Using Taxonomic Indexing Trees to Efficiently Retrieve SCORM-compliant Documents in e-Learning Grids

Wen-Chung Shih
Department of Computer Science, National Chiao Tung University, Taiwan // gis90805@cis.nctu.edu.tw

Shian-Shyong Tseng
(Corresponding author) Department of Computer Science, National Chiao Tung University, Taiwan // sstseng@cis.nctu.edu.tw // Department of Information Science and Applications, Asia University, Taiwan // sstseng@asia.edu.tw

Chao-Tung Yang
Department of Computer Science and Information Engineering, Tunghai University, Taiwan // ctyang@thu.edu.tw

ABSTRACT

With the flourishing development of e-Learning, more and more SCORM-compliant teaching materials are developed by institutes and individuals in different sites. In addition, the e-Learning grid is emerging as an infrastructure to enhance traditional e-Learning systems. Therefore, information retrieval schemes supporting SCORM-compliant documents on grid environments are gaining its importance. To minimize the query processing time and content transmission time, our idea is to use a bottom-up approach to reorganize documents in these sites based on their metadata, and to manage these contents in a centralized manner. In this paper, we design an indexing structure named Taxonomic Indexing Trees (TI-trees). A TI-tree is a taxonomic structure and has two novel features: 1) reorganizing documents according to the Classification metadata such that queries by classes can be processed efficiently and 2) indexing dispersedly stored documents in a centralized manner which is suitable for common grid middleware. This approach is composed of a Construction phase and a Search phase. In the former, a local TI-tree is built from each Learning Object Repository. Then, all local TI-trees are merged into a global TI-tree. In the latter, a Grid Portal processes queries and presents results with estimated transmission time to users. Experimental results show that the proposed approach can efficiently retrieve SCORM-compliant documents with good scalability.

Keywords

e-Learning, SCORM, Grid computing, Globus Toolkit, Information retrieval

Introduction

Due to the advances in information technologies and the requirements of courseware, more and more teachers are able and willing to design their own teaching materials and make them accessible on the Web (Iorio, Feliziani et al., 2006; Lanzilotti, Ardito et al., 2006; Sierra, Fernández-Valmayor et al., 2006). In addition, a growing number of large-scale projects aim to construct learning content repositories (Kassahun, Beulens et al., 2006; Kiu & Lee, 2006). For example, in 2002, the National Science Council of Taiwan approved a resolution on the "National Science and Technology Program for e-Learning," planning to spend $120 million within a 5-year period. These educational contents are mainly based on Sharable Content Object Reference Model (SCORM), which has become a popular standard for creating sharable and reusable teaching materials for e-Learning. With the popularization of e-Learning, how to find and reuse these existing materials becomes an important issue.

Grid computing systems (Foster & Kesselman, 1997; Foster, 2002) are transparent resource-sharing infrastructures, which can overcome the limitations in traditional e-Learning platforms, such as scalability, interoperability, availability, etc. Also, grid computing technologies provide possibilities for supporting innovative applications of e-Learning. For example, a medical college can provide students with three-dimensional simulation of human body anatomy using high performance grid computing systems, which is beyond the ability of traditional e-Learning platforms. Therefore, more and more effort has gone into the field of e-Learning grid, using grid technologies in the context of e-Learning. Among these, ELeGI (European Learning Grid Infrastructure, 2004-2008) is the most representative project with respect to e-Learning Grid (Gaeta, Ritrovato et al., 2003). Its goal is to address and advance current e-Learning solutions through use of geographically distributed resources as a single e-Learning environment.

With the promising development of e-Learning grid, there will be a great demand to find desired teaching materials from multiple repositories in the e-Learning grid. A traditional approach is to implement a meta-search engine on top of these distributed repositories (Yu, Liu et al., 2002). When the meta-search engine receives a query, it distributes the query to the local repositories, and then collects the returned results and presents them to users. However, this traditional approach doesn't consider characteristics of SCORM and grid computing to speed up the retrieval process. For example, SCORM-compliant documents are associated with nine categories of metadata. When processing a query with a restrictive filter, such as "documents about insects," the meta-searching approach will search the whole databases. In fact, a better approach is to search only the "insects" category, which will obviously reduce the searching time. For another example, because common grid middleware is implemented with a tightly-coupled master-slave model, it is efficient to use centralized management schemes. The traditional meta-searching approach is not designed for this model, and is inefficient in the process of query dispatching and results collecting, which involves synchronization and communication overhead. Hence, there is an urgent need to design a tailor-made approach to be suitable for e-Learning grids.

The goal of this work is to minimize the time of query processing and content transmission when retrieving SCORM-compliant documents in a grid. To speed up the searching process, our idea is to use a centralized index, which is generated by reorganizing the existing documents based on a bottom-up approach, because this approach is suitable for the master-slave grid model and can effectively collect the information of existing documents from all sites in the grid. Furthermore, the indexing structure stores metadata and structural information, which increases the efficiency and precision of searching. To speed up the transmission process, the other idea is to present the ranked results with estimated transmission time, which is derived from grid monitoring tools. In this way, the document which has high ranking score but has a long estimated transmission times (maybe due to a low-bandwidth link) can be avoided by users.

In this paper, we address the problem of retrieving SCORM-compliant documents on e-Learning grids. To efficiently manage documents in the grid, we have designed an indexing structure named Taxonomic Indexing Trees (TI-trees). A TI-tree is based on an existing taxonomic schema and has two novel features: 1) reorganizing documents according to the Classification metadata such that queries by classes can be processed efficiently and 2) representing each document by a term-weighting vector, where the term-weighting includes structural information. In this way, an appropriate weighting scheme can be used to utilize the structural information in the SCORM-compliant documents. In addition, the cost of constructing, merging and maintaining TI-trees is not expensive, but the benefits are significant. The overall process of this approach is composed of a Construction phase and a Search phase. In the former, a local TI-tree is built from each Learning Object Repository. Then, all local TI-trees are merged into a global TI-tree. In the latter, a Grid Portal processes queries and presents results to users. After the user specifies the desired documents, the Portal retrieves this document from the target site for the user.

The primary contribution of this paper is the design of an efficient approach to retrieving SCORM-compliant documents on grid environments. To the best of our knowledge, this topic has not been addressed in previous work. Second, real-time information of network bandwidth is taken into consideration for estimation of transmission time on grid environments. Finally, we have built a prototype on a grid test-bed, and experimental results show that this approach is scalable with respect to storage requirements and time complexity.

Preliminaries and Related Work

In this section, we review background knowledge about SCORM, Grid computing and information retrieval. Then, related researches about this work are surveyed.

Sharable Content Object Reference Model (SCORM)

To share and reuse teaching materials, several standards have been proposed recently. Among these, SCORM (http://www.adlnet.org/) is the most popular standard for learning contents. It was proposed by the U.S. Department of Defense's Advanced Distributed Learning (ADL) organization in 1997. This standard consists of several specifications developed by IEEE LTSC (Learning Technology Standards Committee, http://ltsc.ieee.org/wg12/), IMS (Instructional Management System, http://www.imsproject.org/), AICC (Aviation Industry CBT Committee,

http://www.aicc.org/), etc. The SCORM specifications are a composite of several specifications developed by international standards organizations. In a nutshell, SCORM is a set of specifications for developing, packaging and delivering high-quality education and training materials whenever and wherever they are needed (Nitto, Mainetti et al., 2006; Su, Tseng et al., 2006). In SCORM, content packaging scheme is proposed to package the learning objects into standard teaching materials. As shown in Figure 1, the content packaging scheme defines a teaching materials package consisting of four components: 1) Metadata, which describes the characteristics or attributes of this learning content; 2) Organizations, which describe the structure of the teaching material; 3) Resources, which denote the physical files linked by each learning object within the teaching material; and 4) the (Sub) Manifest, which describes this teaching material, consisting of itself and other teaching materials.

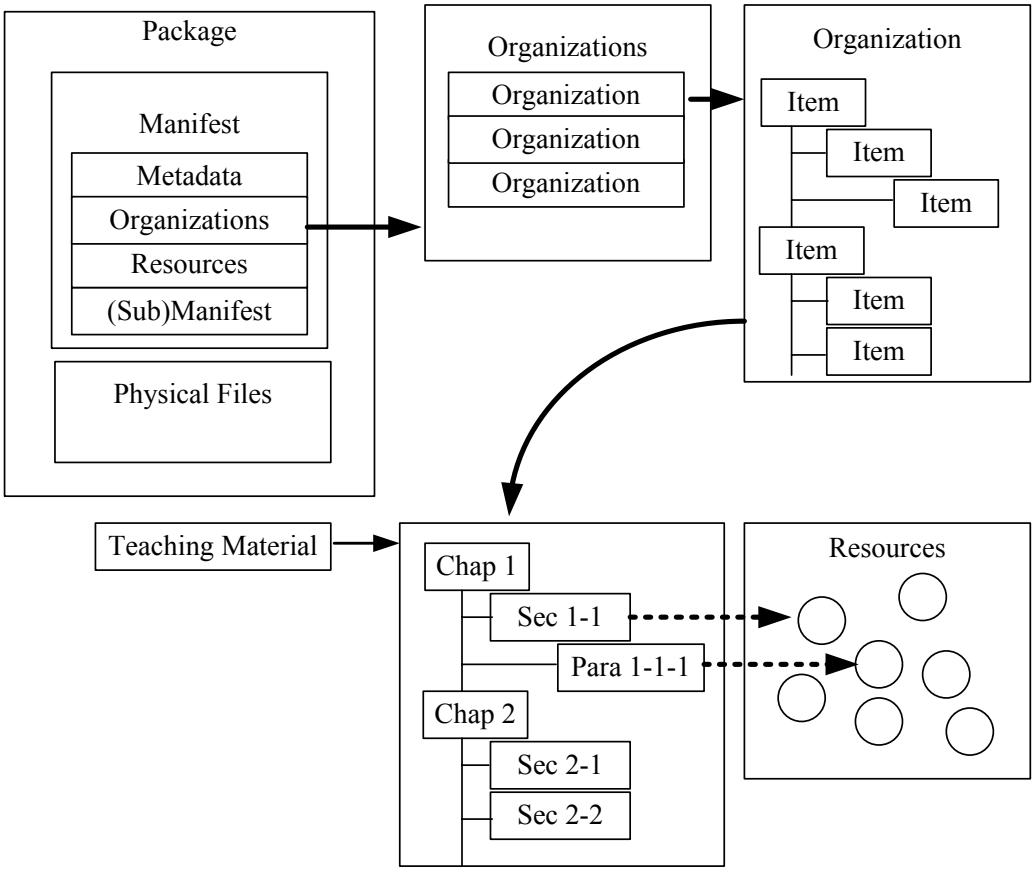

Figure 1. Structure of a SCORM Content Package

SCORM Metadata refers to the IEEE's Learning Object Metadata (LOM), and describes the attributes of teaching materials. IEEE LOM v1.0 includes nine categories: General, LifeCycle, Meta-Metadata, technical, educational, rights, relation, annotation, and classification. In this paper, only the classification category is employed for storage and retrieval of documents. Specifically, the following metadata in this category is used:

- 9.2.1 source: name of classification schema
- 9.2.2.1 id: name of a class
- 9.2.2.2 entry: code of a class

Su et al. proposed a level-wise approach to SCORM content management (Su, Tseng et al., 2005). In this two-phase scheme, structural information is considered. Also, both general and specific Learning Objects can be retrieved according to requests of users. However, this structure doesn't consider the SCORM metadata, and seems too complex to be efficiently used in grids.

There have been numerous studies on Structured Document Retrieval (Trotman, 2004; Trotman, 2005). Researches showed that structured searching can increase precision. Previous work mainly addresses XML and SGML documents. Besides, XML Query Languages, such as XIRQL, XQL, etc., were proposed. However, intra-document structural modeling is not suitable for SCORM-compliant documents.

Grid Middleware

The Globus Toolkit developed by the Globus Alliance and many others all over the world is an open source software toolkit used for building Grid systems and applications (Foster & Kesselman, 1997). A growing number of projects and companies are using the Globus Toolkit to unlock the potential of grids for their causes. The Globus Toolkit has become a popular standard for Grid middleware to handle these four kinds of services:

- Resource management: Grid Resource Allocation & Management (GRAM)
- Information Services: Monitoring and Discovery Service (MDS)
- Security Services: Grid Security Infrastructure (GSI)
- Data Movement and Management: Global Access to Secondary Storage (GASS) and GridFTP

GRAM was designed to provide a single common protocol and API for requesting and using remote system resources, by providing a uniform, flexible interface to local job scheduling systems. The Grid Security Infrastructure (GSI) provides mutual authentication of both users and remote resources using GSI (Grid-wide) PKI-based (Public Key Infrastructure) identities. GRAM provides a simple authorization mechanism based on GSI identities and a mechanism to map GSI identities to local user accounts.

MDS was designed to provide a standard mechanism for publishing and discovering resource status and configuration information. It provides a uniform, flexible interface to data collected by lower-level information providers. It has a decentralized structure that allows it to scale, and it can handle static (e.g., OS, CPU types, and system architectures) or dynamic data (e.g., disk availability, memory availability, and loading). A project can also restrict access to data by combining GSI (Grid Security Infrastructure) credentials and authorization features provided by MDS.

GridFTP is a high-performance, secure, and reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet File Transfer protocol.

Grid computing is considered to be an inexpensive and promising alternative to parallel computing, and it extends conventional parallel and distributed computing by utilizing computers on the Internet to compute (Foster & Kesselman, 1997; Foster, 2002). Consequently, the rise of grid computing provides a potential solution to the e-learning. Researchers have proposed to utilize data grid technologies to share learning materials. However, these approaches focused on the infrastructure of e-learning platforms (Yang & Ho, 2005), and did not address the issue of SCORM-compliant content management.

The Ganglia project grew out of the University of California, Berkeley's Millennium initiative (http://ganglia.sourceforge.net/). Ganglia is a scalable open source distributed system for monitoring status of nodes (processor collections) in wide-area systems based on clusters. It adopts a hierarchical; tree-like communication structure among its components in order to accommodate information from large collections of multiple clusters, such as grids. The information collected by the Ganglia monitor includes hardware and system information, such as processor type, load, memory usage, disk usage, operating system information, and other static/dynamic scheduler-specific details.

Information Retrieval

Inverted file indexing has been widely used in information retrieval (Salton & McGill, 1983; Baeza-Yates & Ribeiro-Neto, 1999; Witten, Moffat et al., 1999; Mittal, Krishnan et al., 2006). An inverted file is used for indexing a document collection to speed up the searching process. The structure of an inverted file consists of two components: the vocabulary and the posting list, as shown in Figure 2. The vocabulary is composed of all distinct

terms in the document collection. For each term, a list of all the documents containing this term is stored. The set of all these lists is called the posting list. However, the structure of a document is not considered in this model.

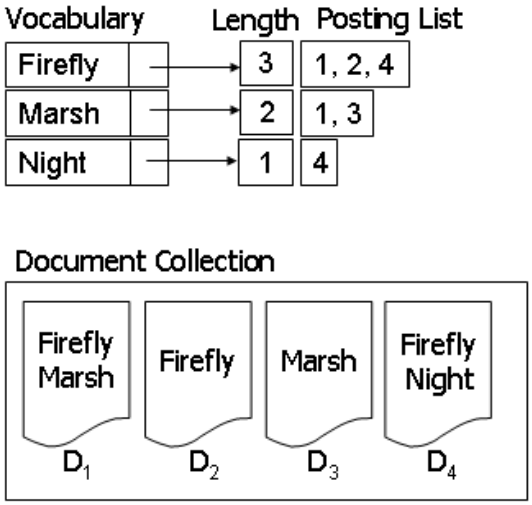

Figure 2. A sample document collection and a corresponding inverted index

Storage requirements of inverted indices (Lee, Yoo et al., 1996) have been evaluated based on B^+-tree and posting list. Five strategies of the index term replication were discussed. This approach is extended to analyze the storage requirement of the proposed approach in this paper. In (Cambazoglu & Aykanat, 2006), 11 different implementations of ranking-based text retrieval systems using inverted indices were presented, and their time complexities were also investigated.

The meta-search approach has been studied in the context of distributed information retrieval (Yu, Liu et al., 2002). This approach consists of Query Distribution and Result Merging phases. Furthermore, the Document Retrieval problem is divided into two sub-problems: Database Selection problem and Document Selection problem. However, in Globus toolkit, a widely-used Grid middleware, centralized indexing is adopted to simplify coordination of simultaneous updates to local indices by different sites.

Related Work

During the past decades, information technologies have advanced at an amazing pace. With the rapid growth of various internet applications, more and more content is available on the web. For example, many large-scale learning object repositories have been built for e-Learning applications, which are open for web-based access. The phenomenon of content overload, referring to the state of having too much content to remain informed about a topic, has emerged as a crucial issue to be addressed. Furthermore, the lack of a method for efficiently retrieving desired content deteriorates this problem.

SCORM-compliant teaching materials can be seen as structured documents. For fast retrieval of information from structured documents, Ko et al. (2002) proposed a new index structure which integrates element-based and attribute-based structure information to represent a document, including three retrieval methods: top-down, bottom-up and hybrid methods. Although this index structure takes information of elements and attributes into account, it is not suitable for management of a huge amount of documents due to its time and storage complexity.

For sharing and reusing teaching materials in different e-learning system, the Sharable Content Object Reference Model (SCORM) has become the most popular international standard among the existing ones. In the LOR, a huge amount of SCORM teaching materials, including associated learning objects, will result in management problems. In (Su, Tseng et al., 2005), a management approach called the *Level-wise Content Management Scheme* (LCMS) was proposed to efficiently maintain, search, and retrieve learning contents from a SCORM compliant LOR. LCMS

includes two phases: the *Construction phase and Search phase*. In the former, the content structure of SCORM teaching materials (Content Package) is first transformed into a tree-like structure, called a Content Tree (CT), to represent each piece of teaching material. Based on Content Trees (CTs), the proposed *Level-wise Content Clustering Algorithm* (LCCAlg) then creates a multistage graph showing relationships among learning objects (LOs), e.g., a Directed Acyclic Graph (DAG), called the *Level-wise Content Clustering Graph* (LCCG). This level-wise approach considers the structural information of SCORM-compliant document for retrieval, but the metadata of content packages has not been utilized to increase its precision. Also, its time and storage complexity has not been addressed in depth.

One of the primary objectives of Semantic Web (Berners-Lee, Hendler et al., 2000), referred to as the next generation Web, is to enable web resources to support ontology-based information retrieval, thus improving its precision by semantic search. However, one important premise to realize the vision of Semantic Web is the publication of substantial Semantic Web content, which requires a tremendous amount of effort for content annotation and ontology building. In (Lee, Tsai et al., 2006), a semantic approach was proposed to use SCORM LOM for the indexing, location, management, and searching of learning objects in a learning object repository. They presented an ontological approach for semantic-aware learning object retrieval. This approach can be embedded to other LOM-based search engines to attain semantic-aware learning object retrieval. However, they focused on ontology-based query expansion while we are focused on content organization and retrieval. Next, they use a fixed ontology developed in their previous research, which can not adapt to the dynamically changing contents in grid environments. Also, the time and storage complexities have not been investigated in their paper.

Bote-Lorenzo et al. (Bote-Lorenzo, Hernández-Leo et al., 2004) addressed two aspects of using educational technology specifications and grid infrastructures. First, they collected collaborative learning designs (scripting) provided by teachers, based on a standard way (IMS-LD). Then, they implemented those units in the form of grid services offered by service providers. In brief, this paper proposed a grid-based CSCL system having these features and showed its applicability by means of a sample collaborative learning scenario. However, their work is focused on scripts of learning design, based on the standard of IMS-LD. Also, learning content management is not the primary issue of that paper.

Table 1. The notation used in this paper

SYMBOL	DESCRIPTION
G	A GRID
N_G	SET OF SITES IN A GRID G
E_G	SET OF COMMUNICATION LINKS IN A GRID G
P_0	MASTER NODE OF A GRID
$BW_I(T)$	BANDWIDTH BETWEEN MASTER AND SITE *I* AT TIME *T*
CP	CONTENT PACKAGE
LOR	LEARNING OBJECT REPOSITORY
V_{CP}	VECTOR REPRESENTING CP
W_I	WEIGHTING ELEMENT I OF CP VECTOR
V	SET OF TERMS IN THE VOCABULARY
CP_J^I	THE *JTH* CP OF LOR *I*
Q	QUERY
V_Q	VECTOR REPRESENTING THE QUERY
C	CLASS ID
$SIM()$	SIMILARITY FUNCTION
N_C	NUMBER OF CLASSES
N_{CP}	NUMBER OF ALL CPS IN THE GRID
N_P	NUMBER OF ALL POSTING ENTRIES IN A TI-TREE
T_I	TIME COST OF STEP *I*
D	DEPTH OF THE NODE IN A TI-TREE
K	NUMBER OF CPS TO BE RETURNED

Approach

In this section, we describe the models of grids and SCORM-compliant documents, and define the problem. Then, the indexing structure of a TI-tree is shown. Finally, the overall process is described. The symbols in Table 1 are used throughout this paper.

Problem Formulation

Typically, grid infrastructure is built with a suite of middleware. Common middleware platforms, such as Globus Toolkits (Foster & Kesselman, 1997) and Condor (http://www.cs.wisc.edu/condor/), are based on a master-slave paradigm. Hence, we represent the grid by a master-slave model. Also, real-time information is included to model the dynamic grid.

Definition 1 (Grid). A grid is a star graph $G = <N_G, E_G>$ that consists of a finite set of node N_G, and a finite set of edges E_G. N_G represents the set of sites in the grid. One node $P_0 \in N_G$ is specified as the master node, and other nodes are slave nodes. Each edge in E_G connects the master node and a slave node. The real-time information of network bandwidth between the Master and Site i at time t is denoted by $BW_i(t)$.

Example 1
An example grid is shown in Figure 3. In this graph, P_0 is the master node and the other 3 nodes, P_1, P_2, and P_3, are slave nodes. In addition, there is a virtual communication link L_i connecting the master node and the slave node P_i. The Grid Monitoring Tool can provide real-time network bandwidth information.

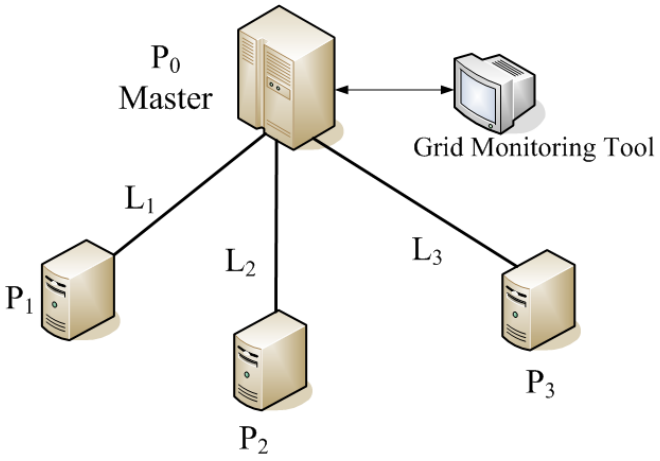

Figure 3. An example grid

In the SCORM standard, a Content Package (*CP*) is defined as a package of learning materials, and a Learning Object Repository (*LOR*) is a database where the *CP*s are stored. In this paper, a *CP* is modeled as a tree to represent the structural information of a CP. To enable content-based retrieval, the well-known Vector Space Model is applied to represent the text content. Also, the Classification metadata is included in this model of *CP* as an attribute to utilize the metadata given by authors. Hereafter, teaching materials, SCORM-compliant documents, and Content Packages are used interchangeably in this paper.

In this paper, the metadata contains only one piece of information: *Classification*. An existing taxonomic schema is assumed, and the value of metadata means the classification ID of the *CP* in this taxonomic schema.

Definition 2 (Content Package). A Content Package (CP) is modeled as a rooted tree, where the leaf nodes contain the content and the internal nodes represent the structural information. The content is represented by a vector. In addition, a CP is associated with a set of Metadata.

Example 2
An example CP is modeled as a trinary tree with three levels, as shown in Figure 4. The leaf nodes contain the content, and the internal nodes represent the structural information. In addition, a CP is associated with a set of Metadata.

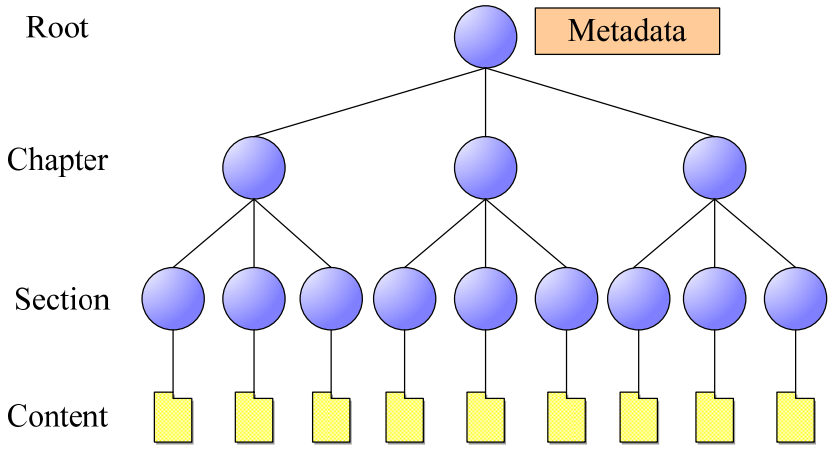

Figure 4. The model of a Content Package (CP)

The *CP* is represented by a term-weighting vector, v_{CP} .

$$v_{CP} = \{w_1, w_2, ..., w_{|V|}\}$$

where *V* means the set of vocabulary and $|V|$ is its size. The term-weighting w_i is evaluated using the extended Vector Space Model proposed by Trotman (Trotman 2005). The idea of this weighting scheme is to emphasize the importance of some structure. For example, the same word appearing in Abstract and the last Chapter of a book has different significance. In this paper, we assume that the first section in each chapter is more important than other sections. Hence, any term appearing in the first section of each chapter will get higher weighting value.

Definition 3 (Learning Object Repository). A Learning Object Repository is a set of Content Packages located in the same site.

Example 3

$$LOR_i = \{CP_1^i, CP_2^i, ..., CP_m^i\}$$

where LOR_i, $1< i < |N_G|$, is the LOR located in the site i of the grid. CP_j^i is the j^{th} CP in LOR_i. Also, each CP can be represented by an ordered tree as the example shown in Figure 4.

We represent the query as a two-tuple: a weight vector and a class id. Its formal definition is as follows.

Definition 4 (Query). A Query is defined by Q= $<v_Q, C>$, where

$$v_Q = \{q_1, q_2, ..., q_{|V|}\}$$

and C is a class ID.

We will now define the notion of similarity between a query and a content package, which means the relevance of the content package to the query.

Definition 5 (Similarity). Let Q be query with query vector v_Q and class C, and CP be a content package. The Similarity sim(Q, CP) is defined by:

$$sim(Q, CP) = v_Q \cdot v_{CP}$$

where the operation is inner product of vectors.

Based on the definitions above, the Grid SCORM Document Search Problem (GSDSP) can be defined as follows.

Definition 6 (GSDRP). Given a query Q = <v_Q, C>, $|N_G|$ learning object repositories, a similarity function sim, and an integer k > 0, the Grid SCORM Document Search Problem is to find the top k relevant CPs, whose class is C, with respect to the query. The criterion is to minimize the query processing time.

Taxonomic Indexing Trees

Taxonomic Indexing Trees are data structures that are designed to support SCORM documents management on grid environments. The main idea is to reorganize SCORM documents according to their associated Metadata, and to utilize the centralized indexing structure for grid environments. The attributes and operations of Taxonomic Indexing Trees are described as follows.

The Dewey Decimal Classification (DDC) system, devised by library pioneer Melvil Dewey in the 1870s and owned by OCLC since 1988, provides a structure for the organization of library collections. Now in its 22nd edition, the DDC is the world's most widely used library classification system (Dewey, 2004). In this paper, we simplify the DDC for the underlying taxonomic schema which the TI-tree is based on.

Definition 7 (Taxonomic Indexing Tree). A Taxonomic Indexing Tree (TI-tree) is a rooted tree having the following properties:

1. Every node x has the following fields:
   x.name: Class Name
   x.id: Class ID
   x.num: Number of Documents
   x.inv: pointer to the Inverted Index
2. Let x be a node in a TI-tree. If y is a child of x, then there exists a relation IS-A between x and y. That is, y IS-A x, which implies that y is a specialization of x.

Example 4
An example TI-tree is shown in Figure 5. The hierarchical relation is based on a taxonomic schema. The root class "Natural Ecology" is divided into two classes: "Terrain" and "Insect." In turn, the two classes are also divided into two sub-classes, respectively. The right child of the root node is "Insect," and its Class ID is ".3." We can see that this node has 2 CPs, and its right child has no CP.

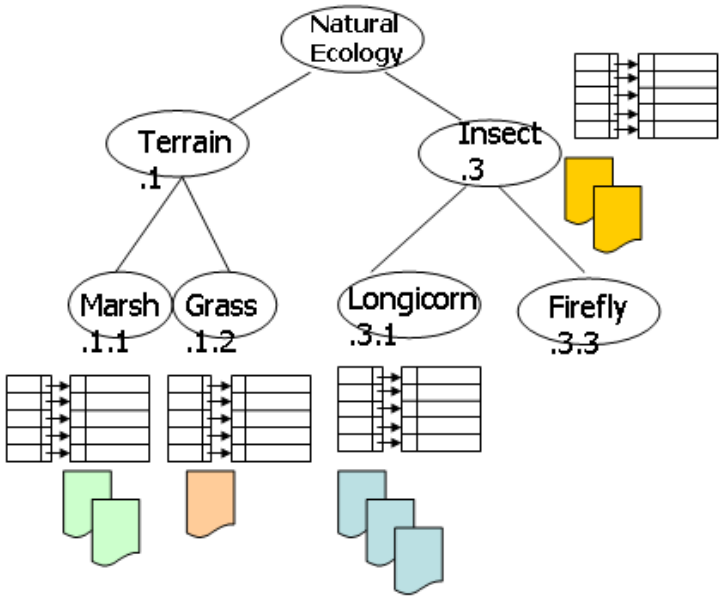

Figure 5. An example of a TI-tree

The operations supported by TI-trees include Searching, Construction, Merging and Insertion, which are described as follows.

. ***Search (*T_0, C*)***
Given a TI-tree T_0 and a class name *C*, this function returns a pointer *x* to a class node in T_0 such that *x.id* = *C*, or *NIL* if no such class belongs to T_0. A common operation performed on a TI-tree is searching for a query in a sub-tree. If the class of the sub-tree is not specified by users, the searching process will start from the root.

. ***Construct (LOR)***
Given a Learning Object Repository *LOR*, a *Construct* operation returns a TI-tree. The resulting TI-tree represents the content packages in the LOR.

. ***Merge (*T_1, T_2*)***
Given two TI-trees T_1 *and* T_2, a *Merge* operation returns a new TI-tree which includes *CP*s of T_1 *and* T_2.

. ***Insert (*T_0, CP*)***
Given a TI-tree T_0 and a content package *CP*, an *Insert* operation inserts the CP into TI-tree T_0.

System Overview

The overall process consists of two phases: Construction Phase and Search Phase, as shown in Figure 6. In Construction Phase, a TI-tree is generated for each local LOR. Then, all local TI-trees are merged into a global TI-tree. Based on the global TI-tree, when users submit their queries, the Grid Portal calculates the similarities between the queries and the global TI-tree. According to the evaluation of similarities, the Grid Portal ranks relevant documents. In addition, estimated transmission time is also shown beside the document. Thus, users can choose suitable documents according to similarity and estimated transmission time.

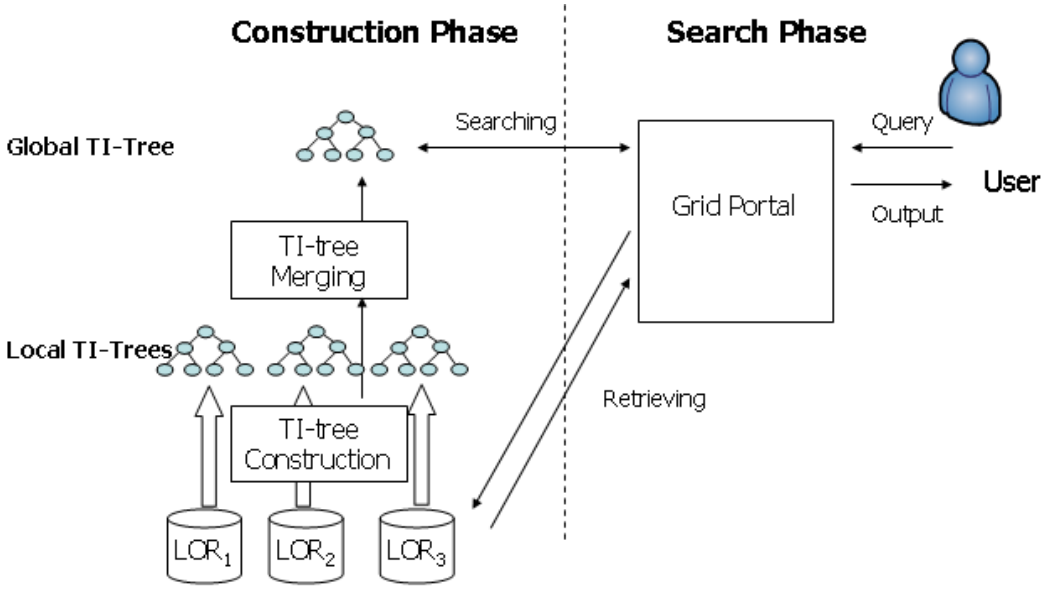

Figure 6. System overview

The Search phase is carried out by the Grid Portal component, which receives queries from users, processes the queries, and presents results to the users. After users specify the desired documents from the returned results, the Portal accesses the site where the documents are stored and retrieves these documents for the users. When a user

submits a query containing terms which do not belong to the Vocabulary, the Portal will suggest some synonymous terms in the Vocabulary. The suggestion is based on a synonym dictionary. Algorithm 1 is used to search the global TI-tree for the top k relevant CPs with respect to a query given by a user.

Algorithm 1. Query Processing

Input:

T_0: the TI-tree to be searched
Q: the query
C: the Class ID associated with the query
k: the number of desired CPs

Output: A sorted list of pointers to the top k relevant CP
Step 1. Search (T_0, C)
Step 2. Scan the term index for each query term.
Step 3. Get the posting lists corresponding to each query term.
Step 4. Scan the posting lists to accumulate similarity scores for each document.
Step 5. Extract the top k documents.
Step 6. Sort the results.

Due to the dynamic grid environments, the network bandwidth might fluctuate. The results are ranked by similarities, and each result is associated with its estimated transmission time. Therefore, the Portal presents the searching results to users with estimated transmission time. The transmission time is estimated by the following formula:

$$T_{trans} = \frac{S_j}{B_i}$$

where T_{trans} is the estimated transmission time, S_j is the storage size of CP j, and B_i is the network bandwidth of Site i.

The Construction phase is carried out in backend, and is transparent to users. First, each site in the grid constructs its TI-tree from the local LOR. Next, this local TI-tree is transferred to the Grid Portal by GridFTP middleware. Then, these TI-trees gathered from all sites are merged into a global TI-tree at Grid Portal.

The operation **Construct (LOR)** is implemented by Algorithm 2.

Algorithm 2. TI-tree Construction

Input:

LOR: the LOR to be converted to a TI-tree

Output: A TI-tree
Step 1. Initialize class nodes of a new TI-tree.
Step 2. For each class node, allocate a term index.
Step 3. Scan the LOR. For each CP, update the attributes, then scan each term and update the postings.

After all the local TI-trees are built and transferred to Grid Port, the global TI-tree is generated as follows, which indexes all the documents in the grid. The operation **Merge (T_1, T_2)** is implemented by Algorithm 3.

Algorithm 3. TI-tree Merging

Input:

T_1, T_2: the two TI-trees to be merged

Output: A TI-tree
Step 1. Initialize class nodes of a new TI-tree.
Step 2. For each class node, allocate a term index.
Step 3. Scan each local TI-tree. Update the attributes of nodes and the postings of the global TI-tree.

The process above is for creating a complete TI-tree on a LOR. However, when a new document is added into a local LOR, it is inefficient to rebuild the whole TI-tree. The operation **Insert (T_0, CP)** is implemented by Algorithm 4.

Algorithm 4. TI-tree Insertion

Input:

T_0: the TI-tree to be inserted to

CP: the CP to be inserted

Output: A TI-tree containing CP

Step 1. Search the local TI-tree for the target class node, and update the posting.

Step 2. Transfer the attributes to the Portal.

Step 3. Search the global TI-tree for the target class node, and update the posting.

Analysis

Analysis of storage requirements and time complexity can help us understand the performance of TI-trees. Because analytical results heavily depend on methods of implementation, fix-length attributes of nodes and index terms are represented by static array, and Posting Lists are implemented with Linked Lists, as shown in Figure 7.

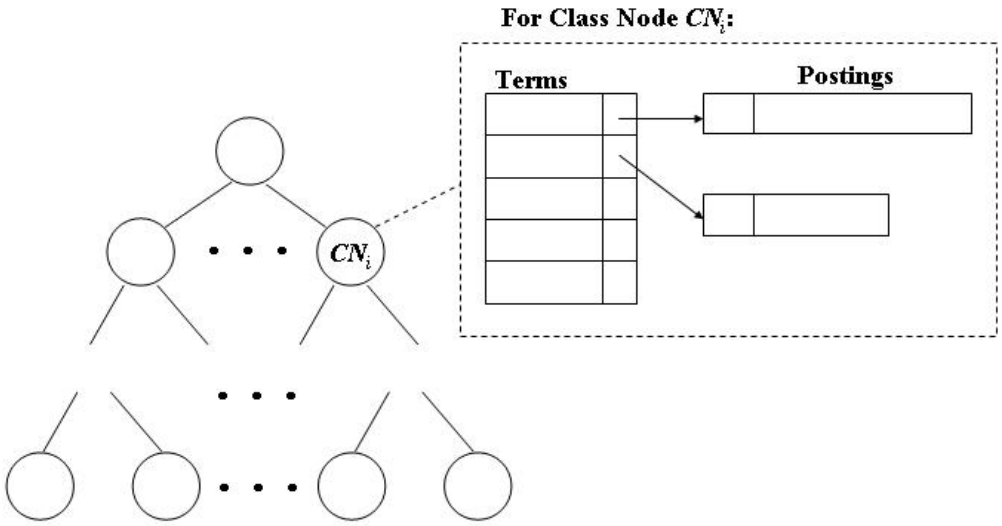

Figure 7. TI-Tree Indexing implemented with array

To simplify our discussion, we assume that the whole structure of a TI-tree is stored in main memory, without disk accesses.

Storage Requirement

The storage requirement of a TI-tree structure, $S_{TI\text{-}tree}$, is the product of the number of class nodes and the sum of storage required for a class node, term index, and posting lists.

$$S_{TI\text{-}tree} = N_C \times (S_C + S_{inv} + S_{post})$$

The number of class nodes, N_C, depends on the assumed taxonomic schema. The storage requirement of fix-length attributes in a class node is represented by S_C. Hence, S_C is a constant.

Term index is straightforwardly implemented with static array.

$$S_{inv} = |V| \times S_{term}$$

where $|V|$ is the number of terms and S_{term} is the space for a term entry.

Posting lists are also implemented with static array.

$$S_{post} = N_p \times S_{ent}$$

where N_p is the number of postings and S_{ent} is the space for a posting entry.

The storage requirement of this implementation is $O(N_C*(S_C+|V|+N_P))$. Assume N_C is a constant, $N_C << |V|$, and $N_C << N_P$. Then, $S_{TI\text{-}tree} = O(|V|+N_P)$, which is asymptotically equal to the storage requirement of the traditional inverted index scheme.

Time Complexity of Search Phase

We denote the time cost of step i as T_i. Costs for query processing are analyzed as follows.
Step 1. A query contains a class ID, which specifies a class node in the TI-tree. To search the global TI-tree for the target class node, a pointer is pointed to the root node, and this pointer traverses the path down to the target node. Let D denote the depth of the target node. $T_1 = O(D)$.
Step 2. Term index is implemented with a static array. To find a term, this array is accessed sequentially from the first element to the next element. In worst case, $T_2 = O(|V|)$.
Step 3. Each term q_j in the query is considered in turn to get the corresponding posting lists. $T_3 = O(|Q|)$.
Step 4. Scan the posting lists to accumulate similarity scores for each document. This step involves N_P reading and writing. $T_4 = O(N_P)$.
Step 5. Extract the top k documents. $T_5 = O(N_{CP})$.
Step 6. Sort the results with array implementation. $T_6 = O(k^2)$.

The running time of this implementation is $O(D + |V| +|Q| + N_P + N_{CP} + k^2) = O(|V| + N_P + N_{CP})$. The time complexity of query processing for the TI-tree approach is $O(|V| + N_P + N_{CP})$, which is linear.

Time Complexity of Construction Phase

Costs for constructing a local TI-tree are as follows.
Step 1. Initialize class nodes of a new TI-tree. Hence, $T_1 = O(N_C)$.
Step 2. For each class node, allocate a term index and posting lists. $T_2 = O(N_C * (|V| + N_P))$.
Step 3. Scan the LOR. For each CP, scan each term and update the postings. $T_3 = O(N_{CP})$.
The running time of this implementation is $O(|V| + N_P + N_{CP})$, which is linear.

Costs for generating a global TI-tree are as follows.
Step 1. Initialize class nodes of a new TI-tree. $T_1 = O(N_C)$.
Step 2. For each class node, allocate a term index and posting lists. $T_2 = O(N_C * (|V| + N_P))$.
Step 3. Scan each local TI-tree. Update the postings of the global TI-tree. $T_3 = O(|N_G| * N_P)$.
The running time of this implementation is $O(|V| + N_P + |N_G| * N_P) = O(|V| + |N_G| * N_P)$. Assume that $|N_G| << N_P$. Then $O(|V| + |N_G| * N_P) = O(|V| + N_P)$, which is linear.

Costs for inserting a new CP are as follows.
Step 1. Search the local TI-tree for the target class node, and update the posting. $T_1 = O(D+N_P)$.
Step 2. Transfer the attributes to the Portal. Assume $T_2 = O(1)$.
Step 3. Search the global TI-tree for the target class node, and update the posting. $T_3 = O(D+N_P)$.
The running time of this implementation is $O(D+N_P)$, which is linear.

Experimental Results

In this experiment, we have implemented a grid portal for query submission and results presentation. In addition, results are shown with estimated transmission time. The corpus is composed of synthetic SCORM-compliant Documents. The experiments investigate Searching efficiency, Retrieving efficiency and Precision. Also, questionnaire is used to understand the degree of satisfaction of users. First, we introduce the grid test-bed. Then, the

implementation of the prototype is described. Next, experimental results about query processing time and scalability are presented. Finally, results of questionnaires are reported.

Grid Test-bed: TIGER Grid

A metropolitan-scale Grid computing platform named TIGER Grid (standing for Taichung Integrating Grid Environment and Resource) has been built in a project leaded by Tunghai University (Yang, Li et al., 2005; Yang, Han et al., 2006). The TIGER grid interconnects computing resources of universities and high schools and shares available resources among them, for investigations in system technologies and high performance applications. This novel project shows the viability of implementation of such a project in a metropolitan city. The TIGER Grid computing platform consists of three universities and two high schools, all located in Taichung, Taiwan. The purpose of constructing such a grid infrastructure was to share computational resources of each institution.

The educational institutions participating in this project are Tunghai University (THU), Providence University (PU), HsiuPing Institute of Tech (HIT), National Dali High School (DL) and Li-Zen High School (LZ). They are interconnected by TANet (Taiwan Academic Network) with bandwidth of 1Gbps. The TIGER Grid platform consists of 33 computing nodes, with 58 CPUs of different speed and total storage of more than 2TB, in 2006 (Yang, Han et al. 2006). All these institutions are in Taiwan, and each is at least 10 Kilometers away from THU geographically. All machines in this grid have Globus 3.2.1 or above installed. Figure 8 shows the real-time status of the grid test-bed acquired by the monitoring tool, Ganglia. In this paper, we make use of the features of the Ganglia monitoring system and the monitoring utilities in the Globus Toolkit to realize resource monitoring. As shown in this figure, the number of available CPUs is 101, which are contributed by 12 schools. The real-time loading information and the capacity of available memories are also illustrated in this snapshot.

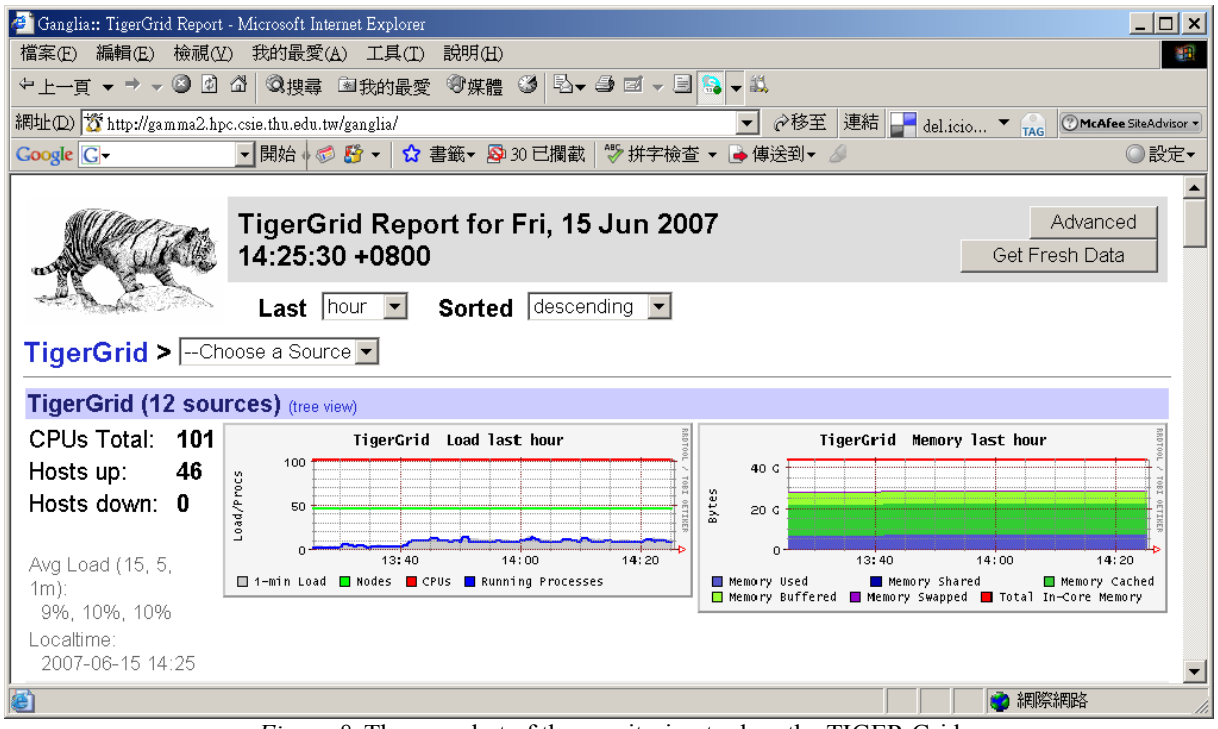

Figure 8. The snapshot of the monitoring tool on the TIGER Grid

In this paper, we have built a grid test-bed by using part of the TIGER Grid. The master node is at Tunghai University (THU), and the slave nodes are located at Tunghai University (THU), Providence University (PU), Li-Zen High School (LZ), and Hsiuping Institute of Technology School (HIT). Figure 9 shows the topology of our grid test-bed, and the specifications of the grid test-bed are shown in Table 2. The purpose of the grid portal is to allow users to access resources via a friendly web page interface.

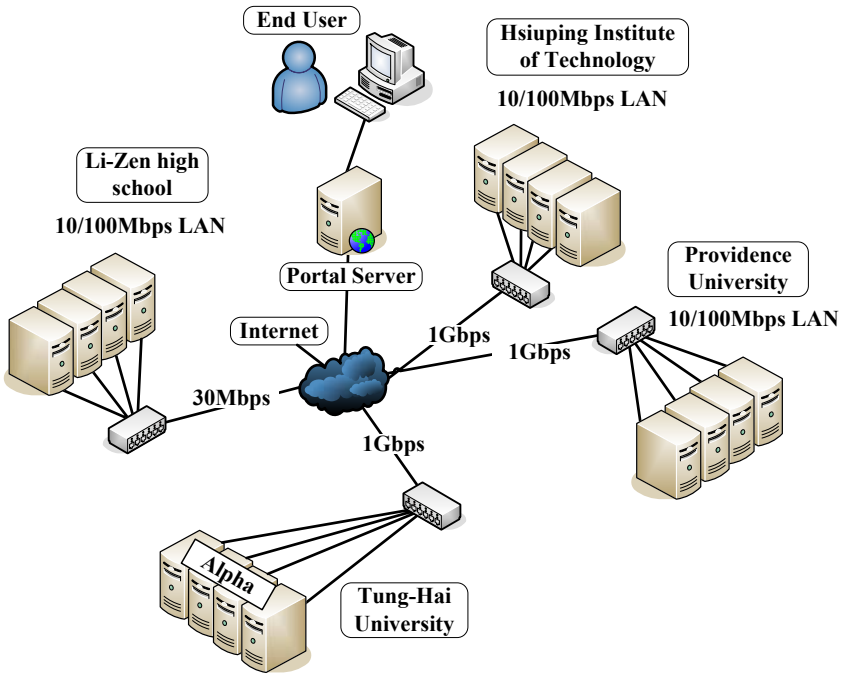

Figure 9. The topology of our grid test-bed

Table 2. Specifications of computing resources on the test-bed

SITE	HOST	CPU TYPE	CLOCK (MHZ)	RAM	NIC	LINUX KERNEL	GLOBUS VERSION
THU	DELTA1	INTEL PENTIUM 4	3001	1GB	1G	2.6.12	4.0.1
	DELTA2	INTEL PENTIUM 4	3001	1GB	1G	2.6.12	4.0.1
	DELTA3	INTEL PENTIUM 4	3001	1GB	1G	2.6.12	4.0.1
	DELTA4	INTEL PENTIUM 4	3001	1GB	1G	2.6.12	4.0.1
LZ	LZ01	INTEL CELERON	898	256MB	10/100	2.4.20	3.2.1
	LZ02	INTEL CELERON	898	256MB	10/100	2.4.20	3.2.1
	LZ03	INTEL CELERON	898	384MB	10/100	2.4.20	3.2.1
	LZ04	INTEL CELERON	898	256MB	10/100	2.4.20	3.2.1
HIT	GRIDHIT0	INTEL PENTIUM 4	2800	512MB	10/100	2.6.12	3.2.1
	GRIDHIT1	INTEL PENTIUM 4	2800	512MB	10/100	2.6.12	3.2.1
	GRIDHIT2	INTEL PENTIUM 4	2800	512MB	10/100	2.6.12	3.2.1
	GRIDHIT3	INTEL PENTIUM 4	2800	512MB	10/100	2.6.12	3.2.1
PU	HPC09	AMD ATHLON XP	1991	1GB	1G	2.4.22	3.2.1
	HPC10	AMD ATHLON XP	1991	1GB	1G	2.4.22	3.2.1
	HPC11	AMD ATHLON XP	1991	1GB	1G	2.4.22	3.2.1
	HPC12	AMD ATHLON XP	1991	1GB	1G	2.4.22	3.2.1

Prototype and Evaluation Design

To evaluate the proposed approach, we have implemented a web-based prototype, named "e-Learning Grid Resource Portal," for the e-Learning grid test-bed. As shown in Figure 10, users can submit queries in this web page. Then, the TI-tree is accessed to find related teaching materials. After that, the desired contents are retrieved from local sites and returned to the user. The retrieved content packages are ranked by their similarities to the query. Also, the estimated transmission time of each content package is listed. All programs are implemented in the Java language. Because of the dynamic nature of grid environments, experiments are repeated 5 times, and the average values are reported. All experiments are conducted in dedicated mode.

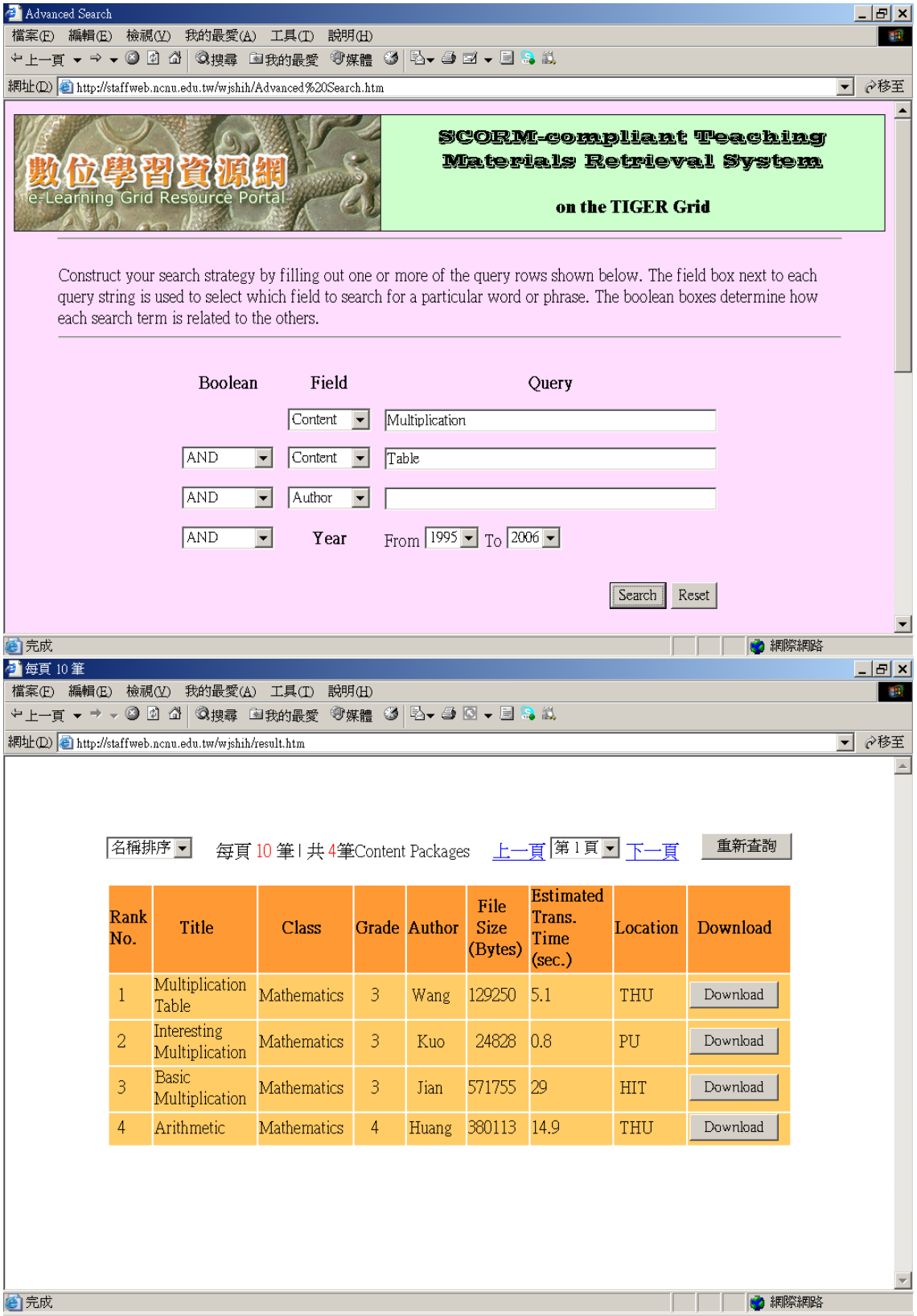

Figure 10. Query submission and results presentation of the prototype

We adopt the method in (Su, Tseng et al., 2005) to generate synthetic learning contents. All the synthetic teaching materials were generated by setting three parameters: 1) V: The dimension of the feature vectors of the teaching materials (TM); 2) D: the depth of the content structure of the TM; 3) B: the upper and lower bounds of the included sub-section for each section of the TM. Four synthetic LORs are built, with V = 15, D = 3, and B = [5, 10], and stored in the grid, which are listed in Table 3. The four LORs contain 2,400,000 SCORM-compliant documents, which are converted from Web pages related to educational domains. After stop-word cleansing, there remains 4,730,384 distinct index terms. The total size of these LORs is around 40 GB.

Some of the Content Packages in the LORs are retrieved and adapted from existing repositories on the Internet, such as http://learning.edu.tw/mainpage.php. Currently, these LORs are only available to primary-school teachers who participate in this evaluation. However, in the near future, we plan to place the prototype and the LORs on the web for public access and large-scale evaluation.

Table 3. Specifications of learning object repositories

SITE	NO. OF DOCUMENTS	SIZE (GB)
THU	900,000	15
LZ	300,000	5
HIT	600,000	10
PU	600,000	10

We simplify the Dewey Decimal Classification (DDC) system to serve as the taxonomy. After refinement, there were 10, 50, and 250 class nodes in levels 1, 2 and 3 of the TI-trees, respectively.

We have implemented two algorithms for this experiment. The first algorithm is a traditional meta-searching approach, named DIR (Distributed Information Retrieval). The other is the approach using TI-trees, named GIR (Grid Information Retrieval).

Experiments on Execution Time

The purpose of this experiment is to show that the proposed GIR is faster than the traditional approach DIR, by comparing the average time of query processing. 20 queries generated randomly were used to compare the performance of DIR and GIR. Figure 11 illustrates the processing time of queries for implementations of DIR and GIR. The average query processing time of GIR is 1.1 second less than that of DIR. The main reason may be that GIR using the TI-tree approach can effectively speed the searching processing. However, DIR dispatches the query to four sites, and merges the results returned from these sites, thus slowing down the process.

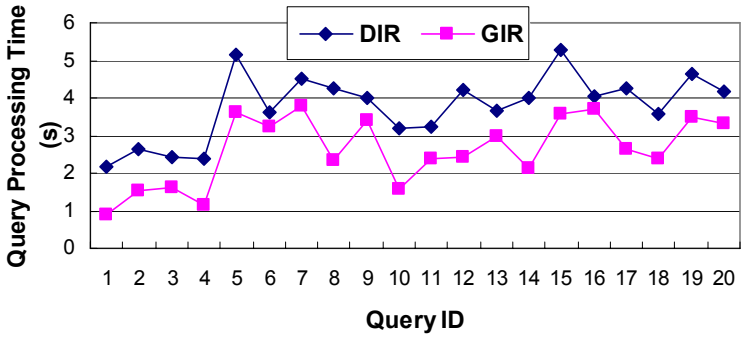

Figure 11. The query processing time for 20 queries

The difference of query processing time obtained in Figure 11 is 1.1 second, which seems not significant. One reason may be duo to the short length of queries. In the following experiment, we compare the average query processing time of queries with different length (number of terms). We generated three kinds of queries with lengths of 5, 10

and 15 terms, respectively. As shown in Figure 12, when the query length is 15, the difference of query processing time for DIR and GIR is 7.1 second. Although further experiments are needed to justify the real reason, we conjecture that the difference of performance will become more significant when the scale gets larger. This conjecture is further verified in the next experiment.

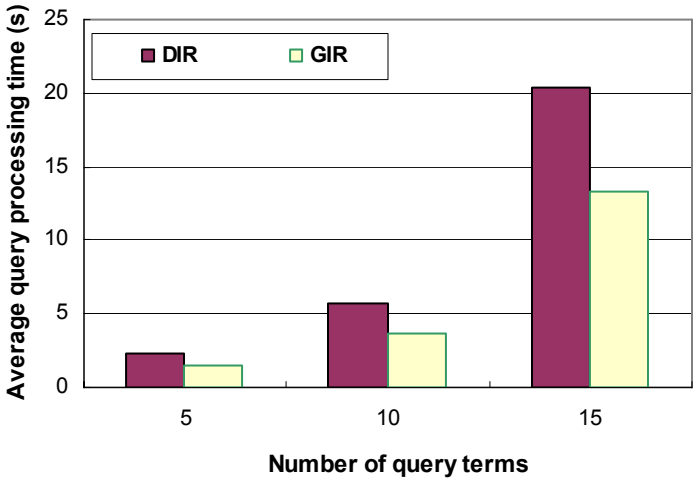

Figure 12. The average query processing time for different numbers of query terms

Experiments on Scalability

This experiment aims to address the influence of the scale of LOR sizes. We generate four combinations of LORs with doubled sizes, as shown in Table 4. The number of query terms is 10. Figure 13 shows the average query processing time for collections of different numbers of documents. According to this figure, there is almost no scalability problem for GIR. However, scalability begins to become an issue when we further increase the size of document collection for DIR. In general, the proposed approach is more scalable. This is mainly because the global TI-tree decreases the searching time.

Table 4. Combinations of learning object repositories

COMBINATION OF SITES	ORIGINAL NO. OF DOCUMENTS	DOUBLE NO. OF DOCUMENTS
LZ	300,000	600,000
LZ+HIT	900,000	1,800,000
LZ+HIT +PU	1,500,000	3,000,000
LZ+HIT +PU+THU	2,400,000	4,800,000

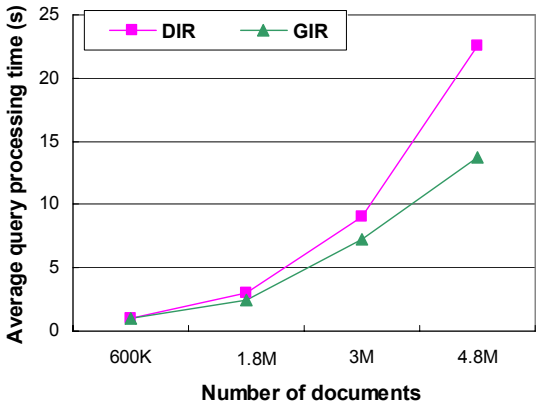

Figure 13. Scalability for increasing numbers of documents

Results of questionnaires

In the experiment, 12 participants are invited to use our prototype system for retrieving documents. These users include 12 primary school teachers in Taichung and Nantou, in Taiwan. They are asked to answer two questions after they use this prototype. The first question is concerning the precision of the retrieved results, and the second question is regarding the perceived performance of transmission. The results are illustrated in Figure 14. In this figure, the score given by a user ranges from one to five. "One" means "Very Unsatisfactory" and "Five" means "Very Satisfactory". The average of satisfaction for "Search Time" is 4.6, and that for "Precision" is 3.9. This experiment shows that our system could be efficient and helpful to users. After reading comments from these teachers, we find that they are satisfied with the response time of the prototype.

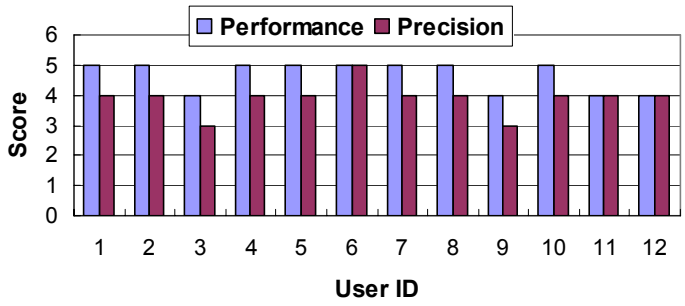

Figure 14. The running time for different numbers of classes

Conclusion

With the rapid development of grid technology, more and more LORs are constructed and need to be connected to share the content. Efficient content retrieval can reduce the response time and thus attract more users to utilize the e-learning systems. In this paper, we have designed an approach to retrieve content packages on grid environments. In addition, a grid test-bed was built to implement the system. Experimental results show the proposed approach is efficient, scalable and suitable for grid environments.

The issue of learning content management affects the developers of educational systems and educators who implement the systems. By means of a detailed description of the proposed approach, this paper can help educators use the technology and the tool to rapidly find the desired learning contents. On the other hand, developers can understand characteristics of learning contents through the introduction of the methodology. Therefore, the proposed approach provides obvious advantages for Distance Education stakeholders. Students can remotely access the desired learning contents stored in grid environments fast and accurately, thus facilitating their learning. For teachers, effective and efficient retrieval of various teaching materials is helpful to sharing and reusing of these contents.

The existing methods of information retrieval can not satisfy the request of rapidly and accurately retrieving desired learning contents from large-scale repositories, such as grids. The proposed approach attains rapidness and precision by a Taxonomic Indexing tree. With this technology, the desired learning contents can be efficiently retrieved, thus advancing the sharing and reusing of learning contents. For example, one application of this technology is teaching material design. To support individualized and adaptive learning, teachers are encouraged to develop various teaching materials according to different requirements. However, traditional methodologies for designing teaching materials are time-consuming. To speed up the development process of teaching materials, one idea is to use a rapid prototyping approach which is based on reusing existing contents.

The construction of Taxonomy-Index Trees is based on a taxonomy for rapid searching and retrieval. In our future work, the proposed method will be extended to an ontology-based approach to retrieving learning content. Next, an ontology-based framework of content management, which consists of ontology building phase, indexing phase and search phase, will be designed and implemented. With this methodology, we hope to provide an effective and

efficient system for learning content retrieval, thus advancing the further development of e-Learning. In this paper, we focused on the management of learning contents which have three types of attributes: textual contents, metadata and structural information. SCORM is one of the standards which satisfy the requirements. In fact, improving the proposed method to consider the learning design of contents, such as IMS-Learning Design, is another interesting issue, which is also our future work.

Acknowledgements

This research was partially supported by National Science Council of Republic of China under the number of NSC95-2520-S009-007-MY3 and NSC95-2520-S009-008-MY3.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*, New York: ACM Press.

Berners-Lee, T., Hendler, J., & Lassila, O. (2000). Semantic Web. *Scientific American, 1* (1), 68-88.

Bote-Lorenzo, M. L., Hernández-Leo, D., Dimitriadis, Y. A., Asensio-Pérez, J. I., Gómez-Sánchez, E., Vega-Gorgojo, G., et al. (2004). Towards Reusability and Tailorability in Collaborative Learning Systems using IMS-LD and Grid Services. *International Journal on Advanced Technology for Learning, 1* (3), 129-138.

Cambazoglu, B. B. & Aykanat, C. (2006). Performance of query processing implementations in ranking-based text retrieval systems using inverted indices. *Information Processing and Management, 42*, 875-898.

Dewey (2004). *Dewey Decimal Classification*, Retrieved Dec. 07, 2006, from http://www.oclc.org/dewey/about/default.htm.

Foster, I. (2002). The Grid: A New Infrastructure for 21st Century Science. *Physics Today, 55* (2), 42-47.

Foster, I. & Kesselman, C. (1997). Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications and High Performance Computing, 11* (2), 115-128.

Gaeta, M., Ritrovato, P., & Salerno, S. (2003). ELeGI: The European Learning Grid Infrastructure. *Paper presented at the 3rd International LeGE-WG Workshop: GRID Infrastructure to Support Future Technology Enhanced Learning,* Dec. 3, 2003, Berlin, Germany.

Iorio, A. D., Feliziani, A. A., Mirri, S., Salomoni, P., & Vitali, F. (2006). Automatically Producing Accessible Learning Objects. *Educational Technology & Society, 9* (4), 3-16.

Kassahun, A., Beulens, A., & Hartog, R. (2006). Providing Author-Defined State Data Storage to Learning Objects. *Educational Technology & Society, 9* (2), 19-32.

Kiu, C. C. & Lee, C. S. (2006). Ontology Mapping and Merging through OntoDNA for Learning Object Reusability. *Educational Technology and Society, 9* (3), 27-42.

Ko, S. K., & Choy, Y. C. (2002). A structured documents retrieval method supporting attribute-based structure information. *Paper presented at the 2002 ACM Symposium on Applied Computing,* Mar. 10-14, 2002, Madrid, Spain.

Lanzilotti, R., Ardito, C., Costabile, M. F., & Angeli, A. D. (2006). eLSE Methodology: a Systematic Approach to the e-Learning Systems Evaluation. *Educational Technology & Society, 9* (4), 42-53.

Lee, M. C., Tsai, K. H., & Wang, T. I. (2006). An Ontological Approach for Semantic-Aware Learning Object Retrieval. *Paper presented at the the Sixth International Conference on Advanced Learning Technologies (ICALT),* Jul. 5-7, 2006, Kerkrade, the Netherlands.

Lee, Y. K., Yoo, S.-J., Yoon, K., & Berra, P. B. (1996). Index structures for structured documents. *Paper presented at the the 1st ACM international conference on digital libraries,* Mar. 20-23, Bethesda, Maryland, the United States.

Mittal, A., Krishnan, P. V., & Altman, E. (2006). Content Classification and Context-Based Retrieval System for E-Learning. *Educational Technology & Society, 9* (1), 349-358.

Nitto, E. D., Mainetti, L., Monga, M., Sbattella, L., & Tedesco, R. (2006). Supporting Interoperability and Reusability of Learning Objects: The Virtual Campus Approach. *Educational Technology & Society, 9* (2), 33-50.

Salton, G., & McGill, M. J. (1983). *Introducion to Modern Information Retrieval*. New York: McGraw & Hill.

Sierra, J. L., Fernández-Valmayor, A., Guinea, M., & Hernanz, H. (2006). From Research Resources to Learning Objects: Process Model and Virtualization Experiences. *Educational Technology & Society, 9* (3), 56-68.

Su, J.-M., Tseng, S.-S., Wang, W., Weng, J.-F., Yang, J. T. D., & Tsai, W.-N. (2006). Learning Portfolio Analysis and Mining for SCORM Compliant Environment. *Educational Technology & Society, 9* (1), 262-275.

Su, J. M., Tseng, S. S., Wang, C. Y., Lei, Y. C., Sung, Y. C., & Tsai, W. N. (2005). A Content Management Scheme in SCORM Compliant Learning Object Repository. *Journal of Information Science and Engineering, 21* (5), 1053-1075.

Trotman, A. (2004). Searching structured documents. *Information Processing and Management, 40*, 619-632.

Trotman, A. (2005). Choosing Document Structure Weights. *Information Processing and Management, 41*, 243-264.

Witten, I. H., Moffat, A., & Bell, T. C. (1999). *Managing gigabytes: compressing and indexing documents and images* (2nd Ed.), San Francisco, California: Morgan Kaufmann.

Yang, C. T., Han, T. F., Shih, W. C., Chiang, W. C., & Chang, C. H. (2006). Metropolitan-Scale Grid Environment: The Implementation and Applications of TIGER Grid. *Paper presented at the Frontiers of High Performance Computing and Networking – ISPA 2006 Workshops,* Dec. 4-6, 2006, Sorrento, Italy.

Yang, C. T., Li, K. C., Chiang, W. C., & Shih, P. C. (2005). Design and Implementation of TIGER Grid: an Integrated Metropolitan-Scale Grid Environment. *Paper presented at the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT),* Dec. 5-8, 2005, Dalian, China.

Yang, C. T. & Ho, H. C. (2005). An e-Learning Platform Based on Grid Architecture. *Journal of Information Science and Engineering, 21* (5), 911-928.

Yu, C., Liu, K. L., Meng, W., Wu, Z., & Rishe, N. (2002). A Methodology to Retrieve Text Documents from Multiple Databases. *IEEE Transactions on Knowledge and Data Engineering, 14* (6), 1347-1361.