ROLE OF WEB SERVICES FOR GLOBALLY DISTRIBUTED INFORMATION RETRIEVAL SYSTEMS IN A GRID ENVIRONMENT (IMPLEMENTATION AND PERFORMANCE ANALYSIS OF A PROTOTYPE)

by

Sairam Chilappagari

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Computer Science

Committee:

[illegible] Dr. J. Mark Pullen, Thesis Director

Sanjeev Setia Dr. Sanjeev Setia, Committee Member

[illegible] Dr. Hakan Aydin, Committee Member

[illegible] Dr. Hassan Gomaa, Department Chair

[illegible] Dr. Lloyd J. Griffiths, Dean,
The Volgenau School of Information Technology and Engineering

Date: 07/31/08

Summer Semester 2008
George Mason University
Fairfax, VA

Role of Web Services for Globally Distributed Information Retrieval Systems in a Grid Environment (Implementation and Performance Analysis of a Prototype)

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Sairam Chilappagari
Bachelors of Science
Jawaharlal Nehru Technological University, 2005

Director: Dr. J. Mark Pullen
Department of Computer Science

Summer Semester 2008
George Mason University
Fairfax, VA

DEDICATION

This thesis is dedicated to my parents, Shyam Sunder and Rohini, who have been with me every step of the way, from the day one of my life journey. Thank you for all the love, guidance, and support that you have always given me, helping me to succeed in all my endeavors. Thank you for everything.

I would also like to dedicate this work to Dr. J. Mark Pullen, Professor of Computer Science, George Mason University and Director, C4I Center for introducing me to the world of research.

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to my supervising professor, Dr. J. Mark Pullen, for his valuable guidance throughout my research. He has instilled in me a passion towards scientific research that will prove to be vital as I continue to pursue my academic interests. I am also grateful to the Open Grid Forum (OGF) for encouraging me to do the research in Grids.

I would like to thank Gregory Newby, Arctic Region Supercomputing Center, University of Alaska Fairbanks and Nassib Nassar, Etymon Systems, Inc. for their numerous original ideas form part of this work. I am immensely thankful to Ian Forster, University of Chicago. He has been a role model for me in the area of Distributed Parallel Computing and Grid Computing. I specially thank to NETLAB team, specifically Nick Clark, for his great help during the entire work.

My heartfelt thanks go to my parents and to my friends. This work could not have been accomplished without their support.

TABLE OF CONTENTS

Page

LIST OF FIGURES .... **vii**

LIST OF ACRONYMS .... **viii**

ABSTRACT .... **i**

1. INTRODUCTION **1**
- 1.1. Grid Computing and Grid Services 2
- 1.2. SOA and Web Services 3
- 1.3. Need for an Information Retrieval System in Grid Environment 5
- 1.4. Role of Web Services in Information Retrieval System 6

2. GOALS OF THIS THESIS **8**
- 2.1. Brief about the application 8
- 2.2. Goals 9

3. LITERATURE SURVEY **11**
- 3.1. Service Oriented Architecture (SOA) 11
  - 3.1.1. Architectural Components 12
  - 3.1.2. Issues and Challenges 13
- 3.2. Web Services 14
  - 3.2.1. Web Service Architecture 14
  - 3.2.2. Purpose of the Web Service Architecture 20
  - 3.2.3. Comparison between Web Services and Standalone Applications 20
- 3.3. Grid Computing 22
  - 3.3.1. The Grid Architecture 23
  - 3.3.2. Types of grids 25
- 3.4. Grid Services 26
  - 3.4.1. Web Services Resource Framework (WSRF) 28
  - 3.4.2. Relationship between Web services and Grid Services 28
- 3.5. Information retrieval (IR) 30
  - 3.5.1. Different Types of Information retrieval 31
- 3.6. Amberfish (Text-Based Information Retrieval System) 36
  - 3.6.1. Brief about the Application 36

4. ARCHITECTURAL BLOCK DIAGRAM 39

5. A GLOBALLY DISTRIBUTED INFORMATION RETRIEVAL SYSTEM IN A GRID ENVIRONMENT 41

5.1. Introduction 41
5.2. System Requirements 42
5.2.1. General Implementation Requirements 42
5.2.2. Security Requirements 46
5.3. System Overview 50
5.3.1. Design Considerations of the System 51
5.3.2. Architectural Components and Interfaces 52
5.4. Benefits of the IR Systems 63
5.5. Usage Scenario 66

6. DETAILED DESIGN OF THE SYSTEM 68

6.1. Use Case Diagrams 70
6.2. Class diagrams 75
6.3. Sequence Diagrams 78

7. PERFORMANCE EVALUATION AND ANALYSIS .... Error! Bookmark not defined.

7.1. IR systems for a Grid environment .... **Error! Bookmark not defined.**
7.2. Workload Model .... **Error! Bookmark not defined.**
7.2.1. Effect of Grid middleware on the performance of the System .... **Error! Bookmark not defined.**
7.2.2. Effect of Job Scheduler (Sun N1 GE) On Overall Performance of the System 86
7.2.3. Effect of number of Nodes in the Grid on the Performance of the system 91
7.2.4. Amberfish as stand alone application vs. Amberfish as Web Service 93
7.2.5. Query Evaluation Measurements 95
7.2.6. Server Connection and Performance 96

8. CONCLUSIONS AND FUTURE WORK 97

8.1. Conclusions 97
8.2. Future Work 98

REFERENCES 100

LIST OF FIGURES

Figure Page

1.1 Message Interaction between the end user and service provider……………………….03

1.2 Message Interaction between the end user and service provider……………………….04

3.1 Web Service Architecture……………………………………………………………15

3.2 SOAP Message format…………………………………………………………….18

3.3 Layered Grid Architecture……………………………………...…………………22

4.1 Architectural block diagram of IR System in a Grid Environment………………….37

5.1 Collection Manager Service components and interaction with other services………52

5.2 Index/Search Service components and interaction with other services……………...55

5.3 Components of Index / Search Service……………………………………………56

5.4 Query Processor Service components and interaction with other services…………..59

5.5 Format of the input query sent to Search Interface…………………………………..60

6.1 Interaction between the user/client and the Searching procedure………………….....68

6.2 Interaction between the administrator and the Indexing procedure………………….69

6.3 Implementation of the Index / Search interfaces…………………………………….70

6.4 Interaction between the administrator and the Indexing procedure………………….71

6.5 Implementation of the Collection Manager interface………………………………..72

6.6 Class interactions between the interface……………………………………………..73

6.7 Service Class, which creates the instance from which connection to the service will be created……………………………………………………………………………..74

6.8 Util (Abstract class) and the Config class, which are used by all interfaces………...70

6.9 Flow of interactions between the client/user and the IR system components……….71

7.1 Relationship between terms per query and Evaluation time...………………….…….80

7.1 Comparison between the IR systems running as Web Service Vs Grid Service…….84

7.2 Effect of Job Scheduler on the performance of an IR system………………………..89

7.3 Effect of the number of nodes connected to the Grid on the performance of an IR system…………………………………………………………………………….92

7.4 Effect of the number of nodes connected to the Grid on the performance of the IR system……………………………………………………………….…….....94

7.5 Relationship between terms per query and Evaluation time………………………....95

LIST OF ACRONYMS

[1] GIR – Grid Information Retrieval
[2] API – Application Programming Interface
[3] IR – Information Retrieval
[4] DR – Data Retrieval
[5] XML – eXtensible Markup Language
[6] SOA – Service Oriented Architecture
[7] SOAP – Simple Object Access Protocol
[8] WSDL – Web Services Description Language
[9] WSRF – Web Services Frame Work
[10] WSsec – Web Services Security
[11] URI – Uniform Resource Identifier
[12] OGSA – Open Grid Service Architecture
[13] OGSI – Open Grid Service Infrastructure
[14] HTTP – Hyper Text Transport Protocol
[15] FTP – File Transport Protocol
[16] RMI – Remote Method Invocation
[17] RSS - Really Simple Syndication
[18] GA - Grid Architecture
[19] GIR-WG - Grid Information Retrieval-Working Group
[20] QP – Query Processor
[21] CM – Collection Manager
[22] IS – Index/ Search Service
[23] OGF – Open Grid Forum
[24] GT4 – Globus Toolkit 4.
[25] W3C - World Wide Web Consortium

ABSTRACT

ROLE OF WEB SERVICES FOR GLOBALLY DISTRIBUTED INFORMATION RETRIEVAL SYSTEMS IN A GRID ENVIRONMENT
(IMPLEMENTATION AND PERFORMANCE ANALYSIS OF A PROTOTYPE)

Sairam Chilappagari, M.S.

George Mason University, 2008

Thesis Director: Dr. J Mark Pullen.

Information retrieval is an important aspect of any data storage system. As the data storage capability increases there will always be need for an information retrieval (IR) system, which caters to the growing needs of information search and retrieval. IR System in a Grid environment provides a common hierarchical solution to the information retrieval problem for globally distributed networked systems. It sits on top of Grid middleware and provides a general purpose solution by implementing Web Services in the process of document search.

In this thesis, I present a detailed study of the issues related to the information retrieval in Grid Environment; from the basic need for a grid information retrieval system to the working model that enables locating information resources. This document covers the capabilities and limitations of the system and the role of Web Services in the course of developing an information retrieval system for the Grid environment.

1. INTRODUCTION

Things were simpler in the past. One could focus on the science rather than needing to have the skills of an information technology professional with expertise in arcane computer data analysis tools. In the current world data has spread in heterogeneous systems across the world. In order to access the data, we need an information retrieval system which enables us to retrieve the data in an efficient way over a largely distributed networked environment otherwise known as Grid. We continue to develop new methodologies and technologies to solve the complex problems in the field of science and engineering. Always the requirements of these solutions, however, tend to supersede the capacity of the contemporary technology. As a matter of fact the complexity and the nature of industrial problems are much more process intensive and memory consuming to address by a conventional single system approach.

For example Particle physics, DNA analysis and High-energy physics are some of the scientific domains which require higher computational power and storage capability in order to analyze and provide a statistical analysis of incoming streams of data. The analysis of incoming data can be facilitated by computing technologies such as high performance computing, supercomputing and distributed computation. These require high computational power and large scale data storage systems. In this scenario, the problem of retrieving the data in distributed environment is a challenging issue. As we don't yet know the problem resolution capabilities of distributed computing, we continue to explore its concepts entering into a new era of massively powerful grid-based solutions. To address this problem, and in-order to access the stored data, we need an efficient information retrieval system.

1.1. Grid Computing and Grid Services

Grid can be defined as *an infrastructure that will provide us with the ability to dynamically link together resources as an ensemble to support the execution of large-scale, resource-intensive, and distributed applications* [2]. The main purpose of the Grid Computing discipline is networking unlimited number of ubiquitous computing devices, connections and services within a Grid.

The concept of grid computing grew out of *attempts to build large-scale distributed applications that group remote supercomputers, databases, and online instruments into tools used by groups of collaborators* [32]. This new innovative thought is simply a massively large processing and storage utility. Grid Computing is capable of adding an unlimited number of computing devices into any grid environment. It allows using distributed computers as a parallel computing resource. Web Services is an emerging technology for distributed computing using Internet and other standard protocols. Grid Services is a merger of the two: it is a Grid technology for parallel computing using Web Services protocols as basic communication.

The worldwide business demand requiring complex problem solving capabilities has driven dynamic collaboration of many resources to work together to solve these problems. Grid computing solutions are formed using wide variety of technologies and open standards like OGSA, OGSI, SOAP, WSDL, etc. Grid provides highly scalable, highly secure and extremely high-performance mechanism to discover and access the remote resources in seamless manner. Unlike the cluster, a grid can connect to heterogeneous systems to operate with each other. In other words, *a computational grid* *is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities* [4].

Later work broadened this definition with more focus on coordinated resources sharing and problem solving in multi-institutional virtual organizations.

1.2. SOA and Web Services

A service-oriented architecture (SOA) can be defined as the process of integrating the loosely coupled interoperable services and applications [7]. In SOA, the system is decomposed into a collection of network-connected components. In this structure, the applications are composed dynamically from the deployed and available services in the network. The most commonly understood SOA architectures are the World Wide Web and Web Services. The Architecture of Web had evolved even before the term SOA was found. This internet model is sufficient for client-server interaction. The following diagram shows how the Web user receives agent-based interaction

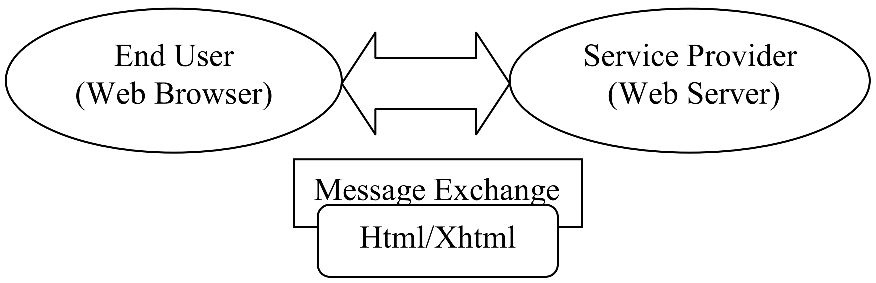


Figure 1.1 Message Interaction between the end user and service provider in Web based applications

However, But when the interaction process becomes complex, (e.g. business-to-business), the above mentioned architectural model should be modified so as to support message exchanges between the applications of choice.

There comes the concept of more generalized solution, Web Services. The Web services architecture extends the conventional interaction process by adding the interoperability and expressiveness of eXtensible Markup Language (XML). By introducing XML encoding, the interaction pattern has evolved to connect any consumer to any service provider [67]. Here the message interaction is done asynchronously for better interoperability between the consumer and service provider, in other words a very loosely coupled system.

In order to better understand the SOA, consider the architecture with more emphasis on four major components service description, registration of the service, discovery of the service and usage of the published service. Figure 1.2 shows different interactions among the components in SOA.

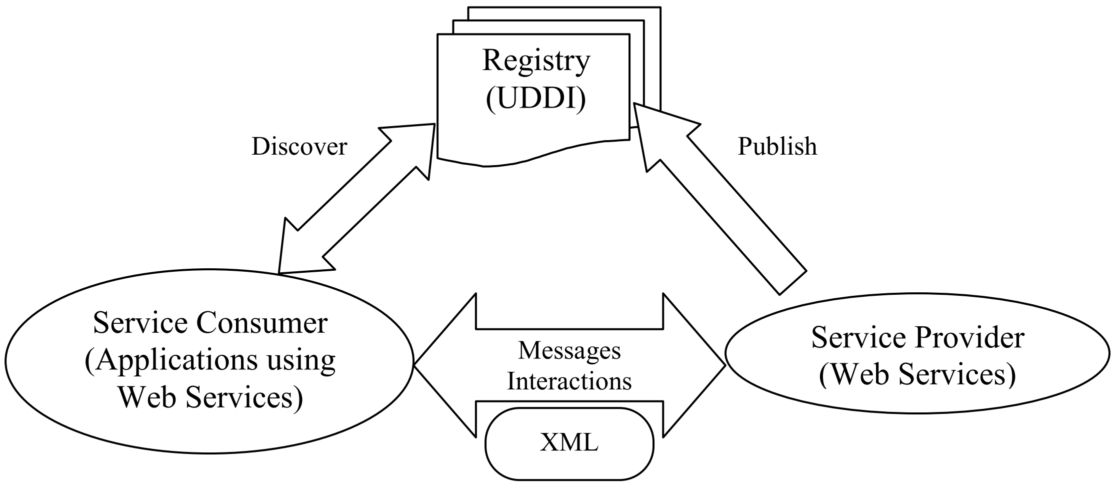


Figure 1.2 Message Interaction between the Consumer and Service Provider in SOA systems

This is a simple, yet very powerful concept for providing a communication channel between the consumer and service provider. A service provider develops a web services and publishes the service for the consumers with necessary message format and transport binding. The service provider can register this service and its description with one or more registries. Consumer can discover the service from the registry or directly establish a connection with service provider and start sending the messages in a format that is defined by service provider. The above mentioned structure is the basic implementation of SOA. It can be further extended by adding security and other management features. In this thesis, I present the role of these services and their impact on information retrieval process in a grid environment.

1.3. Need for an Information Retrieval System in Grid Environment

Information retrieval (IR) is the science and practice of identifying documents or sub-documents that meet information needs [17]. In other words, the basic need of an information retrieval system is to match an information need against a database with the purpose of retrieving the information that matches the need from the collection of files that from the database. It can be used by end users, applications and other services as subordinate part, to provide documents that match information needs. Building an information retrieval system is an effort to integrate information retrieval techniques with the power of grid computing.

Generally grid applications require the handling of huge amounts of data (typically in the terabyte or petabyte range) so they need different types of data handling tools such as data mining, information retrieval. Because of the amount of data many grid projects deal with, hierarchical and distributed processor architectures are needed for an information retrieval system.

Consider a scenario where a grid is connected with heterogeneous systems distributed across the world and not every internal document is available online. In this case if a client from one corner of the world wants to access those files from the other corner of the grid, there must be a mechanism which enables the system to access the data that is not available publicly. This case explains the situation where we need an information retrieval system (a service) which can integrate present sources of information such as web services, database servers, portals, etc into a single secure, scalable system.

These services are independent from other services, and being Web Services, they are distributed and can be created dynamically and in any virtual organizations. This provides a facility to integrate with other forms of information retrieval implementations such as ontology based retrieval and it can also be sub-part of other services.

1.4. Role of Web Services in Information Retrieval System

Information Retrieval (IR) can be defined as *the process of searching and retrieving the documents from a given database based on the input query* [33]. The IR Framework for Grid environment was originally devised as a set of interfaces for the Java programming language implement the various components of the system such as query processor, indexer and collection manager in a way that would allow them to be easily

linked together into a single system. Initially, it was intended that this hybrid system would run on a single machine. However the advent of Web Services offers a different approach for implementing the framework which is more suited to the distributed nature of the various components involved in the grid information retrieval (GIR) system. Many of the components in the GIR framework can be mapped to Web Services – for example, Query Processor Web Service may take a query and convert it to a suitable form.
The Application Programming Interface (API) for the Web Service may then take a suitable query and return a set of relevant documents using a particular algorithm. This hierarchical structure of information retrieval system not only enables to develop individual components of the system on their own platforms but also provides a way to integrate together over the Internet to build a complete system

Another reason for using Web Services as part of retrieval process is the ease with which they can be implemented independently and integrated together minimal difficulty. Each component of the GIR is a Web Services and they are independent of each other and, being Web Services, they are distributed. Since they can be initiated and called from anywhere, they can be integrated in any combination of virtual organizations. This also provides a facility to extend the GIR, by providing abstraction and modularity to integrate the enhancements. Since they are autonomous the enhancement algorithms for any component can be easily integrated to the existing system. For example if we want to add an extension of semantic-based search technique, we need only to wrap the query processor with an extra wrapper which provides this capability.

2. GOALS OF THIS THESIS

This work concentrates on the role of web service in grid computing for information retrieval processes, which is used to integrate all the components of Information retrieval system query processor, indexer and collection manager. An introduction to which is provided in this chapter. More detailed explanation about each component, implementation details and about the terminology used in this work is provided in the later chapters.

2.1. Brief about the application

This Information retrieval system applies the concepts of grid computing to IR in order to provide a general infrastructure for distributed networks. It also brings the capabilities of IR to grid computing. The idea of this system is to define an IR system in terms of three functional components, implemented as web services: the Collection Manager Service (CM), the Indexing/Searching service (IS), and the Query Processing service (QP). This architecture was proposed by Grid Information Retrieval – Working Group (GIR-WG) and presented at GGF-5 conference. These services are autonomous, and being Web Services, they are distributed. Since they can be created dynamically and in any combination of virtual organizations, they can be used to create new IR systems or link existing ones together in an interoperable network of IR services. These three components concentrate on different steps of information retrieval process. The Query Processor takes the input text to search for from a user and accesses the database, indexed by Indexer. The Collection Manager informs the Indexer when to re-index the database after new documents are added to grid.

2.2. Goals

The primary goal of this document is to show how web/grid services can be useful in implementing an IR system in grid environment. In order to achieve this, we apply the concepts of grid computing to information retrieval process and to provide a general infrastructure for distributed networks. Our purpose is to utilize grid computing advantages as applied to the science of information retrieval.

Information Retrieval in a Grid Environment can be defined as a set of web/grid services which, together, constitute a complete IR system [17], composed principally of the three components Query Processor, Data Indexer and Collection Manager. The main aim of developing the system is to develop these three components as Web Services so that the structure of the system provides a whole new level of abstraction. Different groups can work on different components of the Grid Information Retrieval components and integrate them in a straightforward way.

The basic process described by this thesis is:

- gather network-based documents
- build data in an indexed form for retrieval, using the Indexer
- determine post-indexing term and document weights

- use Query processors, to take input queries from users and merge the result sets
- rank the results from different sources and merging them
- organize or present the result set to users
- update the indexer frequently, using the Collection manager to keep track of the information added or updated in grids
- include the Grid security infrastructure in all the steps of information retrieval process.
- make efficient use of computational resources

In many cases, Grid IR will allow communication between two or more components (e.g., multiple Collection Managers communicating with an Indexer). Anticipated individual services include crawlers, indexers and search and presentation engines. Solutions for most of the IR techniques are available, although some do not scale well or are less amenable to the distributed processing of the Grid. The IR system presented here is capable of handling extremely large collections (billions of documents). Grid IR will benefit from past experiences in networked IR. For example, the Z39.50 offers the ability to send a query to multiple IR engines. GridIR will take Z39.50 further by layering IR on the Grid security and authentication infrastructure, and by providing sophisticated techniques for merging and ranking the results from the engines.

3. LITERATURE SURVEY

The development of the grid information retrieval system pertinent to this work utilizes various existing grid computing technologies and underlying frameworks. This chapter contains a brief description of these technologies, along with an introduction to the text-based information retrieval system (Amberfish) that has played an important role in laying a foundation for developing a Web Service model to implant an information retrieval system in grid environment.

3.1. Service Oriented Architecture (SOA)

The goal of the SOA is to collaborate different services which are published and available to use. Implementing SOA is essential to provide agility to the existing system and this flexibility promised by Web Services. Microsoft defines SOA as, *it's not a product. It is a design philosophy that informs how the solution should be built*. Often people use the words SOA and Web Services interchangeably. While it is true that Web Service is an implementation methodology using specific standards and language protocols to execute on a SOA solution, the two are distinct. These benefits are delivered not by just viewing service architecture from a technology perspective and the adoption of Web Service protocols, but require the creation of a Service Oriented Environment that is based on the following key principles [7] :

- Service is the first and most important part of SOA. Web Services can be used to make the services available using the set of protocols by which Services can be published, discovered.
- SOA is not just an architecture of the services seen from a technology perspective; it's a philosophy to ensure policies, practices, and frameworks.

Service orientation is a means for integrating resources across heterogeneous systems. Each resource whether an application, service or data storage system, can be accessed as a service. These capabilities are available through interfaces; complexity occurs when one service differs from another in their implementation of services (e.g.: operating system or communication protocols they use). Service orientation uses standard protocols and conventional interfaces usually Web services to facilitate to provide interoperability among diverse services. Specifically, SOA allows the underlying service capabilities and interfaces to be composed into processes. Each process is itself a service, one that now offers up a new, aggregated capability. Because each new process is exposed through a standardized interface, the underlying implementation of the individual service providers is free to change without impacting how the service is consumed.

3.1.1. Architectural Components

SOA architectural components can be broadly categorized into three categories

- *Service providers*. A Service provider publishes and maintains a stateless service which accepts the input from the consumer and provides the outputs.

- *Service consumers*. A service consumer, interested in using one or more of the services provided by service providers.
- *Service repository.* A service repository contains the collection of descriptions of the services. Service providers register their services in this repository and service consumers access the repository to discover the services being provided.

3.1.2. Issues and Challenges

The demand for a robust architecture such as SOA, which responds to integrate internal or external service systems, is not without its challenges. Some of the challenges are described below.

Service identification, identifying services and determining corresponding service providers is a critical and the first step in architecting a service-oriented solution.

*Service location***,** finding an ideal location for the service to execute.

*Service domain definition***,** Classifying services into logical domains simplifies the architecture by reducing the number of components to be addressed.

Service packaging defines how is existing functionality within legacy mainframe systems to be re-engineered or wrapped into reusable services.

Service orchestration, given service exists because there's at least one instance of a service consumer initiating the request for that service.

*Service routing***,** enables these architectures to provide location transparency while ensuring acceptable system performance levels.

Service governance, is how the service provider exercise governance processes to administer and maintain services.

Service messaging standards adoption, Messaging standards specific to vertical industries enforce standardization on a set of data elements and message formats.

In addition to these integrating systems in SOAs requires identity management, authorization in a distributed environment, Service Aggregation, and Entity Aggregation.

3.2. Web Services

A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [48]. Web Services offer a mean to communicate among loosely coupled systems by providing interoperable protocols for Security, Reliable Messaging and Transactions. These specifications are built over the XML and SOAP standards.

3.2.1. Web Service Architecture

In case of standard Web access, the site or the service is identified by its domain name (a symbolic representation for its IP address). On the other hand a Web Service system is identified by a URI, whose public interfaces are defined and described by XML. Other software systems or services can communicate with these systems by using

XML messages defined by the services. This agrees with the above definition by W3C [48]. In this section we describe some of the technologies that are critical in creating Web Services and the role they fill in relation to this architecture. Web Service architecture involves many layered and interrelated technologies. There are many ways to visualize these technologies, just as there are multiple ways to build and use Web services. The Web Service architecture is built around the XML technologies and they are independent of the underlying transport mechanism. The following diagram shows the layered approach to Web Service Architecture.

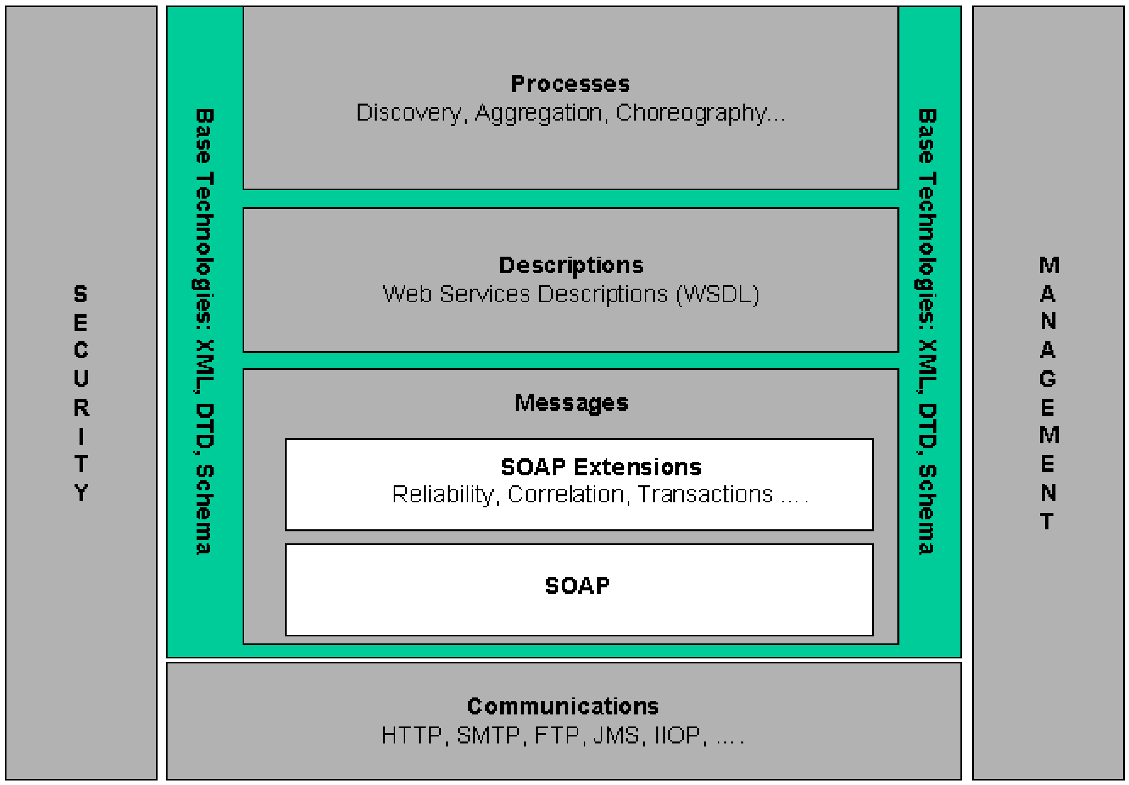

SECURITY
Base Technologies: XML, DTD, Schema
Processes
Discovery, Aggregation, Choreography...
Descriptions
Web Services Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions ....
SOAP
Base Technologies: XML, DTD, Schema
MANAGEMENT
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...

Figure 3.1 Web Service Architecture [48]

Communication between the services and the consumer is done from the base layer of this architecture. The messages are packaged and exchanged using envelopes including SOAP and extension models. Different SOAP header messages are included in these extensive models for message correlation, transactional capabilities, message readability and service addressing. There is a high level description for every message exchange that happens between service and the consumer. This description is given by any description language of choice, and the most commonly used descriptive language is *Web Service Descriptive Language* (WSDL).

One can build different kinds of technologies using this architectural model. These technologies can be infrastructural, middleware solutions or high-end applications. Other noticeable features of this architecture are the vertical pillars of Security and Management which are spread across all the layers of the architecture. Security and Management are two features, required at all levels of horizontal architecture components. Some of the technologies used in this architecture are explained below.

XML, Related Technologies and Relevance to Web Services

Understanding the basics of XML is very important to develop interoperable solutions. We know from the previous section that Web Services communicate with each other by message exchange. These messages are defined in terms of XML. The importance of XML lies in its concepts of standard-based, flexible and extensible data format. Web Service messages are defined by using *XML infoset*, *XML schema, and XML* *name space standards*. XML significantly reduces the burden of deploying the many technologies needed to ensure the success of Web Services.

Infoset specification provides a set of definitions for use in other specifications that need to refer to the information in an XML document. In other words it defines a set of information items and associated properties that comprise an abstract description. The XML Infoset specification provides for a consistent and rigorous set of definitions for use in other specifications that need to refer to the information in a well-formed XML document. These definitions help the standards and tools to validate XML documents. Another important concern that may arise is the serialization format of the XML documents for transport and binding. There are multiple serialization formats available for XML today including simple XML text streams and binary XML streams. Serialization of the XML Infoset definitions of information may be expressed using XML 1.0. However, this is not an inherent requirement of the architecture. The flexibility in choice of serialization format(s) allows for broader interoperability between agents in the system. In the future, a binary encoding of the XML infoset may be a suitable replacement for the textual serialization. Such a binary encoding may be more efficient and more suitable for machine-to-machine interactions.

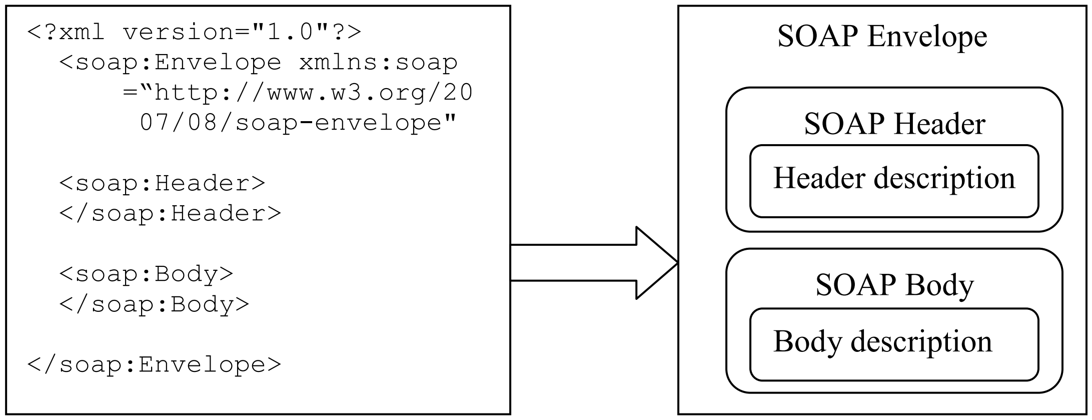
SOAP

The Simple Object Access Protocol is a light weight, XML-based mechanism for creating structured data packages that can be exchanged between any network applications. SOAP messages can be carried by a variety of network protocols; such as

HTTP, SMTP, FTP, RMI/IIOP, or a proprietary messaging protocol. The SOAP specification defines the following fundamental components:

- An envelope that defines a framework for describing the message structure
- A set of encoding rules for expressing instances of application-defines data types
- A convention for representing remote procedural calls and responses
- A set of rules for using SOAP with HTTP
- Message exchange pattern such as request-response, one-way, and peep-to-peer conversations

SOAP is currently being used as the standard for XML messaging including enveloping and exchanging messages. It is also very flexible and extensible offering the capability to add-on standards and application- defined extensions. The wide acceptance of SOAP is based on the fact that it builds upon XML infoset. The format of SOAP message is formally defined in SOAP 1.2; specifically the messaging framework specification. Figure 3.2 illustrates the simple SOAP structure.



```
<?xml version="1.0"?>
  <soap:Envelope xmlns:soap
      ="‟http://www.w3.org/20
       07/08/soap-envelope"

  <soap:Header>
  </soap:Header>

  <soap:Body>
  </soap:Body>

</soap:Envelope>
```

Figure 3.2 SOAP Message format.

WSDL

Web Service Description Language (WSDL) is a XML language for describing Web Services. It enables us to describe the service and in turns enables the client to consume these services in standard way without having much knowledge about lower level protocol exchange binging including SOAP and HTTP. This high level abstraction on the services limits the human interaction and enables the automatic generation of proxies for Web Services, which can be static or dynamic. The messages themselves are described abstractly and bound to a well defined network protocol and a message format. Irrespective of the language implementations on sender and receiver side, if both the sender and receiver agree on the service description, (e.g. WSDL file), the implementations behind the Web services can be anything. The abstract service definition components are Types, Messages, Operations, PortType and Binding. The WSDL binding elements are used to specify the agreed grammar for input, output and fault messages. These are defined by WSDL for SOAP and HTTP.

There are two kinds of message encoding: literal encoding and SOAP encoding. Literal encoding specifies that the message will be constructed according to the specification in XML schema, where in SOAP encoding it occurs according to the SOAP-encoding rules defined in the SOAP specification. Other important concept in WSDL architecture is the message transfer format. The messages can be transferred as document or RPC parameters. Document means that the message is part the body of the message. And RPC means that the parts are just parameters of the call. These message formats should confirm to the SOAP semantic.

3.2.2. Purpose of the Web Service Architecture

Web Services provide a standard a way to interoperate between different software applications and services, running on different platforms. Here we intended to provide role of Web Services, and define its place within a largely distributed network system. The WSA provides a conceptual model and a context for understanding Web services and the relationships between the components of this model.

Fundamentally, the Web Services architecture is about interoperability: it identifies those global elements of the global Web Services network that are required in order to ensure interoperability among Web Services.

3.2.3. Comparison between Web Services and Standalone Applications

Web Services are generally considered to be thin clients as compared to standalone applications. Before the development of Web Services, Standalone applications used to be deployed on to a single machine that does not depend on any other infrastructure for processing. There were significant problems with this implementation. Some of them are deployment, scalability, fault tolerance, and maintenance.

Deployment

One major challenge with the standalone application is that it has to be installed in every machine present in the distributed environment in order to put the machines to good use. If the dependencies of the product change, the deployment should be done in all machines adding an extra overhead.

Scalability

Scalability refers to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added [56]. As the requirement of processor capacity grows, there is no easy way to satisfy the needs other than buying new hardware in the case of standalone applications. Also, the service will be unavailable momentarily when it is migrated from one processor to the other. In case of a Web Service, this can be solved easily by adding more machines to the existing infrastructure.

Fault Tolerance

There is a risk of single point failure in standalone applications; if anything goes wrong with the processor, the service is down. However, in a Web Service, one of the child nodes can take over the responsibility of the head node in case of any failure.

Maintenance

Even a small requirement change can lead to lot of problems due to the effort required to re-install the application. With the introduction of Web Services, maintenance has become more convenient with the n-tier architecture deployment infrastructure.

3.3. Grid Computing

The evolution of Grid Computing has started as an important branch in the Computer Science field by proving itself different from distributed computing because of its focus on the resource sharing, coordination and high performance computing. Grid Computing is intended to solve the problems with resource sharing among the individual or groups. Examples include the Particle Physics Data Grid, which involves a widely distributed team of physicists working to analyze the data streaming out of large international particle physics projects. Other disciplines such as astrophysics, genomics and proteomics, and chemical engineering are building many other Grid applications according to their needs. To enable the collaboration of distributed teams to build these applications, grids require an architecture which uses the services to manage the resources provided to them. These grid resources include power, data storage, hardware requirements and other applications. In this scenario, a grid needs to have services which provide resource discovery, authentication, authorization, and access mechanisms. Some of these services are explained below.

• *Security services*: There should be a way to authenticate Grid users before accessing the information from the grid. And this requires separate services to define what the user is authorized to do. Also privacy is also one of the serious security concerns for grid.

• *Scheduling services and resource brokers*: When multiple users need to use a distributed set of resources or applications in a grid (such as advanced instruments and databases) in a linked computation, there should be a fair mechanism to allocate the services or resources among the users.

• *Information and data services*: Many large scientific applications require a way to store and retrieve huge amount of data distributed around the world. This data must be indexed to make information accessible to the user.

3.3.1. The Grid Architecture

The concept of virtual organization is the key to Grid Computing. A new architectural model was developed for the establishment, management and cross organizational resource sharing within virtual organization. This architecture is known as the G*rid Architecture* (GA*). GA identifies the basic components of the grid system and defines the purpose and functions of such components and indicates how each of these components interacts with each other* [1]. The main attention of this architecture is on interoperability among the resource providers and users which is required to establish the sharing relationships. This interoperability requires common protocols at each layer of the architectural model. The layered architectural model of GA provides mechanisms, interfaces, schema and protocols at each layer, by which users can negotiate, establish, manage and share the resources. Figure 3.3 illustrates the layers in GA with specific capabilities at each layer. It also compares the corresponding Internet Protocol (IP) layers, for the purpose of providing more clarity to the specifications.

The Fabric Layer defines all the physical and logical resources that can be shared between the users in a grid. These resources include computational resources, data storage, networks, catalogs and other system resources.

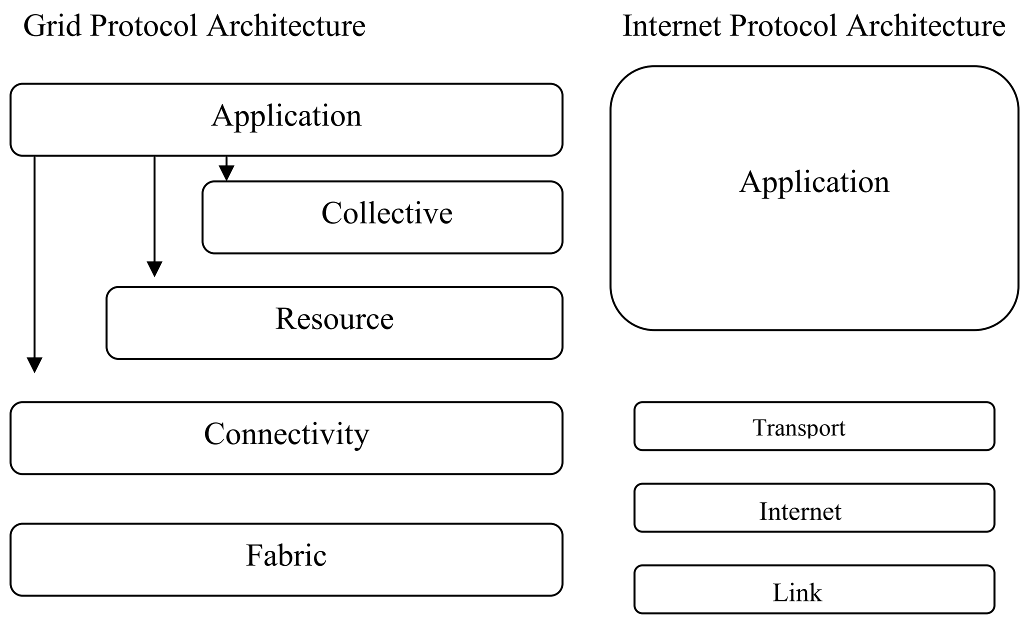


Figure 3.3 Layered Grid Architecture [1]

The Connectivity Layer defines the protocols used for communication and authentication in a grid specific networking environment. Here the communication protocols cover the issues related with transport, routing, naming. These protocols ensure the data exchange between the fabric layers of respective resources.

The Resource Layer utilizes the protocols (mainly communication and security protocols) required by the connectivity layer to control the secure negotiation, initiation, monitoring the sharing of operations across individual resources.

While the resource layer manages the individual resources, *The Collective Layer* is responsible for global resource management and the message exchanges among the resources. This layer implements a wide variety of services, such as discovery, co-allocation, scheduling and brokering services, monitoring and diagnostic services, data replication services, work load management and authorization services.

The Application layer provides an Application Programming Interface (API) for user applications by utilizing the services defined by each layer described above. Such applications can directly use all the resources available in the grid with proper authorization.

3.3.2. Types of grids

Grid computing can be used in a different ways to address various kinds of application requirements. Building a grid is purely application specific. There are three primary types of grids which are described below, though, there are no hard boundaries among these grid types and a grid may be a combination of two or more types.

However, the decisions made regarding the application development for the grid environment will depend on the type of grid you are using. The three different types of grids are described below.

Computational grid

A *computational grid* is focused on setting aside resources specifically for computing power. In this type of grid, most of the machines are high-performance servers. These types of grids generally used in the fields such as distributed supercomputing, high-throughput computing and on-demand computing.

Scavenging grid

A *scavenging grid* is most commonly used with large numbers of desktop machines connected in a distributed environment to enable the resource sharing among available machines. Owners of the desktop machines are usually given control over when their resources are available to participate in the grid.

Data grid

A *data grid* is responsible for storing and providing access to data across multiple organizations. Users need not to bother about data storage location as long as they have access to the data. As an example, you may have two organizations doing the same research, each with unique data. A data grid would allow them to share, manage their data, and manage security issues such as authorization to access the data.

3.4. Grid Services

Open Grid Services aim for the integration of services across distributed and heterogeneous virtual organizations with disparate resources and relationships. The Open Grid Services Architecture (OGSA) addresses these challenges by integrating grid computing and Web Services technologies. The basic purpose of OGSA is to provide a standard mechanism for creating, naming, and discovering the grid service instances. In addition to supporting the integration with native platforms, it also provides location transparency and multiple protocol bindings for the service instances. This enables the system to handle lifetime management, reliable invocation, authentication, authorization and delegation of the services.

The OGSA defines grid as an extensible set of services that may be aggregated in different ways to meet the requirements of virtual organizations. These services vary from information discovery through state registration and management for secure, reliable service creation to handling the allocation of resources. OGSA assigns a well defined set of standard interfaces for managing the service instances. These interfaces are discussed below [3].

Discovery, to discover available resources and services

Dynamic service creation, to create and manage new service instance

Lifetime management, to reclaim services and resources associated with failed operations

Notification, for asynchronous notification of changes in the state among collection of dynamic, distributed services

Authentication allows the identify of individuals and services to be established for policy enforcement.

Reliable invocation guarantees that a service has received a message. It gives the foundation for higher-level-per-operation semantics, such as transaction.

These standards and the protocol bindings increase the generality of the architecture without compromising the functionality. At the same time, OGSA has its downside. It is very bulky, has a very complex architecture, is highly object-oriented and lacks the support from WSDL 2.0 extensions [7]. These issues with uprising Web Services standards introduced a new framework WSRF.

3.4.1. Web Services Resource Framework (WSRF)

In order to support the new standards of Web Services such as WS-Addressing, OGSI proposed a framework which partitions its functionality into five distinct specifications [66].

WS-Resource Properties, associates stateful resources and their properties with Web Services. It includes operations to receive, change and delete resource properties.

WS-Resource Lifetime creates or destroys a WS-Resource immediately or in the future.

WS-Renewable Reference retrieves a new endpoint reference to a service.

WS-Service Group manipulates collections of Web Services.

WS-Base Fault describes types for error reporting.

WS-Notifications uses publish and subscribe technologies for notification.

These specifications capture all the functionality provided by OGSI, but do so in way that integrates with evolving Web Services standards. In addition, the WSRF expresses the OGSI definition in a more consistent way more familiar to Web Services developers in genera [66].

3.4.2. Relationship between Web services and Grid Services

In this chapter we have described about Web Services and Grid Services along with their respective architectures. The basic purpose of these architectures is the creation of interoperable services and their respective applications. As described above, Grid Computing is the process of sharing resources among the distributed system in virtual organizations. This involves the interoperable access to resources. The evolution of Grid

Computing into Grid Services chooses Web Services technology as its underlying tool for defining these interoperable resources. The basic reason for this is that the base of Web Services is a set of interoperable open protocols.

As we have discussed the both Web Services and Grid Services in previous sections, we can now draw relationship and differentiate each of them. A key consideration is the ability of an application maintaining its state information. For example, a purchase order system keeps a user's order information between the interaction for the purpose of verifying, changing or updating the order information. This state information may be local or stored in external sources such as databases. To understand this distinction it is very important to understand how state is managed in Web Service scenarios. We can classify state management of the services into two types:

Interaction aware state information, where the interactions are done between the client and service using a simple cookie, session ID, or complex correlation information. The advantage of this type of state information is that the server side is not managing any state information of client.

Application aware state information, where the services are aware of each client's state information. Each service creates a new client specific instance and a reference to this instance will be passed to a client interaction with the service without passing correlation information. These services typically referred as stateful services.

3.5. Information retrieval (IR)

Information retrieval (IR) is the science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hyper textually networked databases such as the World Wide Web [68]. Unfortunately the word information can be very misleading; many people use the word data retrieval (DR) interchangeably with information retrieval (IR). To make clear distinction between these two, the following discussion puts forward some of the distinguishing properties of DR and IR.

Matching: DR retrieves all the documents which are the exact match for the query. Where as IR retrieves on the basis of partial or the best match.

Inference: Inference in DR is done using deduction whereas IR uses induction.

Model: DR uses deterministic model and IR probabilistic.

Query Language: the query language for DR will generally be of the artificial kind, one with restricted syntax and vocabulary; IR implements natural language processing as query language.

Query Specification should be complete in DR where in IR it need not to be complete.

Items Wanted: results are retrieved on exact matching basis in DR; in IR the search is done on relevant basis.

Error Responses: DR is more sensitive to errors in the sense that, an error in matching will not retrieve the wanted item which implies a total failure of the system. In IR, small errors in matching generally do not affect performance of the system significantly.

3.5.1. Different Types of Information retrieval

In this section we will discuss the different types of information retrieval techniques, variations and extensions of these methods which are part of modern IR process. Here we examine traditional text retrieval systems including full text scanning and the new developments on approximate and inversion based searching methods. This document also covers the method of using signature files and clustering approaches [38].

Full Text Scanning

This is the most straight forward way of searching the documents that contain a certain search string called substring test. Here the string is considered to be the sequence of characters without unnecessary characters. The algorithm for sub string test is as follows:

- Compare all the characters of the query string to the corresponding characters of the document;
- If mismatch occurs, right shift the search string by one position and continue the search until the string is found or the end of the document is reached.

Although this is the simplest algorithm to implement, it is slow. If m is the length of the search string to be searched and n is length of the document, then it takes O(m*n) comparisons. Knuth Morris and Pratt developed an alternate algorithm that needs O(m+n) comparisons. Their idea is to right shift the search string by more than one character to whenever a mismatch occurs. The fastest known algorithm was proposed by Boyer and Moore Their idea is to perform character comparisons from right to left if a mismatch occurs the search string may be shifted up to m positions to the right The number of comparisons in the worst case is n + m. Searching algorithms that can tolerate typing errors have been developed by Wu and Manbe [38].

The advantage of the full text scanning method is that it requires no space overhead and minimal effort on insertions and updates. The disadvantage is the bad response time, which might be severe for large data bases. Therefore full text scanning is usually carried out by special purpose hardware or it is used in cooperation with other access methods that restrict the scope of searching. These kinds of retrieval systems are best suited for Grid like environments.

Signature Files

The signature file approach has attracted much interest due to its efficiency in comparisons compared to full text scanning. In this method each document yields a bit string using hashing on all its words and superimposed coding. These signatures are stored in a sequential order in a separate file, which is smaller than the original file and can be searched much faster. Files and Huskey [42] applied this method on a database of bibliographic entries. They implemented a stop list to avoid the common words and a procedure to reduce each non common word to its stem. They also implemented a numeric procedure as a hashing function in place of a lookup table. Harrison [43] used the signature file approach in order to make the substring testing more efficient. He suggests using consecutive letters (n-grams) as input to the hashing function. Roberts [44] used a one level signature le for a telephone directory application, where the signature file is stored in a 'bit-slice' manner; the bits are stored in a consecutive manner. Although this structure makes updates difficult it reduces the I/O cost for retrieval. He suggests creating the signatures in such a way that terms that appear frequently in queries are treated specially.

Two level signature files have been suggested by J.J. Rocchio [45] with improved search speed. The advantage of this system are the simplicity of its implementation, efficiency in handling the insertion operations, handling the queries with partial words, ability to support a growing file, tolerance to spelling errors. The disadvantage is long response time when the file is large.

Inversion

In this method each document is represented in terms of key words that describe the content of the document in retrieval process. These words are indexed and arranged in an alphabetical order. Each word points to the documents in which that word occurs. This is the basic approach for many commercialized systems. More sophisticated methods are used to organize the index file, such as hashing, B- trees, or combination of these two. This combination uses two levels for the index file. Here the words starting with same pair of letters are stored together in the second level. First level contains the pointer to the second level.

The main disadvantage of this type of retrieval is the storage overhead, which can reach up to 300% of the original file. In addition to this, updating, re-organizing the index and merging lists take huge amount of cost if they are too long. And the advantages are that, the system is simple to implement and fast. It also supports the synonymous search.

Vector Model and Clustering

The basic idea of clustering is to group similar documents together to form clusters. The idea behind the clustering is grouping the similar documents is that the documents tend to be more relevant to search for the same request. Also grouping similar documents accelerates the searching procedure.

Clustering also can be applied to terms in addition to the documents. Terms can be grouped to form the classes of co-occurring terms. Co-occurring terms are usually relevant to each other and sometimes synonymous. This grouping will enable to forming a thesaurus automatically which in turn makes the searching procedure more efficient. Document clustering is a combination of two different procedures. First, the cluster generation and second, cluster search. And these two procedures are discussed below.

Cluster Generation Methods

In the cluster generation method each documents is represented as a vector and operates on vectors or points of a t-dimensional space. As each document is processed and some keywords are assigned to it. This is the indexing procedure and it can be carried out either manually or automatically. An automatic indexing procedure usually uses the following dictionaries.

- A negative dictionary that is used to remove common words.
- A suffix and prefix list that help to reduce each word to its stem.
- A dictionary of synonyms that helps to assign each word stem to a concept class.

The main criterion for efficiency is the time required for clustering. Space requirements are usually neglected in the performance analysis of the cluster generation methods.

Many cluster generation methods have been proposed. Unfortunately no single method meets both requirements for soundness and efficiency. There are two classes of methods.

- 'sound' methods, that are based on the document similarity matrix.
- Iterative methods, which are more efficient and proceed directly from the document vector.

Custer Searching

Searching in a clustered file is considered to be a simpler task than cluster generation. The input query is a t-dimensional vector and it is compared with the cluster centroids. The searching starts from the most similar clusters, i.e. those whose similarity with the query vector exceeds a threshold. A cluster-to-query similarity function has to be selected; generally people prefer to use the cosine function is preferred.

The vector representation of queries and documents allows relevance feedback mechanism which increases the effectiveness of the search user pinpoints the relevant documents among the retrieved ones and the system reformulates the query vector and starts the searching from the beginning The usual way to carry out query re-formulation is by adding vector addition to the query vector, based on the weighted vectors of the relevant documents and by subtracting the non-relevant ones. Experiments indicate that this method gives excellent results

3.6. Amberfish (Text-Based Information Retrieval System)

Amberfish is general purpose text retrieval software, developed at Etymon Corporation by Nassib Nassar. The main features of this text retrieval system are indexing the semi structured text, structured queries allowing generalized field/tag paths, efficient indexing, and relatively low memory requirements during indexing, hierarchical result sets, allowing modular indexing which provides a way to search across multiple databases and Text Retrieval Conference (TREC) [47] format results. In the process of indexing, Amberfish creates a set of files that collectively known as a database. Once the database has been created, the original text can be searched via database. The following sections explain more details about this system including the role of Amberfish in Grid-based information retrieval systems.

3.6.1. Brief about the Application

The basic features of the Amberfish are indexing and searching. In addition to these it also provides the following features.

- Provides the built-in support for XML documents using the Xerces library.
- Supports Boolean queries, right truncation, and phrase searching.
- Uses relevance ranking and incremental indexing.
- Supports for multiple documents per file.

It is very easy integrate this tool with other UNIX tools. Amberfish uses an open source database "postgresql" as its backend tool. Indexing and Searching features are explained below.

Indexing

The easiest way to understand Amberfish is to index and search a small collection of text files. As an example, consider the Amberfish command:

```
$ af -i -d dbName -C -v *.txt
```

"The 'af' command is used for indexing/searching the files. Here '-i' is used for indexing purpose. 'dbName' indicates the database file that you want to create after indexing, so that the search can done over that particular database. '-d' option is used here for this. The list of files is expected to be at the end of the line. The '-C' option indicates that we want to create a new database. If the database name already exists, it automatically over writes the old one. Finally, the `-v' option tells af to print information about the indexing process as it goes along. In order to provide the user more control over indexing, it also have another option, `-m'. It may be added to the command to increase the amount of memory used for indexing, which helps to reduce indexing time. The indexing process creates a group of files beginning with `dbName'. These are the index files that make up the database. We can use the same 'af' tool with this database to search the files we have indexed" [24]. It also has an option by which we can linearize the database once the indexing is done. And the command,

```
af -L -d dbName.
```

Linearizing reduces the amount of time taken to search, but at the same time linearizing takes lot of time to process. The main disadvantage of linearizing the database is that it modifies the database so that no additional documents can be added later.

Searching

In this section only basic features of searching are described taking the following example.

```
$ af -s -d dbName -Q 'gmu & (computer science | electronics)'
```

The `-s' option with ‘af’ command used for search operation. The `Q' option specifies a Boolean search query, which represents what you are searching for. The symbols, `&' and `|', are Boolean operators meaning `and' and `or' respectively. The above mentioned query can be represented in English as, find all documents that contain the word, `gmu', and also contain the word ‘computer science’ or ‘electronics’. The search string is case-insensitive. Gmu, GMU or gmu are all the same. Amberfish also provides some other features, such as, phase searching, searching the specific fields within documents, XML type documents. The full description of Amberfish Retrieval System is described in reference [27].

4. ARCHITECTURAL BLOCK DIAGRAM

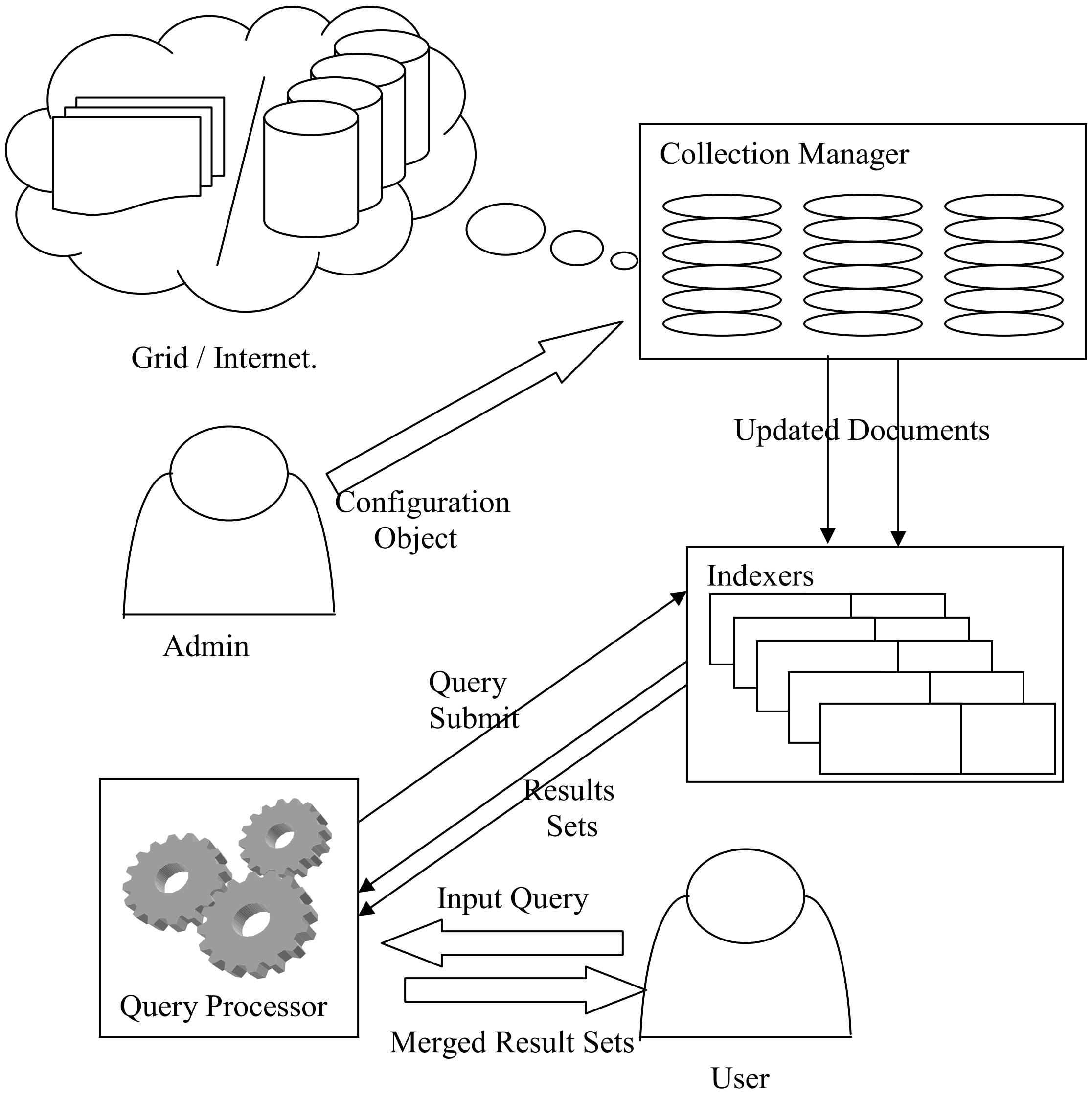


Figure 4.1 Architectural block diagram of Information Retrieval System in a Grid Environment

Figure 4.1 shows the different components and interfaces present in the IR system. An abstract of each of the components of the system are presented below.

Collection Manager (CM):

• Keeps track of updates or additions of documents in the each node.

• Notifies the indexer when it is time to re-index.

Indexer/Searcher Service (IS):

• Indexes all the documents present in the grid.

• Maintain searchable databases for querying document sets

Query Processor (QP)

• Preprocesses queries and if required, it expands the query.

• Provides one access point to multiple indexers.

These three are the basic components of the IR system in a Grid environment. Each of these components and related interfaces are explained in detail in the next chapters.

5. A GLOBALLY DISTRIBUTED INFORMATION RETRIEVAL SYSTEM IN A GRID ENVIRONMENT

Grid Information Retrieval is a concept which integrates the Grid Computing with IR techniques to build a powerful information retrieval system. It adds a whole new perspective to the information retrieval trends in grids. GRID IR is currently a proposed working group of the Open Grid Forum and will use the Open Grid Service Architecture to specify a set of IR Services, which will be implemented in grid environment.

5.1. Introduction

In the last decade, there has been a proliferation of implementations of information retrieval systems and the corresponding IR architectures. The main purpose of this thesis is to provide a means to integrate all the useful features of all these existing architectures and services into one model that serves as common platform for interoperability of current systems. This could be achieved by implementing all the components of IR system as Web Services. Here we present a prototype of a particular system which implements each component as a Web Service, all of which are integrated together to work as fully functional information retrieval system. This is similar to distributed computing except it is more finely refined implementation for task monitoring and synchronization among the nodes present in the grid. Since Grid computing has a security model built in by default, we also can implement desired security features in all levels in the information retrieval system. The requirements and the overview of the system are given in the following sections.

5.2. System Requirements

This section covers detailed description of the requirements for a grid information retrieval system.

5.2.1. General Implementation Requirements

The following describes the general requirements for any information retrieval system in a grid environment.

Distributiveness

The system should cater the needs of IR process in a network connected to more than two computing nodes, so that we can benefit the power of grid computing.

Modular

The system must clearly defined roles for each component (module) of the system. Modularity level of the system depends on the architecture and the network infrastructure. As the number of modules increase, it becomes more flexible, but at the same time, the complexity of the system increases. An effective design must have a balance between these two.

Message Passing

There has to be a way by which all the modules can talk with each other for creating a link, sending and receiving the data once the connection is created.

Asynchronous notifications

One module might request the notification from the other after an event or an action taking place. This way the sink between any two modules can be maintained.

Event-Driven Operation

If any module sends an asynchronous notification, the response should be immediate. Since every component in the system is a Web Service, each service can communicate to others and every message is considered as a separate event.

Service Persistence

It requires that a service is considered to be persistent if it remains active for some time after initial creation of the connection in order to serve the subsequent operational requirements. There has to be a mechanism where you can destroy the inactive services in a timely manner.

Query Persistence

A persistent query is defined as a query against a particular dataset of indices and result set which are updated through scheduling or triggered event such as index update. Consider an example where a new document is available to a collection, which may trigger the dependent indexer to update itself to include the newly added document. In turn the Indexer updates all the datasets that are dependent processes maintaining persistence such that the results may be updated in real-time efficiently.

Meta Services

It requires that all modules must describe access mechanism and the semantic description of all methods defined. These access methods and the argument data types are defined by using WSDL file. All data set must describe its content type, collection and the query holdings.

Data Collection and Descriptive Services

It provides a method to describe an abstract data collection to users and other services. This service is considered to be input for a process that results in collection manager. The description might include specific URLs, message transfer and, crawling rules.

Scheduling Services

It provides a method to schedule periodic updates to the existing datasets. Any update in the document collection should invoke this scheduling service in order to update the all indexers to include newly added or updated documents.

Data delivery Service

There has to be a mechanism for transferring the metadata of the documents from collection manager to indexers.

Document Content Type

The system must have the capability of handling different types of document formats. Specifically, a system must be able to define its transferring data format and should be compatible with all the components. For example, a collection may contain text, html, xml or audio type documents. But indexer may not support the audio files. So they require meta services to list the supported data formats.

Document Transformation Capabilities

It provides a way to change a file from one content type to another content type.

Index Generation Service

It is the basic feature of the IR system which actually indexes all the documents in a searchable database (datasets), provided by the collection manager. These datasets should be available to users in order to access the information.

Query Service

It provides an interface from which user can input the queries to the indexer for searching purpose. These queries are generally expressed in terms of keywords, normalized forms or as regular expressions.

Result Set

It contains metadata of all the relevant documents, which matches with the input query. It contains the links to the actual documents.

Query types

These are mechanisms through which we specify the arbitrary category for a query, providing a way to search based on some particular criterion. For example, to search the documents related to sports and avoid all other datasets from searching, and thereby speed up the process of searching the datasets. To assure the interoperability, there must be a basic query type supported by the system.

Result Set Delivery Service

These services are required to provide a way of retrieving query results from datasets provided by indexer. This will allow a user to retrieve from more than one range of records from the set. Each result retrieved should point to the original location of the document. At the same time there has to be a way to delete the result sets remotely.

Index Scan Services

It supports browsing the content of an index. This will provide the keywords that appear in an index, which helps in better query formation.

Support for peer to peer communication

The communication model should support the P2P. For example there should be a way to communicate between two participants of the system for a given task.

Support for client-server communication

Client must know a way to send the requests to the server and receive the responses from it. The communication model should support client-server communication.

Distributed Searching

The system must able to utilize the power of grid computing by distributing a single query across multiple search modules.

Merging the Result

It should able to merge all the results retrieved from one or more Indexers.

5.2.2. Security Requirements

Security is another important feature in the information retrieval process. Issues related to security at all levels of the system must be addressed. An IR system in the grid environment must assure the same level of security that Grid provides as underlying infrastructure, such as OGSA with ongoing work with Web Services security (WSsec).

Security for the information management and network systems has three basic principles, which are defined to achieve the goals of a good security infrastructure.

i. Confidentiality

The mail goal of this feature is protecting the system from unauthorized access to the information. Two important aspects of this are authentication and identification. By using these two mechanisms we can prevent the unauthorized access to the data.

ii. Access Control

The system should avoid giving full access to the information for all. It should control the information access according to the permissions.

iii. Integrity

Integrity is preventing the accidental or intentional the unauthorized access to system data. In other words the system data can only be accessed and altered by only authorized parties.

The following are part of above mentioned security management issues. The system must satisfy the requirements of mentioned below.

Authentication

Before giving the permission to access the information in grid, the system must identity the user and the connection to the services should be mutually established during any session. The term "user" here indicates a single user, group or another service.

Authorization

It is the process of giving permission or privilege to someone to access the information. It is away to identify the users who are using the system. By implementing this feature we can prevent the misuse of the system from intruders.

Hosting environment authorization

Generally the system is built on the technologies such as Web Services, Globus Toolkit 4 (GT4), Grid Services, which typically run in a hosting environment, such as an operating system, Java run-time environment, Tomcat container, etc. Access to these environments may need to be restricted.

Service-level Authorization

There may be an authorization mechanism at lower levels of the service, which restricts the authorized user from accessing all features of the service. IR in the Grid should enforce this mechanism to prevent the access to the system without proper permissions. The service responses may be asynchronous in cases, where a human is responsible for giving authorizations to users.

Index-level authorization

There may be an authorization mechanism to ensure the authorized access to a specific index or dataset. The system must enforce the authorization decision as either "allow" or "deny".

Record-level authorization

There may be an authorization mechanism to ensure the authorized access to a specific record within an index. As each component in the system is a Web Service, we can merge the result set of the system with the Web results. This result set may contain the merger of internal as well as the external links. In this case the IR system in Grid does not define any control access to such external links. Also it does not provide the second level record authorization, which means it is not capable of authorizing only part of the record. For example: it can not allow the user to access only some pages in a single document. The user either has permission to access the document or doesn’t.

Identity management

System should be able to manage the identities associated with the services. This includes the features like establishment and maintenance.

Single sign-on

The system should eliminate the authentications at different levels of operations. It should maintain a single sign-on procedure which maps between local identity and a Grid identity. It is important that the identity and credential integrates with the user's already existing local security policies.

Delegation

Users should be able to access related services through this service; hence users should be able to delegate his credential through the service.

Credential Renewal

Generally in grids, temporary pr proxy credentials are created for any user to access any kind of data or service that is available on that grid. IR in grids requires a mechanism to renew the temporary credential when it expires.

Intrusion Detection System

In general effective security requires that there should be a way to detect intrusions and breaches. System actions, administrative actions and policies should be monitored, to detect and deal with attacks.

Logging and Audit

System should maintain a log file of all user actions and system activities. This is vital to audit for dealing with intrusions and security threats. Policies should be clearly defined regarding when logs must be retained or disposed are outside the scope of this discussion.

5.3. System Overview

For the past two decades, the Information Retrieval field has developed well beyond the basic operation of indexing text and searching for the useful documents. Present research in the IR field incorporates modeling, document categorization, systems architecture, user interfaces, data visualization, filtering, languages, natural language processing, semantic search, and other related topics [33]. Despite its maturity, until the past decade, this field was a narrow area of interest mainly to librarians and information experts. Many information retrieval systems (search engines) are available and are capable of retrieving the information efficiently over the net. But an owner has sensitive data, spread across the world, who wants to search the data without publishing it to the public, finds an Information Retrieval system in the Grid environment finds to be useful. Such a system can search all the documents available in a grid and is able to merge the results with the information that is available on net retrieved by any search engine. This allows benefit from both the features of normal monolithic search engine and also an IR system in a distributed Grid environment.

The fundamental tasks of IR system in grid are described below:

- Collecting documents available on grid into a single database having many data sets

- Indexing the documents to make the search efficient
- Querying one or more document collections and merging the search results

The rest of this chapter covers each of these features in detail and assumes this particular model of a distributed IR system is segmented into three fundamental components based on the proposal given by GridInformationRetrieval–WorkGroup (GIR-WG). GIR defines interfaces that correspond roughly to each of the three fundamental IR tasks, respectively:

(1) Collection Manager

(2) Index/Search

(3) Query Processor.

This architecture was proposed with the OGSA architecture in mind. At the same time this architecture and specification seek to be neutral in relation to platform.

5.3.1. Design Considerations of the System

One of the primary motivations for developing an Information Retrieval system for Grid environments using Web Services is that the current IR systems are becoming too complex and massive [17]. In addition to this, there has been is a constant increase in capacity of private data storage for monolithic implementation. The reason for considering this specific approach is to enable greater modularity in the next generation of IR systems. Generally, monolithic search engines use web crawlers and spiders to index the data that is available on net; they cannot search for the data available on local machines connected in a network. In this scenario there is a need for an information retrieval system which can search the information present in local machines and merge the results with web search results from the net. Implementing the IR system in the Grid would considerably increase available processing power, since current monolithic systems are restricted by the need to provide instant response to queries of extremely large data collections. It is necessary to consider the scalability of the interaction between the systems over the network in grids. It also is important to consider the security features throughout the system. This model combined with a distributed architecture can enable greater sharing of semi-public information among organizations by supporting their highly customized access constraints.

5.3.2. Architectural Components and Interfaces

This section of the chapter provides a detailed description of the Services (implementation of the interfaces) and the relationships among them. Consider a simple case, where each service implements a single interface. Here, the service is nothing but a Web Service which implements one or more interfaces of the system. For better understanding, we consider this simple case. This enables us to provide different services such as CM (Collection Manager) service, QP (Query Processing) services, and IS (Indexing Service)

Collection Manager

The Collection Manager mainly focuses on collecting and organizing the documents which have to be indexed, searched, and retrieved by using the Indexing service. CM retrieves the documents from various locations and provides to the client depending on the client requests. The client here represents the Indexing Service, by using which the documents are indexed and searched. Any change in the set of collection can be notified to Indexing Service by CM. This change can be either update of existing documents or adding new documents to the collection set. In other word CM acts like a virtual document manager that notifies one or more Index Services upon the collection set updates. Simultaneously, the Index Service has an ability to retrieve the documents from one or more CM Services.

The CM Service accepts documents as input and also gives documents as output. It behaves like a warehouse, bringing all the documents to local system and sending them to off to the required location. Input and output communications in CM are controlled by setting the rules for the CM Service. The configuration of the system can be controlled by the administrator for the IR system in the Grid. The person need not to be an administrator for the local system, these actions (configuring the CM Services) can be performed by IR system function calls of the CM Service. It can also be performed by Indexing Service. Consider an example, creating an instance of Indexing Service enables the system to create an instance of CM Service, or interacts with already created instance of CM service. By using this instance we can configure the CM Services. As we can configure the services using the instance, anybody who can create an instance can control the collection sets using that instance. And because of this, we can’t really differentiate the roles of administrator and user in the IR system in Grid environment. The access control of these services may vary, depending on the location from which the user is accessing.

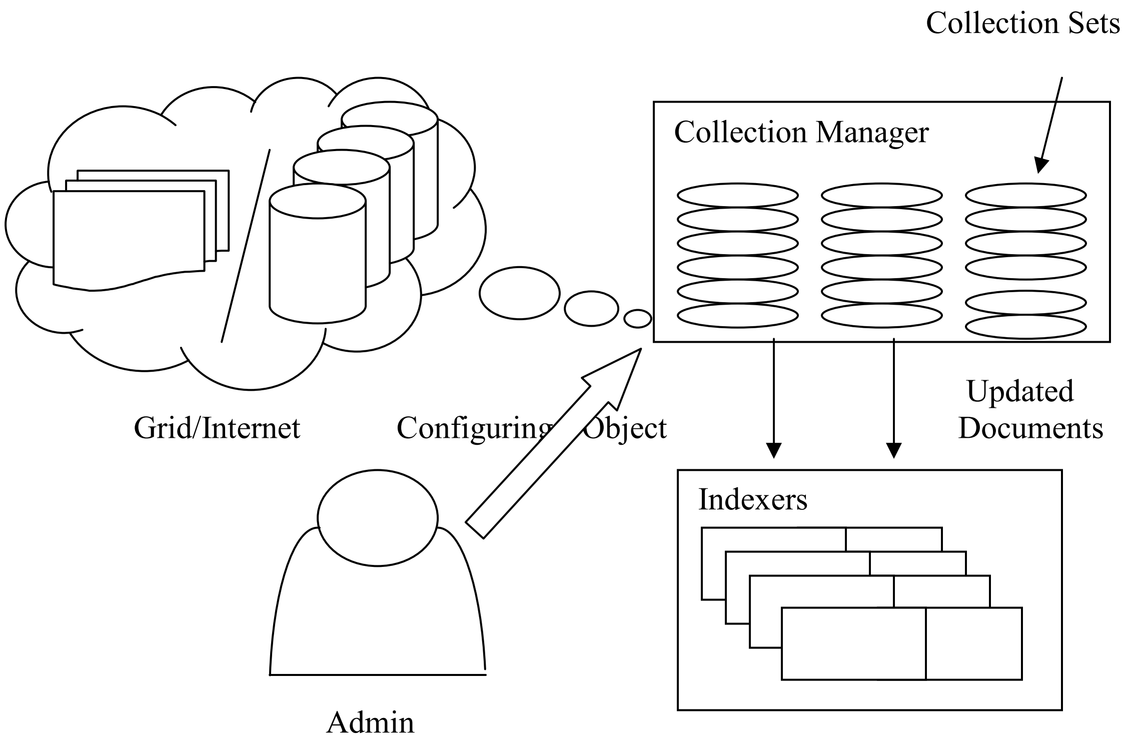


Figure 5.1 Collection Manager Service components and interaction with other services

Figure 5.1 shows the relationship among CM Service and other Services in IR system such as Index Service and also indicates the components of CM service. The CM Service generally takes the input documents from the remote server by using TCP/IP based protocols such as HTTP and FTP. Nevertheless, these documents also may be used to provide input to CM as events. These events can be compared as Really Simple Syndication (RSS) feeds in Web Technology. Depending upon the configuration of CM

Service, it searches the QP and IS to retrieve the documents from the result sets. In general CM Service is scheduled to gather the documents from specific servers but sometimes it checks for document updates on a regular schedule instead. CM retrieves the document based on certain criterion and the criterion is specifies as part of configuring the CM Service.

The CM service uses two types of mechanisms to provide output to the clients, "Pull Service" and "Push Service". Pull service is used when client requests for the document from CM. Push service is used when output is provided to the client by a triggered event under certain circumstances, or a client has a request notification at a regular scheduled time, or some internal or external event has occurred which triggers the CM Service to send the documents to the client (here the Indexing Service). For example, if there are any new documents are added to the database or if the document location is changed, the CM sends notification to client that new documents are available. These rules are specified by configuration.

Index/Search

The Index/Search Service (IS) supports the basic features of a traditional IR system. It accepts documents, indexes them, and provides searching capabilities on those documents. In addition, it also provides a common search interface. IS need not be concerned about collecting the documents from various existing sources, as CM takes care of that. The main purpose of the IS Service is to create and manage the collected documents and provide them in a structured form, as needed to provide searching capabilities. These structures are normally referred to as an “index.” IS also provides the capability of searching the indexed collection sets. This feature is supported by a “Searching Interface”. Searching capability is supported by both Query Processor and Indexing Service. So in order to make the searching feature accessible to both these Services, it is named after “Searching Interface” and can be implemented by these services independently.

The IS component is concerned with indexing and searching. The document collection needed for indexing process is managed by CM Service notifies the indexer for event driven updates. The Query Processor distributes the searching side of the IR system. The implementations of these two procedures are highly interrelated; it is very difficult to separate their implementations, but architecturally they are very distinct and easy to separate. In any IR system, the Indexing Service is the most significant and basic interface and that one has to implement. Since Indexing is the heart of information retrieval, implementing IS as Web Service will definitely add more modularity to the system. The most important feature of the indexing architecture is the distributing of the document collection to one or more CM components. The job of maintaining different input documents from different sources of IS may be reduced by preprocessing or normalization process in the CM.

Figure 5.2 shows the relationship among IS with other Services in IR system and indicates the components of IS Service.

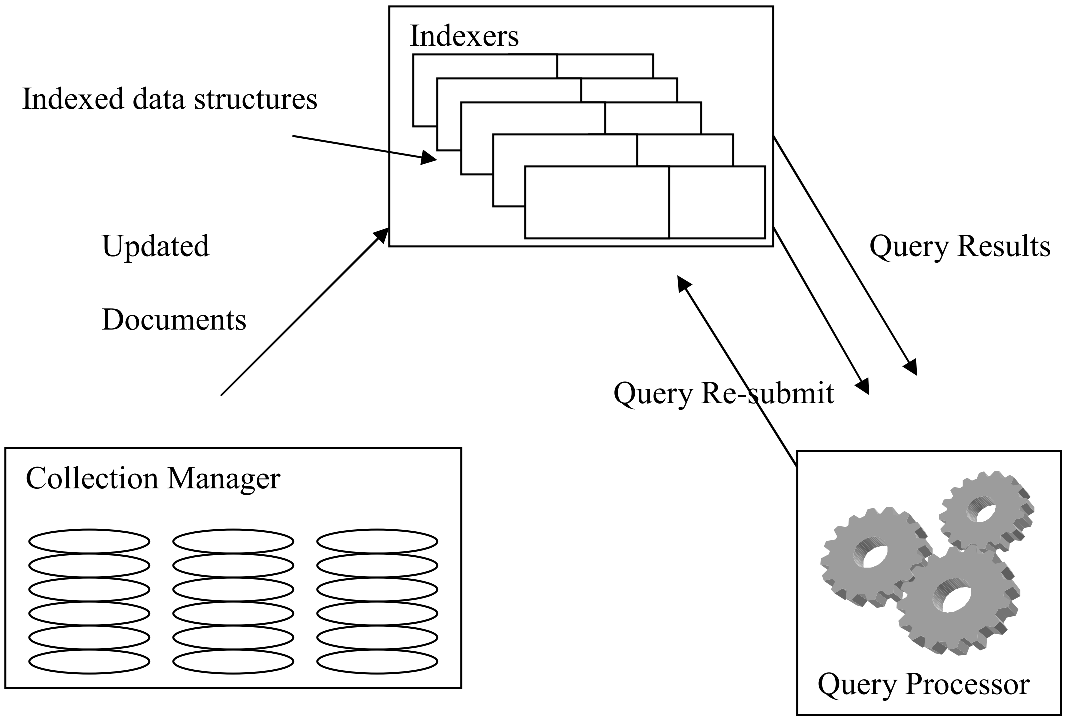


Figure 5.2 Index/Search Service components and interaction with other services

IS service is considered to be the heaviest component in the IR system and has the most implementation complexity. In developing my prototype of IR system in the Grid environment, I used the third party software called Amberfish as IS. Amberfish is an application used for indexing the documents in a local system and a Web Service is implemented to retrieve the indexed documents. This application can be incorporated in the implementation of IR system in Grid to retrieve the data at node level. Once the results are retrieved from different indexers located in different virtual organizations, they can be merged using the QP Service.

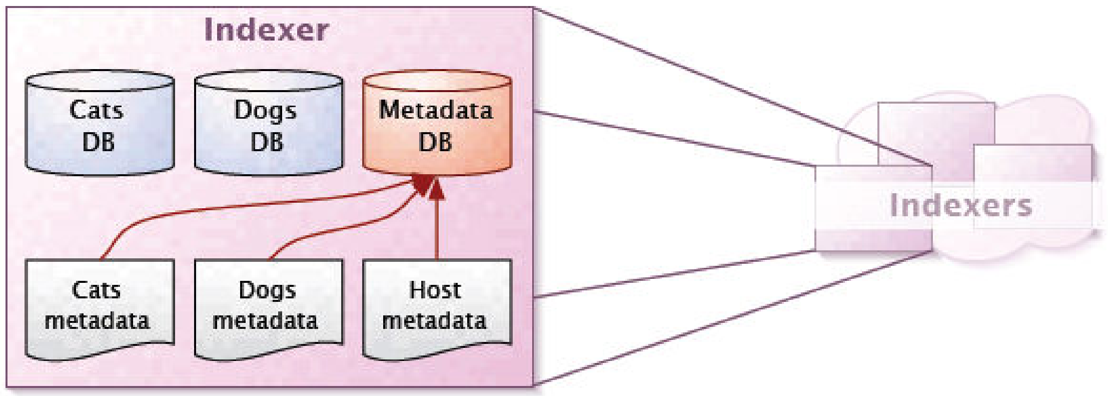


Figure 5.3 Inner components of Index Service [Gir-WG presentation, 65]

Figure 5.3 represents the inner details of the Indexer. It has a two level data retrieval structure. The IS component consists of collection of data structures consisting metadata and the links pointing to the actual location. For the purpose of easy retrieval it also maintains the metadata of metadata. As the structure of IS Service is complex, the most architecturally significant features of IR system are placed in the CM and QP components, so as to distribute the overall complexity of implementation. The IS Service is configured in the same way as CM Service, with its relationship to CM being the only source of documents.

There are again two mechanisms that IS uses for indexing the documents. First, polling in which IS polls one or more CM's for document updates. Second, an event-driven process, in which IS may receive notification from one or more CMs about the document updates. In both the cases IS notifies one or more QPs that the collection has changed, after updating its local index to reflect the changes. Features other than the indexing are mentioned in the configuration, which include re-indexing schedule, document types. Once the indexing is done, IS organizes all the documents and maintains them in databases. These databases may be searched individually or in combination depending on the input query. The above mentioned IS search interface is based on the Z39.50 IR standard [48].

Query Processor

The Query Processor (QP), it is the third architectural component in IR system in a Grid Environment. The QP is a point of entry for a client wishing to search for the information in Grid Environment. It accepts the input queries from the client and then preprocesses the query before sending it to IS, if needed. Here the client can be a single user or a group or another service. It may submit the query to one or more ISs, in which case it is also concerned with merging result sets from those IS's. As discussed in in the previous section, QP and IS provide an identical search interface. With regard to searching criteria QP acts like a virtual IS Service, but without indexing capability of its own. It acts as searching gateway to one or more IS's; although it is not intended to save as a searching service on its own, the interface is common for both IS and QP so it can function as a searching service locally. However it is always preferable that this service should be served though IS Service interface, even if no indexing functions are provided through IR system.

Another purpose of QP is to act as an interface to a collection of ISs and CMs which can be managed independently and dynamically of the QPs. The important concept here is that any arrangement of CMs, ISs, and QPs is only one of potentially multiple virtual GIR organizations. QP can correspond to a single IS or many IS components. Unless QP is explicitly in a query role, it is always considered that the search is being done by using search interface via IS.

In reality a QP can serve as an IS, because both have identical interfaces for searching. Implementing both QP and IS as Web Services, it provides a flexibility for setting up an integrated system dynamically, by creating an instance of only a QP. This is a flexible compromise with respect to the traditional monolithic model of distributed searching, where searching is performed through a single point entry. In Z39.50 model [48], it searching operation is done on the user side in a distributed environment. IR system in a Grid Environment provides a flexibility of implementing either of these approaches depending on the system requirements. Figure 5.4 represents the relationship among QP Service with other Services in IR system and also illustrates the components of QP Service.

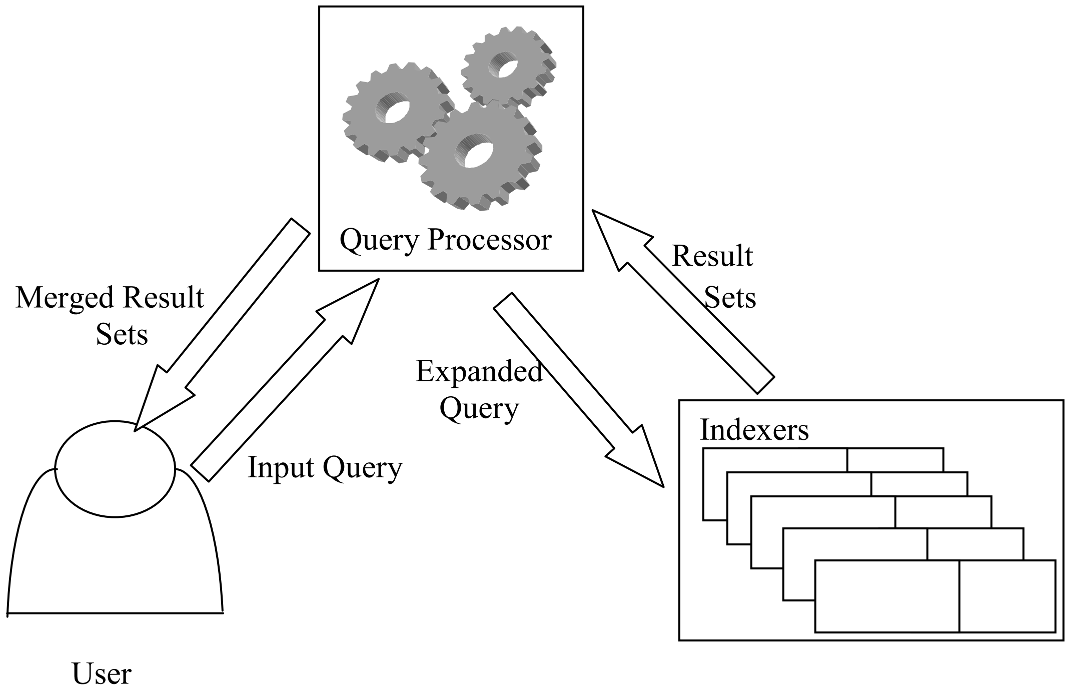


Figure 5.4 Query Processor Service components and interaction with other services

QP can be implemented as a Web Service to provide for distributed searching in Grid environment. The idea behind this is to manage the processing and distribution of queries to one or more IS Services. QP gets the responses from more than one IS and merge them into a single result set. So that the user gets only one result set from different virtual organizations connected to that particular grid. Thus, from the point of view of a user in searching for information in existing Grid systems, QP is not at all different from IS. The key problem comes while processing the query. QP distributes the queries to the Searching Interface and merges the returned results. Figure 5.5 shows the format in which QP sends the query to search interface.

```
<Query>
    <SearchPhrase>
        George Mason
        AND (university OR college)
        AND (fairfax OR virginia)
    </SearchPhrase>
    <SearchFileTypes>
        <Type>HTML</Type>
        <Type>TXT</Type>
        <Type>XML</Type>
    </SearchFileTypes>
    <Date>
        <Start>2000-01-01</Start>
        <End>2007-12-31</End>
    </Date>
</Query>
```

Figure 5.5 Format of the input query sent to Search Interface.

QP also supports the function "standing query," by which is means a query that is automatically reissued to a search interface, based on specific criterion. This can be either in the event of collection update, or on basis of regular schedule. It is another important aspect of IR systems in information filtering and QP can be used to implement this feature. Generally IR implements two types of queries, Fixed type query and Boolean type query.

Fixed type query, it is the concept in which a query is applied to a stream of documents. Boolean type query, it is the concept in which the relevance judgment is made on each document as being either relevant or irrelevant. As we already know that the documents themselves are not maintained by QP or the search interface of IS, QP returns the results as references to the documents. The documents can be retrieved from the client node either by communicating with the CM which has the collection of documents stored or by retrieving the documents directly through URL in case of document from web.

In general, the interactions among components of an IR system are asynchronous; hence all the operations such as managing, updating and searching are executed through event-driven notifications. For an IR system that follows the Z39.50 standards, all interfaces in IR system must support a feature, "Explain." (the term is borrowed from Z39.50.) It explains the details about a service, such as its supported capabilities, the type of content available and searchable. The following are features of the "Explain":

- What document collections those are available for searching?
- Which subset of searching capabilities are supported?
- What metadata elements are directly searchable?

5.4. Benefits of the IR Systems

IR systems in grids fundamentally allow user required information needs to be matched to documents by document collections, indexes and query engines, all of which exist as Web services. Generally all IR systems have only one method for indexing and query processing. These systems treat all documents equally. However, nearly all components of an IR system can run in parallel. An IR system implemented in grids will enable amalgamation of document collections in order to increase the performance. When the security features of grid are included, document collections can be "published" to the grid, including access control lists that can limit who can query the database so that the results are shown only to the users who have the right permission access control. The following are the advantages of implementing IR in Grid Computing:

- The use of divide and conquer approach, implementing this technique to collections, indexing and querying processors combined with parallel execution of each of these components will definitely improve the over all performance of the system, allowing IR on larger collections or otherwise these techniques would be too slow or complex on non-Grid systems;
- The ability to retrieve and tune huge collections of documents.
- Security at all levels of the system, based on separately implementation as web Services.
- Implementing each component as Web Service increases the flexibility of applying multiple algorithms or IR techniques to the same data set, and then provides merging and ranking methods to determine the overall ranking from the merged results.
- Provides a framework for maintaining and updating collections and an infrastructure for changeable index-able content.

Other advantages of the Grid IR system when compared to monolithic search engines are:

- User queries will run only against those collections of documents with a possibility of possessing relevant documents. This process eliminates the search against all the collections which have least or no relevance to the document search. For example person wants to search the information about top universities might want to avoid searching the databases related to the wonders of the world.

- Every collection is customized according to the qualities of the respective collection. Customization of collections include a full range of query processing, document processing, different IR mechanisms such as key word weight, and IR models such as Boolean, Full-Text retrieval, Vector Space, and Latent Semantic Indexing.
- Generally, monolithic search engines run each query against billions of documents which are pre-indexed, where as an IR system in a grid runs the query against far smaller collections. which enables more complex queries to run.
- Since the current model of Grid IR supports collections of smaller size and localization to a particular source in a virtual organization, it can be updated very quickly without re-indexing all the collections. This eliminates the delay as experienced among the harvest runs exhibited by other search engines. Thus it is reasonable to expect the capacity of the system to exceed any search engine.
- Grid computing integrated with Web Services provides a notification mechanism by which one event can trigger the other. For example adding or updating new documents in collection manager will trigger an event of re-indexing of that particular collection. This model supports features such as standing queries and information filtering.

5.5. Usage Scenario

This section describes a sample usage scenario of the information retrieval system in a Grid environment. One typical application need might be an alternative to monolithic search engines for cases when such engines do not offer sufficiently sophisticated capabilities, or when security constraints limit their use. Within in an organization, there may be different franchises at different geographical locations across the world, and each has its own data sources such as units, departments, each of these may have different sets of data of different data types to represent the routine activities. For example, a business like online selling store might have a technical product support unit that has textual documents in XML markup or other formats, and a manufacturing unit that keeps track of part numbers and inventories and the manufacturing facilities.

In this case, if a user wants to access a document related to manufacturing processes or materials. For this, IR in grid environment offers two important capabilities over alternatives retrieval systems: integrity, and security. Since the datasets related to manufacturing and technical documents not available to the outside world, they could not be searched by using public search engines. By creating two separate IR indexes, perhaps with different IR systems or settings, search is enabled. Here an IR system offers the capability of search across both datasets and merge results based on the capabilities of each IR system. Because IR operates within the secure Grid computing environment, it is possible to associate different control access to different data sets depending on the sensitivity of the information.

Another important use case scenario that should be considered is information filtering. IR in grid environment offers a capability to set up long-running queries against dynamic datasets, a process known as information filtering. In other words if any system includes the features such as resource monitoring and events tracking apart from the regular searching option this is known as information filtering. In such a system document indexes will be automatically updated with an automated trigger action define by collection manager. If any new documents are added to the system, CM will trigger an action to re-index the current data sets. Thus, if any inventory is updated in above mentioned example on a constant basis we can trigger an action of re-indexing the datasets accordingly. This keeps the data sets always updated.

6. DETAILED DESIGN OF THE SYSTEM

This chapter presents the detailed design of a IR system in the Grid environment using the Use Case, Class and Sequence diagrams. The goal from this is to produce a model of the components involved in an IR system which later will need to be built. The representations of the components that are to be used in the target system need to be designed.

UML Approach

Software design is a process that gradually changes as various new, better and more complete methods with a broader understanding of the whole problem in general come into existence. There are various representations of methods in software design and they are as follows:

- Use case Diagram
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- State Chart Diagram
- Component Diagram
- Deployment Diagram

Here, here we use only first three approaches to describe the design of the system. The following explain each of these three approaches in brief.

Use case Diagrams:

A Use Case Diagram consists of use cases and actors and shows the interaction between the use cases and actors.

- The purpose is to represent the system requirement from user's perspective.
- It must be remembered that the use cases are the functions that are to be performed in the module
- An actor could be the end-user of the system or it could be another external system.

Class Diagram:

This is one of the most important diagrams in software development. Every class in the diagram is divided into three layers. One has the name, the second describes its attributes and the third its methods. The private attributes are represented by a padlock to the left of the name. Main features include

- The relationships are drawn between the classes.
- Developers use the Class Diagram to define the classes.
- Analyses use it to show the details of the system.

Sequence Diagram:

The purpose is to show the flow of functionality through a use case with information flow sequenced in time.. In other words, we can call it mapping processes in terms of data transfers from the actor through corresponding objects.

- To represent the logical flow of data with respect to a process.
- It must be remembered that the Sequence Diagram displays objects and not the classes.

6.1. Use Case Diagrams

Use Case diagram for Search Procedure in IR system for a Grid.

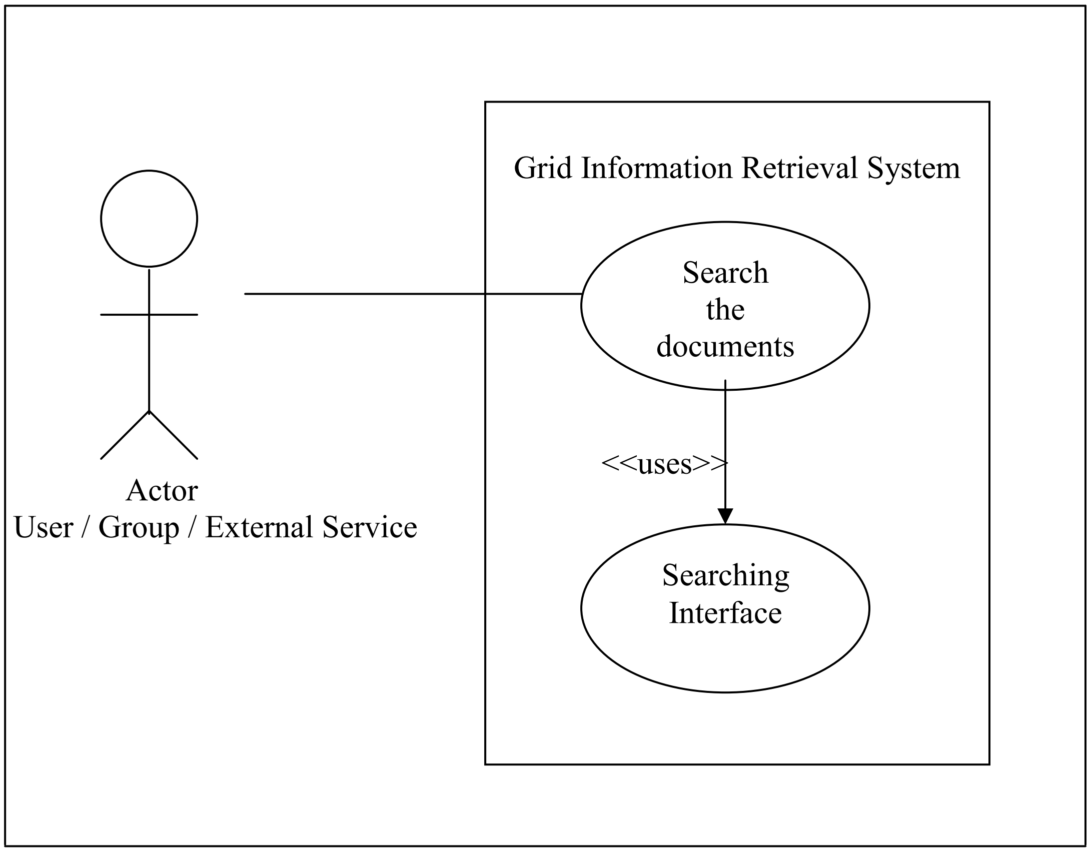

Grid Information Retrieval System
Search the documents
<<uses>>
Searching Interface
Actor
User / Group / External Service

Figure 6.1 Interaction between the user/client and the Searching procedure

Figure 6.1 shows the interaction between the actor and the Search Service. Here actor can be a single user or a group of users or any external service. An actor calls the Search Interface. The user sends the query to QP by interacting with GIR service which in turn sends the query to the Searching Interface to retrieve the documents. This Searching Interface is implemented by QP.

Use Case diagram for Index Procedure in IR system for a Grid.

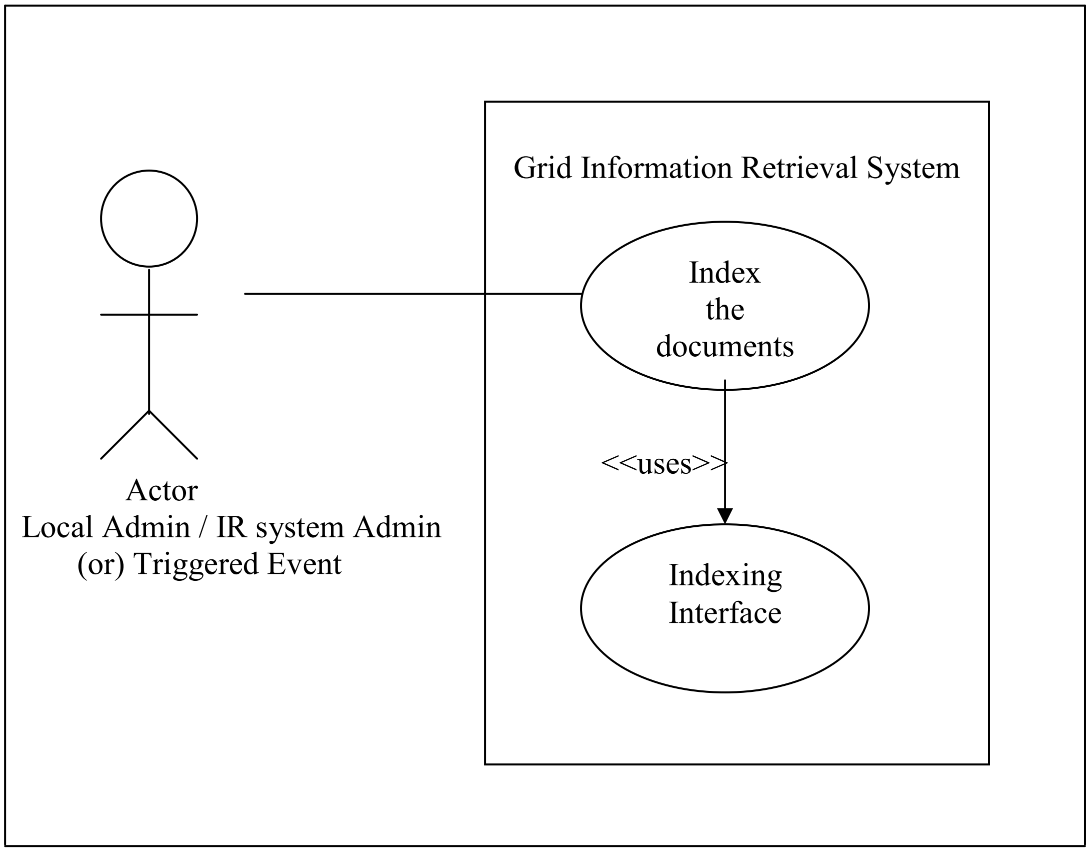


Figure 6.2 Interaction between the administrator and the Indexing procedure

Figure 6.2 shows the interaction between the actor and the Indexing Service. Here actor can be a local/IR system administrator or any triggered event. When an actor wants to index the list of documents, he sends a request to the GIR service with a list to index; which in turn sends it to the Indexing Service. Triggered action can be a scheduled event or forced signal sent by the Collection Manager when the collection is updated.

Use Case diagram for Search / Index Interfaces in IR system for a Grid.

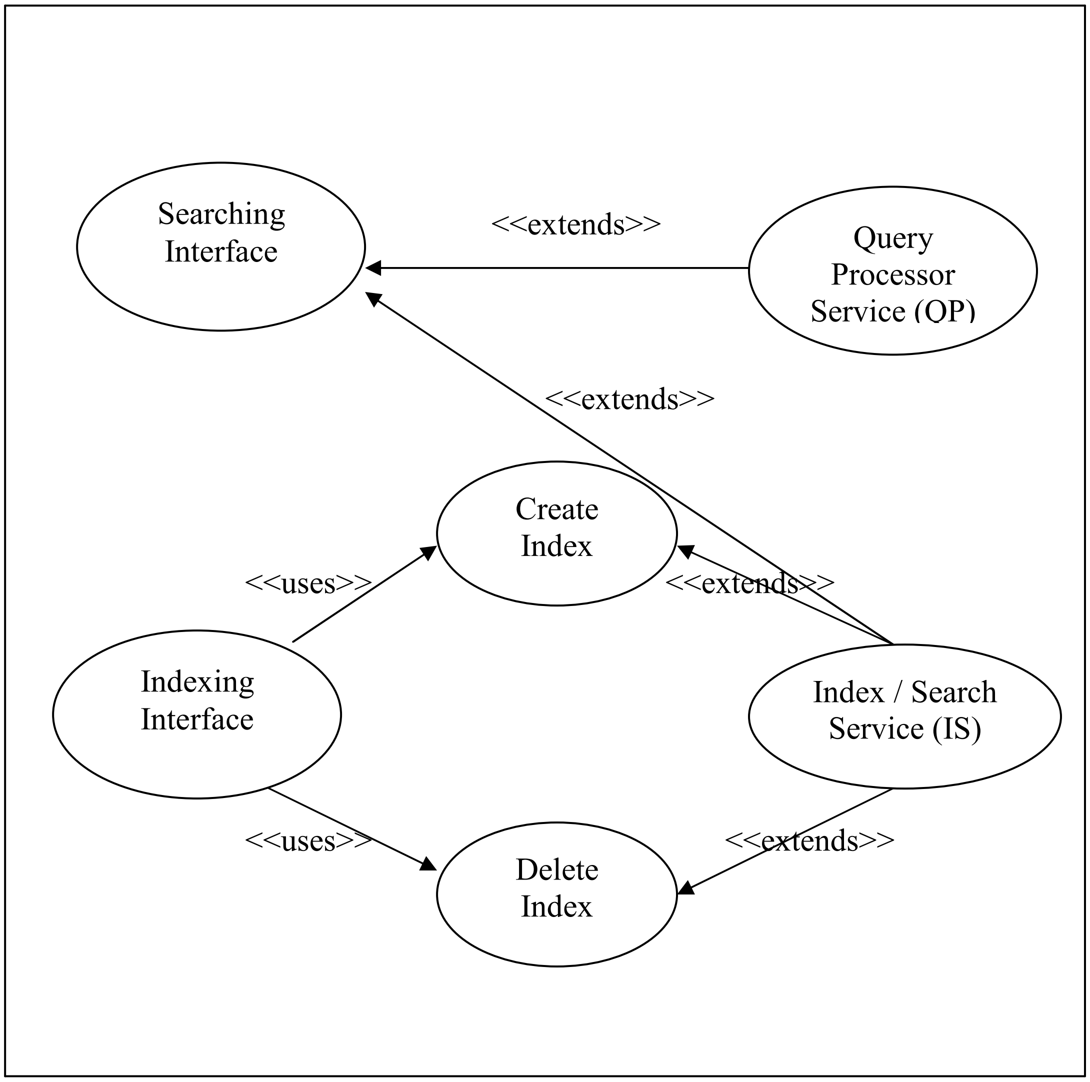


Figure 6.3 Implementation of the Index / Search interfaces

Figure 6.3 shows the implementation of the Index/ Search interfaces by corresponding Services. Here Search Interface is implemented by both QP Service and IS and IS implements the Indexing Interface (used for creating and deleting the index) in addition to Searching Interface.

Use Case diagram for Collection Manager in IR system for a Grid.

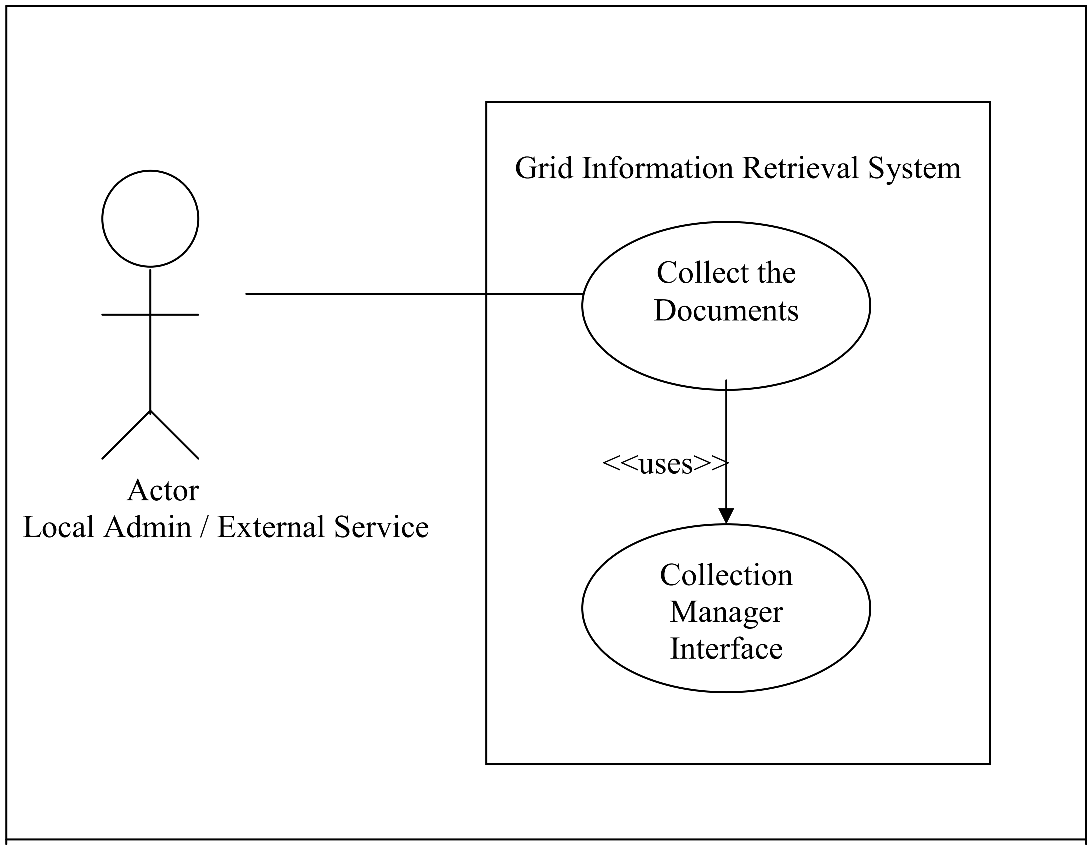

Grid Information Retrieval System
Collect the Documents
<<uses>>
Collection Manager Interface
Actor
Local Admin / External Service

Figure 6.4 Interaction between the administrator and the Indexing procedure

Figure 6.4 shows the interaction between the actor and the Collection Manager Service. Here the actor can be a local/IR system administrator or any external service. When an actor wants to add or update the existing collection sets, sends a request to the GIR service with the list to add or update. The CM in turn schedules an event which triggers the re-indexing task.

Use Case diagram for Search / Index Interfaces in IR system for a Grid.

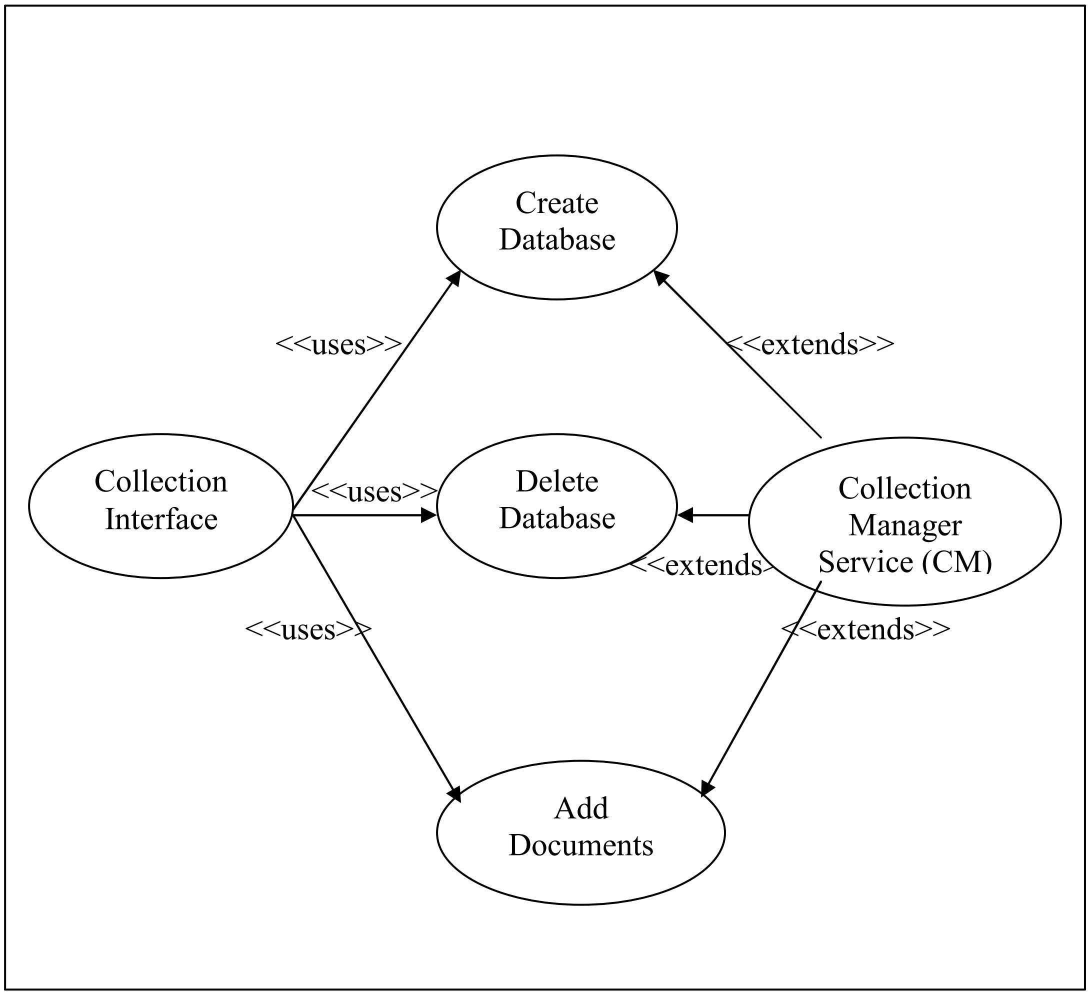
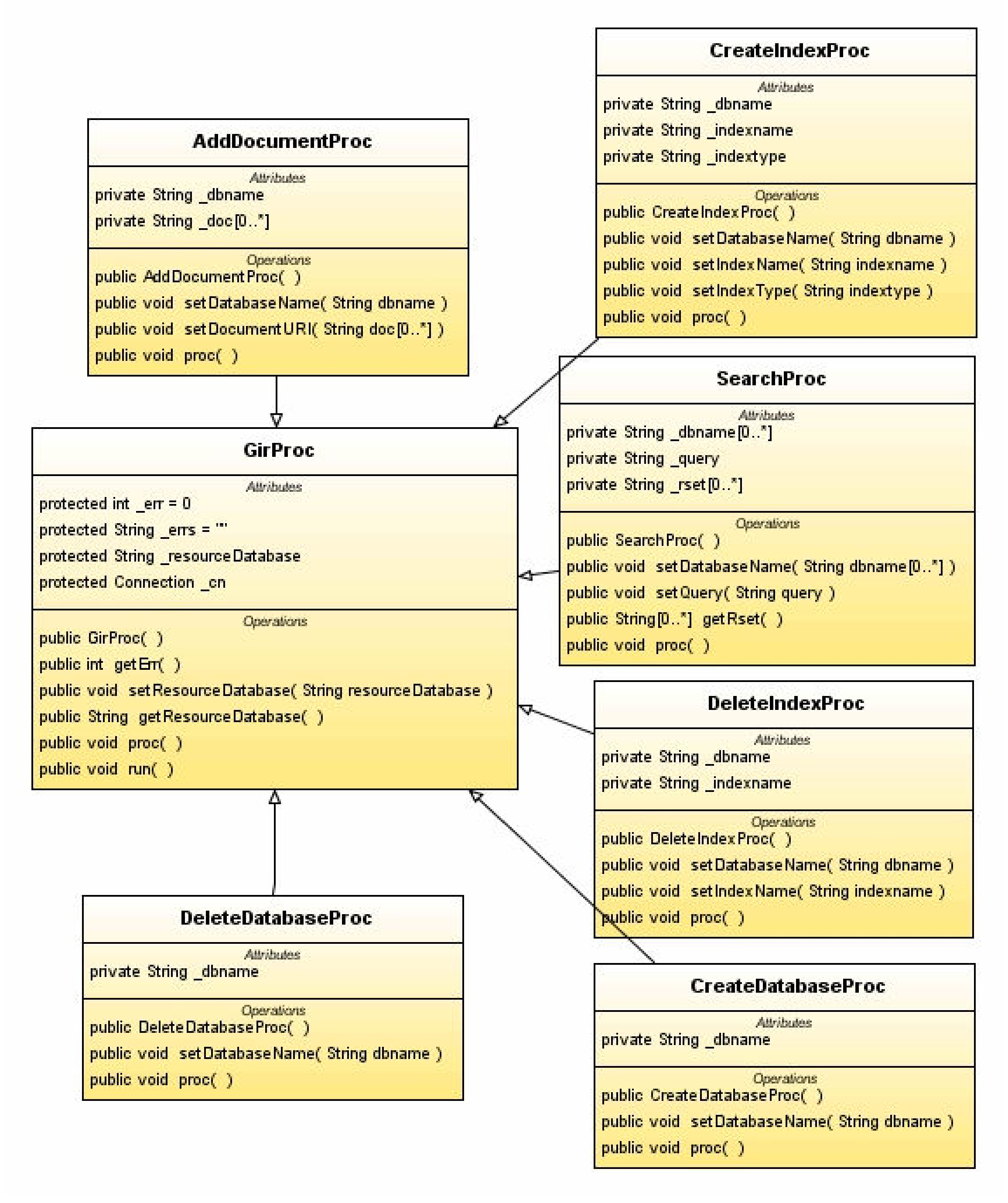


Figure 6.5 Implementation of the Collection Manager interface

Figure 6.5 shows the implementation of the Collection interfaces by CM Services. Here the CM Service implements the Collect Interface. It is used to add and delete databases to the collection sets and It can also add new documents to the existing database using Add Documents interface.

6.2. Class diagrams

Class diagram for the interactions between the interfaces.



CreateIndexProc
Attributes
private String _dbname
private String _indexname
private String _indextype
Operations
public CreateIndexProc( )
public void setDatabaseName(String dbname)
public void setIndexName(String indexname)
public void setIndexType(String indextype)
public void proc( )
AddDocumentProc
Attributes
private String _dbname
private String _doc[0..*]
Operations
public AddDocumentProc( )
public void setDatabaseName(String dbname)
public void setDocumentURI(String doc[0..*])
public void proc( )
SearchProc
Attributes
private String _dbname[0..*]
private String _query
private String _rset[0..*]
Operations
public SearchProc( )
public void setDatabaseName(String dbname[0..*])
public void setQuery(String query)
public String[0..*] getRset( )
public void proc( )
GirProc
Attributes
protected int _err = 0
protected String _errs = ""
protected String _resourceDatabase
protected Connection _cn
Operations
public GirProc( )
public int getErr( )
public void setResourceDatabase(String resourceDatabase)
public String getResourceDatabase( )
public void proc( )
public void run( )
DeleteIndexProc
Attributes
private String _dbname
private String _indexname
Operations
public DeleteIndexProc( )
public void setDatabaseName(String dbname)
public void setIndexName(String indexname)
public void proc( )
DeleteDatabaseProc
Attributes
private String _dbname
Operations
public DeleteDatabaseProc( )
public void setDatabaseName(String dbname)
public void proc( )
CreateDatabaseProc
Attributes
private String _dbname
Operations
public CreateDatabaseProc( )
public void setDatabaseName(String dbname)
public void proc( )

Figure 6.6 Class interactions between the services

Class diagram for the Service Class

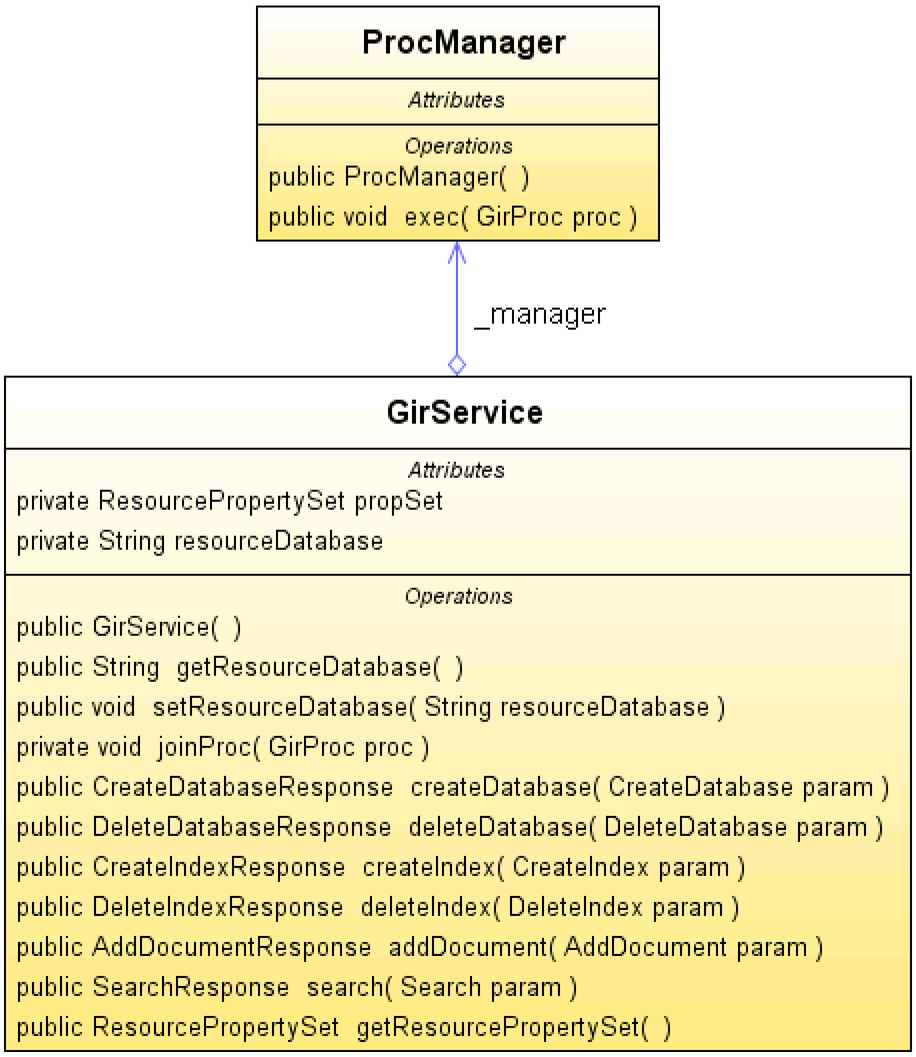

ProcManager
Attributes
Operations
public ProcManager( )
public void exec(GirProc proc)
_manager
GirService
Attributes
private ResourcePropertySet propSet
private String resourceDatabase
Operations
public GirService( )
public String getResourceDatabase( )
public void setResourceDatabase(String resourceDatabase)
private void joinProc(GirProc proc)
public CreateDatabaseResponse createDatabase(CreateDatabase param)
public DeleteDatabaseResponse deleteDatabase(DeleteDatabase param)
public CreateIndexResponse createIndex(CreateIndex param)
public DeleteIndexResponse deleteIndex(DeleteIndex param)
public AddDocumentResponse addDocument(AddDocument param)
public SearchResponse search(Search param)
public ResourcePropertySet getResourcePropertySet( )

Figure 6.7 Service Class, which creates the instance from which connection to the service will be created

Figure 6.6 shows the interaction between the interfaces implemented by QP, IS and CM services. All interfaces are derived from GirProc class. It is a class which creates a thread for each event. Figure 6.7 shows the GirService class which actually acts as an interface between user and the IR system; it creates the binding and links the types.

Class diagrams for the Static Members Classes used by all interfaces.

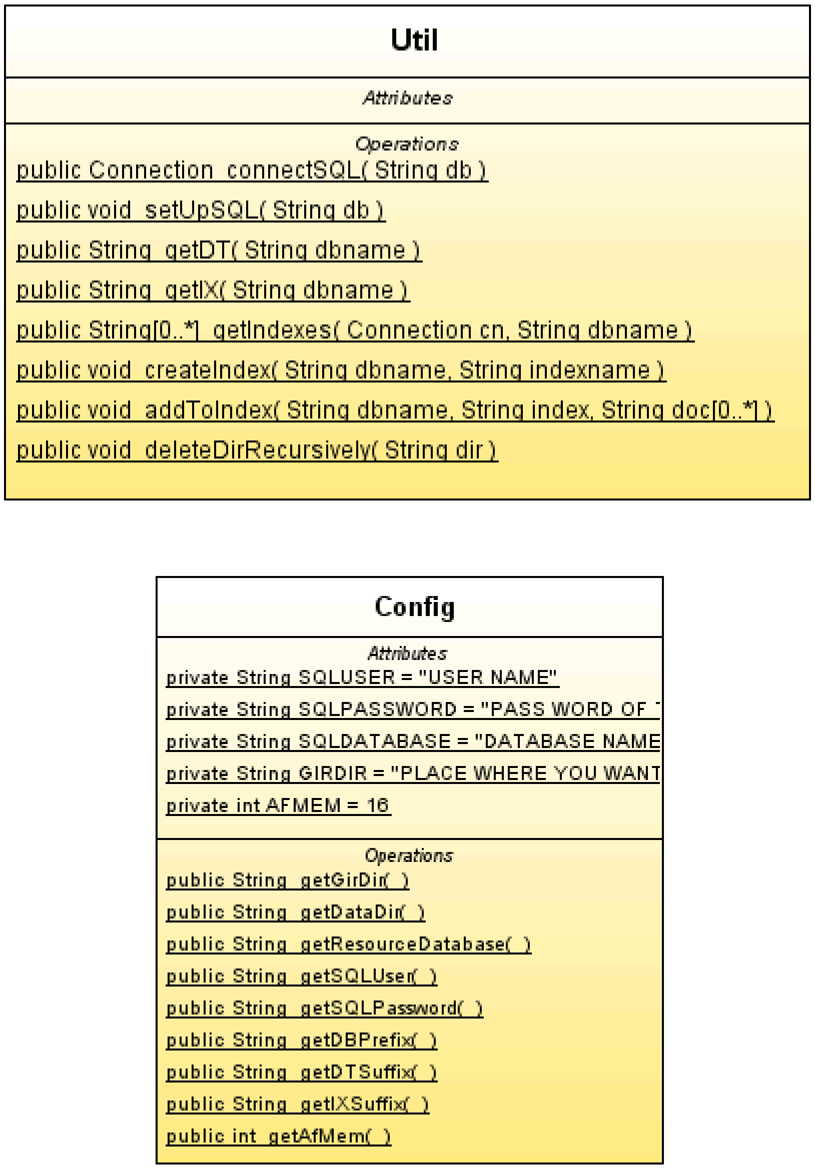

Util
Attributes
Operations
public Connection connectSQL(String db)
public void setUpSQL(String db)
public String getDT(String dbname)
public String getIX(String dbname)
public String[0..*] getIndexes(Connection cn, String dbname)
public void createIndex(String dbname, String indexname)
public void addToIndex(String dbname, String index, String doc[0..*])
public void deleteDirRecursively(String dir)
Config
Attributes
private String SQLUSER = "USER NAME"
private String SQLPASSWORD = "PASS WORD OF
private String SQLDATABASE = "DATABASE NAME
private String GIRDIR = "PLACE WHERE YOU WANT
private int AFMEM = 16
Operations
public String getGirDir( )
public String getDataDir( )
public String getResourceDatabase( )
public String getSQLUser( )
public String getSQLPassword( )
public String getDBPrefix( )
public String getDTSuffix( )
public String getIXSuffix( )
public int getAfMem( )

Figure 6.8 Util (Abstract class) and the Config class, which are used by all interfaces

Figure 6.8 represents two static class diagrams, which are the static classes and are used by all services for basic operations such as get username, password, and etc.

6.3. Sequence Diagrams

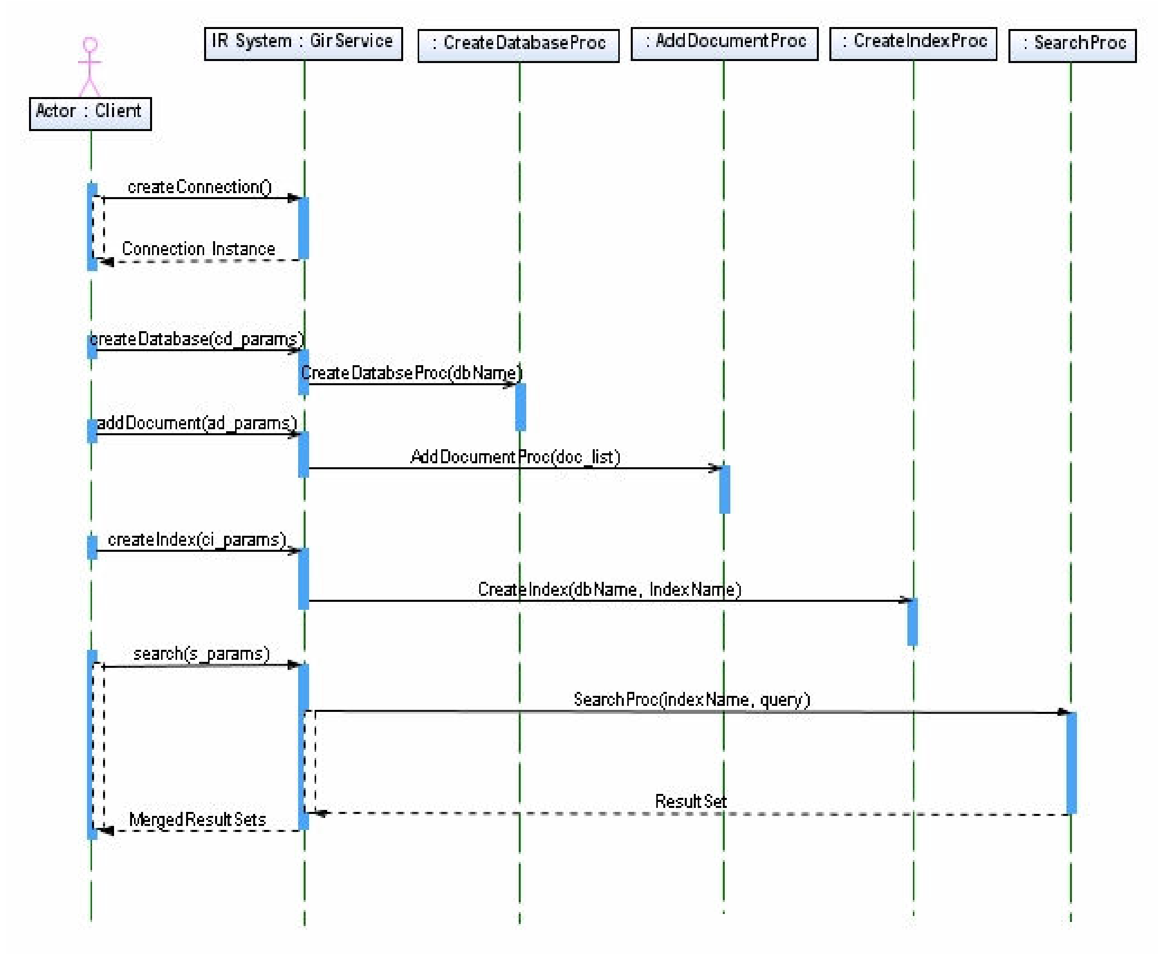


Figure 6.9 Flow of interactions between the client/user and the IR system components

Figure 6.9 shows the flow of actions in IR system between different interfaces. In the first interaction service creates an instance for the client. All the interaction are done using GIR Service interface. When user accesses the system for any service, first it contacts the GIR Service; which in turn accesses the corresponding interface that is requested for.

7. PERFORMANCE EVALUATION AND ANALYSIS

In today's world, Information Retrieval (IR) systems are increasingly being used by thousands of users on larger databases. The systems allow users to connect to a single database either locally or remotely. In this case, the performance of the system is inversely proportional to the number of users and resource demands [40]. To sustain better performance with respect to the constant increase in needs, Grid Computing offers a solution based on distributed computation. It provides for better sustained performance by spreading work across the nodes in a network enabling the use of parallel computation. However, due to the mix of different I/O and CPU intensive applications, IR systems present unique problems for system architects and developers. Another important concern of an IR system is the overhead of the Grid middleware in addition to Network issues.

7.1. IR systems for a Grid environment

As part of this research, I designed and implemented a distributed information retrieval system in a Grid Environment. The following specifies the features of the system.

- It provides multiple simultaneous connections between Clients and Servers.
- An input query from the user is given to Query Processor (QP). It modifies the query, if necessary, and forwards it to Indexer Service for searching.

- The Indexer searches for the documents from the indexed databases present and returns the links to the documents maintained by collection manager.
- QP merges all the results from different indexers and presents the results to the user.

Early performance tests used two different grids located in GMU, Fairfax. One is the Hydra Grid; a cluster consists of 30 Linux machines running Fedora Core 3. Each node computer contains dual 3.2 GHz Intel Xeon processors and 2GB of RAM. They are connected to a head node that also has two 3.2GHz Xeons and 2GB of RAM. The second one is the GMU C4I Center's NETLAB Micro Grid, a cluster of six machines out of which three are Linux machines running Fedora 6 with 1.6 GHz Athelon processor and 2GB of RAM. The other three are running on Windows XP with 2.4 GHz Intel P4 processor and 1 GB RAM. The indexing process converts the documents into XML, parses the XML, indexes them, and then stores them in a searchable database format. The IR system has achieved an average sustained indexing rate of about 3000 to 3500 records per second on average. Larger, more complex records were tested with 640 MB of Text Encoding Initiative (TEI) encoded documents. The testing was done using LAN from GMU, Virginia and also using an SSH remote connection from Seattle, Washington. The following section of the document describes the performance-related issues of the Grid Information Retrieval system.

7.2.Workload Model

A workload model is required to address the performance related issues of an IR system. To accurately model the system, I conducted different experiments each concentrating on different parameters that affect the performance of the IR system.

1. Effect of Grid middleware on the performance of the system.
2. Effect of Job Scheduler (Sun N1 GE) on overall performance of the system.
3. Effect of number of nodes in a grid on performance of the system.
4. Effect of Amberfish on the performance of an IR system running as a Web Service compared to running as a standalone application.

The following sections cover these four experiments in detail. In order to asses the reliability and the performance of the software accurately, we must eliminate the external factors which cause the experiment results to vary. So I made the following assumptions while conducting the experiments:

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- Topology doesn't change.
- The network is homogeneous.

7.2.1. Effect of Grid middleware on the performance of the System

Since the performance of distributed Grid applications is dependent on operation of the underlying grid middleware, it is very important to analyze grid middleware's behavior and performance in order to evaluate the Grid application performance.

In order to measure the performance of the grid middleware (GT4), I used two different tools for two distinct purposes. The first, Java Instrumentation Suite (JIS) [49], contains tools to trace running Java applications on a JVM as well as the tools used for parsing and merging different files. The second, Linux Trace toolkit (LTT) [50], is a fully functional system developed for Linux operating system to identify all the processes running on the OS.

The grid middleware, GT4 container generally uses two major sets of threads. The ServiceDispatcher thread and Service threads. The first one, Service Dispatcher handles the client connections and responsible for sending the requests to a RequestQueue. The second thread set, Service threads receive the requests from the RequestQueue and process them. The creation of these threads generally is dynamic depending on the server load. This experiment describes the effect of Gird Middleware (performance of the RequestQueue and ServiceDispatcher) on the overall system performance. GT4 is generally integrated with a job scheduler to run the service as a grid service. In this case, sun grid engine is used as job scheduler to run the Grid Services.

In the test, I created some jobs that overload the CPU that uses a for-loop to create 100% CPU load. This executed 100 instances of these in parallel, which automatically overloads the CPU at system level. Running these jobs not only reduces the performance losses in some test cases. In the test that I ran on hydra cluster with 30 machines, Globus Toolkit 4 was able to execute 285 jobs on average before any of the jobs failed. Another important item noticed was that the increased number of jobs executed decreased the response time. Globus Toolkit 4 continues trying to execute all the jobs as soon as possible despite system loading. The tendency of executing the jobs in this fashion often results in the negative effect of locking up the Grid Middleware and causing the response time to increase drastically. It reached a point where my experiment job execution time was nearly 52 seconds on average for a process which generally takes 5 seconds to complete.

In order to show the effect of grid middleware on the overall system performance, I compared the performance of the system running as a Web Service to the performance of the system running as a Grid Service using GT4.

The main purpose of this experiment was to find the effect of Grid Middleware in the overall performance of the Information retrieval system. The result is illustrated in figure 7.1. In figure 7.1, the X-axis represents the number of queries submitted to the system at once and the y-axis represents the Average Query evaluation time (QET). QET can be defined as the average time taken by an IR system from the submission of the query until results are available. I compared the QET between the service run as Web Service and the service run as Grid Service. The difference between these two is the overhead of the Grid middleware.

I assumed the network capacity to be consistent and there is no latency in the network throughout the experiment and the number of terms in the query has no effect on the retrieval time. In order to achieve this, I conducted the experiment with the same number of terms in all the queries.

I started my experiment, submitting 10 queries. When the IR system was running as Web Service it took 18 seconds on average for retrieving the results. When the same number of queries were submitted to the IR system running as a Grid Service it took 27 seconds. The difference between the retrieval times is considered as an overhead of the grid middleware. It is 9 seconds in this case.

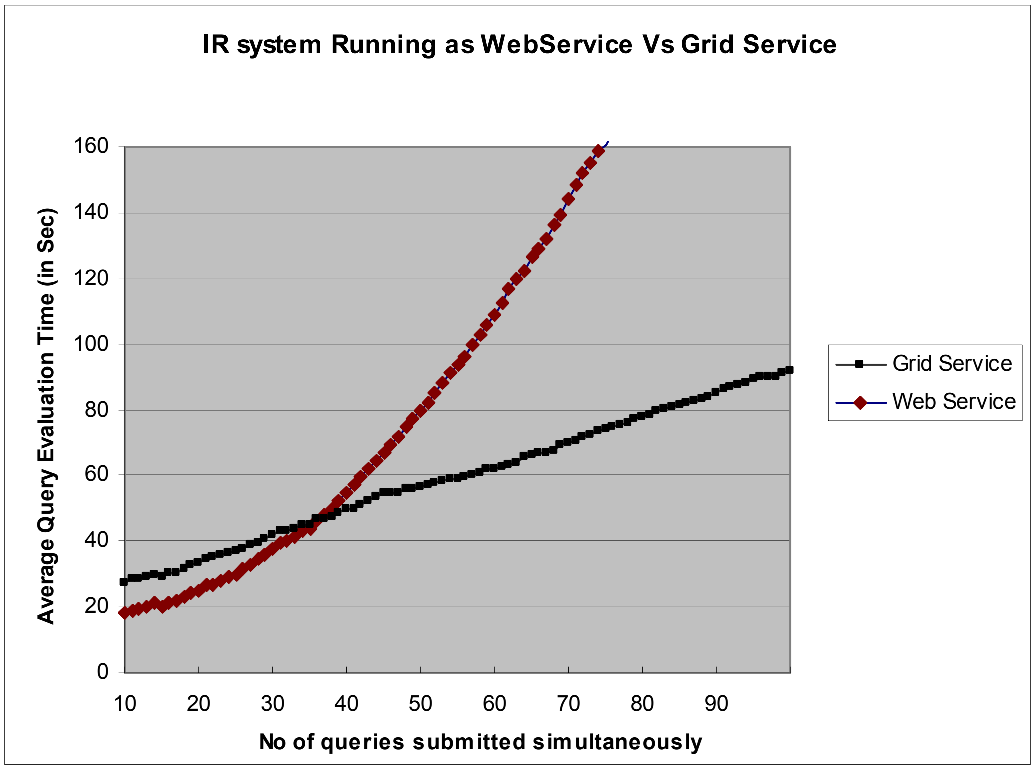


Figure 7.1 Comparison between the IR systems running as Web Service Vs Grid Service

Consider a Web Service hosted on a single machine in a web container; in this case, retrieval time depends on the number of requests that the container can handle simultaneously. Query retrieval time increases exponentially in case of Web Services. This behavior can be seen in the above graph. All the queries are placed in a waiting queue and will be served according to the implemented job scheduling algorithm. As the number of requests increases, waiting time also increases which in turn increases the retrieval time. The above graph clearly indicates the exponential increase in the Query Evaluation Time with increase in the number of queries in case of Web Service. On the other hand when an IR system is implemented as a Grid Service, there will be an overhead in the communication between the nodes and also an overhead of merging the results. As soon as a request arrives, it will be added to the RequestQueue. At this point the ServiceDispatcher is responsible to delegate the job to the scheduler to be assigned to an idle resource. Grid middleware is responsible to poll the job scheduler to see if there is an idle CPU to which the job can be delegated. If all the machines are busy serving the requests, the size of the RequestQueue increases. This causes an increase in the waiting time for a request, which again increases the Query Evaluation Time.

What makes Grid Service very different from a Web Service is the ease at which the system can be scaled by increasing the number of nodes connected to the main node. After a certain point the overhead of Grid middleware is small compared to the retrieval time. In Figure 7.1, it is possible to see the gradual increase in retrieval time for Grid Service as the number of queries submitted to the system per unit time increases. At the same time, retrieval time for the Web Service increases exponentially.

As the number of queries submitted to the system increased, the Grid Service performance improved as compared to the Web Service, The Web Service was hosted on a single root machine; the IR system performed better than the Grid Service until either the container reached to the threshold or the CPU has reached its 100% utilization. Generally the performance of Web Services depends on the configuration of the machine on which it is running, whereas the performance of the Grid Services depends on the number of nodes that are connected to the root node in addition to the root node itself. Grid middleware provides a platform which enables execution of jobs in parallel. If the server is powerful enough to handle the incoming requests in a timely fashion, Web Services are always preferred and they do perform better. However if the incoming traffic to the service is unpredictable and is increasing over time and also the job can be divided into smaller jobs Grid Services definitely would provide a less expensive mechanism through parallel computing while allowing to add increasing numbers of machines to the cluster without excessive overhead.

7.2.2. Effect of Job Scheduler (Sun N1 GE) On Overall Performance of the System

The main purpose of scheduling is to run the jobs in parallel in a distributed environment. To perform this on different nodes of a distributed system is a challenging area of research. Even though extensive research has been done in this area [50, 52, 53, 54], it is still not well understood how to schedule and execute parallel efficiently jobs in a distributed environment [53]. By distributing tasks among the processors, we can improve the performance of the system by minimizing communication overheads and/or maximizing resource utilization. I used Sun Grid Engine as a job scheduler while developing the system. SGE has queues located in server nodes which have attributes characterizing the properties of the different servers. A user may request certain execution features at submission time such as memory, execution speed, and available software licenses, etc. Submitted jobs wait in a holding area where its requirements/priorities are determined. It only runs if there are queues (servers) matching the job requests. The following section briefly explains the different job scheduling algorithms [73] implemented in the Sun Grid Engine [70].

Sun N1 Grid Engine 6.0 Scheduling Algorithms

The current system uses the Sun N1 Grid Engine as its job scheduler and is integrated with the Sun grid-middle ware GT4. Sun Grid Engine uses different types of advanced scheduling algorithms like priority, shared tree policy, functional policy, urgency policy, and unified ticketing. Each of these scheduling algorithms is described below [73].

- *Priority:* Each job is given a priority number. The job with highest priority is scheduled to execute first. There are two types of priority scheduling: Pre-emptive and Non-pre-emptive.
- *Share Tree Policy:* Users can distribute the tickets to which they are currently entitled using different shares assigned via ***'–js'***. If all jobs have the same job share value then the tickets are distributed evenly. Otherwise, jobs receive tickets relative to the different job shares. Job shares are treated like an additional level in the share tree in the latter case.

- *Functional Policy:* The job share can be used to weigh jobs within the functional job category. Tickets are distributed relative to any uneven job share distribution treated as a virtual share distribution level underneath the functional job category. If both the Share Tree and the Functional Policy are active, the job shares will have an effect in both policies and the tickets independently derived in each of them are added up to the total number of tickets for each job.
- *Urgency Policy:* There are three different kinds of urgency related sub-policies and they are ResourceUrgency, WaitTimeUrgency and DeadlineUrgency. Expensive assets such as software licenses can be assigned a higher static ResourceUrgency value to ensure they are used as frequently as possible. Wait Urgency increases as jobs spend waiting for execution. This can be used to prevent very low-priority jobs from being 'starved'. DeadlineUrgency increases as a preset job dispatch deadline approaches. Jobs with approaching dispatch deadlines will rise in priority.
- *Unified Ticketing:* All of the various policies that use 'tickets' (share tree, functional policy and override policy) are combined into a unified policy simply by normalizing each value to arrive at a relative priority.

In addition to the above mentioned scheduling algorithms there are four more scheduling algorithms implemented in Sun Grid Engine which can be used according to need.

- Improved Priority based scheduling
- Resource reservation & backfilling
- Override Policy
- Deadline

All of my experiments used the default job scheduling algorithm, FCFS priority scheduling. The primary focus of this experiment was to evaluate the role of job scheduler in the performance of the overall system, but not to evaluate the intricacies of the job scheduling algorithms. In order to eliminate the intricacies of analyzing the algorithms that affect job scheduling of processes, the experiments which compare the overall performance of the system were conducted with and without running Job Scheduler.

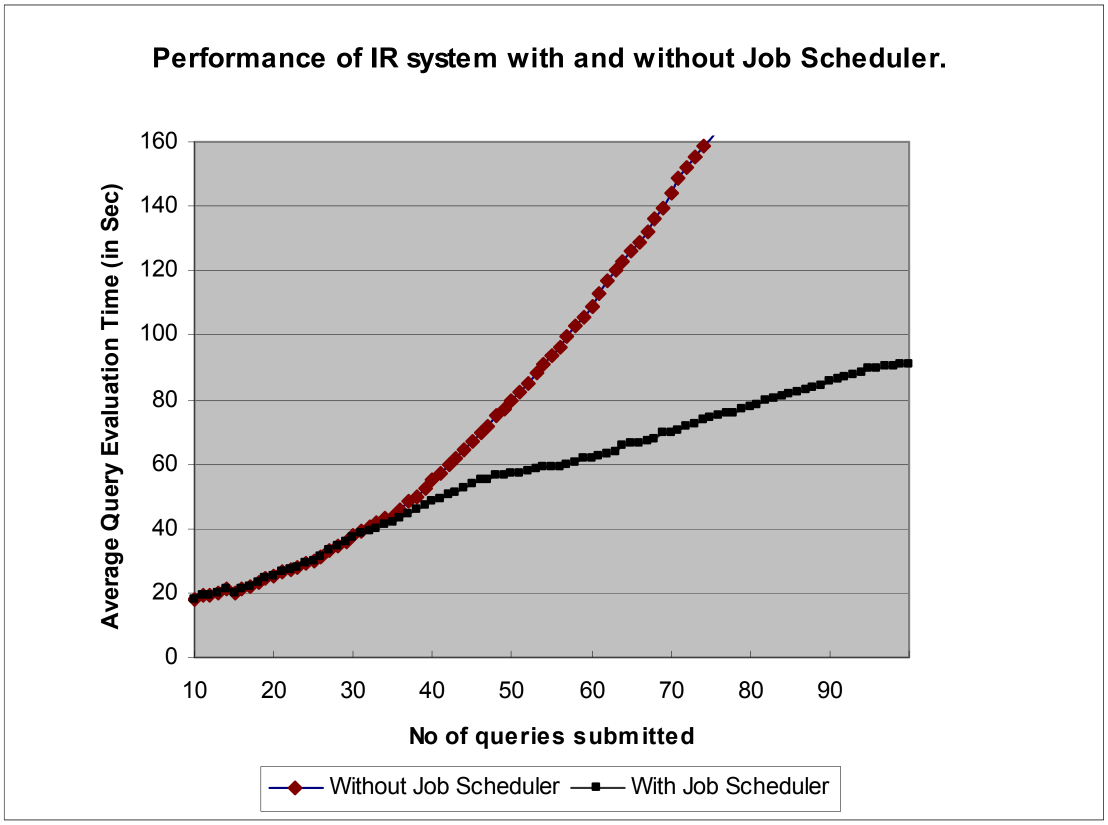


Figure 7.2 Effect of Job Scheduler on the performance of an IR system.

When a request is submitted to grid middleware, Globus returns an error if the arguments are not valid or if there are no resources available. Otherwise, Grid middleware Submits jobs to SGE after building the job wrapper script by getting the necessary information from the RSL (Resource Specification Language) variables Poll. This poll links the present status of jobs running in SGE with the Globus appropriate message allowing the user to know the status of running jobs parsing the qstat.

Figure 7.2 shows the performance difference between the systems with and without using job scheduler. The number of queries submitted to the IR system is plotted against X-axis and Average QET is plotted against Y-axis. The experiment started by submitting 10 queries simultaneously; both the systems took exactly the same amount of time, i.e. 18 seconds to retrieve the results. As the number of queries submitted to the system was increased, retrieval time also increased. This increased the load on the head node on the system that was not using the Job Scheduler. All the requests were served by the head node, which increased the CPU utilization of head node. Once the CPU reaches its 100 % CPU utilization the requests are either trashed or added into the RequestQueue which lead to an exponential increase in the retrieval time. Whereas the system using the Job Scheduler was never serving the requests by itself, instead it was passing the requests to job scheduler to handle them for it. Job scheduler is responsible to balance the load among the nodes which are idle at that point and distribute among them. Performance of the two systems was same until the root node's CPU utilization reached its threshold. Afterwards, the time taken to finish the jobs by the system without the job scheduler increased exponentially. However, the one with the job scheduler divides the jobs to the nodes present in the Grid, and the graph shows the linear increase in the retrieval time in this case. This leads us to an interesting fact of performance of a system where the job scheduler depends on the number of nodes attached to the system. The following experiment tries to find out the effect of the number of nodes on the performance of the system.

7.2.3. Effect of number of Nodes in the Grid on the Performance of the system

The Hydra Grid consists of 30 systems connected to the cluster. In order to use only 5, 10 or 20 of the systems, I created Grinder [71] scripts which overload the CPUs. These scripts create a heavy load on CPUs and by setting the threshold on the CPU utilization. One can exclude the nodes from being used by the scheduler once they reach the threshold. The following figure describes the behavior of the performance of the IR system with change in the number of nodes connected to the Grid. The number of queries submitted to the IR system is plotted on the horizontal axis and the Average QET is plotted on the vertical axis.

Figure 7.3 indicates the relationship between queries submitted versus average query evaluation time of an IR system with varying number of nodes (5, 10, 20 and 30 nodes are connected to Grid). The experiment started with one query and noted the query retrieval time while varying the number of nodes connected to the system. It continued by increasing the number of queries submitted to the system by one and recorded the values till the number of queries submitted to the system reached 90.

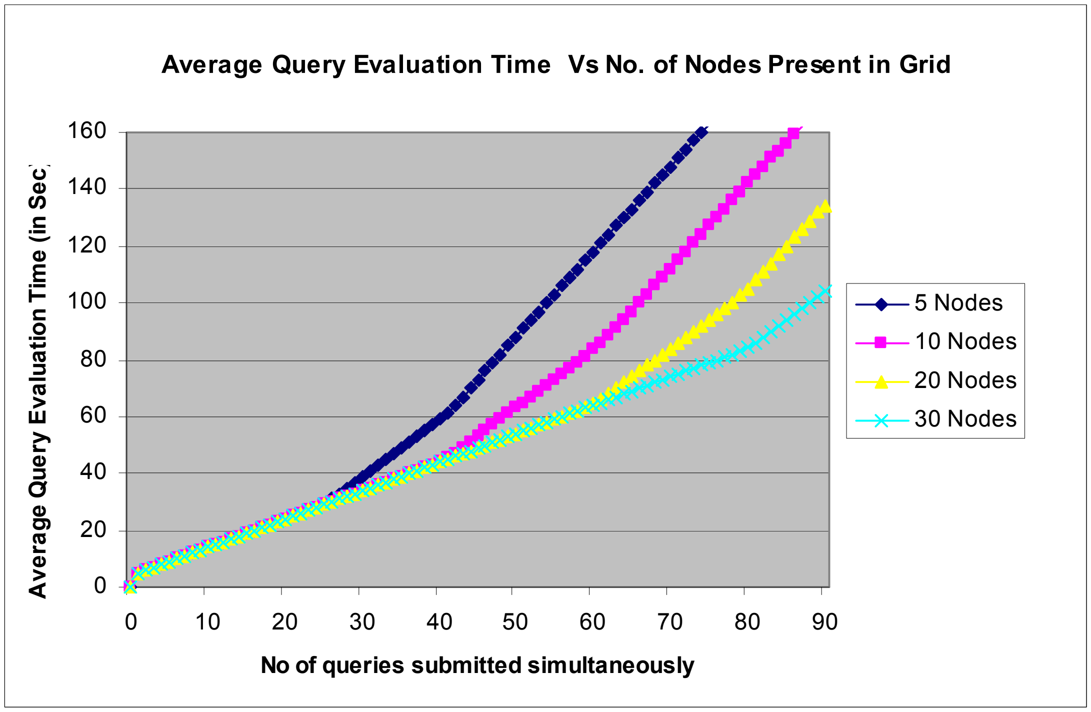


Figure 7.3 Effect of the number of nodes connected to the Grid on the performance of an IR system.

The results were consistent with the working of the Grid Service. First, the request goes to Grid middleware which in turn polls the SGE to determine whether it can provide any resource to perform the operation. As soon as it finds a resource, ServiceDispatcher will pass the request to SGE, which again passes to the respective available resource for serving. All systems, each having a different number of nodes, served the requests well until one reached a point where the SGE could find a resource for serving the request and add it to the WaitingQueue in SGE. This resulted in adding waiting time to the retrieval time. In the experiment, all cases have the same performance until the number of queries has reached 25, after which the retrieval time for 5 nodes increased exponentially. The same trend continued for 10 and 20 until the number of queries reached 39 and 59 respectively. This experiment shows how adding extra nodes to an existing Grid increases the performance of the system. It showed that the number of nodes connected to the Grid is indirectly proportional to the Average QET. Thus, the number of nodes increases as the Average QET decreases.

7.2.4. Amberfish as standalone application vs. Amberfish as Web Service

The main focus of this experiment was to compare the query results retrieval time of the Amberfish as a standalone application compared to Amberfish as a Web Service. This will help us in evaluating the overhead of implementing Amberfish as a Web Service as opposed to a standalone application.

This experiment was conducted on a single node eliminating the overhead of job scheduler and the communication between the nodes using Grid middleware. As a result, performance is totally dependent on the power of the CPU on which the service is hosted or the application is run. As the terms per query increases, the average QET for both standalone application and the Web Service increases.

The experiment started by submitting one query to both the systems. Here, the number of terms per query is plotted on the horizontal axis and the Average QET is plotted on the vertical axis. The standalone application took 2 seconds to retrieve the result where as Web Service took 8 seconds. The difference of 6 seconds is considered to be an overhead of the Web Service. As the number of terms per query increased, the difference between the retrieval times between the two systems also increased. It was observed that

Amberfish as a standalone application always takes lesser time when compared to the Amberfish Web Service. The results of the experiment are shown in figure 7.4.

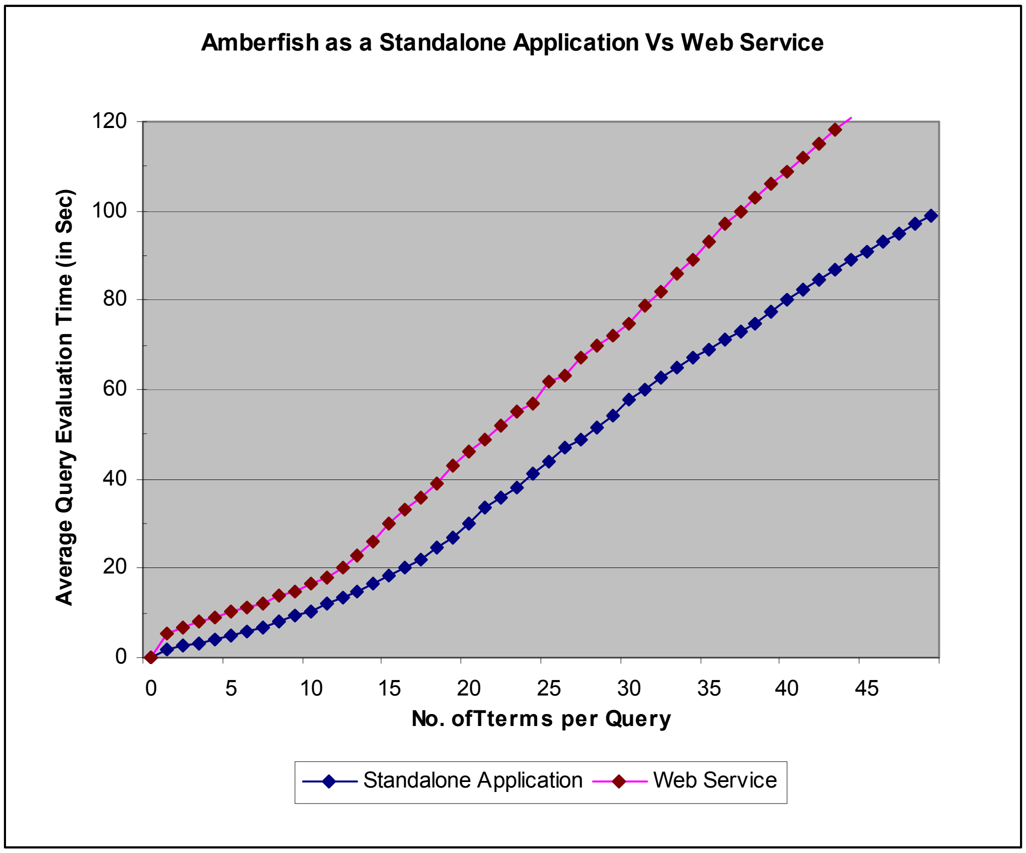


Figure 7.4 Effect of the number of nodes connected to the Grid on the performance of the IR system.

Considering the implementation details of Amberfish as Web Service, the only thing that differentiates it from a standalone application is the overhead of technologies like Soap, WSDL, and XML used by Web Services for communication purposes. This raises the question of why we need a Web Service when we already have a standalone application which is taking less time to finish the same job. The answer to this question lies in the overall ease of assembling systems from Web Services. Section 3.2.3 describes the comparison between Web Services and Standalone applications.

7.2.5. Query Evaluation Measurements

A query operation consists of creating a query from the input provided by the user, evaluating the query, and ranking the documents that match the query.

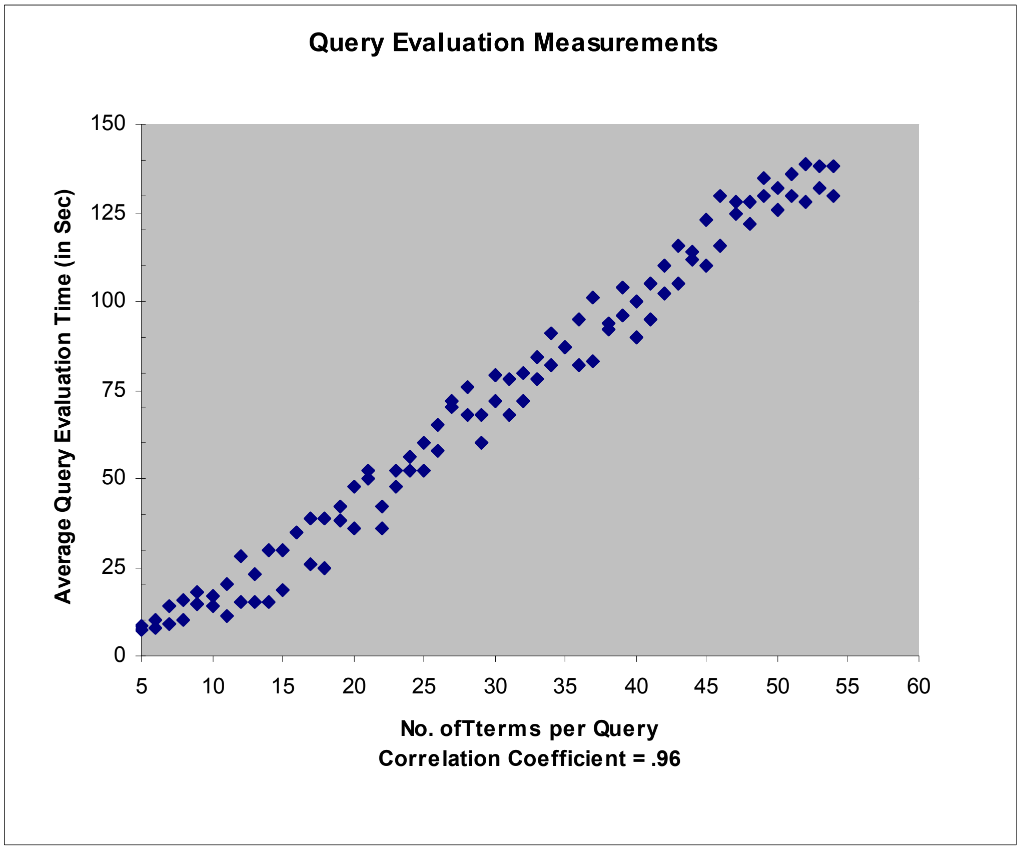


Figure 7.5 Relationship between terms per query and Evaluation time.

The process is very complex so I conducted experiments for the time required to evaluate a query and found that the query evaluation time is directly related to the number of terms, constraints on the query as shown below. The experiment is conducted on the set of 10,000 documents.

While conducting experiments, I considered the set of queries with different number of query terms to evaluate the QET. Figure 7.5 graphs the relation between the length of the query and query evaluation time. This considers all the query terms to be different from each other, a lack of network latency, and a constant network capacity during the experiment. The result indicates a strong linear association between query length and query evaluation time with very high correlation, value 0.96.

7.2.6. Server Connection and Performance

The network capacity between the client and the server hosting the IR service in a Grid is also an important factor to consider for analyzing the performance of a system. We can eliminate the connection establishment time and only consider the throughput time since the connection is created only at the beginning of a session and maintained throughout the session. When the server receives a message, it performs two operations. Merging the result sets and sending the results back to the client. Time to send the results back to the client is constant but varies from connection to connection. The time required to merge result sets was shown in the previous experiments to be a linear function of the size of the data (retrieved from the indexers).

8. CONCLUSIONS AND FUTURE WORK

The architecture presented in this thesis is intended for technically skilled people mainly researchers, who don't want to re-invent the realm the fundamentals of IR, and at the same time want to use modern and capable code base in their own services.

8.1.Conclusions

The amalgamation of Web Services and Grid Computing offers a good architecture for the evolution of new methods of Information Retrieval. It offers a various advantages including the information retrieval from non-publicly available documents within a Grid. By using power of Grid technologies and adding modularity to the design of Web Services, offers new techniques and powerful information retrieval systems.

We envision a future where people implement their own IR systems that are trained to their own information needs, preferences and interests. Implementing Web Services for each individual components of the IR system enables building each component separately with later easy integration. Also the existing algorithms can be replaced with other state of the art algorithms for each component without changing the design of the system.

The major advantage of this system is that it reduces the overall cost for building an IR system using expensive servers or clusters. Currently existing systems can be added as nodes in a virtual organization, using the power of Grid Computing technology.

Apart from all of these, IR system in the Grid Environment provides dynamic load balancing mechanism unlike the conventional system, by using under utilized computing machines connected to the virtual organization.

8.2.Future Work

The system needs to be evolved with the new releases of future versions of the IR system with roles defined accurately for each component. At present there is an ambiguity in defining the roles for all component of the system. We want to concentrate on defining clear definitions to each interface and component. Also, web-based job submission services need to be added for searching the information in Grids.

It is possible to take advantage of maintaining the statefulness of Grid Services to implement grid elements that have knowledge of the history of users and their needs. Taking these into consideration and using this information in merging and ranking algorithms can make the system intelligent.

All the security features mentioned in the section 5.2. are not implemented in a prototype model, but would like to implement these in future versions of the system

Presently the system is still in a prototype model. A future goal is integrating all components into a single package to provide it as software. The current system has the capability of searching only in a grid; it would be better to have a web-search feature available in the system so that it merges the results from grid with the web-based search results.

As the prototype uses the Amberfish (text retrieval tool for UNIX environment) as an indexing tool, it can only index on UNIX based machines now; it would be better of it were platform independent.

The success described above in, prototyping an IR system in a Grid Environment under the current assumptions provides great motivation for such future work.

REFERENCES

REFERENCES

[1] Ian Foster, Carl Kesselman and Steven Tuecke, *Anatomy of the Grid*, ANL, ISI, 2001.

[2] Fran Berman Geoffrey Fox Tony Hey*, The Grid: Past, Present, Future*, 2003.

[3] Ian Foster, Carl Kesselman Jeffrey M. Nick and Steven Tuecke, *The Physiology of the Grid*, ANL, ISI and IBM, 2002.

[4] Ian Foster, Carl Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, Calif., 1998.

[5] David De Roure, Mark A. Baker, Nicholas R. Jennings and Nigel R. Shadbol, *The Evolution of the Grid*, Universities of Portsmouth and Southampton, UK, 2003.

[6] D. Berry, A. Djaoui, A. Grimshaw, B. Horn, *Open Grid Services Architecture*, Ian Foster, Argonne, H. Kishimoto, Fujitsu and A. Savva, Fujitsu, 2005

[7] Joshy Joseph and Craig Fellenstien, *Grid Computing – On Demand Series*, Book, Prentice Hall Professional Technical Reference, 2004.

[8] Ian Foster, *The Grid: A New Infrastructure for 21st Century Science*, 2002.

[9] Malcolm Atkinson, *Towards a Grid Architecture Roadmap*, The UK e-Science Architecture Task Force, 2002.

[10] Dr. Rayburn, *7 Things to Know About Grid Computing,* Educause Learning Initiative, 2006

[11] David De Roure, Nicholas Jennings and Nigel Shadbolt, *The Semantic Grid: A Future e-Science Infrastructure*, University of Southampton, UK, 2004.

[12] Reagan Moore and Chaitan Baru, *Virtualization Services for Data Grids*, San Diego Supercomputer Center, University of California, San Diego, 2003

[13] Geoffrey Fox, Dennis Gannon and Mary Thomas, *Overview of Grid Computing Environments*, GGF, 2002

[14] Julian Bunn and Harvey Newman Caltech, *Data Intensive Grids for High Energy Physics*, California Institute of Technology, 2003.

[15] Book (Collection of papers), *Grid Computing – Making the Global Infrastructure a Reality*, Edited by F. Berman, G. Fox and T. Hey, 2002.

[16] Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L. and White, *The Sourcebook of Parallel Computing*, San Francisco, Morgan Kaufmann Publishers, 2002.

[17] N. Nassar, G. Newby, K. Gamiel, M. Dovey and J. Morris, *Grid Information Retrieval Architecture*, GGF Grid Information Retrieval Working Group, 2004.

[18] N. Nassar, G. Newby, K. Gamiel, M. Dovey and J. Morris, *Grid Information Retrieval Requirements*, GGF Grid Information Retrieval Working Group, 2003-04.

[19] Matthew J. Dovey, Kevin Gamiel, *GRID Information Retrieval*, Oxford e-Science Centre and MCNC, 2001.

[20] Matthew J. Dovey, *Music GRID – a Collaborative Virtual Organization for Music Information Retrieval Collaboration and Evaluation*, Oxford University, 2002.

[21] Nicholas J. Belkin, W. Bruce Croft, *Information filtering and information retrieval: two sides of the same coin?*, ACM Press, 1992.

[22] Marios D. Dikaiako, Rizos Sakellario and Yannis Ioannidis, *Information Services for Large-Scale Grids - A Case for a Grid Search Engine*, Institute on Knowledge and Data Management, 2005.

[23] Hechuan, Dubin, and Sanli Li, *Grid Services Performance Tuning in OGSA*, Grid Research Group, Tsinghua University, 2003.

[24] Nassib Nassar, *The Amberfish Text Retrieval System*, Etymon Systems, Inc, 2006.

[25] Michel Jouvin and Irakli Mandjavidze, *A Hierarchical GRID Information Service based on MDS 2.1*, IN2P3 LAL, CEA Saclay, France, 2006.

[26] Marios Dikaiakos, Yannis Ioannidis and Rizos Sakellariou, *Search Engines for the Grid: A Research Agenda*, CrossGrid project, European Union, 2001.

[27] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes and Jun Zhang, *Similarity Search for Web Services*, University of Washington, Seattle, 2004.

[28] Christopher T. Fallen and Gregory B. Newby, *A Result-set Merging Experiment*, Arctic Region Supercomputing Center, 2006.

[29] Christopher T. Fallen and Gregory B. Newby, *Distributed Web Search Efficiency by Truncating Results*, Arctic Region Supercomputing Center, 2006.

[30] Ian Foster, Karl Czajkowski, Donald F. Ferguson, Jeffrey Frey, Steve Graham, Tom Maguire, David Snelling and Steven Tuecke, *Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF*, Proceedings of the IEEE, vol. 93, no.3, 2005.

[31] Malcolm Atkinson, David DeRoure, Alistair Dunlop, Geoffrey Fox, Peter Henderson, Tony Hey, Norman Paton, Steven Newhouse, Savas Parastatidis, Anne Trefethen, Paul Watson, and Jim Webber, *Web Service Grids: An Evolutionary Approach*, TAG revised version, 2004.

[32] Steffen Staab, *Web Services: Been There, Done That?*, University of Karlsruhe, IEEE Intelligent Systems, 2003.

[33] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley Longman Publishing Co. Inc., 1999.

[34] Sloot Peter, Hoekstra G. Alfons, Priol Thierry, *Advances in grid computing -- EGC 2005*, European Grid Conference, Amsterdam, The Netherlands, 2005

[35] Silva Vladimir, *Grid computing for developers*, Hingham, Mass., Charles River Media, 2006.

[36] R. R. Larson, J. McDonough, P. O'Leary, L. Kuntz, and R. Moon. Cheshire II, *Designing a next-generation online catalog. Journal of the American Society for Information Science*, 47(7):555–567, 1996.

[37] Pratap Pattnaik, Kattamuri Ekanadham and Joefon Jann, *Autonomic Computing and GRID*, IBM Research, 2004.

[38] Christos Faloutsos and Douglas W. Oard, *A Survey of Information Retrieval and Filtering Methods,* University of Maryland, College Park, 1995.

[39] Mary Thomas Texas and Jay Boisseau, *Building Grid Computing Portals: The NPACI Grid Portal Toolkit,* Univ. of Texas, 2003.

[40] Clifford A. Lynch, *Measuring system performance and performance analysis in public access information retrieval systems*, Pages: 177 - 183, American Library Association, 1998.

[41] Fidel Cacheda, Victor Carneiro, Vassilis Plachouras and Iadh Ounis, *Performance Comparison of Clustered and Replicated Information Retrieval Systems*, Yahoo Research, 2007.

[42] J.R. Files and H.D. Huskey, *An information retrieval system based on superimposed coding,* Proc. AFIPS FJCC 35:423-432, 1969.

[43] M.C. Harrison, *Implementation of the substring test by hashing,* CACM, 14(12) 777-779, 1971.

[44] C.S. Roberts, *Partial match retrieval via the method of superimposed codes,* Proc. IEEE, 67(12), 1979.

[45] J.J. Rocchio, Performance indices for document retrieval In G. Salton editor, *The SMART Retrieval System - Experiments in Automatic Document Processing,* Prentice Hall Inc, EnglewoodCliffs, New Jersey, 1971

[46] R. Sacks-Davis and K. Ramamohanarao, *A two level superimposed coding scheme for partial match retrieval,* Information Systems, 1983

[47] Iadh Ounis, Maarten de Rijke, Craig Macdonald, Gilad Mishne, Ian Soboroff, *Overview of the TREC-2006 Blog Track,* TREC, 2006

[48] National Information Standards Organization, *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification,* An American National Standard, 2002.

[49] B. Yagoubi, Y. Slimani, *Task Load Balancing Strategy for Grid Computing*, Journal of Computer Science 3 (3): 186-194, 2007

[50] D. Carrera, J. Guitart, J. Torres, E. Ayguadé, and J. Labarta, *Complete instrumentation requirements for performance analysis of web based technologies*, IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). Austin, USA, March 2003.

[51] K. Yaghmour and M. R. Dagenais, *Measuring and characterizing system behavior using kernel-level event logging*, Usenix Annual Technical Conference, San Diego, CA, June 2000.

[52] G. Jost, H. Jin, J. Labarta, J. Gimenez, and J. Caubet, *Performance analysis of multilevel parallel applications on shared memory architectures,* 2003.

[53] Ramon Nou, Ferran Julià, David Carrera, *Monitoring and analysing a Grid middleware Node*, Computer Architecture Departmenr, Technical University of Catalonia, Spain

[54] Helen D. Karatza, *Performance Analysis of Gang Scheduling in a Distributed System under Processor Failures,* Department of Informatics Aristotle University of Thessaloniki, 2001

[55] Helen D. Karatza and Ralph C. Hilzer, *Performance Analysis of Parallel Job Scheduling in Distributed Systems*, Proceedings of the 36th Annual Simulation Symposium, 2003

[56] J.J. Rocchio, *Performance indices for document retrieval In G. Salton editor, The SMART Retrieval System - Experiments in Automatic Document Processing,* Prentice Hall Inc, EnglewoodCliffs, New Jersey, 1971

[57] Grid Information Retrieval Working Group
http://www.gir-wg.org/

[58] Etymon Systems – Text Retrieval System (Amberfish).
http://www.etymon.com/tr.html

[59] Information Retrieval Reference Book
http://www-csli.stanford.edu/~schuetze/information-retrieval-book.html

[60] Wikipedia - Information Retrieval
http://en.wikipedia.org/wiki/Information_retrieval

[61] Globus Alliance
http://www.globus.org

[62] SOA and Web Services Architecture
http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html

[63] Microsoft - SOA Overview
http://www.microsoft.com/biztalk/solutions/soa/overview.mspx

[64] W3School Organization – Web Services
http://www.w3.org/TR/ws-arch/

[65] W3School Organization - WSDL
http://www.w3.org/TR/wsdl20/

[66] Web Services Resource Framework - OASIS
http://www.globus.org/wsrf/

[67] W3School Organization - XML
www.w3.org/XML/

[68] SUN - SOA and Web Services
http://java.sun.com/developer/technicalArticles/WebServices/soa/

[69] IBM – Web services and SOA architecture.
http://www.ibm.com/developerworks/webservices/library/ws-soa-messagingstandard/?S_TACT=105AGX04&S_CMP=HP

[70] Sun Grid Engine
http://www.sun.com/software/gridware/

[71] Grider
http://grinder.sourceforge.net/

[72] The Eight Fallacies
http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing

[73] Sun Grid Engine Scheduling Algorithms
http://bioteam.net/dag/gridengine-6-features.html

CURRICULUM VITAE

Sairam Chilappagari attended the Jawaharlal Nehru Technological University, Hyderabad, India where he received his Bachelors of Technology in Computer Science with honors, in May 2005. He started graduate studies at George Mason University, Fairfax, USA in Fall 2005. He started working as a Research Assistant in Netlab, C4I center from Dec, 2005 and received OGF-19 Student Scholarship for doing the research in the field of Grid Computing. He received his Master of Science in Computer Science in August 2008.