, t)
         For all candidates c ∈ Ct do
              c.count++

    Lk = { c ∈ Ck | c.count ≥ Minimum Support }

Final frequent Sets = ∪k Lk;
```

**Figure 21:** Pseudo-code for Apriori algorithm

# 5 Grid System

## *5.1 Experimental Setup*

### 5.1.1 The Grid Environment in details

The components used to develop a small experimental environment are shown in the following figure:

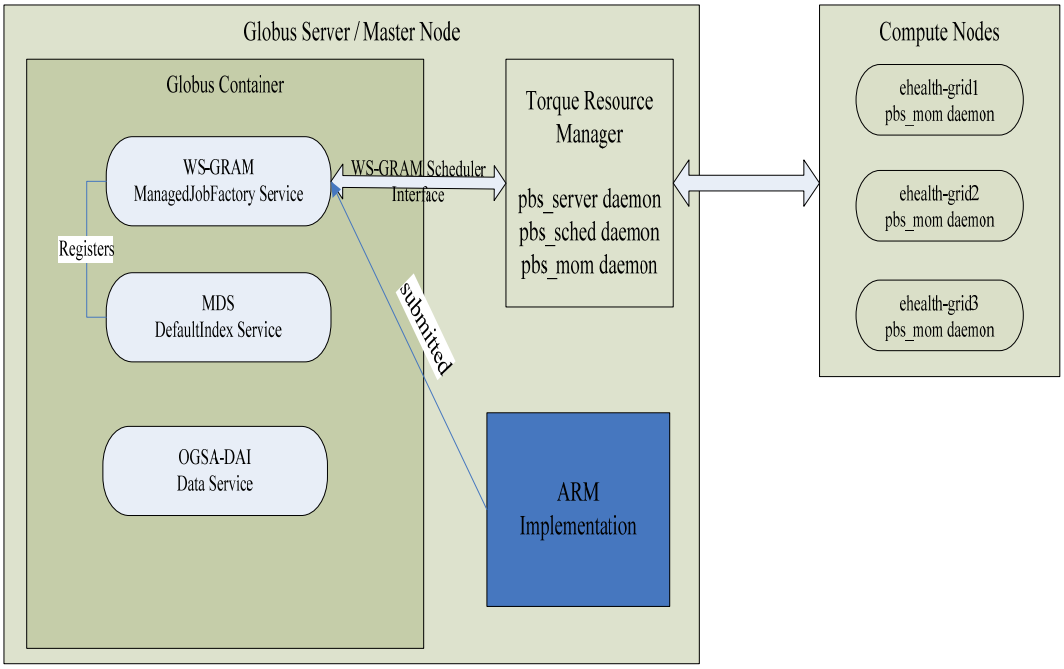



**Figure 22:** The components of the grid system

All available Web Services are hosted by the Globus Container. Access to these services is possible through the Endpoint Reference (EPR) of each service. For example the indexing service has the following EPR: *https://globus_host:8443/wsrf/services/DefaultIndexService*. Each EPR is combined of two parts. The first part: https://globus_host:8443/wsrf/services is the same for every service. The second is the service's name. A view of all the services in the container of our Grid is shown below:

```
[globusroot@syrios globusroot]$ globus-start-container
Starting SOAP server at: https://139.91.190.22:8443/wsrf/services/
With the following services:

[1]: https://139.91.190.22:8443/wsrf/services/TriggerFactoryService
[2]: https://139.91.190.22:8443/wsrf/services/DelegationTestService
[3]: https://139.91.190.22:8443/wsrf/services/SecureCounterService
[4]: https://139.91.190.22:8443/wsrf/services/IndexServiceEntry
[5]: https://139.91.190.22:8443/wsrf/services/DelegationService
[6]: https://139.91.190.22:8443/wsrf/services/InMemoryServiceGroupFactory
[7]: https://139.91.190.22:8443/wsrf/services/mds/test/execsource/IndexService
[8]: https://139.91.190.22:8443/wsrf/services/mds/test/subsource/IndexService
[9]: https://139.91.190.22:8443/wsrf/services/SubscriptionManagerService
[10]: https://139.91.190.22:8443/wsrf/services/TestServiceWrongWSDL
[11]: https://139.91.190.22:8443/wsrf/services/SampleAuthzService
[12]: https://139.91.190.22:8443/wsrf/services/WidgetNotificationService
[13]: https://139.91.190.22:8443/wsrf/services/AdminService
[14]: https://139.91.190.22:8443/wsrf/services/DefaultIndexServiceEntry
[15]: https://139.91.190.22:8443/wsrf/services/CounterService
[16]: https://139.91.190.22:8443/wsrf/services/TestService
[17]: https://139.91.190.22:8443/wsrf/services/InMemoryServiceGroup
[18]: https://139.91.190.22:8443/wsrf/services/SecurityTestService
[19]: https://139.91.190.22:8443/wsrf/services/ContainerRegistryEntryService
[20]: https://139.91.190.22:8443/wsrf/services/NotificationConsumerFactoryService
[21]: https://139.91.190.22:8443/wsrf/services/TestServiceRequest
[22]: https://139.91.190.22:8443/wsrf/services/IndexFactoryService
[23]: https://139.91.190.22:8443/wsrf/services/ReliableFileTransferService
[24]: https://139.91.190.22:8443/wsrf/services/mds/test/subsource/IndexServiceEntry
[25]: https://139.91.190.22:8443/wsrf/services/Version
[26]: https://139.91.190.22:8443/wsrf/services/NotificationConsumerService
[27]: https://139.91.190.22:8443/wsrf/services/IndexService
[28]: https://139.91.190.22:8443/wsrf/services/NotificationTestService
[29]: https://139.91.190.22:8443/wsrf/services/ReliableFileTransferFactoryService
[30]: https://139.91.190.22:8443/wsrf/services/DefaultTriggerServiceEntry
[31]: https://139.91.190.22:8443/wsrf/services/TriggerServiceEntry
[32]: https://139.91.190.22:8443/wsrf/services/PersistenceTestSubscriptionManager
[33]: https://139.91.190.22:8443/wsrf/services/mds/test/execsource/IndexServiceEntry
[34]: https://139.91.190.22:8443/wsrf/services/DefaultTriggerService
[35]: https://139.91.190.22:8443/wsrf/services/TriggerService
[36]: https://139.91.190.22:8443/wsrf/services/gsi/AuthenticationService
[37]: https://139.91.190.22:8443/wsrf/services/TestRPCService
[38]: https://139.91.190.22:8443/wsrf/services/ManagedMultiJobService
[39]: https://139.91.190.22:8443/wsrf/services/RendezvousFactoryService
[40]: https://139.91.190.22:8443/wsrf/services/WidgetService
[41]: https://139.91.190.22:8443/wsrf/services/ManagementService
[42]: https://139.91.190.22:8443/wsrf/services/ManagedExecutableJobService
[43]: https://139.91.190.22:8443/wsrf/services/InMemoryServiceGroupEntry
[44]: https://139.91.190.22:8443/wsrf/services/AuthzCalloutTestService
[45]: https://139.91.190.22:8443/wsrf/services/DelegationFactoryService
[46]: https://139.91.190.22:8443/wsrf/services/DefaultIndexService
[47]: https://139.91.190.22:8443/wsrf/services/ShutdownService
[48]: https://139.91.190.22:8443/wsrf/services/ContainerRegistryService
[49]: https://139.91.190.22:8443/wsrf/services/TestAuthzService
[50]: https://139.91.190.22:8443/wsrf/services/CASService
[51]: https://139.91.190.22:8443/wsrf/services/ManagedJobFactoryService
```

**Figure 23:** A list of all grid services running in the globus container

**GRAM services (WS-GRAM)**

The parallelARM service was developed mainly for discovery purposes. It registers to the default indexing service of GT4. These way clients have the ability to discover that the specified data mining tool exists in the Grid. The actual implementation of the parallel ARM algorithm is an executable that will be submited through WS-GRAM to the local resource manager. The ManagedJobFactory Service will create a job resource for every job we submit and each job resource will be monitored until completion by the ManagedJob Service. GRAM assumes a job execution model by which a job proceeds through a series of states, from initially *Unsubmitted* to eventually *Done* or *Failed*. The following table shows GRAM's job states.

| State | Description |
|---|---|
| Unsubmitted | Job has not yet been submitted. |
| StageIn | Job is waiting for stage in of executable or input files to complete. |
| Pending | The local scheduler has not yet scheduled the job for execution. |
| Active | Job is executing. |
| Suspended | Job execution has been suspended. |
| StageOut | Job execution has completed; output files are being staged out. |
| CleanUp | Job execution and stage out have completed; clean up tasks are underway. |
| Done | Job has completed successfully. |
| Failed | Job failed |

**Table 2. GRAM job states**

A diagram of the transition between states is shown in the following figure.

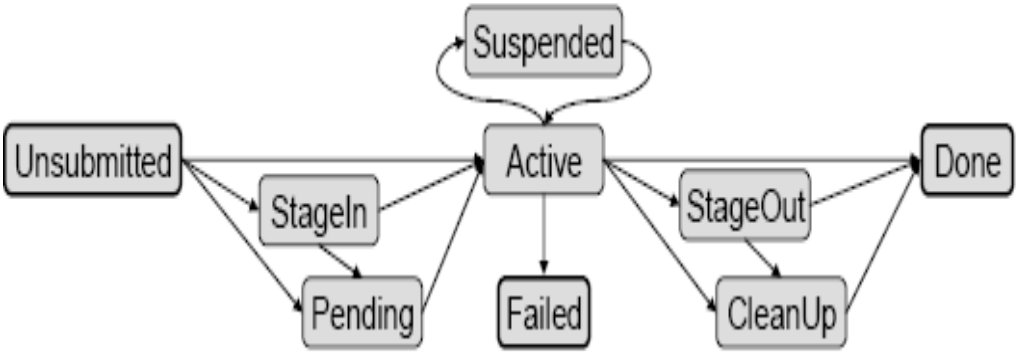



**Figure 24:** State transition diagram for GRAM jobs

In our case the executable is located on the target computer, input data is passed as arguments, and output data has been written to files on the target computer. However, there will also be cases in which the executable or other files need to be *staged in* to the target computer and/or output files must be *staged out*. We specify file staging by adding file transfer directives to the job description. These directives follow the RFT syntax, which enables third-party transfers. Each file transfer must therefore specify a source URL and a destination URL. URLs are specified as GridFTP URLs (for remote files) or as file URLs (for local files). For

example, in the following example of a stage-in request, the source URL is a GridFTP URL (here, gsiftp://job.submitting.host:2888/tmp/mySourceFile) that resolves to a source document accessible on the file system of the job submission machine (here, /tmp/mySourceFile). At runtime, the Reliable File Transfer service used by the GRAM service on the remote machine fetches the remote file using GridFTP and writes this file to the specified local file (here, file:///$GLOBUS_USER_HOME/my_file, which resolves to ~/my_file).

```
<fileStageIn>
        <transfer>
                <sourceUrl>gsiftp://submit.host:2888/tmp/mySourceFile</sourceUrl>
                <destinationUrl>file:///${GLOBUS_USER_HOME}/my_file</destinationUrl>
        </transfer>
</fileStageIn>
```

The symbol ${GLOBUS_USER_HOME} is one of four that the GRAM server will substitute at run time, as follows:

- GLOBUS_USER_HOME: The path to the home directory for the local account/user
- GLOBUS_USER_NAME: The local account the job is running under
- GLOBUS_SCRATCH_DIR: An alternative directory where a file system is shared with the compute nodes that a user might want to use. Typically it will provide more space than the users default HOME dir. This directory's value may contain ${GLOBUS_USER_HOME}, which will be replaced with value of that substitution.
- GLOBUS_LOCATION: Path to the Globus Toolkit installation

We use the job description below to illustrate not only job staging but also a range of further job description language features. The submission of this job to the GRAM services causes the following sequence of actions:

- The /bin/echo executable is transferred from the submission machine to the GRAM host file system. The destination location is the HOME directory of the user on behalf of whom the job is executed by the GRAM services (see <fileStageIn>).
- The transferred executable is used to print a test string (see <executable>, <directory> and the <argument> elements) on the standard output, which is redirected to a local file (see <stdout>).
- The standard output file is transferred to the submission machine (see <fileStageOut>).
- The file that was initially transferred during the stage-in phase is removed from the file system of the GRAM installation (see <fileCleanup>).

```
1 <job>
2     <executable>my_echo</executable>
3     <directory>${GLOBUS_USER_HOME}</directory>
                                4     <argument>Hello world!</argument>
5     <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
6     <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
7     <fileStageIn>
8       <transfer>
9         <sourceUrl>gsiftp://submit.host:2888/bin/echo</sourceUrl>
10        <destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
11      </transfer>
12    </fileStageIn>
13    <fileStageOut>
14      <transfer>
15        <sourceUrl>file://${GLOBUS_USER_HOME}/stdout</sourceUrl>
16        <destinationUrl>gsiftp://submit.host:2888/tmp/stdout</destinationUrl>
17      </transfer>
18    </fileStageOut>
19    <fileCleanUp>
20      <deletion> <file>file://${GLOBUS_USER_HOME}/my_echo</file> </deletion>
```

**Figure 25:** Example of a job description

A GRAM server is installed to enable Web services access to a compute resource. Installation involves two principal configuration steps.

- *Local scheduler interface*. GRAM depends on a local mechanism for starting and controlling jobs. If the fork-based GRAM mode is to be used, no special software is required. For batch scheduling mechanisms, such as Condor, SGE, PBS, and LSF, the local scheduler must be installed and configured for local job submission prior to deploying and operating GRAM. In our system we have installed TORQUE, a resource manager based on openPBS.
- *Authorization to user mapping*. GRAM depends on an authorization callout to determine both whether a specific request should be allowed and the local user as which an authorized request should be executed. By default, GRAM looks for the file /etc/grid-security/grid-mapfile, which is assumed to contain a set of one-line entries, each specifying the DN of an authorized user and the local user as which they are to execute. For example: "/O=Grid/OU=GlobusTest/OU=simpleCAfoo.bar.com/OU=bar.com/CN=John" john

There are several security issues concerning GRAM operations. An incoming job management request is subject to multiple levels of security checks:

- WS-Security mechanisms are used to validate the credentials associated with the request and thus to authenticate the requestor.
- Authorization is performed via an authorization callout. Depending on configuration, this callout may consult a gridmapfile, a SAML server, or other mechanism.
- If authorization succeeds, then it also yields the local identity under which the job is to execute. The Unix utility sudo is used to invoke local resource management mechanisms under this user id, thus enabling the application of site authorization mechanisms.

So, in order for a job to execute, the user who submits the job must be authorized (in our case with a grid-map-file) and it must have an account to all compute elements. GRAM has been designed to minimize the privileges required and to minimize the risks of service malfunction or compromise. Services operating in the GT4 Java container do not require special privileges to perform their operations: privileged operations are performed exclusively via sudo functions. To protect users from each other, jobs submitted by different users are typically

executed in separate local security contexts: e.g., under specific Unix user IDs based on details of the job request and authorization policies. (Dynamic account management mechanisms can be used to generate such security contexts in an on-demand manner) To assist with normal accounting functions as well as to further mitigate risks from abuse or malfunction, GRAM uses a range of audit and logging techniques to record a history of job submissions and critical system operations.

**Information services (MDS4)**

The Monitoring and Discovery System (MDS4) component of GT4 can streamline the tasks of monitoring and discovering services and resources in a distributed system. *Monitoring* is the process of observing resources or services (e.g., computers and schedulers), for such purposes as fixing problems and tracking usage. A user might use a monitoring system to identify resources that are running low on disk space, in order to take corrective action. *Discovery* is the process of finding a suitable resource to perform a task: for example, finding a compute host on which to run a job. This process may involve both finding which resources are suitable (e.g., have the correct CPU architecture) and choosing a suitable member from that set (e.g., the one with the shortest submission queue).

The following figure shows the MDS4 deployment of our grid environment. Through this deployment we provide both static and dynamic data (e.g. queue lengths, architecture types) relevant to selecting the best resources to use for a particular job. End-users via a web interface, meta-scheduling system or other applications can use this deployment to find the resources that best meet their needs.

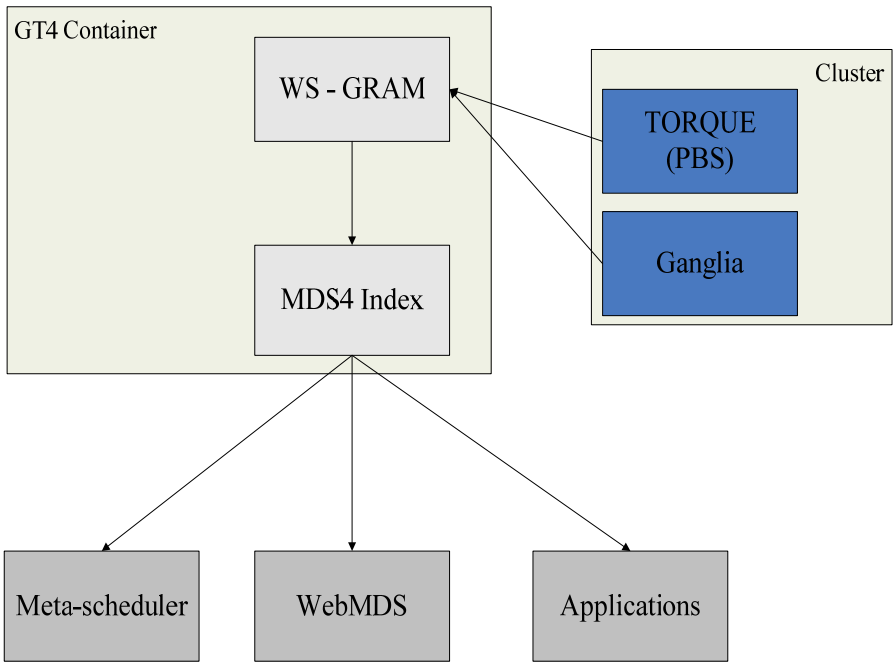



**Figure 26:** MDS4 deployment

In our case the information published by MDS4 default indexing service is utilized by a client application. Through the client GUI the user may retrieve information like the one shown bellow:

Service Group EPR

- Address: https://139.91.190.22:8443/wsrf/services/DefaultIndexServiceEntry
- GroupKey: 16450366
- EntryKey: 32221547

Member Service EPR

- Address: https://139.91.190.22:8443/wsrf/services/ManagedJobFactoryService
- ResourceID: PBS

Content

- AggregatorConfig:
  - GetResourcePropertyPollType:
    - PollIntervalMillis: 60000
    - ResourcePropertyName: glue:GLUECE
- AggregatorData:
  - GLUECE:
    - ComputingElement:
      - Name: long
      - UniqueID: long
      - Info:
        - TotalCPUs: 4
      - State:
        - EstimatedResponseTime: 0
        - FreeCPUs: 4
        - RunningJobs: 0
        - Status: enabled
        - TotalJobs: 0
        - WaitingJobs: 0
        - WorstResponseTime: 0
      - Policy:
        - MaxCPUTime: 0
        - MaxRunningJobs: 0
        - MaxTotalJobs: 0
        - MaxWallClockTime: 0
        - Priority: 0

**Figure 27:** Information published by MDS

## Security

Security tools are concerned with establishing the identity of users or services (authentication), protecting communications, and determining who is allowed to perform what actions (authorization), as well as with supporting functions such as managing user credentials and maintaining group membership information. GT4 provides distinct WS and pre-WS authentication and authorization capabilities. Both build on the same base, namely standard

X.509 end entity certificates and proxy certificates, which are used to identify persistent entities such as users and servers and to support the temporary delegation of privileges to other entities, respectively.

In order for users to acquire certificates we have set up the simple-CA package in our system. The SimpleCA package provides a simple certification authority that a user can install and use to issue credentials to Globus Toolkit users and services. This package is meant for use in situations where the user wants public key credentials, for example in order to test GT's operation, but does not have access to a proper certification authority.

A certificate of authority must be generated for each hosting environment. In addition, each client wishing to access a grid hosting environment (server) must request a CA and deploy it in their Globus installation in order to properly access the grid server. The first step of this procedure is to setup the simple-CA package and provide the necessary information regarding the installation. For example we must provide the unique subject name of this Certificate Authority (i.e. /O=Grid /OU=GlobusTest /OU=simpleCA-syrios /CN=globususer1) and an email address where requests will be sent. Furthermore we must provide a PEM (Privacy Enhanced Mail) pass phrase for the CA. This pass phrase will be requested by the CA every time the grid-administrator wants to sign a certificate request. The next step is to request and sign a host certificate and then copy it into the appropriate directory for secure services. The certificate must be for a machine with a consistent name in DNS. We should not run it on a machine using DHCP where a different name could be assigned to our computer.

Each user who wishes to access the available grid services must first request a certificate and get it signed by the grid-environment host. The user will be prompted to supply a unique password. This password is necessary for the generation of a proxy certificate. Finally we must add authorization for every user of the grid. The easiest way to do this is by creating a grid-map-file. Each line of this file must contain two pieces of information: the subject name of the user and the account name it will map to. In our system the grid-map-file has the following entry:

*"/O=Grid /OU=GlobusTest /OU=simpleCA-syrios /CN=globususer1" globususer1*

**Data Services**

The data service is responsible for accessing clinical information from relational or XML databases. It is a point of contact for clients who wish to access, query or update a data resource. It offers a document-oriented interface which accepts OGSA-DAI perform documents from clients and returns response documents to them. A data service can expose zero or more data service resources. A data service resource implements the core OGSA-DAI functionality. It accepts perform documents from data services, parses and validates them, executes the data-related activities specified within them and constructs response documents. Each data service resource can support interaction with a single database.

Deploying and exposing a data service resource via an OGSA-DAI WSRF data service is a three-step process.

1. Deploy an OGSA-DAI data service. This data service initially exposes 0 data service resources. Data services can expose 0 or more data service resources. It is up to us whether we expose 1 data service resource per service or expose multiple data service resources per service.

2. Deploy a data service resource. The data service resource contains information about a data resource and the activities clients can perform on that data resource via a data service.
3. Add a data service resource to a data service. This instructs a data service to expose a data service resource allowing clients to interact with it.

### 5.1.2 TORQUE Resource Manager

TORQUE (Tera-scale Open-source Resource and QUEue manager) [39] is a resource manager that provides control over jobs submitted to compute nodes. TORQUE is based mainly on OpenPBS with scalability, fault tolerance, and feature extension patches contributed by many different sites. Main components of TORQUE are server, scheduler and job executor. All of the parts have its own daemon programs (*pbs_server, pbs_sched and pbs_mom* respectively) that can be run on the machines that are designated to provide specific functionalities.

The cluster from the TORQUE's perspective using standard monitoring tool looks as follows:

```
[globusroot@syrios globusroot]$ pbsnodes -a
syrios.ics.forth.gr
     state = free
     np = 1
     ntype = cluster
     status = arch=linux,uname=Linux syrios 2.4.20-8 #1 Thu Mar 13
17:54:28 EST 2003 i686,sessions=4079 4137 4144 4146 4163 4165 4167
4170 4172 4174 4185
4429,nsessions=12,nusers=2,idletime=11,totmem=1557976kb,availmem=13981
68kb,physmem=513792kb,ncpus=1,loadave=0.05,netload=825313,state=free,r
ectime=1136804641

ehealth-grid1.ics.forth.gr
     state = free
     np = 1
     ntype = cluster
     status = arch=linux,uname=Linux ehealth-grid1 2.4.20-8 #1 Thu Mar
13 17:54:28 EST 2003
i686,sessions=2410,nsessions=1,nusers=1,idletime=169,totmem=777320kb,a
vailmem=731892kb,physmem=255248kb,ncpus=1,loadave=0.08,netload=220104,
state=free,rectime=1136804654

ehealth-grid2.ics.forth.gr
     state = free
     np = 1
     ntype = cluster
     status = arch=linux,uname=Linux ehealth-grid2 2.4.20-8 #1 Thu Mar
13 17:54:28 EST 2003
i686,sessions=2422,nsessions=1,nusers=1,idletime=162,totmem=777320kb,a
vailmem=731392kb,physmem=255248kb,ncpus=1,loadave=0.07,netload=218072,
state=free,rectime=1136804662

ehealth-grid3.ics.forth.gr
     state = free
     np = 1
     ntype = cluster
     status = arch=linux,uname=Linux ehealth-grid3 2.4.20-8 #1 Thu Mar
13 17:54:28 EST 2003
i686,sessions=2422,nsessions=1,nusers=1,idletime=149,totmem=777320kb,a
vailmem=731344kb,physmem=255248kb,ncpus=1,loadave=0.10,netload=214782,
state=free,rectime=1136804660
```

**Figure 28:** The cluster from the TORQUE's perspective

## 5.2 Clinical domain description

The clinical information from the Primary Health Care Centre Information System (PHCCIS) [21] form the clinical domain which will be used as the data set for knowledge extraction techniques. The Primary Health Care Centre Information System implements a generic and broad-spectrum patient record to be used by general practitioners. The principle functionality of the system is the electronic storage and management of patient data that are produced during the communication about the patient between two or more (health care) individuals. This communication is called encounter. The most usual encounter is a visit at the Health Care Centre.

The PHCC Information System provides two different views of the electronic patient record:

- The visit oriented view which consists of a list of encounters/visits implemented as a 2-dimension table. Each row represents a distinct encounter/visit. For each encounter visit the examinations, diagnosis and assessments that have to be done are stored. Specifically an encounter could contain a number of examinations such as a clinical exam, a gynaecological exam, a biochemical exam, a blood analysis, as well as diagnosis and the therapeutic management of the healthcare personnel.
- The problem oriented view. The patient record is divided into a series of sections each of which is given the heading "problem" or more generally "previous diagnosis". The problem may be defined by an ICD9 code or described by free text. Items and data in patient record are grouped under one or more problem headings. When a problem is selected, information relevant to that problem is viewed. Every problem is associated with one or more episodes. An episode may be an: a) Episode of problem: Many problems are not active all time but they have a cycle of activity. The period where the problem is reported to be active may be defined as an episode of the problem. b) Service episode: a collection of events aggregated during an interval bounded by start and stop times.

A large portion of the information stored in the PHCC system would not be useful in a data mining procedure. For this reason only specific "observations" are extracted from PHCCIS and stored in XML document format. The table bellow shows the clinical attributes which will participate in Association Rules Mining.

| **Domain** | **Attributes** |
|---|---|
| OBSERVATION | BIOCHEMICAL_EXAM, BLOOD_EXAM, CLINICAL_EXAM , MEDICAL_HISTORY, PATIENT, PROBLEM_OF_MEDICAL_HISTORY , VISIT |
| BIOCHEMICAL_EXAM | ACHOLERITHRINI, ALBUMIN, ALKALINE_PHOSPHATASE, ANOXI_GLUCOSE_S0, ANOXI_GLYCOZE_S120, ANOXI_GLYCOZE_S60, ANOXI_GLYCOZE_S90, A_AMYLASE, B12, BLOOD_SUGAR, CALCIUM, CHOLESTEROL, CPK, CREATINITE, FERRITIN, FERRUM, FOLIC_ACID, GGT, GLOBULIMS, HDLCHOLESTEROL, LDH, LDLCHOLESTEROL, NATRIUM, OCHOLERITHRINI, PHOSHORUS, POTASSIUM, PSAMMOSARCOMA, SGOT, SGPT, TOTAL_LIPIDS, TOTAL_PROTEINS, TRIGLYCERIDES, UREA, URIC_ACID |
| BLOOD_EXAM | EOSIN, HCT, HGB, LYM, LYMPHOCYTE, MCH, MCHC, MCV, MONOCELLULAR, MPV, PCT, PDGF, RBC, RDM, TKE1ORA, WBC |
| CLINICAL_EXAM | BEATSPERMIN, BREATHSPERMIN, DIASTOLICPRESSURE, HEIGHT, SYSTOLICPRESSURE, TEMPERATURE, WEIGHT |
| MEDICAL_HISTORY | DRINKINGID, SMOKINGID |
| PATIENT | GENDERID |
| PROBLEM_OF_MEDICAL_HISTORY | ICD9DISEASEID |
| VISIT | SYMPTOMID |

**Figure 29:** Clinical domains and their attributes

### 5.2.1 Categorization of clinical items

The data involved in a mining procedure are usually numerical, a case which applies in clinical information as well. However we are particularly interested in the meaning of that numerical data. For example the value of cholesterol is considered high if it is above 45 and normal if it is below 45.

Therefore it is essential to categorize the numerical values of data into logical values which can be used to extract meaningful association rules. In order to perform this task we use a simple text document like the one shown bellow:

```
########BIOCHEMICAL_EXAM######## Comment

SMOKINGID 1-1 SMOKING_YES
SMOKINGID 2-2 SMOKING_NO
SMOKINGID 3-3 SMOKING_STOPPED


ALBUMIN 3-5 ALBUMIN_NORMAL
ALBUMIN 0.001-2.999 ALBUMIN_LOW
ALBUMIN 5.001-1000 ALBUMIN_HIGH

ANOXI_GLUCOSE_S120 0.001-200 ANOXIGLUCOSES120_NORMAL
ANOXI_GLUCOSE_S120 200.1-100000 ANOXIGLUCOSES120_HIGH
```

**Figure 30:** Example of a domain semantics text document

Each clinical item has a specific domain of numerical values. For example the item "ALBUMIN" can take values between 0.0 and 1000.0. With the above document it is possible to divide the entire domain of an item into sub-domains and assign a logical value to each sub-domain. Therefore when the value of "ALBUMIN" in a biochemical examination is 4 then it is considered to be NORMAL.

The structure of the above document must adopt the following two rules:

- The symbol "#" denotes that the whole line is a comment and it must be ignored.
- The assignment of a numerical domain to a logical value must obey the following structure: <item's name> <domain> <item's name>_<category>. Those three elements must be divided by space (or spaces) and each line can contain only one triad.

A document that does not obey the above rules will not be parsed properly, resulting to catastrophic effects for the discovery of association rules.

## *5.3 A sequential algorithm for association rules mining*

The figure below shows the pseudo-code of a sequential algorithm for association rules mining. This algorithm uses a procedure like apriori-gen for candidate generation. The code below is pretty much the same like apriori but it uses a prefix-tree data structure for storing candidate/frequent sets and an optimization which reduces the size of the database by a significant factor.

```
L0 = ∅
k = 1
C1 = {{i}| i∈ I }  // all 1-itemsets
while ( Ck ≠ ∅ ) do
   { Count support for all candidates in Ck }
    for all transaction T∈ D
        for all k-subsets t ∈ T
            if(∃ c ∈ Ck ) then
               c.count++

    Lk = {c∈Ck | c.count ≥ Minimum Support }

   { Remove irrelevant 1-itemsets from database }
    if (k == 1) then
       for all transactions T∈D
           for all 1-itemsets t ∈ T
              if ( t ∉ L1) then
                 delete t

    Ck+1 = apriori-like-gen( Lk)
    k++
return L = ∪k Lk
```

**Figure 31:** Sequential algorithm for determining frequent sets

Each counting phase of the above algorithm has to scan the entire database. And every transaction has to be checked to find out which candidates it supports. In order to perform this procedure more efficiently we store candidate sets in a prefix-tree, which is explained in the next section.

**Complexity of Association Rule Mining**

Association Rule Mining involves generating all association rules in the data set that have a support greater than minimum support (the rules are frequent) and that have a confidence greater than minimum confidence (the rules are strong). The complexity of the two steps mentioned above in the *worst case scenario* is:

1. Find all frequent itemsets having minimum support. The search space for enumeration of all frequent itemsets is $2^m$, which is exponential in m, the number of items.
2. Generate strong rules having minimum confidence, from the frequent itemsets. We generate and test the confidence of all rules of the form X→Y where Y is subset of X and X is frequent. Because we must consider each subset of X as the consequent, the rule-generation step's complexity is $O(r \cdot 2^l)$, where *r* is the number of itemsets, and *l* is the longest frequent itemset.

*Therefore the computational power provided by the proposed GRID infrastructure can be affectively used to handle the computational needs of Association Rule Mining. Note that if the number of available computational resources is equal with the longest frequent itemset then the complexity of the mining procedure is reduced from exponential to linear.*

### 5.3.1 Prefix Tree

In this tree structure each node contains information (e.g. counters) about a specific itemset. Each edge in the tree is labelled with an item. If we want information about an itemset we just follow the edges labelled with the items in the set starting from the root of the tree. The figure bellow shows a complete prefix tree containing all the subsets of {1, 2, 3, 4, 5}.

Accessing information for a given set becomes very ease with the use of this structure. For example, if we want to access the counters for set {1, 2, 3} we begin from the root of the tree and follow the edges labelled with the items in the given set. The root of the tree represents the empty set and contains the number of total transactions in the database. Another important feature of this structure is that it stores both candidate and frequent sets. When candidates are counted and determined to be frequent they simply remain in their proper position in the tree and become frequent sets. The rest of the candidates are removed from the tree. Candidate sets have a count equal to zero before being processed. Two properties characterize this data structure:

1. The counts of node along a path are non increasing
2. If a set is frequent and therefore present in the tree, then all its subsets have to be in their proper place in the tree also.

From the first property we conclude that a set cannot occur more frequently in the database than any of its subsets. The second property is not restricted to the sets on the path to the root which naturally have to exist. For example if set {1, 2, 3} is in the tree, {1, 2} and {1} are clearly present because they are along the path but by property 2, {1, 3} and {2, 3} must also be in the tree.

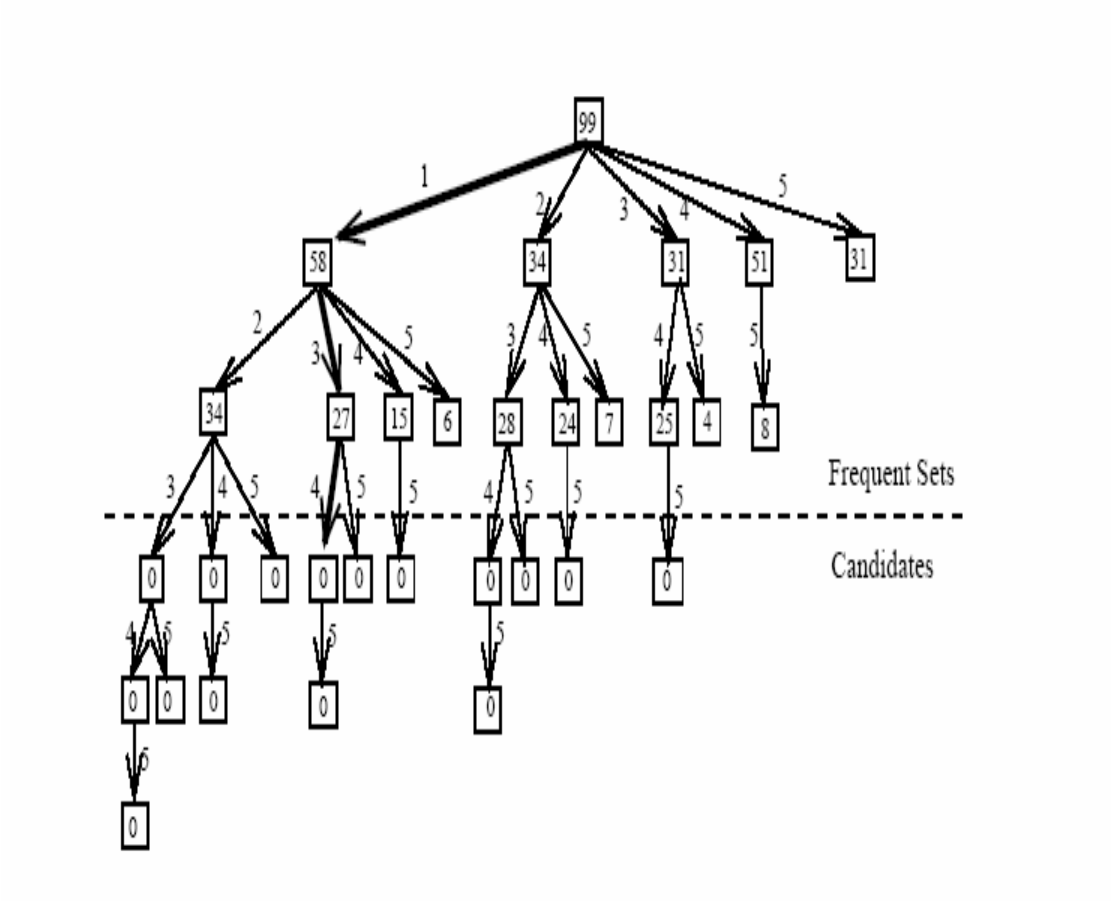



**Figure 32:** A complete prefix tree for 5 items

### 5.3.2 Computing support for candidates

As mentioned before the ARM algorithm is applied on clinical data stored in XML files. So the lack of a database system prevents us from processing the necessary information with the use of SQL queries. Instead we read the XML file using the SAX parser. This limitation results in the following procedure for determining the support of candidates.

The algorithm calls the apriori-like-gen procedure for candidate generation and stores them in the prefix tree. The next step is to parse the entire XML file and determine the support of each candidate. It is clear that this procedure can not be done separately for every candidate. So we store the candidates in an array and search the entire XML file only once.

Assume that we are in the first pass (k=1) of the algorithm and apriori-like-gen generates the 1-itemsets candidates {1}, {2}, {3}, {4}, {5}. Those are stored in the prefix tree and in the array, as shown bellow. After that the search procedure follows which determines the support for each candidate.

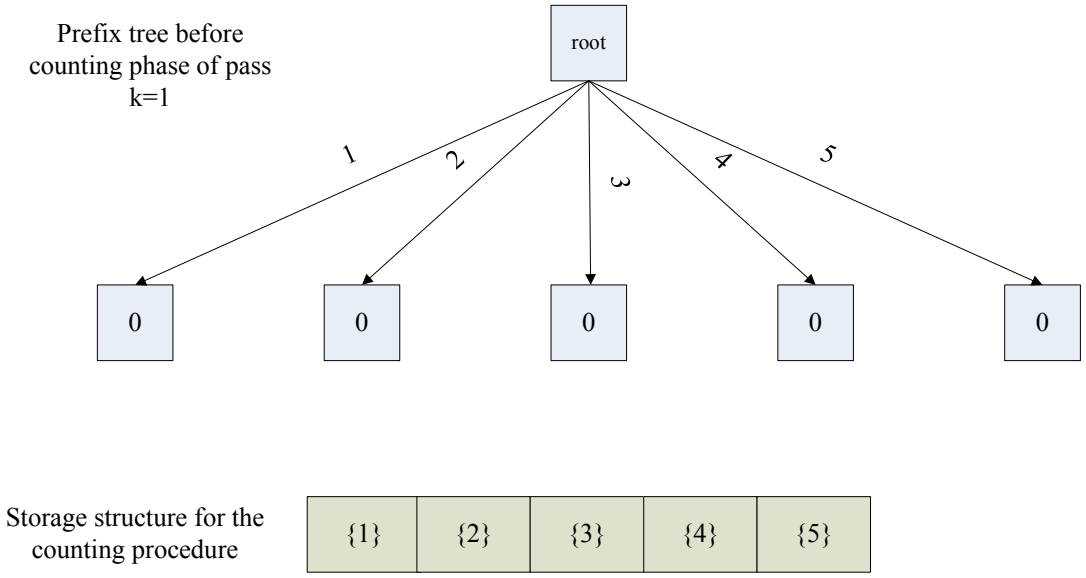



**Figure 33:** The prefix-tree and the storage structure before the counting phase in pass k=1

The next step is to initialize the search procedure which will determine the support for each candidate. For every transaction in the XML file we check which of the candidates are supported and increment the appropriate counter. A graphical representation of the search is shown bellow.

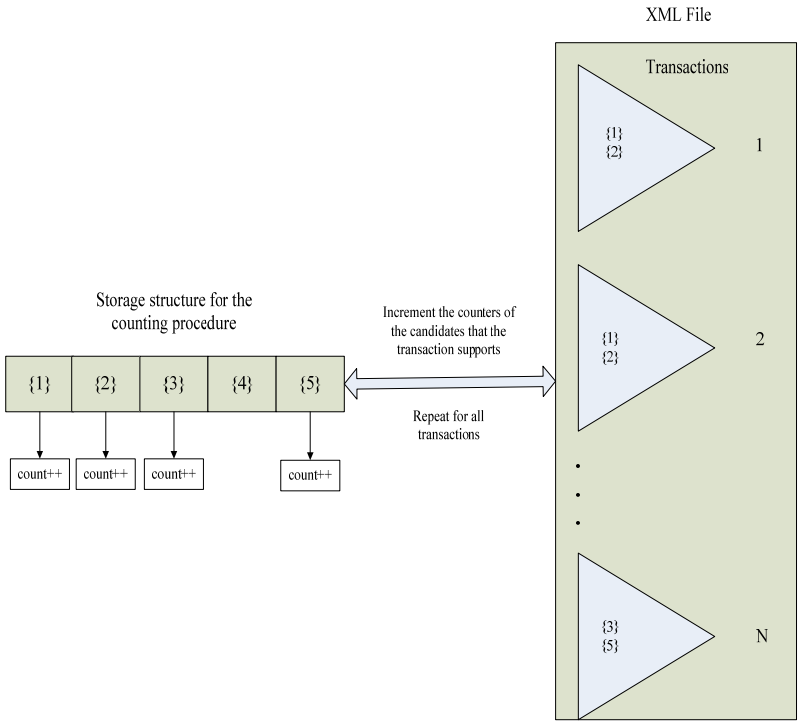



**Figure 34:** Graphical representation of the search procedure

Suppose that after the first pass of the algorithm the candidates found to be frequent are {1}, {2}, {3} and {5}. So the prefix tree and the candidates-array take the form shown bellow.

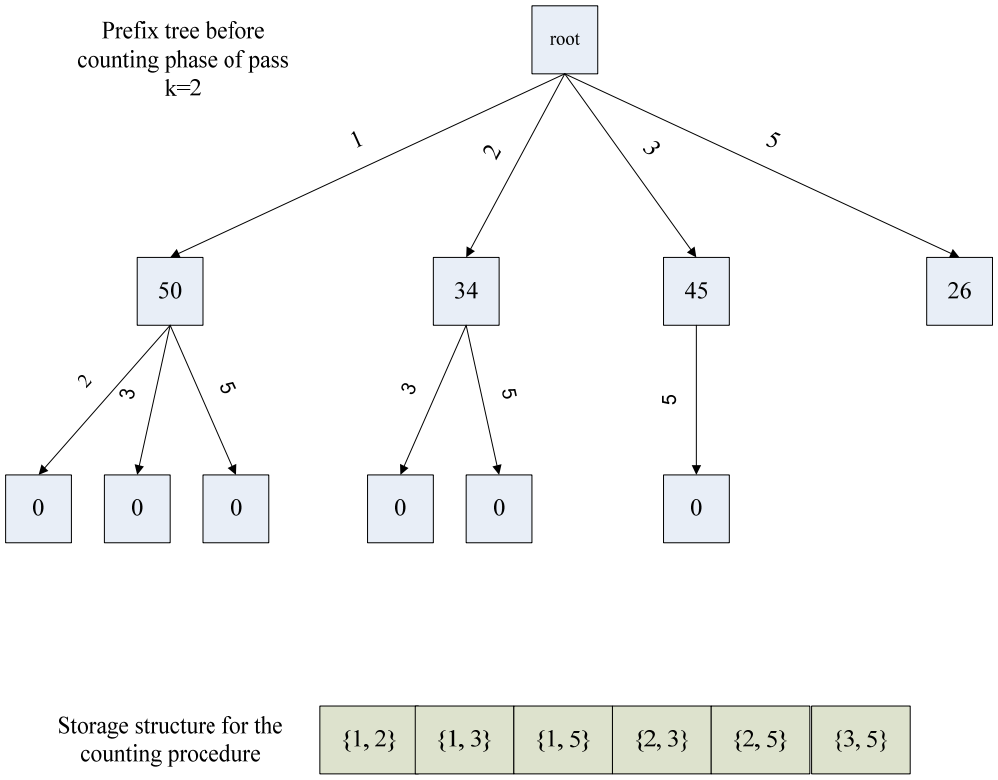



**Figure 35:** Prefix tree and candidates-array before counting phase of pass k=2

The counting procedure is executed again to determine the support of the new candidate sets. When apriori-like-gen generates an empty set for new candidates then the algorithm terminates and the formed prefix tree contains all the frequent itemsets. Each node of the final prefix tree contains necessary information (e.g. counters) about the frequent sets.

### 5.3.3 Removing useless items and transactions

The major problem with the sequential algorithm is that it always has to read the entire XML file in every pass, although many items and transactions are no longer needed after the first pass. One optimization for reducing the overhead of this problem is to remove from the XML file all items and transactions that will no longer be useful in later passes of the algorithm.

Assume that we have 20 individual items and wish to find association rules for the following items, {1}, {2}, {3}, {4} and {5}. In the first pass the support for the above items will be computed and according to minimum support it will be determined which are frequent. Let's suppose that {1}, {2} and {5} are frequent. Therefore the rest of the initial items cannot participate in an association rule and are no longer useful to the algorithm. So before continuing to the remaining passes we can perform a transformation of the original XML file in order to remove all useless items and transactions. Although such a transformation could take some time, it is preferable to perform it because it reduces the initial size of the XML file by a significant factor. The resulting decrease makes the later passes of the algorithm much faster. Figure 35 shows an example of the optimization.

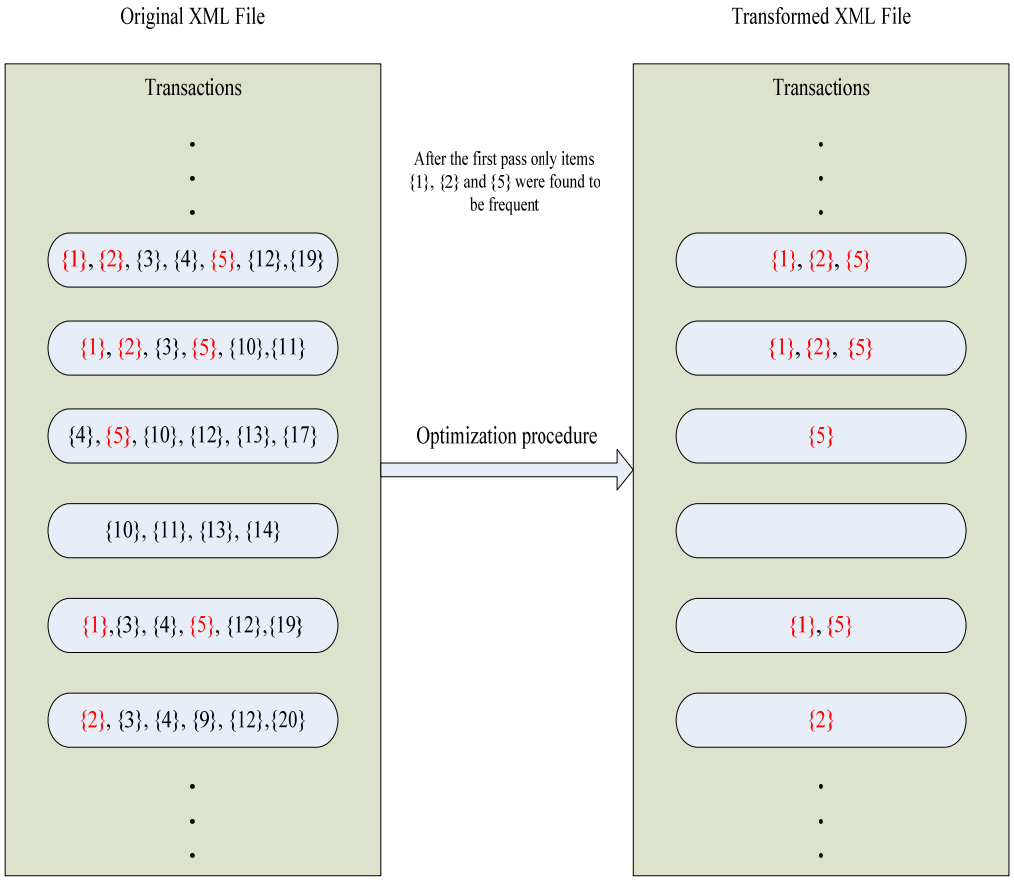



**Figure 36:** Transforming the original XML file according to the frequent 1-itemsets

From the above example it is obvious that the resulting XML file can be much smaller in size from the original. The decrease percentage depends on the number of items selected to participate in association rules and the underlying data of the XML file. Because both factors can vary in each case it is very difficult to come up with a general assumption of the decrease percentage. However from the experiments contacted in our system there was a decrease up to 80% from the original data.

## 5.4 Parallel algorithm for association rules mining

The sequential algorithm described in the previous sections has one major drawback regarding performance. As mentioned before the counting phase of each pass requires a full scan of the entire database. Transforming the original XML file may result in faster processing for later passes of the algorithm but still, if the XML file is extremely big in size, the problem remains.

The Grid described in chapter 3 provides computational resources which can be used to achieve faster results. In this section we describe a parallel version of the ARM algorithm that utilizes resources of the grid environment. In the parallel algorithm for ARM we split the initial XML file to smaller files which contain a portion of the original data. The number of the files is equal to the number of the computational resources (compute nodes). Afterward each compute node applies the sequential algorithm in its private data. Figure 36 bellow shows the pseudo-code for the parallel ARM algorithm:

```
C1 = {{i} | i∈I }          // all 1-itemsets
k=1
while ( Ck ≠ ∅ ) do
    for each proseccor i=1, 2, …, p do in parallel
        for all transactions T∈D{i}        // the private data of processor i
            for all k-subsets t⊆T
               if ( ∃c∈Ck : c = t) then c.count(i) ++
    end parallel

    {send count results to the master node}

    if ( proseccor i=0)  then          // master node
        for each c∈Ck
            c.count = ∑_{i=1}^{p} c.count(i)
    Lk = { c∈Ck | c.count ≥ miimum support }
    Ck+1 = generate_candidates(Lk)
    k ++

return L = Y k Lk
```

**Figure 37:** Pseudo-code for parallel ARM

The sequential ARM algorithm alternates between candidate generation and support counting steps, a pair of which we call one pass. Counting support for candidates can easily be done in parallel, each processor evaluating its private data. However the decision of which candidates to accept as frequent and which to discard cannot be done correctly without knowledge of the global support for these candidates. For that reason, every processor after the counting step

sends its results to the master node. The latter is responsible for adding the counts for all candidates and determining which are frequent. Then it creates the candidates for the next pass and the procedure repeats again. This scheme is depicted in the figure below:

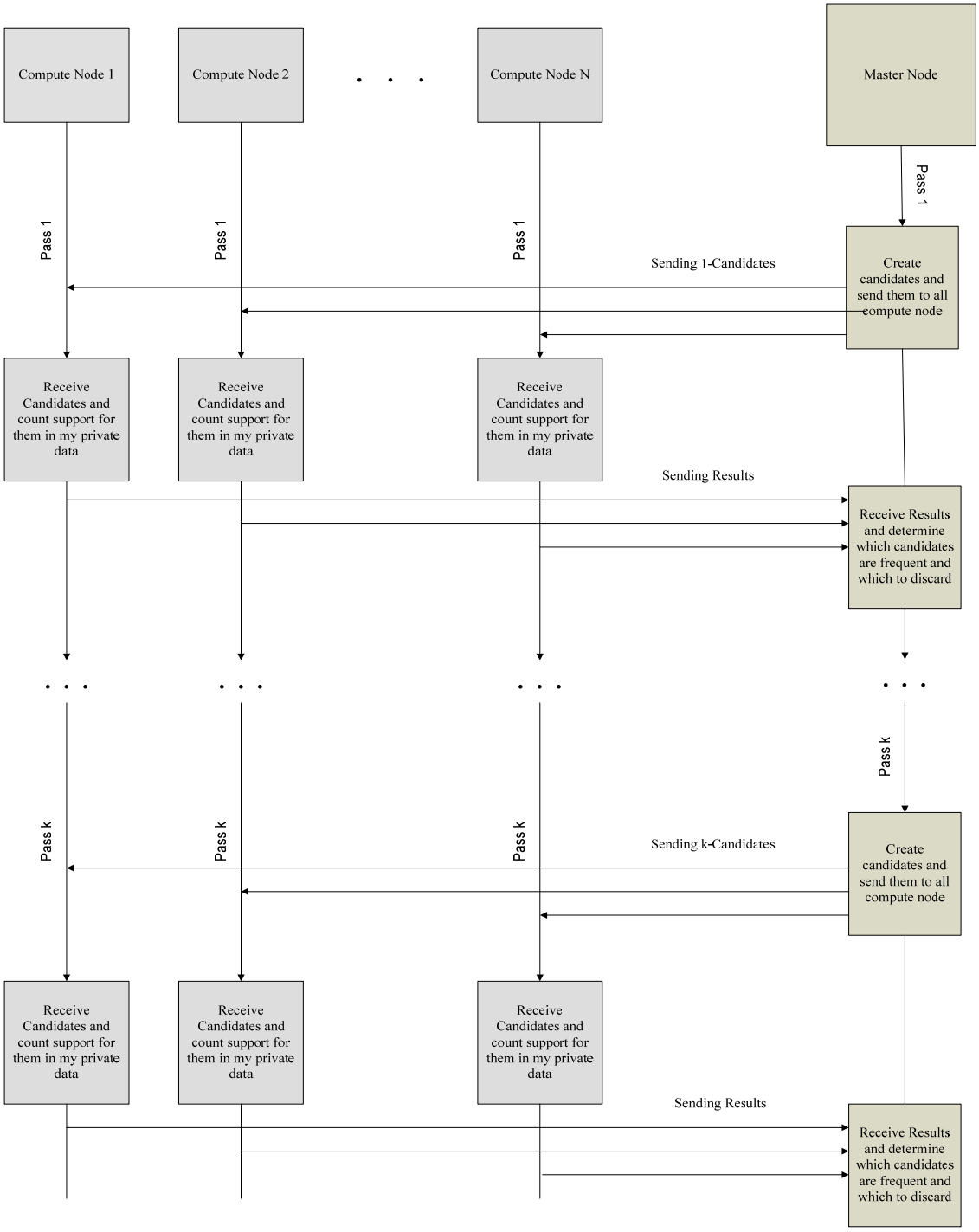



**Figure 38:** Parallel ARM algorithm scheme

In each pass the master node sends to all compute nodes the array-structure shown in section 5.3.2, which contains the candidates. Afterwards, all compute nodes calculate in parallel the support of those candidates according to its private data. As soon as a compute node finishes this procedure, it sends the results back to the master node. The next pass of the algorithm

cannot begin until all compute nodes send their results and the master node determines which candidates are frequent and which to discard.

The optimization described in section 3.2.3 is applied in the parallel algorithm as well. Therefore after the first pass, each compute node transforms the original XML file containing its private data. This results in a much smaller XML file, where all items and transaction no longer useful to the algorithm, have been omitted.

## 5.5 Handling missing values of items

The data mining algorithm described in the previous sections is applied in clinical data. For this type of data it is very common many attributes not to have a value. For example, assume that a patient has to make some blood examinations and that there are 10 different blood tests. So there are 10 attributes in the database to hold the values of those tests. Usually the patient will not perform all examinations but a certain number of them. Suppose that only 4 of the 10 tests are contacted. Therefore the transaction describing the examinations of the specific patient will contain values only in 4 out of 10 attributes. This phenomenon poses the question of how to handle missing values in the ARM algorithm.

Missing values can affect the support of an itemset. Assume attribute A which can take two values A-high and A-low. The support for A-high is *supp(A-high)* = "*number of transactions that support A-high" / "number of total transactions"* and the support for A-low is *supp(A-low)* = "*number of transactions that support A-low" / "number of total transactions"*. This scheme does not take into account the attribute's missing values. As a result an itemset which is frequent may be discarded. Another way of counting support is to divide the number of transactions that support an attribute with the number of transactions that the specific attribute has a value.

Assume, for example, that we have a total 100 transactions and that A-high, A-low are supported by 40 transactions, each. Minimum support is set to be 0.5. According to the first way of counting support we have the following results:

- Supp(A-high) = $\frac{40}{100} = 0.4 \leq 0.5$ the item is found to be infrequent
- Supp(A-low) = $\frac{40}{100} = 0.4 \leq 0.5$ the item is found to be infrequent

But with the second way we have:

- Supp(A-high) = $\frac{40}{80} = 0.5 \geq 0.5$ the item is found to be frequent
- Supp(A-low) = $\frac{40}{80} = 0.5 \geq 0.5$ the item is found to be frequent

The purpose of the ARM algorithm is to find associations between items in clinical data. Unfortunately the phenomenon of missing values may result to failure of the procedure. In order to avoid this situation we follow the second strategy for counting support.

## 5.6 Association Rules Generation

The outcome of the procedure described in the previous sections is the construction of a prefix-tree that contains all frequent itemsets. Information, like the support for each itemset is stored in the corresponding node of the tree. Therefore the construction of the actual association rules is now feasible. Suppose we want to discover the existence of an association rule (or rules) that involve the itemset {1, 2, 3}. In order to do this we must first construct all

possible rules that can be derived from this set and then examine if its confidence is greater or equal than the specified minimum confidence. For the set {1, 2, 3} all possible association rules are the following:

- 1 → 2, 3
- 2 → 1, 3
- 3 → 1, 2
- 1, 2 → 3
- 1, 3 → 2
- 2, 3 → 1

In order for rule R: {1 → 2, 3} to be valid the following must be true:

$$conf(R) = \frac{totalSupport}{leftSupport} = \frac{\sup(\{1,2,3\})}{\sup(\{1\})} \geq \min Conf$$

The pseudo-code for generating association rules is shown in the figure bellow.

```
V = ∅  // contains the valid association rules

for all itemsets c ∈ prefixTree do
    R = generate_possible_rules( c )

    for all rules r ∈ R do
        conf = compute_confidence ( r )
        if conf ≥ minimum_confidence do
            V = V Y r

return V
```

**Figure 39:** Pseudo-code for generating association rules

An itemset with n items has $2^n - 2$ possible association rules. So it is obvious that as the size of the prefix-tree increases so does the number of possible association rules that has to be examined. For this reason it would be more efficient to parallelize this procedure.

One characteristic of the prefix-tree structure is that each branch of the tree can be used separately from the others to generate association rules. Therefore it is possible to divide the tree in a vertical manner, as the figure bellow shows, and process each part independently.

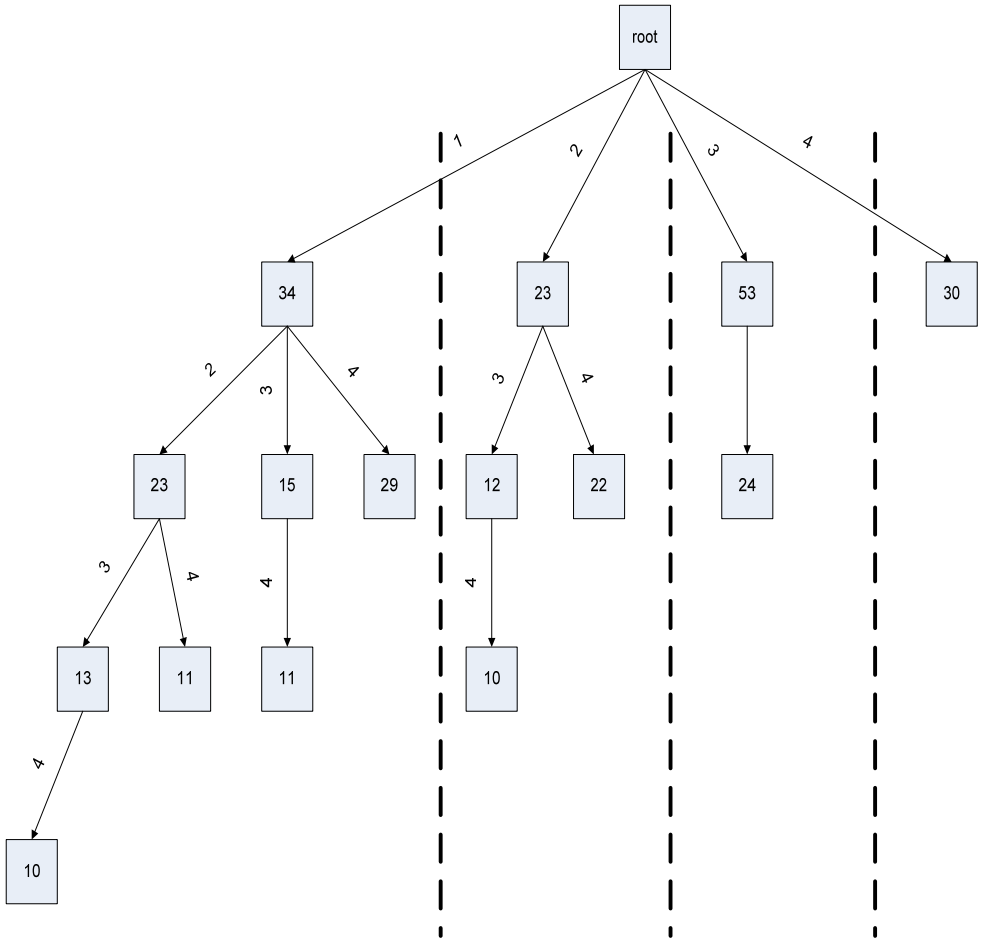



**Figure 40:** Divided prefix tree with four items

Each branch of the tree can be assigned to a computing node for processing. Suppose we want to examine if the rule R: {3 → 1, 2} is valid. In order to do this we must know the support for itemset {1, 2, 3} and the support for itemset {3}. For the first set this information can be found in the first branch of the tree but the support counter for the second set is stored in the third branch. Therefore all computing nodes must have a copy of the entire prefix-tree.

If the width of the prefix tree is greater than the number of the available computing nodes it is not possible to assign each branch in a specific node. Suppose there are 4 computing nodes and the prefix tree has 10 branches. The first four branches are assigned to each compute node. When a node finishes his branch it starts processing the next available. This procedure will end when all branches are examined. So, it is necessary to keep track of the branches that have already been processed. This task is assigned to the master node. The following figure depicts this algorithmic scheme.

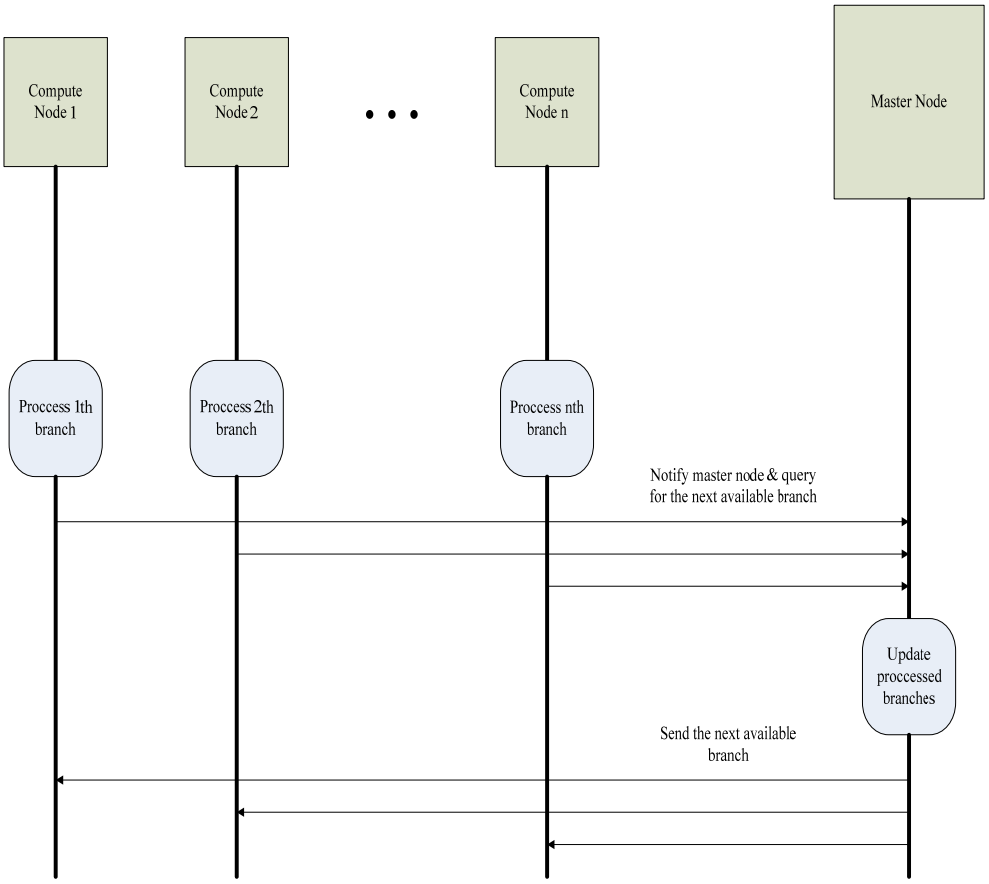



**Figure 41:** A parallel approach for generating association rules

In real life situations the resulting prefix tree is not so big in size. Therefore the above parallel approach may be slower than the sequential one. This occurs because of the communication overhead. So, if the width of the prefix tree is smaller than a specified threshold then the above algorithm is executed only on the master node.

## 5.7 Graphical User Interface

The figure bellow shows the graphical user interface developed for the needs of this master thesis.

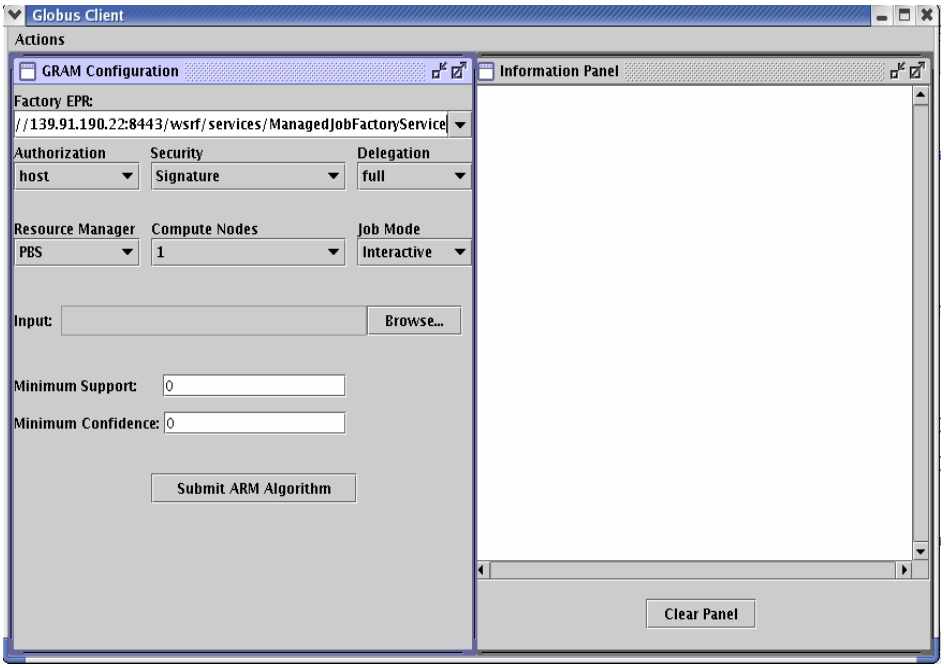



**Figure 42:** Grid System – Graphical User Interface

With the above GUI the user is able to perform the following actions:

- Query for available data sources and retrieve information in XML document format
- Query for available computational resources
- Configure all necessary parameters for the ARM algorithm
- Configure all necessary parameters for the GRAM services
- Submit the data mining algorithm for execution onto the grid infrastructure
- Retrieve notifications regarding the submitted job state
- Show the results of the ARM algorithm

**GRAM Configuration**

Before submitting a job to the grid it is required to configure properly all the necessary parameters that the GRAM services will need.

- **EPR (Endpoint Reference).** GRAM services are deployed in a Globus container (server). In order to contact these services it is necessary to use some form of addressing. In the Web Service Resource Framework each service can be invoked by using the endpoint reference which has the form of a URI (Uniform Resource Identifiers). EPRs are used to address a specific WS-Resource which is a pairing of a Web Service and a resource. In Globus Toolkit 4 the Web Service responsible for submitting and monitoring jobs is called "ManagedJobFactoryService". Each globus installation has its own default GRAM service which is deployed automatically when starting the globus container. The EPR for contacting the

"ManagedJobFactoryService" of our globus installation is the following: *https://139.91.190.22:8443/wsrf/services/ManagedJobFactoryService*.

- **Authorization.** This parameter can take two values: *host* or *self*. When using host authorization the client will authorize an invocation if the service has a host credential. The client must be able to resolve the address of the host to the hostname specified in the host credential. With self authorization the client will authorize an invocation if the service's identity is the same as the client. In our case only host authorization will allow a valid invocation of the "ManagedJobFactoryService" because all services in the globus container are deployed using host credential.
- **Message Security.** This parameter specifies the message protection level. There are two choices: *"sig"* for XML signature and *"enc"* for XML encryption. With the XML signature method a message is signed with the given credentials. With the XML encryption method a security context is first established between a client and the service which is then used to sign/verify/encrypt/decrypt messages.
- **Delegation.** There are three choices regarding the delegation policy of our grid client. These are: *full, limited or none.* The client may delegate credentials to the server. These credentials can then be used by the server (or processes initiated by the server) on the client's behalf. If the user chooses the "*none*" option then no delegation will be performed between the client and the server. In "*full*" delegation the server has complete rights on the client's credential. However in "*limited*" delegation the server cannot use the client's credentials for process invocation.
- **Job Mode.** The user can choose to submit the job in a *batch* or an *interactive* mode. In batch mode the application does not wait for started job to complete and it does not destroy the started job service on exit. Instead the job's handle is printed in the *information panel* which can be used to query for the job's state. On the contrary when interactive mode is used the application will not respond to any user actions until termination of the job. In addition state notifications will be displayed by the *information panel* to inform the user about the job's state.
- **Resource Manager.** The actual management of the submitted jobs will not be performed by the GRAM services. An external software component called resource manager is responsible for handling this task. GRAM provides the necessary interface for controlling three very popular resource managers: PBS, LSF and Condor. The TORQUE resource manager is installed in our grid experimental environment. TORQUE is based on openPBS so for this parameter the default choice is: *PBS*. However there is another choice: *fork*. By enabling this choice the submitted job will not be passed to a resource manager for execution. Instead the job will be executed in the globus installation machine with the use of the *fork* program. The second approach does not utilize the available computational resources of the grid system.

**ARM algorithm configuration**

The parallel ARM procedure requires certain input arguments to run properly. These arguments, which are depicted bellow, can be configured by the user through the graphical user interface.

- **"data.xml" file.** This file contains the data that will be mined by the parallel ARM procedure.
- **"data.int" file.** This file contains the domain values for every attribute of the clinical information.

- **"selectedAttr.txt" file.** This file contains the names of the attributes which the user has chosen to participate in the ARM procedure. This means that the procedure will search for association rules only between those attributes.
- **Minimum support.** This parameter is vital for the ARM algorithm and it can take values between 0.0 – 1.0
- **Minimum confidence.** This is the second parameter that the ARM algorithm will need to generate association rules. It can take values between 0.0 – 1.0

Before submitting the ARM procedure for execution in the grid, it is crucial to configure properly all the above input arguments. The picture bellow shows the GUI that allows the user to perform this configuration. The graphical components involved with the above parameters are circled in red.

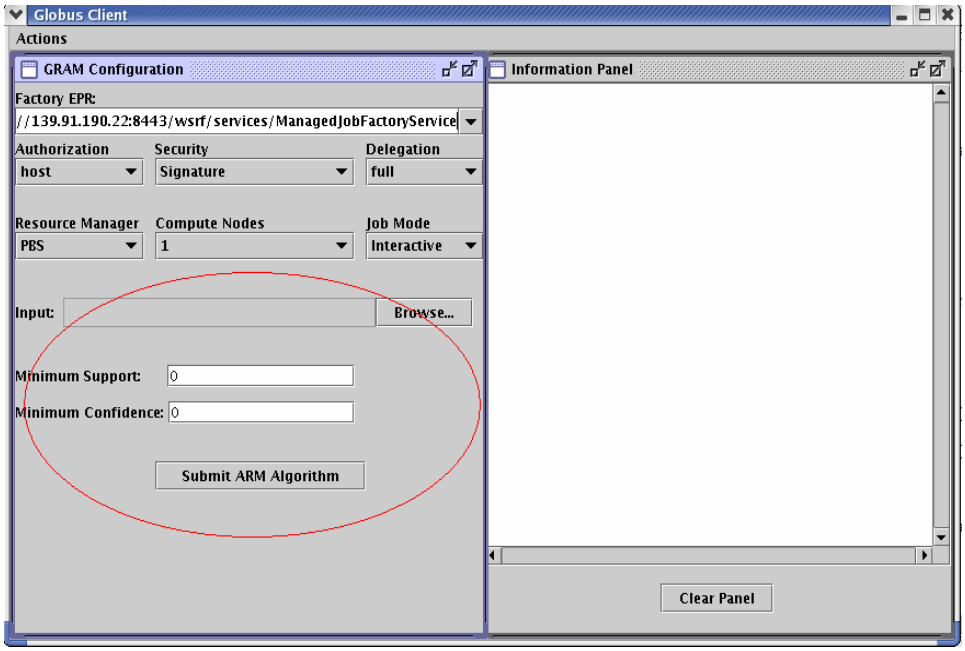



**Figure 43:** Graphical components involved in the ARM configuration

The "browse" button shown above displays a dialog window for browsing the local file system. This way the user is not restricted in his home directory. Instead he has the flexibility to search for the data that will be processed by the algorithm, in the entire file system. Except from the XML file that contains the clinical information there must be two more files in the same directory. The domain semantics file, which was mentioned above, and a DTD file that describes the data. All three files must have the same name. For example if the clinical information is in a file named "archanes.xml" then the files "archanes.int" and "archanes.dtd" must be present in the same directory.

Once the user selects the XML file containing the data the application will search in the same directory for the corresponding DTD file. The latter will be parsed and a new interface like the one shown bellow will appear.

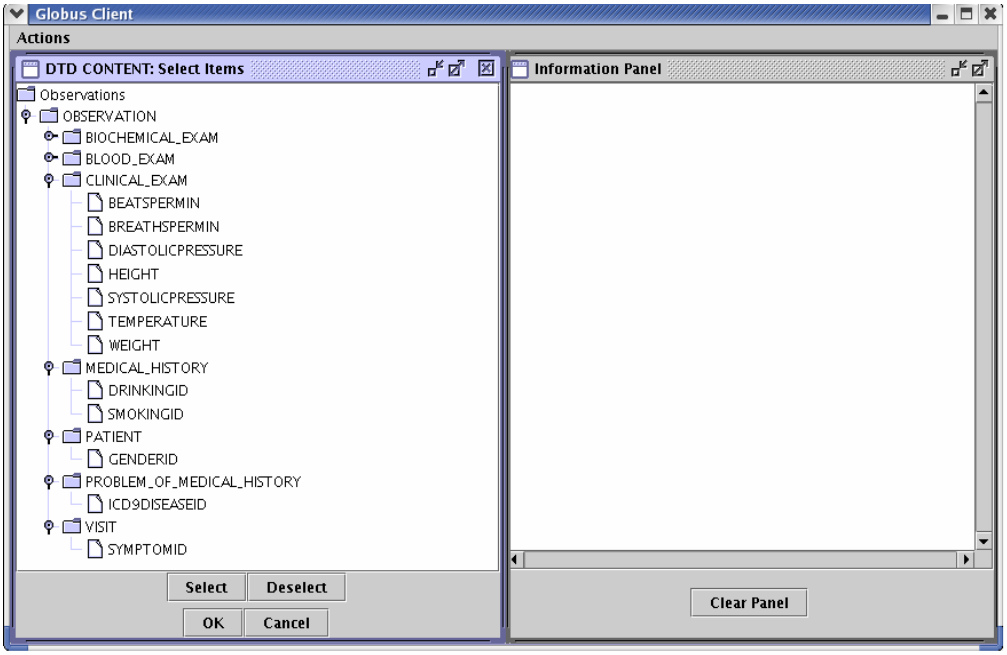



**Figure 44:** DTD content interface

Through the above interface the user has the ability to select (or deselect) the attributes that will participate in the ARM algorithm. The selected attributes are stored in a file named "selectedAttr.txt".

The "information panel" shown below displays information about the current state of the submitted job.

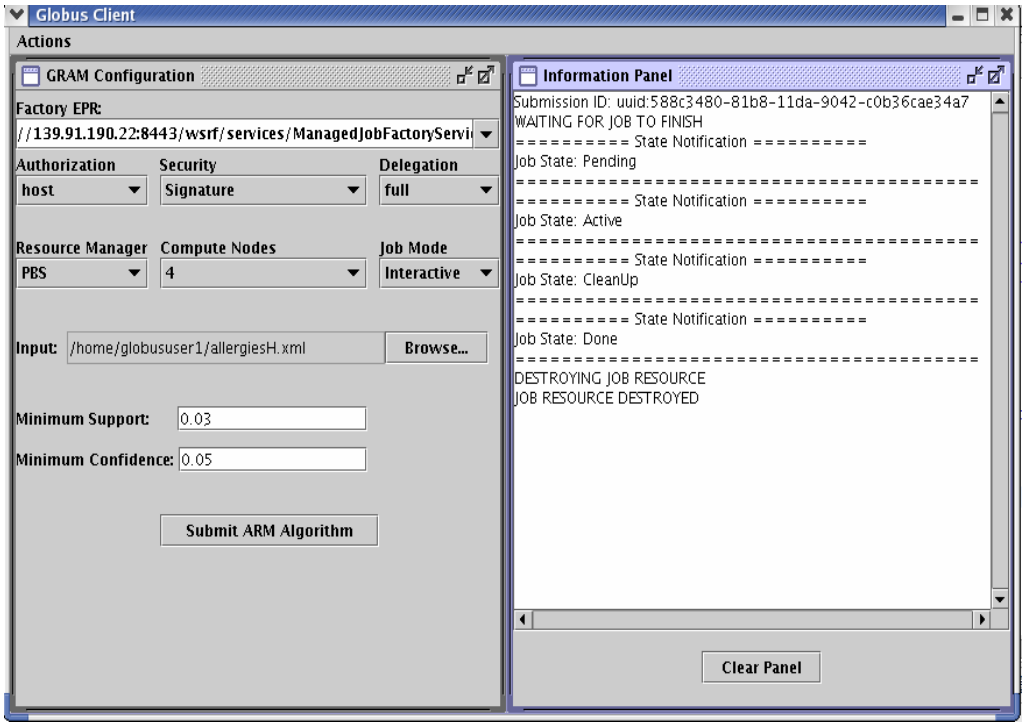
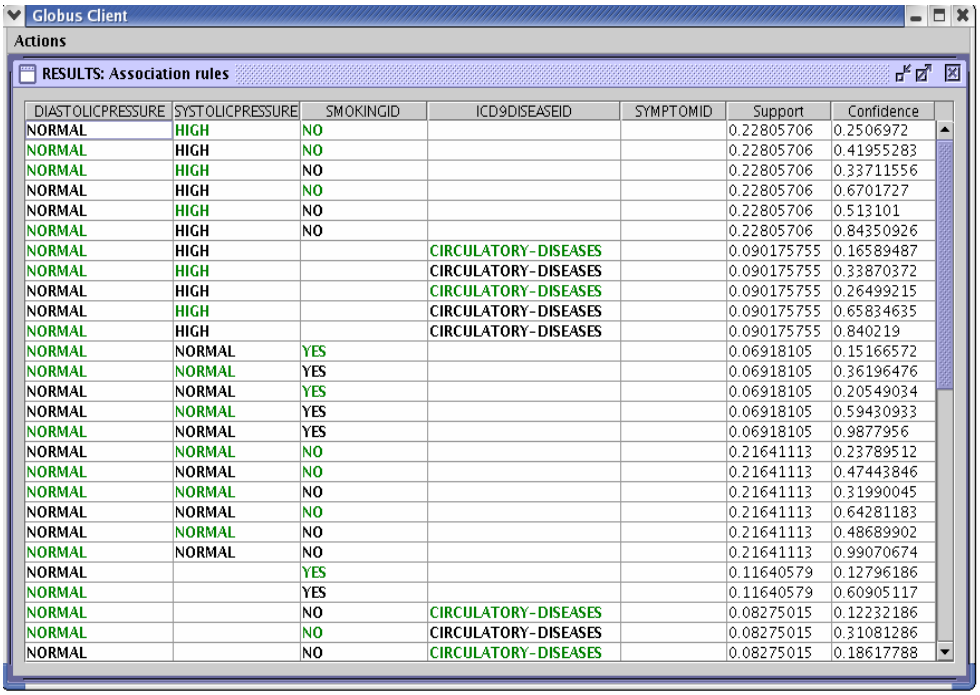



**Figure 45:** Information panel interface

State notification like the ones shown above are only displayed if the user chooses to submit a job in an interactive mode. If all goes well the final state notification will be: "*Done*". This means that the job was submitted and executed properly. If the parallel ARM algorithm finds any association rules, it stores them in a file named "results.txt". This file is parsed by the application and its content is displayed through the following interface:




| DIASTOLICPRESSURE | SYSTOLICPRESSURE | SMOKINGID | ICD9DISEASEID | SYMPTOMID | Support | Confidence |
|---|---|---|---|---|---|---|
| NORMAL | HIGH | NO | | | 0.22805706 | 0.2506972 |
| NORMAL | HIGH | NO | | | 0.22805706 | 0.41955283 |
| NORMAL | HIGH | NO | | | 0.22805706 | 0.33711556 |
| NORMAL | HIGH | NO | | | 0.22805706 | 0.6701727 |
| NORMAL | HIGH | NO | | | 0.22805706 | 0.513101 |
| NORMAL | HIGH | NO | | | 0.22805706 | 0.84350926 |
| NORMAL | HIGH | | CIRCULATORY-DISEASES | | 0.090175755 | 0.16589487 |
| NORMAL | HIGH | | CIRCULATORY-DISEASES | | 0.090175755 | 0.33870372 |
| NORMAL | HIGH | | CIRCULATORY-DISEASES | | 0.090175755 | 0.26499215 |
| NORMAL | HIGH | | CIRCULATORY-DISEASES | | 0.090175755 | 0.65834635 |
| NORMAL | HIGH | | CIRCULATORY-DISEASES | | 0.090175755 | 0.840219 |
| NORMAL | NORMAL | YES | | | 0.06918105 | 0.15166572 |
| NORMAL | NORMAL | YES | | | 0.06918105 | 0.36196476 |
| NORMAL | NORMAL | YES | | | 0.06918105 | 0.20549034 |
| NORMAL | NORMAL | YES | | | 0.06918105 | 0.59430933 |
| NORMAL | NORMAL | YES | | | 0.06918105 | 0.9877956 |
| NORMAL | NORMAL | NO | | | 0.21641113 | 0.23789512 |
| NORMAL | NORMAL | NO | | | 0.21641113 | 0.47443846 |
| NORMAL | NORMAL | NO | | | 0.21641113 | 0.31990045 |
| NORMAL | NORMAL | NO | | | 0.21641113 | 0.64281183 |
| NORMAL | NORMAL | NO | | | 0.21641113 | 0.48689902 |
| NORMAL | NORMAL | NO | | | 0.21641113 | 0.99070674 |
| NORMAL | | YES | | | 0.11640579 | 0.12796186 |
| NORMAL | | YES | | | 0.11640579 | 0.60905117 |
| NORMAL | | NO | CIRCULATORY-DISEASES | | 0.08275015 | 0.12232186 |
| NORMAL | | NO | CIRCULATORY-DISEASES | | 0.08275015 | 0.31081286 |
| NORMAL | | NO | CIRCULATORY-DISEASES | | 0.08275015 | 0.18617788 |

**Figure 46:** Results visualization interface

For every rule discovered by the ARM procedure the items comprising the head of the rule are printed with black color and the items comprising the body to the rule are printed with green color. The support and confidence of each rule is also displayed.

## Actions menu

The "actions menu" has the following three choices:

- **Query Data Sources.** This choice displays a new interface from which the user can query for all available data sources on the grid and retrieve data from those sources in XML document format.
- **Query Job State.** If the user submits a job in a batch mode then it will not be notified automatically about the job's state. Provided a job handle the user can retrieve state notifications through the "query job state" interface.
- **Domain Editor.**

### Query Data Sources interface

The picture bellow shows the "Query Data Sources" interface. Through this interface the user can find out which clinical information systems are integrated to the Grid system. The point of contact for the client that wishes to access, retrieve or update information is an OGSA-DAI data service which is basically a web services deployed in the globus container. Therefore the user must know the end-point reference (EPR) of the data service in the grid system.

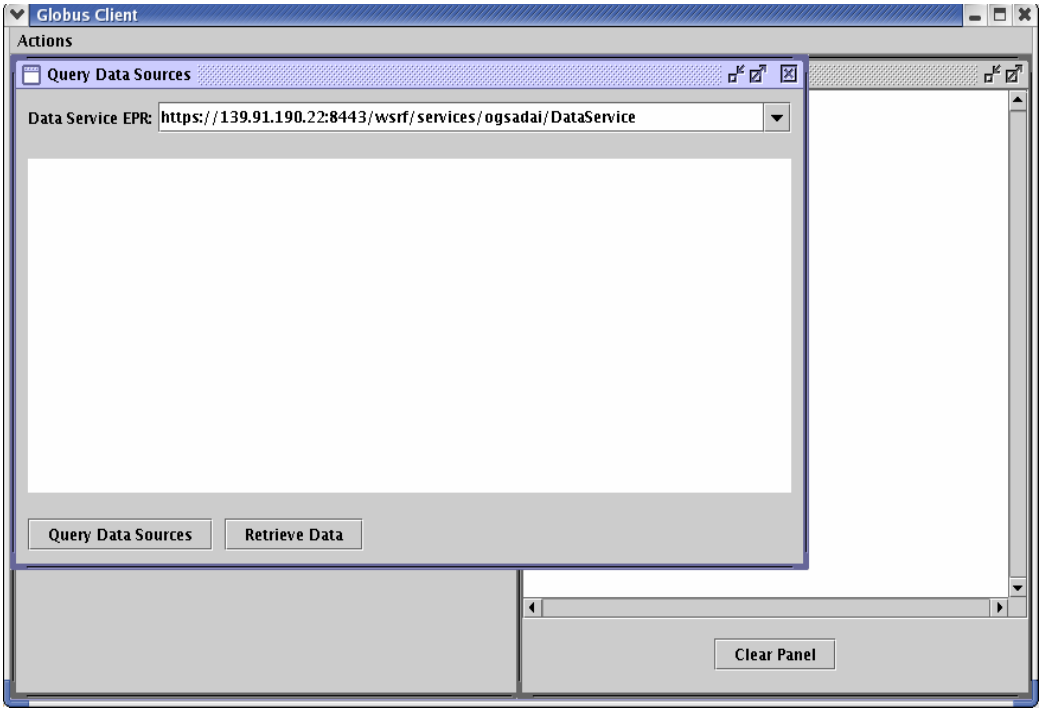



**Figure 47:** Query data sources interface

## Query Job State interface

The user has the option of submitting a job in batch mode. In this case the application will return the job's handle and it will not wait for the job to finish. The only way to keep track of the job's state is through the returned handle. The interface shown bellow was developed for acquiring state notifications from jobs submitted in batch mode.

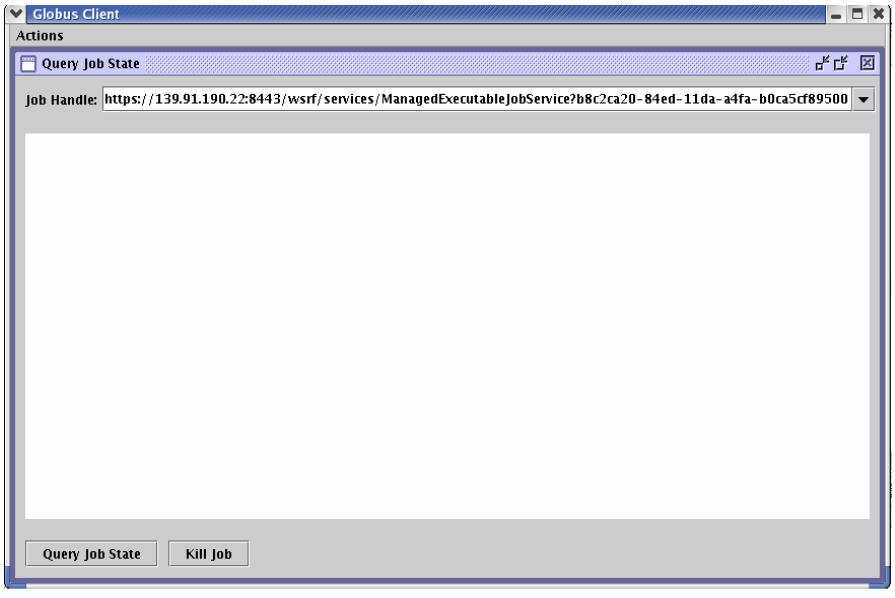



**Figure 48:** Query job state interface

### Domain Editor

Through the domain editor GUI, shown bellow, the user is able to edit the domain semantics file. This interface allows the user to perform the following actions:

- **Delete** an attribute's domain by simply pressing the "Delete" button on the right of the attribute
- **Add** a new domain for an attribute. The user can fill the text fields shown bellow with the minimum and maximum value of a new domain and then press the "Add" button.

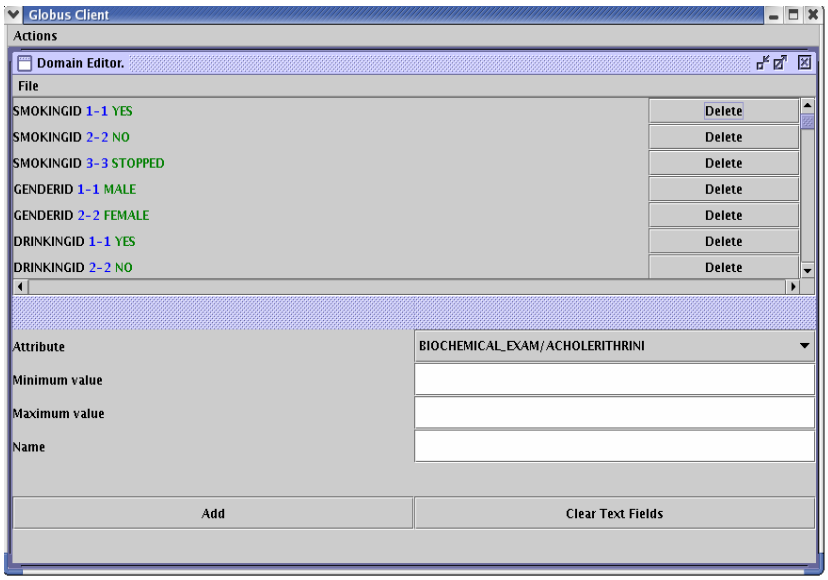



**Figure 49:** Domain editor GUI

## 5.8 Results

The figure bellow shows the resulting running times of the parallel ARM algorithm in relation with the number of computing nodes.

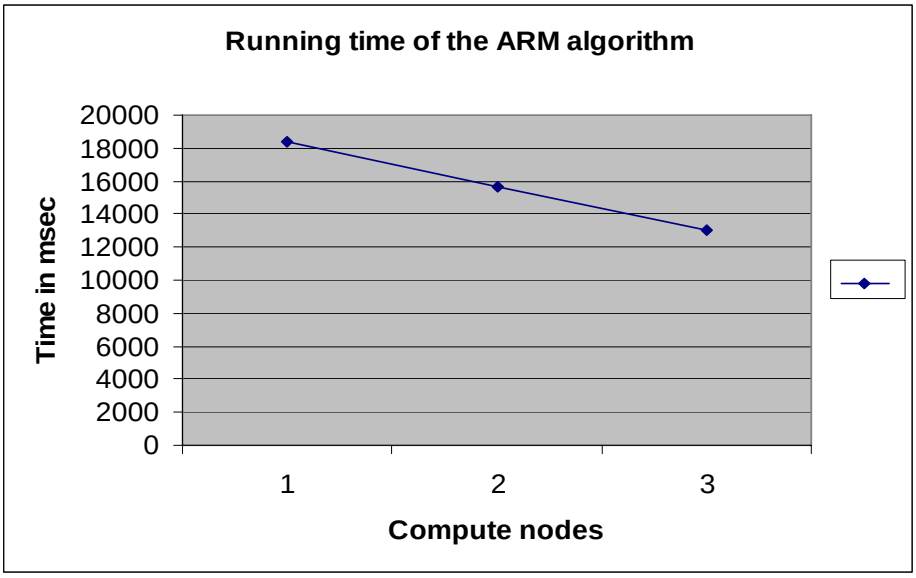



**Figure 50:** Execution times of the ARM algorithm (with 10 clinical attributes) in relation with the number of computing nodes

In the experiments, minimum support was set to 0.6 and the data set contained nearly 30.000 transactions. From the above figure it is clear that the running time of the algorithm decreases as we increase the number of computing nodes. A much larger data set results in decrease of the achieved speedup but it is obvious that the algorithm scale well as we increase the number of processors. Another factor that affects the running time of the algorithm is minimum support. Small values of minimum support result in higher running times because more candidate sets are considered to be frequent.

In order to measure the running time of the algorithm we calculate the time each compute node needs to count the support for every candidate set, for all passes of the procedure. Then we add the highest values of theses times and come up with the total running time of the algorithm. We also conducted every experiment ten times and calculated the average time of execution.

The figure bellow shows the running times of the algorithm including the communication overhead. The communication overhead is affected mainly by the number of attributes selected to participate in the ARM procedure. A larger number of attributes results in large communication size and finally in higher communication overhead regarding time.

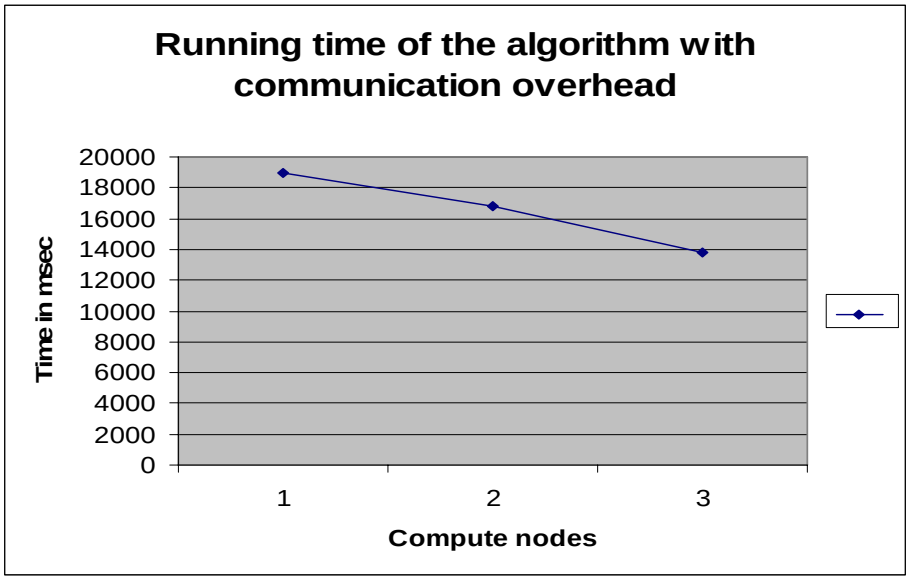



**Figure 51:** Execution time of the ARM algorithm (with 10 clinical attributes) with communication overhead

The resulting running times shown above were derived from experiments where we selected 10 clinical attributes to participate in the ARM procedure. We also conducted experiments where half and all of the clinical attributes were chosen to participate. In those experiments we included the communication overhead and the results are shown in figures 51 and 52.

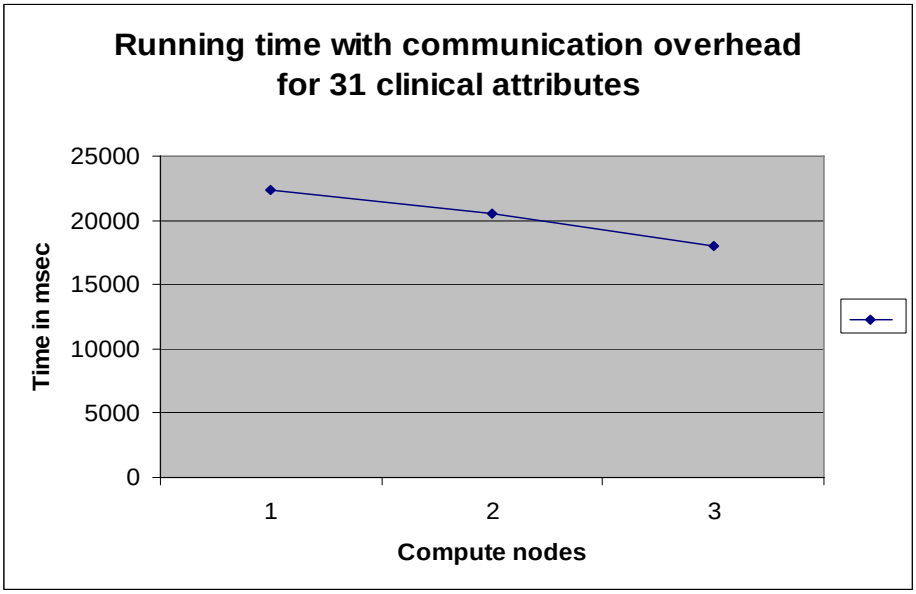



**Figure 52:** Execution time of the ARM algorithm with half (31) clinical attributes

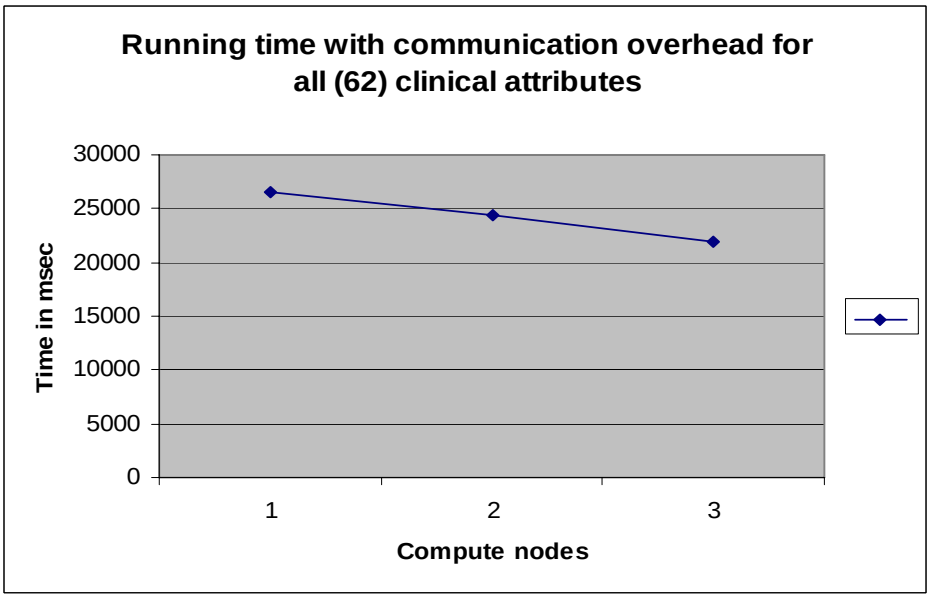



**Figure 53:** Running time of the ARM algorithm with all (62) clinical attributes

## Increasing the number of compute nodes

Using the same minimum support and number of transactions we conducted experiments with five compute nodes for 10 and 31 clinical attributes. The results are shown in the following figures:

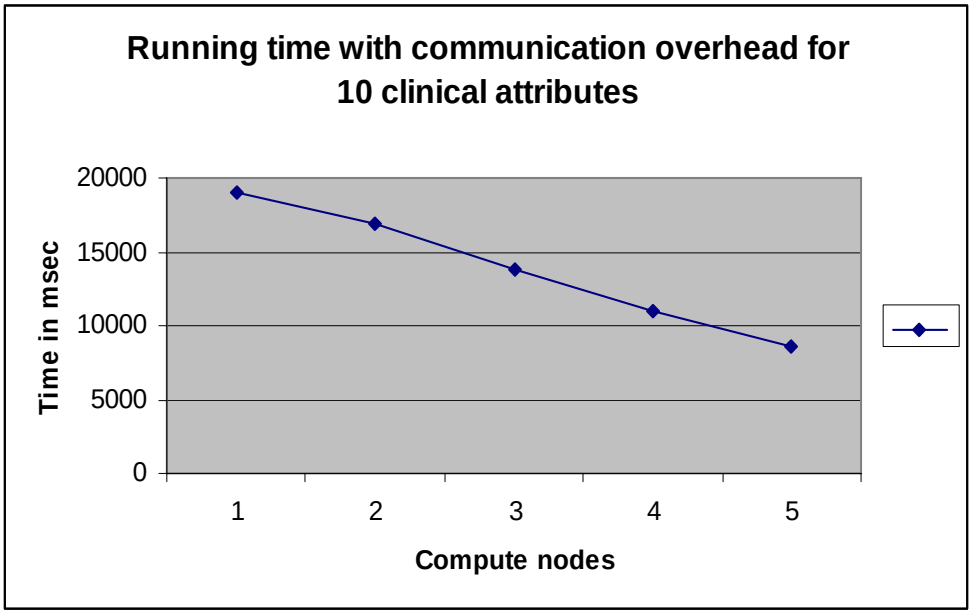



**Figure 54:** Running time of the ARM algorithm with 5 compute nodes for 10 clinical attributes

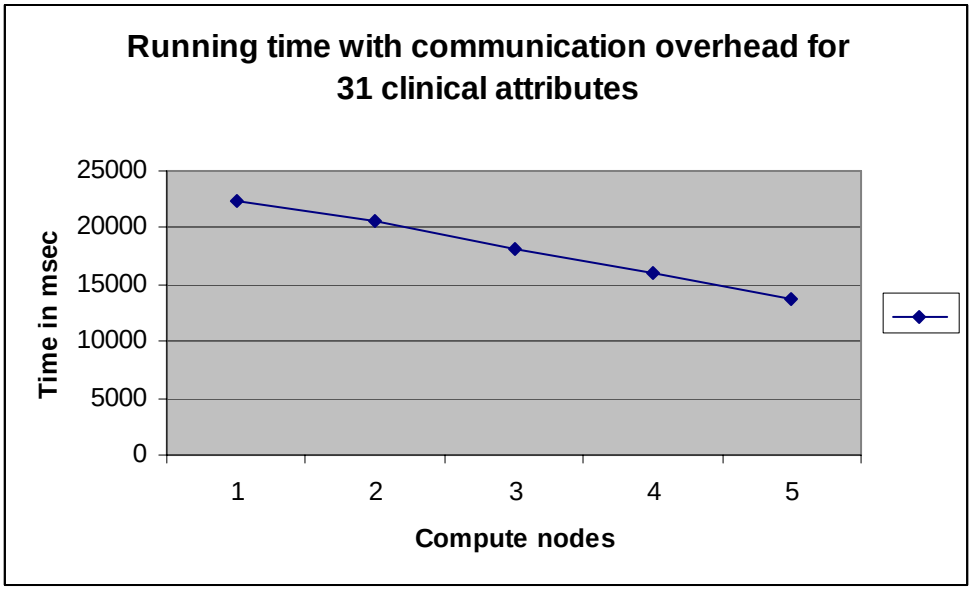



**Figure 55:** Running time of the ARM algorithm with 5 compute nodes for 31 clinical attributes

# 6 Discussion

The Grid environment implemented in this master thesis is mainly used for supporting high-performance computing intensive data mining algorithms. These algorithms are in the form of executables that can be submitted through our Grid environment to the available computational resources. However the grid can be effectively exploited for implementing data intensive and knowledge discovery applications. To succeed in supporting this class of applications, tools and services for data mining and knowledge discovery on Grids are essential. In this section we will discuss approaches for implementing data mining tools as open Grid Services.

## *6.1 Issues for developing data mining tools as grid services*

Since the release of Globus Toolkit 4 which is based on the Web Services Resource Framework (WSRF) the implementation of a Grid Service is in fact an implementation of a stateful Web Service. Writing and deploying a WSRF Web Service is a five steps procedure:

1. Define the service's interface. The interface specifies what the service is going to provide to the outer world. There is special XML language which can be used to specify the interface: the Web Service Description Language (WSDL).
2. Implement the grid service. This is done using the Java programming language. The implementation is "how the service does what it says it does".
3. Define the deployment parameters. This is done with WSDD and JNDI.. This step makes the Web Service available to client connections. It takes all the components of the previous steps and makes them available through a Web Services container. One of the key components of the deployment phase is a file called deployment descriptor which tells the Web Services container how it should publish Web Services. The deployment descriptor is written in WSDD format (Web Service Deployment Descriptor).
4. Compile everything and generate a GAR file. This step takes the tree files created in the previous steps and generates a Grid Archive (GAR file). This GAR file contains all the files and information the Web Services container needs to deploy the service and make it available to the whole world. This procedure is done using the Ant tool.
5. Deploy the service. Finally in the last step a specific GT4 tool unpacks the GAR file and copies the files within (WSDL, compiled stubs, compiled implementation, WSDD) into key locations in the GT4 directory tree.

The above five steps must be followed in order to implement any Grid Service. However the task of implementing Data Mining Tools as Grid Services is much more complicated. Such a task poses a lot of challenges which can be classified into the following:

- **Distributed data:** The data to be mined is stored in distributed computing environments on heterogeneous platforms. Both for technical and for organizational reasons it is impossible to bring all the data to a centralized place. Consequently, development of algorithms, tools, and services is required that facilitate the mining of distributed data.
- **Distributed operations:** In future more and more data mining operations and algorithms will be available on the grid. To facilitate seamless integration of these resources into distributed data mining systems for complex problem solving, novel algorithms, tools, grid services and other IT infrastructure need to be developed.

- **Massive data:** Development of algorithms for mining large, massive and high-dimensional data sets (out-of-memory, parallel, and distributed algorithms) is needed.
- **Complex data types:** Increasingly complex data sources, structures, and types (like natural language text, images, time series, multi-relational and object data types etc.) are emerging. Grid-enabled mining of such data will require the development of new methodologies, algorithms, tools, and grid services.
- **Data privacy, security, and governance:** Automated data mining in distributed environments raises serious issues in terms of data privacy, security, and governance. Grid-based data mining technology will need to address these issues.
- **User-friendliness:** Ultimately a system must hide technological complexity from the user. To facilitate this, new software, tools, and infrastructure development is needed in the areas of grid-supported workflow management, resource identification, allocation, and scheduling, and user interfaces.

In order to solve some of the above challenges it is necessary to develop the following technology components:

- Grid enabled data mining data interfaces and services.
- Data mining tools in the form of grid services.
- Grid enabled data mining workflow management tools.

Grid-enabled data-mining data interfaces and services provide mechanisms that allow users to identify (locate), access, integrate and interface distributed data sources and data-mining programs in a flexible way. This includes seamless access and selection of subsets of data, data transfer, data pre-processing and metadata functionality.

Data Mining Tools implemented as Grid Services encompass all services which perform some sort of computationally expensive calculations in order to produce the desired results. They are opposed to data services which fetch, store, transport, and transform data only. A typical example of a data service use case is the situation, when we can not (because of privacy and security reasons) or would not like to (because of very large data sets) move data from its original location. In such cases the execution of algorithms on the server holding the data or on a server belonging to the organization owning these data is essential for the ability to execute the job at all. This requirement can be satisfied by moving data mining algorithms to a specific server. Therefore, a service has to be developed that initiates the transfer of the algorithm and all related libraries and prompts the resource scheduler to execute the algorithm.

Workflow management tools are necessary in order to act as the major interface for users to the system. They provide graphical creation, loading, and storing of workflows to be executed in the grid. As managing the various aspects of grids such as job submission, job monitoring, and error handling are inherently complex, the tools will provide significant support for the user by insulating him from the details of these aspects. They will also help enforcing complex data mining tasks, which might consist of numerous different steps and usually include accessing, transformation, and pre-processing data as well as applying various Data Mining algorithms, evaluating the results, and storing the models on an appropriate server in the grid or on the client machine.

It is obvious that such a Grid Environment is quite complicated and very difficult to implement. It contains many software components and it would require the development of a very complicated experimental environment. Designing and developing a system of this size would not be realistic; therefore we chose to develop a simpler grid environment in order to explore how grid computing can handle compute-intensive data mining tasks.

# 7 Future Work and Conclusions

## 7.1 Possible extensions of the Grid System

The architecture of the grid environment developed in this master thesis is comprised of three layers: Data Layer, Grid Layer and Knowledge & Discovery Layer. For reasons of simplification we assumed that all databases in the data layer have a common schema. So, all health institutions use a specified common schema to store clinical information. In our case this assumption solves any ontology problems.

In real-life situations where a grid is composed of several health institutions the data layer will have a significant degree of heterogeneity. This poses a major problem for the execution of the available knowledge extraction tools. Assume, for example, that two different health organizations have a slightly different name for a specific blood examination and that a researcher wants to apply a data mining algorithm in both organizations' data. Because of this variation any knowledge discovery operation containing the specified examination would not be possible.

In order to solve this problem we propose the use of ontologies in heterogeneous data integration. This extension of our Grid system is shown in figure 54.

Our approach to heterogeneous data integration will be based on a mediator-wrapper architecture and the use of ontologies/metadata. In particular, the mediator will integrate heterogeneous data sources by providing a virtual view of all this data. Users (including grid tools and services) asking queries to the mediated system do not have to know about data source location, schemas or access methods, because the system will present one shared mediator ontology to the user and users will ask their queries in terms of it.

The development of a Biomedical master ontology involves the analysis of (i) the ontological needs of the domain and the clinical scenarios of relevance and (ii) the ontological foundations and coverage of the existing terminologies and ontologies in the biomedical domain. Based on the analysis of (i) and (ii), it will become possible to craft an ontology that is able to function as a semantic mediator between all systems to be integrated.

In order for the mediator to integrate the various heterogeneous data sources, their object models, terminologies, embedded domain ontologies, hidden semantic information, query capabilities and security information will be analysed. Based on this analysis, a source description will be defined which consists of a local ontology along with a set of metadata, specifying query capabilities and security information. Thus a source description is an abstraction of a particular data source, possibly conveying semantic information not present in the data source schema.

Wrappers are software components, providing source dependent data services to the mediator. Each wrapper receives queries form the mediator in terms of the local ontology, transforms them in the format of the underlying data source, submits the query to the data source and translates the results back to the local ontology schema. Thus a wrapper hides technical details of the data source from the mediator.

When the mediator receives a query from the user in terms of the master ontology, it decomposes the query into subqueries in terms of the local ontologies by taking into account the source descriptions, and sends them to the corresponding wrappers. Then, upon receiving the answers from the wrappers, it translates them in terms of the master ontology, combines the results and sends the final answer to the user. Thus the mediator has to perform the

following subtasks: query translation, query optimization, query decomposition, subquery scheduling, answer translation and answer composition.

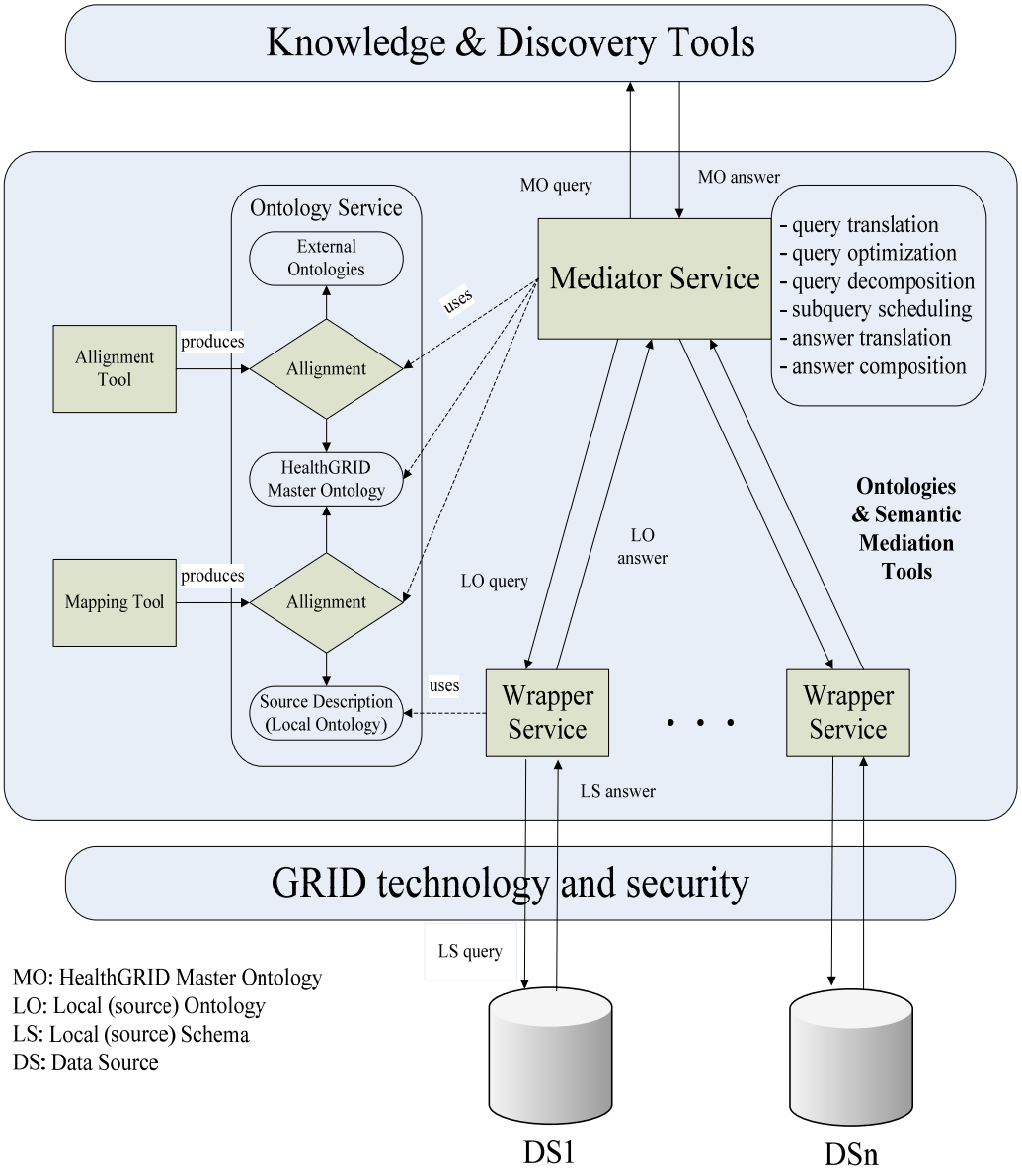



**Figure 56:** The use of ontologies in heterogeneous data integration

In summary, there are two conceptual translation steps: (1) from the master ontology to local ontologies and vice-versa, (2) from the local ontologies to source schemas, and vice-versa. These two steps are performed by the mediator and wrapper components, respectively. Of course, these translation steps require the establishment of the relevant mappings. In particular, for translation step 1, concepts and relationships in the master ontology should be related to these in the local ontologies through a mapping tool. In addition, in the case that a local ontology is linked with an external ontology then the master ontology should also be mapped and aligned with that external ontology, through an alignment tool. For translation step 2, the local ontologies should be mapped to the schema of the corresponding data source. In the case that the source offers limited query capabilities, methods should be written implementing the functionality described in the source description.

## 7.2 Conclusions

Biomedical sciences are facing an exponential growth of the volume of data they need to process and analyze. The movement from manual management of patient information to digital records have generated terabytes of data that need to be stored and made available to different healthcare participants. Researchers, physicians and healthcare centres need access to different sources of medical data usually geographically distributed. Biologists are sequencing more and more genomes and proteins and want to analyze them with sophisticated algorithms that require great computational power. Also compute intensive data mining techniques have become important tools for many biomedical applications in fields such as epidemiology and drug design.

Some of the most important computing needs of biomedical informatics in general are: The management of large volumes of medical data produced in many different centres. Define common standards for their interoperability. Provide to a large community of users secure and efficient access to their content. An infrastructure that would provide the above features it would also allow the association of post-genomic information and medical data and open up the possibility for individualized healthcare.

We have tried to demonstrate that Grid technology is a good response to these needs. More specifically we tried to show that the software components of Globus Toolkit can provide the necessary middleware services to build biomedical applications and address the challenges mentioned above.

The target domain of our work was data mining techniques able to extract useful knowledge from clinical data. We have set up a small grid experimental environment using Globus Toolkit 4 which is now based completely on the Web Services Resource Framework.

Access to clinical information systems was made possible through the OGSA – Data Access & Integration interface; a software component that provides access to relational or XML databases via web services. On top of this infrastructure we implemented a knowledge extraction system through the user is able to mine association rules from clinical data stored in XML document format.

Finally we developed a parallel algorithm for association rules mining using the message passing interface and conducted experiments to measure the execution time of the algorithm in relation to the number of computing nodes. From the experiments we concluded that the algorithm scales well as the number of computational resources increases.

# 8 References


1. I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3), 2001. Available at www.globus.org/alliance/publications/papers.php.
2. I. Foster, What is the Grid? A Three Point Checklist. GRIDToday, July 20, 2002. Available at www.globus.org/alliance/publications/papers.php.
3. I. Foster. The Grid: A New Infrastructure for 21st Century Science. *Physics Today*, 55(2):42-47, 2002
4. I. Foster, C. Kesselman. Computational Grids. *Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure"*, Morgan-Kaufman, 1999.
5. The Globus Alliance. www.globus.org
6. Foster I,Kesselman C. Globus: A Metacomputing Infrastructure Toolkit. International J Supercomputer Applications 1997;11(2): 115-28.
7. M. Humphrey, G. Wasson, K. Jackson, J. Boverhof, M. Rodriguez, Joe Bester, J. Gawor, S. Lang, I. Foster, S. Meder, S. Pickles, and M. McKeown, State and Events for Web Services: A Comparison of Five WS-Resource Framework and WS-Notification Implementations. 4th IEEE International Symposium on High Performance Distributed Computing (HPDC-14), Research Triangle Park, NC, 24-27 July 2005.
8. K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe. The WS-Resource Framework. March 5, 2004. Available at www.globus.org/alliance/publications/papers.php
9. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maguire, T. Sandholm, P. Vanderbilt, D. Snelling; Open Grid Services Infrastructure (OGSI) Version 1.0 Global Grid Forum Draft Recommendation, 6/27/2003. Available at www.globus.org
10. I. Foster, C. Kesselman, J. Nick, S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002. Available at www.globus.org/alliance/publications/papers.php
11. M. Antonioletti, M.P. Atkinson, A. Borley, N.P. Chue Hong, B.Collins, J. Davies, N. Hardman, A.C. Hume, M. Jackson, A. Krause, S. Laws, N. W. Paton, K. Qi, T. Sugden, D. Vyvyan, P. Watson, and M. Westhead. OGSA-DAI Usage Scenarios and Behaviour: Determining good practice**.** *Proceedings of the UK e-Science All Hands Meeting 2004*, September 2004. Available at www.globus.org/alliance/publications/papers.php
12. T. Sugden, A.C. Hume, M. Jackson, M. Antonioletti, N.P. Chue Hong, A. Krause, and M. Westhead. Protecting Application Developers - A Client Toolkit for OGSA-DAI. *Proceedings of the UK e-Science All Hands Meeting 2004*, September 2004. Available at www.globus.org/alliance/publications/papers.php
13. J. Nabrzyski, J.M. Schopf, J. Weglarz (Eds). Kluwer. Grid Resource Management. Publishing, Fall 2003. Available at www.globus.org/alliance/publications/papers.php
14. X. Zhang and J. Schopf. Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2**.** *Proceedings of the International Workshop on Middleware Performance (MP 2004), part of the 23rd International Performance Computing and Communications Workshop (IPCCC)*, April 2004.
15. V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. Meder, L. Pearlman, S. Tuecke. Security for Grid Services. *Twelfth International Symposium on High Performance Distributed Computing (HPDC-12)*, IEEE Press, to appear June 2003. Available at www.globus.org/alliance/publications/papers.php

16. I. Foster, C. Kesselman, G. Tsudik, S. Tuecke. A Security Architecture for Computational Grids**.** *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pp. 83-92, 1998.
17. D. Fenstermacher, C. Street, T. McSherry, V. Nayak, C. Overby, M. Feldman. The Cancer Biomedical Informatics Grid (caBIG™). Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference Shanghai, China, September 1-4, 2005.
18. C. Goble, Chris Wroe, Robert Stevens and the myGrid consortium: *The myGrid project: services, architecture and demonstrator.* Available at www.mygrid.org.uk
19. I. Foster, C. Kesselman. The Grid, blueprint for a new computing infrastructure. Morgan Kaufman, San Francisco, 1999.
20. D.G. Katehakis, C.E. Chronaki, et al., "Towards a Virtual Electronic Healthcare Record: The Patient Clinical Data Directory", Version 3.09, 1999. Available at http://www.ics.forth.gr/~katehaki/material/1998-pcdd/PCDDv3.14.htm
21. R. Agrawal, T. Imielinski, and A. Swami. Database mining: a performance perspective. IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 6, December 1993:914-925, 1993
22. L. Singh, P. Scheuermann, Bin Chen Generating Association Rules from Semi-Structured Documents Using an Extended Concept Hierarchy. Proceedings of the International Conference on Information and Knowledge Management, November 1997.
23. D. Braga, A. Campi, M. Klemettinen, and P. L. Lanzi. Mining association rules from xml data. In *Proceedings of the 4*th *International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2002). September 4-6, Aixen-Provence, France*, 2002
24. I. Cengiz. Mining Association Rules. Department of Computer Engineering & Information Sciences Bilkent ,Ankara ,Turkey.
25. R. Agrawal, T. Imielinski, and A.N. Swami, Mining Association Rules Between Sets of Items in Large Databases, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207-216, Washington, D.C., May 1993.
26. R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules in Large Databases, *Proceedings of the Twentieth International Conference on Very Large Databases*, pp. 487-499, Santiago, Chile, 1994.
27. M. Houtsma and A. Swami, Set-Oriented Mining for Association Rules in Relational Databases, *Proceedings of the 11th IEEE International Conference on Data Engineering*, pp. 25-34, Taipei, Taiwan, March 1995.
28. D. Wai-Lok Cheung, V. T. Ng, A. Wai-Chee Fu, and Y. Fu, Efficient Mining of Association Rules in Distributed Databases, *IEEE Transactions on Knowledge and Data Engineering,* Vol. 8, No. 6, pp. 911-922, December 1996
29. L. Kurgan , K. J. Cios and M. Trombley. The WWW Based Data Mining Toolbox Architecture. Available at http://www.ee.ualberta.ca/~lkurgan/papers/ICNNSC2002.pdf [Acceseed on Jan 2006]
30. W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, Knowledge Discovery in Databases, pages 1-30, AAAI/MIT, 1991
31. M. Schmidt, G. Zahlmann "What are the benefits of Grid Technology for a health care solutions provider?", Siemens Medical Solutions, Proc. HealthGrid conf. January 2003.
32. "Next Generation Grid(s)", European Grid Research 2005 – 2010 Expert Group Report, 16th June 2003, http://www-unix.Gridforum.org/mail_archive/ogsa-wg/pdf00024.pdf.
33. P. Brezany, I. Janciak, A. Wohrer, and A M. Tjoa. GridMiner: A Framework for Knowledge Discovery on the Grid - from a Vision to Design and Implementation

34. D. Lupinski, P. Plaszczak. Integrating various Grid resource managers with GT4 – the early experience, 2005. Available at http://www.cyf-kr.edu.pl/cgw04/presentations/c4-lupinski.pdf
35. I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13, 2005.
36. The OGSA –DAI project. www.ogsadai.org.uk
37. V. Welch. Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective, 2004. Available at www.globus.org/alliance/publications/papers.php
38. M. U. Feyyad. (1996), "Data mining and Knowledge Discovery: Making Sense out of Data", IEEE EXPERT, Microsoft Research. Available at http://www.computer.org/portal/site/ieeecs/index.jsp
39. TORQUE resource manager. www.clusterresources.com
40. J. Corrigan, et al (Eds): To Err is Human: Building a safer health system, Report of the Committee on Quality of Healthcare in America, Washington, Institute of Medicine, 1999
41. Potamias G., Koumakis L., and Moustakis V. (2004). Mining XML Clinical Data: The HealthObs System. Ingenierie des systems d'information, special session: Recherche, extraction et exploration d'information 10:1, 2005.
42. Potamias, G., Koumakis, L. (2004). HealthObs: An Integrated System for Mining Clinical XML-formatted Data. In 2nd International Conference on Information Communication Technologies in Health, Journal for Quality of Life Research (JQLR).
43. Potamias G., Koumakis, L., Charissis, G., Moustakis, V., Tsiknakis, M., and Orphanoudakis, S. (2003). Supporting Population Centered Medical Decision Making: Design Recommendations and Preliminary. In Procs 9th International Conference on Human-Computer Interaction – HCII2003, Heraklion, Crete, pp. 795 – 799.