A Novel PIM System and its Effective Storage Compression Scheme

Liang Huai Yang†, Jian Zhou, Jiacheng Wang
School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310014, China
†Email: yang.lianghuai@gmail.com

Mong Li Lee
School of Computing, National University of Singapore, Singapore
Email: leeml@comp.nus.edu.sg

Abstract—**The increasingly large amount of personal information poses a critical problem to users. Traditional file organization in hierarchical directories is not suited to the effective management of personal information. In order to overcome the shortcomings of the current hierarchical file system and efficiently organize and maintain personal information, some new tools are expected to be invented. In this paper, we propose a novel scheme called *concept space* - a network of concepts and their associations – and use topic map as the underlying data model. We present a materialized view scheme to provide users with a flexible view of the file system according to their own cognition. We also reduce the storage requirement to save space usage of this system by borrowing some ideas from XML data management and contriving a novel and efficient data compression scheme. To demonstrate the effectiveness of the above idea, we have implemented a prototype personal information management system called *NovaPIM* and presented its system architecture. Extensive experiments show that our proposed scheme is both efficient and effective.**

Index Terms—**Personal Information Management, concept space, data compression**

I. Introduction

Personal information management (PIM) refers to the activities people performed to acquire, store, organize and retrieve their items of digital information for everyday use [1]. PIM gained intensive attention in recent years [11, 12,13,14,15,16]. Academic research on personal information tools stems from the early days of Hypertext research including Vannevar Bush's vision of a PIM device called "memex" [10] more than six decades ago, "a memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory." However, most systems only provide a fixed, rigid hierarchical file organization. To make matters worse, there is no alternate way, for example, using different views, to access the personal information.

The goal of PIM[6,7] is to offer easy access and manipulation of all of the information on a person's desktop, with possible extension to mobile devices, personal information on the Web, or even all the information accessed during a person's lifetime. Personal information has a great diversity, which ranges from office documents, PDF documents, emails, XML data, relational data, music files, images, to videos etc. Besides heterogeneity, the data is distributed in laptops, desktop PCs, mobile phones, local systems, email servers and other network systems, leading to *information fragmentation*[2]. The increasingly large amount of personal information poses a critical problem to users. Thus, how to integrate these data is one of the challenges of PIM.

Traditional file organization in hierarchical directories may not be suited to the effective management of personal information because it ignores the semantic associations and bears no connection with the applications that users will run. Further, the physical hierarchical directories can give user only one view.

In this paper, we introduce the notion of *concept space* to manage the collection of personal information objects. Similar to the *view* of relational database, the term *concept* represents user's logical view of the personal information items which may locate in different file directories in the file system. We utilize the graphical data model to organize the concept space where the nodes are the concepts and the edges are the shortcuts to the specific files or hyperlinks to some specific html documents. Consequently, there will be large number of such links in this system. These links share many prefixes and gives us the opportunity to compress the contents.

Based on these ideas, we design a PIM system called *NovaPIM* that provides flexible views of the information items and overcomes the weakness of current file system which has only one monolithic physical organization. *NovaPIM* also use a dictionary based compression scheme to reduce the overheads of the storage space for the contents of the graphical data model. Experiments results verify the effectiveness of this scheme.

The remainder of the paper is organized as follows. Section II reviews briefly the related work on PIM; Section III describes the architecture of our system *NovaPIM*; Section IV addresses the issue of storage

Manuscript received OCT 10, 2011; revised NOV 2, 2011; accepted NOV 4, 2011.
†Correspondent author

compression on *NovaPIM*; and experimental evaluations for our proposed scheme are given in Section V. Finally, we summarize our work in Section VI.

II. RELATED WORK

PIM has attracted much attention for decades since Vannevar Bush's memex[10]. The increasingly large amount of personal information(emails, sms, documents, photos, videos, etc.), available from PCs, mobile phones, PDAs, digital cameras, internet etc., poses a critical problem to users. How to manage and organize this information for personal productivity? As such, much research has focused on this issue in recent years[11,12, 13,14,15,16]. PIM workshops sponsored by NSF (USA) have been held for several years since 2005. Whittaker[28] reviews research on three different information curation processes: keeping, management and exploitation. A series of research prototypes have been proposed in the academic community: *SIS*[20], *Lifestreams*[21], *Agenda*[23], *gIBIS*[24], *Rufus*[25], *iMeMex* [14,13], *SEMEX*[6], *Haystack*[12], *MyLifeBits*[11], etc. Among them, *SIS* and *Lifestreams* are document oriented retrieval system, while the early tools like *Agenda*, *gIBIS* and *Rufus*, and the recent ones like *SEMEX*, *Haystack* and *MyLifeBits*, are based on relational model, and all data are uniformly represented in this data model. This approach can take advantage of the mature technology of RDBMS. As RDBMS depends on rigid relational schema, this approach cannot fully meet the needs of PIM. *iMeMex*, on the other hand, presents an *iDM* data model, which characterize itself with a graph data model to express the data space, provides a formal method to represent a unified view of resources(such as document, directory, relational table, XML document, data stream etc.). The theoretical foundation of *iMeMex* is RDF (Resource Description Framework), *iMeMex*'s architecture consists of three layer: application layer, PDSMS (Personal Data Space Management System) layer, and resource layer, as shown in Figure 1. Stephen[22] gives a good description of PIM issues from the perspective of personal knowledge database. He addresses such issues as data model of personal knowledge database, the theoretical problems involved, and taxonomy of PIM tools.

There exist many PIMS tools in industry but still far from satisfactory. Here we enumerate some popular ones: Microsoft's *OneNote*, Micro Logic's *Info Select* [1] and Thomson's *EndNote*[2]. *OneNote* makes note taking easier, but it is an independent application separated from other applications as email client, Internet explorer, and file system. *Info Select* integrates email and note taking functionalities into it, it is a good supplement to file system[15]. *EndNote* focuses on the management and organization of literature without considering other aspects of PIM.

As PIM involves a diversity of data types with their implicit semantics and lack of associations between data objects, traditional desktop systems are incapable of PIM. Keyword based information retrieval is not sufficient for PIM, hence a flexible querying scheme is desired [26].

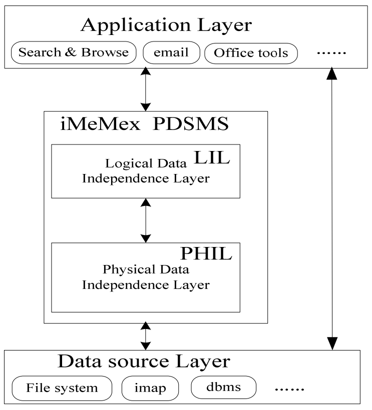


Figure 1. Architecture model of iMeMex PSDMS

iMeMex[14,13] uses RDF as its knowledge representation model. RDF is more "low-level" than the topic maps[19]. In RDF, resources are represented as triplets (subject, predicate, object). In topic maps, topics have characteristics of various kinds: names, occurrences and roles played in associations with other topics. The essential semantic distinction between these different kinds of characteristic is absent in RDF. And more often than not, schema is absent from PIM. Consequently, the data model of topic maps is exploited as our system's underlying model.

III. AN OVERVIEW OF OUR PIM SYSTEM – NOVAPIM

A. The prototype PIM system--NovaPIM

To overcome the shortcomings of the current hierarchical file system in managing personal information, we propose to use *concept* to organize and manage the collection of personal information objects. A *concept* is a logical view that a user uses to organize the information items (files, URLs, emails…) and includes one or more sub-concepts or topics. For each sub-concept or topic, it can be materialized to a file which may contains one or more shortcuts or hyperlinks to the physical files. These concepts form an information space that we call it *Concept Space*, which is a network of concepts and their associations. Based on this idea, we have implemented a prototype PIM system called *NovaPIM* shown in Figure 2. Concepts are visualized as a tree-structure shown on the left-hand tab folders while their relationships are represented as folder and sub-folder or shortcuts/ hyperlinks in a file which can be edited/rendered on the right-hand tab folder. Figure 2 shows an example concept of "Path Compression".

Next, we'll discuss the system architecture and its rationales behind it.

B. System Architecture

Figure 3 illustrates the architecture of our system NovaPIM. It consists of three layers: Application Layer, Concept Space Layer and Resource Layer.

1) Application Layer

[1] http://www.miclog.com/software/

[2] http://www.endnote.com/

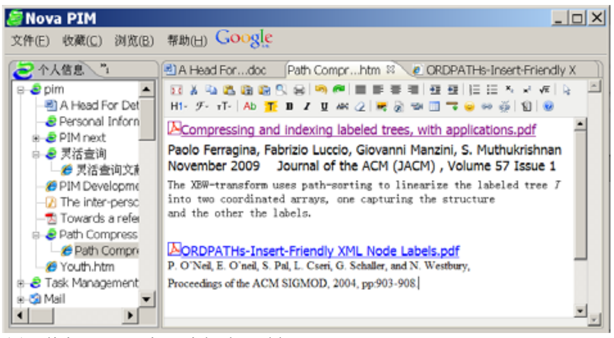
Application Layer is the top layer of the system. It provides various functionalities related to personal information management, such as query and search, email service, task management, agenda scheduling, etc.. The functions may be composed from existing independent applications through application integration, e.g., in place activation of Microsoft office application via OLE techniques. Some have to be constructed from scratch. For example, the capability of flexible querying/search for PIM system is desired, here we envision that the system provides not only the traditional keyword based searching capability but also the structure querying capability, even the DB&IR capability[8]. Much of the user interaction with PIMS involves exploring the data, and users do not have a single schema to which they can pose queries. Consequently, it is important that queries are allowed to specify varying degrees of structure, spanning keyword queries to more structure-aware queries. The query system should be able to exploit both exact matching and approximate matching scheme.

2) *Concept Space Layer*

As stated above, we use *concept* to organize / associate PIM items. Concepts are assumed to be basic constituents of thought and belief, and the basic units of thought and knowledge that underlie human intelligence and communication [17]. Every concept consists of the intension and the extension. The intension of a concept consists of all properties or attributes that are valid for all those objects to which the concept applies. It is an abstract description of common features or properties shared by elements in the extension. The extension of a concept is the set of objects or entities which are instances of the concept, or rather, the extension consists of concrete examples of the concept. All objects/entities in the extension have the same properties or attributes that characterize the concept. A concept is thus described jointly by its intension and extension. All the concepts and their associations form the *Concept Space Layer* which is the core of the PIM system. In essence, it is a graph-data model.

The concept of *concept space* was first proposed from information retrieval perspective by Deng[3] in 1983, where he stated that *Concept Space* was composed of the concepts and the semantic network. Concept reflects the objective nature of things and the characteristics of the general. As we know, semantic network is a knowledge representation scheme involving nodes and links (arcs or arrows) between nodes, where the nodes represent objects or concepts and the links represent relations between nodes. The links are directed and labeled; thus, a semantic network is a directed graph. In theory, this definition is consistent with ours though there is no standard definition of *Concept Space* by now.

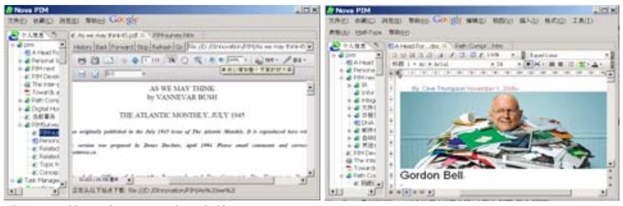
As stated before, the data model of Topic Maps is exploited as our system's underlying model. A type hierarchy is helpful to enforce "is-a" relationship. However, we have no pre-defined schema (or concept hierarchy) in our system. Instead, we use the extensions of concepts. In this respect, it's truly different from the traditional DBMS. In addition, we relax the use of semantic network and take a more practical approach. For instance, we are using synonyms or even generic "related-to" to express relationship between concepts. In the future, we may introduce other "associations" into our system. The reason underlying this preference is detailed below.



Compressing and indexing labeled trees, with applications.pdf
Paolo Ferragina, Fabrizio Luccio, Giovanni Manzini, S. Muthukrishnan
November 2009 Journal of the ACM (JACM) , Volume 57 Issue 1
The XBW-transform uses path-sorting to linearize the labeled tree T into two coordinated arrays, one capturing the structure and the other the labels.

ORDPATHs-Insert-Friendly XML Node Labels.pdf
P. O'Neil, E. O'neil, S. Pal, L. Cseri, G. Schaller, and N. Westbury,
Proceedings of the ACM SIGMOD, 2004, pp:903-908

(a)editing a topic with drag/drop support



(b)application embeddings

Figure 2. Our prototype PIM system—NovaPIM#

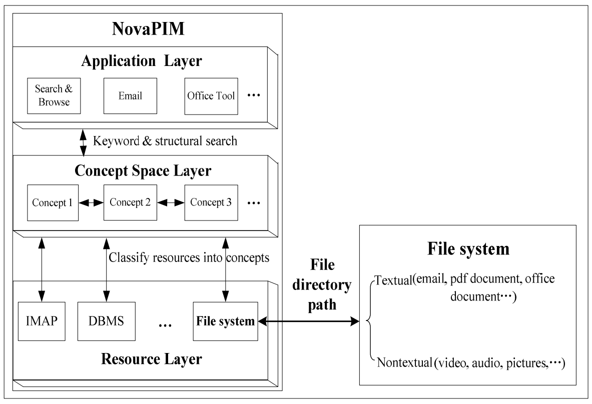


Figure 3. The Architecture of our system NovaPIM

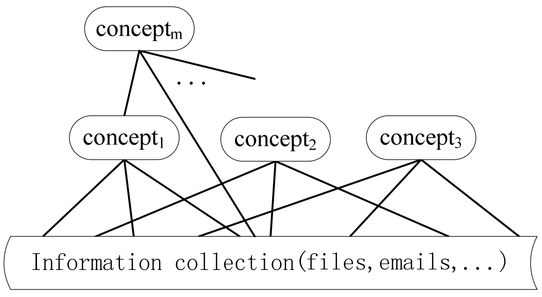


Figure 4. The network of concepts.

During using and maintaining the personal information items, different users may have different view of the same items. In addition, the user's viewpoint of things may evolve as his cognition advances or time goes by. Users may classify/categorize the files according to different considerations (e.g. file content, file type, subject, and time modified/downloaded) in different situations. Hence, it is impractical to force users to use a predefined "semantic network" in managing their information collection. Consequently, *NovaPIM* takes a more practical stand, which allows users to freely define their own concepts by using their own terms. Here, each concept refers to a collection of resources that have similar or related information. *NovaPIM* realizes the associations between concepts through shortcuts/ hyperlinks or folders/subfolders. These concepts form a network as shown in Figure 4. As a result, such a scheme allows users to have flexible views of the same information items, which is reminiscent of the view scheme in RDBMS. As such, this scheme is so powerful that it relieves people from the restrictions of one physical organization allowed in the file system. With this scheme, users can build up a network of concepts with different needs (e.g. the need of work, personal preference or habit). In addition, inductive inference and learning can be exploited to derive relationships between the intensions of concepts based on the relations between the extensions of concepts. Through the connections between extensions of concepts, one may establish relationships between concepts[18].

As a result, the *Concept Space Layer* uses concepts to represent the data/file objects of various types/formats and interconnects them. This abstraction of resources facilitates the user and the *Application Layer*. *Concept Space Layer* acts as the mediator between the *Application Layer* and the physical *Resource Layer* and is the core of PIMS. It achieves the physical data independence through the mapping of *Concept Space Layer/Resource Layer*.

3) Resource Layer

The personal information may be in various forms. It can be a document (in formats as word, txt, pdf, mp3, rmvb, wav, etc.), email, URL, etc. or it can be structured data in DBMS. It is noteworthy to mention that unstructured data comprises the vast majority of data found in an organization, some estimates run as high as 80%[9]. In personal information items, this number becomes even larger. In PIM system, the management of unstructured text is of primary importance. The items may scatter in the different locations/directories in the file system.

C. NovaPIM Implementation and Discussion

We have implemented a prototype system called *NovaPIM* by Eclipse Java to demonstrate the proposed idea. A hyper graph model is adopted for our proposed concept space. Each vertex in the graph is a concept (it is the extent of the concept in our case) and each edge is the association between concepts. Although the underlying theoretical foundation is topic map, it is relaxed in our implementation. When a user uses unstructured data, the data is usually lack of schema or a user may not provide the metadata. That's a big difference from traditional database system which always has a set of predefined schema. As such, *NovaPIM* takes a more practical approach. Currently it is a big challenge to define a view for a concept using the topic map query language, and the topic map query language in *NovaPIM* has a very limited usage for its lack of schema. As a result, we resort to IR. The combination of database technology and information retrieval may be the best rescue.

The extent of a concept is defined and edited via a HTML editor and the hyperlinks therein links to other concepts or physical resources. In the future, we'll introduce concept/view definition language to define concepts. Another consideration is to improve on-demand displaying of contents (concept/view) by active XML[27]. *NovaPIM* combines the tree and flat file to present its hierarchical structure and graph structure. Concepts are stored in XML file and the relationship is realized via ID reference and hyperlinks that will be stated below. In application layer, *NovaPIM* implements the embedding of several applications such as PDF, HTML, Office, media, email, etc., and provides a flexible query scheme which incorporates concept hierarchy, file directory and document content(refer to Figure 2 (b)). *NovaPIM* also realizes email/task association. When an email arrives, the system will check the contents of the mail and compute its similarity with those tasks/topics/subjects already defined.

NovaPIM overcomes the weakness of the current file system. It provides the physical data independence through the mapping between concept space layer and physical resource layer. *NovaPIM* provides a view scheme for user to create his own concept hierarchy according to his cognitions. *NovaPIM* has a drag & drop scheme to define the extent of a concept without changing the file directory structure physically. One only needs to drag and drop the files into a HTML editor(refer to Figure 2 (a)); the shortcuts will be embedded into the editor, e.g., "file:///C:/publication/PIMS/AsWeMay Think.pdf". The extent of a concept is similar in some sense to the materialized view in DBMS.

By this way, some associations of concepts are materialized as shortcuts/hyperlinks to the specific file indicated by a specific file directory path/URL/URI (hereafter, we call this a locator, hyperlink or path for simplicity). A locator is a string which is composed of a series of label names separated by path separator "/" and ends with a file name. With this, we can uniquely determine the file location in the file directory structure/internet, and achieve the mapping from the logical model (Concept Space) to the physical model (file system). The same item/file can be referenced at any number of times with no need of duplicating the document. As the number of concepts increases, there will be large number of such links in this system. These links probably share many prefixes and thus gives us opportunity to compress the contents. The issue of content compression is addressed in next section.

IV. COMPRESSION SCHEME FOR *NOVAPIM*

In *NovaPIM*, the extent of a concept is a collection of hyperlinks/locators. Taking all the shortcuts and hyperlinks as whole, they form a tree. By borrowing the labeling scheme from XML data that are widely used in XML query processing, we achieve the goal of data compression for ***NovaPIM*.** In this section, we adopt the

ORDPATH[4,5] labeling scheme to form a dictionary-based compression method to reduce the overheads of storage space for *NovaPIM*.

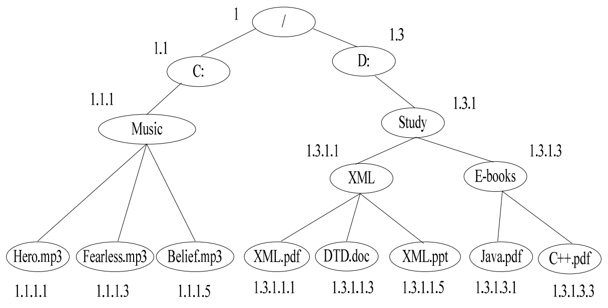


Figure 5. An example of *ORDPATH* encoding

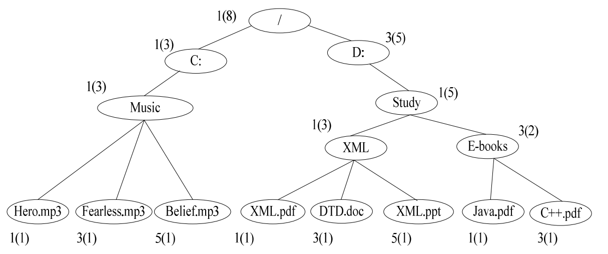


Figure 6. A Variant of *ORDPATH* Encoding

TABLE 1. VALUE TABLE

ORDPATH Encoding	*TAG*	*RefCount*
1	/	9
1.1	C:	4
1.3	D:	4
1.1.1	Music	2
1.3.1	Study	1
1.1.1.1	Hero.mp3	1
1.1.1.3	Fearless.mp3	5
1.1.1.5	Belief.mp3	5
1.3.1.1	XML	3
1.3.1.3	E-books	1
1.3.1.1.1	XML.pdf	1
1.3.1.1.3	DTD.doc	1
1.3.1.1.5	XML.ppt	2
1.3.1.3.1	Java.pdf	1
1.3.1.3.3	C++.pdf	1

A. ORDPATH node labeling scheme

ORDPATH is a prefix-based node labeling scheme, it encodes the parent-child relationship by extending the parent's *ORDPATH* label with a component for the child. An example of the file directory path tree labeling using *ORDPATH* is depicted in Figure 5. For example, 1.3.1 represents a parent node, 1.3.1.3 is its child node. The *ORDPATH* value ("1.3.1.3") with dot separated ordinal values ("1", "3", "1", "3") reflects the successive levels down the path from the root to the node represented. During the initial load, *ORDPATH* assigns only positive and odd integers, even and negative integers are reserved for later insertions. If the newly inserted node is to be added to the right of all the existing children, its label is generated by adding +2 to the last ordinal of the last child. If the newly inserted node is to be added to the left of all the existing children, its label is generated by adding -2 to the last ordinal of the first child. *ORDPATH* supports insertion and update efficiently without relabeling any existing label; it is also efficient to determine the parent-child relationship. In our scenario, we use *ORDPATH* value instead of file directory path. For instance, in Figure 5, the value "1.1.1.1" (7 characters) represents the path "C:\Music\Hero.mp3"(17 characters) which reduces the storage space by 10 characters. Though, the length of the *ORDPATH* label will become long in case of deep trees and trees with large fan-out. Overall, the length of *ORDPATH* value is greatly shorter than that of file directory path.

B. Storage structure

For better managing the path tree, *NovaPIM* requires an efficient storage structure to maintain the compression dictionary. Two alternative ways exist. The first approach is to adopt the adapted *ORDPATH* Value Table. *ORDPATH* scheme uses a table with its schema as R(*ORDPATH*, *TAG*, *NODE TYPE*, *VALUE*), where *ORDPATH* is the encoding value of a file directory path by using *ORDPATH* labeling scheme, *TAG* represents a node label of a locator(from the file directory tree or internet), the other two fields, i.e., *NODE TYPE* and *VALUE*, are not used in *NovaPIM* and can be omitted therefore. In addition, we need another field called *RefCount*. In *NovaPIM*, a *concept* may contain/reference many items from various resources. An item can be referred to by different *concepts* at the same time. "*RefCount*" indicates how many times a file or a document is referenced. Its value is maintained dynamically. If it reaches zero, this entry can be removed from table to save space. When adding a path into the *Concept Space*, e.g., "C:\Music\Hero.mp3", the value of *RefCount* of each corresponding node ("C:", "Music", "Hero.mp3") will be added by 1; When deleting a path, the value of *RefCount* of each corresponding node will be reduced by 1. Hence, the new table schema becomes R' (*ORDPATH*, *TAG*, *RefCount*), one such example table is shown in TABLE 1 for Figure 5. The *Value table* is maintained dynamically. When a new shortcut or hyperlink needs to be encoded, we firstly look up it in this dictionary. If it exists, we replace it with its corresponding *ORDPATH* code. If not, we use *ORDPATH* labeling scheme to add it to the dictionary. When performing add, delete, move and decode, these operations involve traverse a path and need to visit several tuples in the table. Thus it leads to low efficiency.

The second approach is to use the encoding tree as the dictionary but with some adaptation. Each node has the encoding with the prefix removed, and adds *RefCount* attribute. The encoding example for Figure 5 after taking this approach is shown in Figure 6. The *RefCount* for each node is shown within the brackets of its encoding, indicating the total occurrences of this node in different paths. Similar to the *ORDPATH* Value Table, when adding/deleting a path into the *Concept Space*, the value of *RefCount* of each corresponding node will be increased/decreased by 1. The operation of moving a subtree can be seen as the synthesis of adding after deletion. Node insertion doesn't incur the re-encoding of other nodes; node deletion doesn't affect the relationship of ancestor/descendant, parent/child, and siblings. When encoding, we can get the path/locator encoding by traversing and concatenating each node's encoding corresponding to the path/locator labels; when decoding, we combine each node's label by traversing the dictionary tree according to the encodings. This approach can reduce the space overhead of the compression dictionary and is very efficient when encoding/decoding. When the hyperlinks scale is not too large, this approach is a good choice. This paper takes this approach.

Next, we describe the data compression algorithm below.

```
Algorithm: InsertHyperLink
Input: P – a hyper link to be inserted, DT- the dictionary tree
Output: the corresponding ORDPATH encoding of hyper link P, and
    the modified dictionary
1.Parse P into tokens(label names) {P1, P2, P3…} according
  separator (“\” , “/” or others)
2. Let current node $N_{curr}$←the root node of DT;
3. For each label TP in {P1, P2, P3…}, see if TP is among the labels
   of $N_{curr}$ .ChildNodes():
      (1) if exists: increase RefCount by 1 for the corresponding child
          node with TP as its label.
      (2) if not exists, create a new right-most child node with TP as
          its label for $N_{curr}$, encode this new node and set RefCount to
          1.
      (3) $N_{curr}$←the new child node;
4. Combining the ORDPATH encoding values for the nodes
   corresponding to P's label with “.” as their separator and return it.
```

Figure 7. Adding a new *HyperLink*

```
Algorithm: DeleteHyperLink
Input: P - a hyper link to be removed, DT- the dictionary tree
Output: the modified dictionary
1. Parse P into tokens(label names) {P1, P2, P3…} according
   separator (“\” , “/” or others)
2. Let current node $N_{curr}$←the root node of DT;
3. For each label TP in {P1, P2, P3…}, see if TP is among the labels
   of $N_{curr}$ .ChildNodes():
      (1)if exists: decrease RefCount by 1 from the corresponding
         node with TP as its label. If RefCount becomes 0, then delete
         this node and return; else $N_{curr}$←the child node with TP as its
         label;
      (2)if not exists, report error and return;
```

Figure 8. Deleting a HyperLink

C. Data Compression Algorithm

1) Shortcuts/hyperlinks Encoding

When adding a new item into Concept Space, the related information of shortcut or hyperlink of the item may need to be added into the dictionary tree. Here we take shortcut as the example for description. First, we separate a shortcut (e.g. “C:\Music\Hero.mp3”) into its individual parts (“C:”, “Music”, “Hero.mp3”) by the separator “\”; Next, we check whether the leading node (“C:”) exists or not; if exists, check whether the sequent node (“Music”) exists or not within its leading node's children; if the leading node doesn't exist, then encode the leading node into *ORDPATH* value. The remaining node (i.e. “Hero.mp3”) is handled the same way as stated above. Finally, for each node accessed, the value of *RefCount* will be added by 1, the *ORDPATH* value of the last node is the encoding value of the shortcut. The encoding algorithm is depicted in Figure 7.

2) Decoding ORDPATH value into shortcuts/hyperlinks

For a compression system, it is essential to get back the original shortcut/hyperlink by decoding *ORDPATH* value (say X = 1.1.1.1). By removing the rightmost component of X (always an odd ordinal) and then all rightmost even ordinal components [4], we get its parent (here 1.1.1) in *dictionary tree*. Such process continues until up to root. Eventually, we recover the original shortcut/hyperlink by connecting the node successively by adding the separator “\” in between.

3) Delete a shortcut/hyperlink

When deleting a shortcut/hyperlink from its corresponding *concept*, reduce the value of *RefCount* by 1 for all the nodes of shortcut/hyperlink according to *ORDPATH* value by visiting all nodes by the way explained in Section *2)* above. If the value of the *RefCount* becomes 0, the corresponding entry is deleted. The corresponding deletion algorithm is shown in Figure 8.

TABLE 2 EXPERIMENT DATASET 1

Paths	Avg Depth	Avg Length(B)	Nodes	Avg Fanout
500	4.8	33.5	329	46.86
1000	5.3	51.2	1011	21.04
2000	5.3	48.7	1955	11.70
3000	5.2	47.1	2669	10.34
4000	5.1	46.6	3999	7.56
5000	4.6	43.3	5006	8.56
7000	5.3	45.7	6307	15.20
11000	4.9	45.6	10036	9.67

TABLE 3 EXPERIMENT DATASET 2

Paths	Avg Depth	Avg Length(B)	Nodes	Avg Fanout
$5x10^4$	8.7	76.2	50085	12.54
10^5	8.8	75.9	100152	11.95
10^6	7.2	61.5	1000018	14.51

V. Performance Evaluation

A. Experimental Environment and Data Generation

The experiments were performed on an Intel Core 2 Duo 2.2GHz CPU with 2GB memory, running Window 7. We produce two data sets from the real-life file directories shown in TABLE 2 and TABLE 3. The first column indicates the total number of distinct shortcuts/hyperlinks; column two, three and five are the average values of the depth, length and fanout of the collections respectively; and the total number of nodes is given in the fourth column. Note that the fanout is the fanout of the dictionary tree. In TABLE 2, the average depth of the data set is 5, and the average path length is 45.7(Bytes). The data set in TABLE 3 shows that the average path depth is 8, and its average path length is 71.2(Bytes).

B. Experimental results

All results are shown in tables. The acronyms of the table header are explained below. *Paths* is the total number of distinct paths/locators; *RC* is the reference count *RefCount* which indicates the number of times a resource is referred to by different concepts; *ARC* is the average reference count; *DS* is the space occupied by the dictionary tree in KB; *ES* is the space consumed by encoding in KB; *CS* is the space consumed after compressing the data set(hyperlinks) in KB; it consists of two parts: the size of encoded data and the size of the dictionary tree and thus it holds that $CS = DS + ES$; *OS* is the space used without compression in KB; By comparing the compressed space *CS* to the original data size *OS*, we got the compression ratio *CR*(%), i.e., $CR = (OS - CS)/OS*100$.

TABLE 4 EXP. RESULT OF DATASET 1 WITH *RefCount* SET TO 1

Paths	RC	DS	ES	CS	OS	CR
500	1	7	4	11	16.37	33
1000	1	29.42	13.41	42.83	50.00	14
2000	1	51.84	28.99	80.82	95.19	15
3000	1	68.08	39.58	107.66	137.84	22
4000	1	100.71	66.53	167.24	182.03	8
5000	1	133.40	79.59	212.99	211.25	-0.8
7000	1	168.14	96.63	264.77	312.27	15
11000	1	262.31	182.39	444.70	490.03	9

TABLE 5 EXP. RESULT OF DATASET 2 WITH *RefCount* SET TO 1

Paths	RC	DS	ES	CS	OS	CR
$5x10^4$	1	1518	1222	2740	3723	26
10^5	1	3020	2504	5524	7414	25
10^6	1	21365	23975	45340	60048	24

TABLE 6 EXP. RESULT OF DATASET 1 WITH *RefCount* CONFORMING TO NORMAL DISTRIBUTION $N(0, 3^2)$

Paths	ARC	DS	ES	CS	OS	CR
500	1.85	7	7	14	30	53
1000	1.85	29	25	54	92	41
2000	1.89	51	55	106	180	41
3000	1.91	68	75	143	265	46
4000	1.91	100	127	227	346	34
5000	1.92	133	152	285	403	29
7000	1.88	168	182	350	584	40
11000	1.90	262	348	610	937	35

TABLE 7 EXP. RESULT OF DATASET 2 WITH *RefCount* CONFORMING TO NORMAL DISTRIBUTION $N(0, 3^2)$

Paths	ARC	DS	ES	CS	OS	CR
$5x10^4$	1.91	1518	2341	3859	7132	46
10^5	1.91	3020	4796	7816	14206	45
10^6	1.92	21365	45949	67314	115077	42

TABLE 4 ~ TABLE 9 illustrate the effectiveness of our compression scheme on storage space. The experimental results, where one hyperlink is only referred to once by a concept, are shown in TABLE 4 and TABLE 5. In this case, the shared prefixes contribute to the effect. From the results we know that the average compression ratio is about 14% and 25% respectively.

TABLE 6 ~ TABLE 9 give the results when a hyperlink is referred to more than once by several concepts. In TABLE 6 and TABLE 7, the reference count conforms to the normal distribution $N(0,3^2)$, and their average reference count is near 2; in TABLE 8 and TABLE 9, the reference count conform to the normal distribution $N(0,5^2)$ with their average reference count around 3.5.

The experimental results on data set 1 are shown in TABLE 4, TABLE 6 and TABLE 8. With the increasing of the average reference count, their average compression ratio grows accordingly from 14%, 39% to 52%. The experimental results on data set 2 is shown in TABLE 5, TABLE 7 and TABLE 9, their average compression ratio grows accordingly from 25%, 44% to 53% as the average reference count increases.

TABLE 8 EXP. RESULT OF DATASET 1 WITH *RefCount* CONFORMING TO NORMAL DISTRIBUTION $N(0, 5^2)$

Paths	ARC	DS	ES	CS	OS	CR
500	3.41	7	13	20	55	63
1000	3.42	29	45	74	170	56
2000	3.47	51	101	153	330	54
3000	3.52	68	139	207	486	57
4000	3.52	100	233	333	636	48
5000	3.53	133	280	413	741	44
7000	3.47	168	335	503	1076	53
11000	3.49	262.31	637	899	1713	47

TABLE 9 EXP. RESULT OF DATASET 2 WITH *RefCount* CONFORMING TO NORMAL DISTRIBUTION $N (0, 5^2)$

Paths	ARC	DS	ES	CS	OS	CR
$5x10^4$	3.50	1518	4281.7	5799.6	13042.7	55.53
10^5	3.50	3020	8775	11795	25988.3	54.62
10^6	3.51	21365	84019	105384	210434	49.92

VI. CONCLUSION AND DISCUSSION

This paper proposes to use the idea of concept space to manage personal information and exploit topic map as the underlying data model. Based on this, the paper presents the prototype system *NovaPIM*. *NovaPIM* integrates many desktop applications through application embedding and give a solution to the problem of physical data independence. A materialized view scheme is provided to view the file system from different perspectives according to user's own cognition. Users can define his concept through drag & drop without physically changing the directory structure. *NovaPIM* combines both the tree and graph model to organize and manage the data collection. For the diversity of personal data, any single data model is not sufficient. The combination of several data models may be the only right way.

With the help of shortcuts/hyperlinks, we represent the concept space in a graphical model. We adopted the *ORDPATH* label scheme to reduce the storage overheads of file directory path. The experimental results show its effectiveness.

As pointed out before, the lack of schema is the intrinsic nature of PIMS. To solve this issue, data mining and machine learning techniques should come into play to discover the schema among data collection and find the relationships between concepts. An appropriate view definition language is desired to face the schema lack environment. All these are in our future research agenda.

ACKNOWLEDGMENT

This work is supported in part by Zhejiang Provincial NSF Project(Y1090096) and the NSFC Project (61070042).

REFERENCES

[1] M. Lansdale. The psychology of personal information management. Applied Ergonomics, 19(1), 1988, pp.55-66.
[2] W. Jones. Finders, keepers? The present and future perfect in support of personal information management. First Monday,2004,http://www.firstmonday.dk/issues/issue9_3/jones/index.html.
[3] L. H. DENG. Library and Information Mathematics, Northeast Normal University, 1983.
[4] P. O'Neil, E. O'neil, S. Pal, L. Cseri, G. Schaller, and N. Westbury, "ORDPATHs: Insert-Friendly XML Node Labels", Proceedings of the ACM SIGMOD, 2004, pp.903-908.
[5] R. Alkhatib and M. H. Scholl. Compacting XML Structures Using a Dynamic Labeling Scheme. BNCOD, 2009, pp.158-170.
[6] X. Dong and A. Halevy. A Platform for Personal Information Management and Integration. CIDR, 2005.
[7] S. T. Dumais, E. Cutrell, J. J. Cadiz E., G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: A system for personal information retrieval and re-use. SIGIR, 2003, pp.72-79.
[8] S. Chaudhuri, R. Ramakrishnan, G. Weikum. Integrating DB and IR Technologies: What is the Sound of One Hand Clapping?. CIDR, 2005.
[9] C. C. Shilakes and J. Tylman, "Enterprise Information Portals", Merrill Lynch, 16 November, 1998.
[10] V. Bush. As we may think. Atlantic Monthly, 176(1), 1945, p:101-108.
[11] J. Gemmell, G. Bell, R. Lueder, SM Drucker, C. Wong. MyLifeBits: Fulfilling the Memex vision. Proc. of the 10th ACM International Conference on Multimedia, 2002, pp.235-238.
[12] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, V. Sinha. Haystack: A customizable general-purpose information management tool for end users of semistructured data. CIDR, 2005, pp.13-26.
[13] J.P. Dittrich, M. Antonio, M. Salles. iDM: A unified and versatile data model for personal dataspace management. VLDB, 2006, pp.367-378.
[14] L. Blunschi, J. Dittrich, O. R. Girard, S. K. Karakashian, and M. A. V. Salles. A Dataspace Odyssey: The iMeMex Personal Dataspace Management System. CIDR, 2007, pp.114-119.
[15] W. Jones, J. Teevan. Personal Information Management. Communications of the ACM, 49(1), 2006, pp.40-42.
[16] D. K. Barreau. Context as a factor in personal information management systems. Journal of the American Society for Information Science, 46(5), 1995, pp.327-339.
[17] Y. Y. YAO. Concept Formation and Learning: A Cognitive Informatics Perspective. Proceedings of the Third IEEE International Conference on Cognitive Informatics, 2004, pp. 42–51.
[18] Y.Y. Yao. A step towards the foundations of data mining. Data Mining and Knowledge Discovery: Theory, Tools, and Technology V, B.V.Dasarathy(Ed.), The International Society for Optical Engineering, 254-263, 2003.
[19] Topic Maps - XML Syntax. http://www.isotopic-maps.org/sam/sam-xtm/2006-06-19/
[20] S. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, D. C. Robbins. Stuff I've seen: a system for personal information retrieval and re-use. SIGIR conference, 2003, pp.72–79.
[21] S. Fertig, E. Freeman, and D. Gelernter. Lifestreams: An alternative to the desktop metaphor. In Conference Companion on Human Factors in Computing Systems: Common Ground, 1996, pp. 410–411.
[22] S. Davies. Still Building the Memex. Communications of the ACM, 2011, 54(2):80-88.
[23] S. J. Kaplan, M. D. Kapor, E. J. Belove, R. A. Landsman, and T. R. Drake. Agenda: A personal information manager. Commun. ACM 33, 7 (July 1990), pp.105–116.
[24] J. Conklin and M. L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. ACM Transactions on Office Information Systems, Vol. 6, No. 4, October 1988, pp.303-331.
[25] K. Shoens, A. Luniewski, P. Schwarz, J. Stamos, J. Thomas. The Rufus System: Information Organization for Semi-Structured Data. In VLDB, pp.97-107, 1993.
[26] W. Wang, A. Marian, T. D. Nguyen. Unified Structure and Content Search for Personal Information Management Systems. International Conference on Extending Database Technology, pp. 201-212 , 2011.
[27] S. Abiteboul, O. Benjelloun, T. Milo. Positive Active XML. PODS Conference, 2004, pp.35-45.
[28] S. Whittaker. Personal Information Management: from information consumption to curation. Annual review of information science and technology (ARIST), Vol. 45 (2011), pp. 3-62.

Liang Huai Yang is a professor at Zhejiang University of Technology. He received the BSc. degree in Information Science (Department of Mathematics) in 1989 and the PhD degree in Computer Science from Peking University in 2001. He assumed a research fellow position at National University of Singapore during 2001~2005. He has published about 40 papers in major conferences and journals in the database field. He has served on the program committee of some database conferences, and as reviewers of some journals such as Information Sciences, Information Systems, International Journal of Electronics and Computers, etc.

Lee Mong Li is an Associate Professor and Assistant Dean in the School of Computing at the National University of Singapore (NUS). She received her Ph.D. in Computer Science from NUS in 1999. She was awarded the IEEE Singapore Information Technology Gold Medal for being the top student in the Computer Science program in 1989. Mong Li joined the Department of Computer Science, National University of Singapore, as a Senior Tutor in April 1989 and was appointed Fellow in the School of Computing in February 1999. She was a visiting Fellow at the Computer Science Department, University of Wisconsin-Madison, from September 1999 to August 2000 and Consultant at Quiq Incorporated, USA from June to August 2000. Her research interests include the cleaning and integration of heterogeneous and semi-structured data, database performance issues in dynamic environments, and medical informatics. Her work has been published in database conferences such as ACM SIGMOD, VLDB, ICDE and EDBT, data mining conference ACM SIGKDD and database conceptual modeling conference (ER).