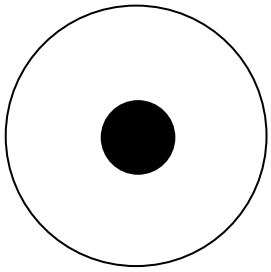


TIR

The IPSI BgD Transactions on Internet Research

Multi-, Inter-, and Trans-disciplinary Issues in Computer Science and Engineering

A publication of
IPSI Bgd Internet Research Society
New York, Frankfurt, Tokyo, Belgrade
July 2006 Volume 2 Number 2 (ISSN 1820-4503)

Special Issue: Selected Research Results from the School of Electrical Engineering, University of Belgrade, Serbia

Table of Contents:

A Proposed Hybrid Approach for Patent Modeling
Scekic, Ognjen; Popovic, Djordje; and Milutinovic, Veljko 3
A Reflective Memory System for Personal Computers
Tomasevic, Milo; Protic, Jelica; Savic, Savo; Jovanovic, Milan; Grujic, Aleksandra; and Milutinovic, Veljko 7
Assembly Language in Modern Technologies still Faster than HLL: Myth or Reality
Micic, Milos; Etinski, Maja; and Milutinovic, Veljko 13
One Approach of Efficient Management of Zillion Signatures
Rudan, Sasa; Kovacevic, Aleksandra; Babovic, Zoran; Jovic, Darko; Milutinovic, Veljko; and Milligan, Charles 17
Cache Clearing System
Babovic, Zoran; Jovic, Darko; Cakarevic, Vladimir; Milosavljevic, Ivan; Stevanovic, Marija; Minic, Predrag; and Milutinovic, Veljko. 22
Data Assurance in a Conventional File System
Rudan, Sasa; Kovacevic, Aleksandra; Milligan, Charles; and Milutinovic, Veljko. 27
Data Mining: A Brief Overview and Recent IPSI Research
Radivojevic, Zaharije; Cvetanovic, Milos; and Milutinovic, Veljko 32
Internet Application Testing
Prijic, Aleksandar; Jovic, Darko; and Milutinovic, Veljko 38
Patent Maps: A Simpier Way to Search Patents in the Light of Prior Art
Micanovic, Mina; Milutinovic, Veljko; and Milligan, Charles 43
SwanLink: Mobile P2P Environment for Graphical Content Management System
Popovic, Jovan; Bosnjakovic, Andrija; Minic, Predrag; Korolija, Nenad; and Milutinovic, Veljko 47
New Modifications of Selection Operator in Genetic Algorithms for the Traveling Salesman Problem
Radovic, Marija; and Milutinovic, Veljko 53
Designing of an XPath Engine for P2P XML Store
Jovic, Darko; and Milutinovic, Veljko 59

Call for Papers for the IPSI BgD Conferences

The IPSI BgD Internet Research Society

The Internet Research Society is an association of people with professional interest in the field of the Internet. All members will receive this TRANSACTION upon payment of the annual Society membership fee of €100 plus an annual subscription fee of €1000 (air mail printed matters delivery).

Member copies of Transactions are for personal use only

IPSI BGD TRANSACTIONS ON INTERNET RESEARCH

www.internetjournals.net

STAFF

Veljko Milutinovic, Editor-in-Chief	**Aleksandra Jankovic, Journal Manager**
Department of Computer Engineering University of Belgrade POB 35-54 Belgrade, Serbia (381) 64-1389281 (tel)	**Bojana Milasinovic, Journal Manager**
	Ana Naumovic, Journal Manager
	IPSI BgD Internet Research Society
vm@etf.bg.ac.yu	tir@internetjournals.net

EDITORIAL BOARD

Adeli, Hojjat	**Gonzalez, Victor**	**Milligan, Charles**
Ohio State University, Ohio,	University of Oviedo, Gijon,	StorageTek, Colorado,
USA	Spain	USA
Blaisten-Barojas, Estela	**Janicic, Predrag**	**Milutinovic, Veljko**
George Mason University, Fairfax, Virginia,	The Faculty of Mathematics, Belgrade,	IPSI, Belgrade,
USA	Serbia	Serbia
Crisp, Bob	**Jutla, Dawn**	**Neuhold, Erich**
University of Arkansas, Fayetteville, Arkansas,	Sant Marry's University, Halifax,	Research Studios Austria, Vienna,
USA	Canada	Austria
Domenici, Andrea	**Karabeg, Dino**	**Piccardi, Massimo**
University of Pisa, Pisa,	Oslo University, Oslo,	Sydney University of Technology, Sydney,
Italy	Norway	Australia
Flynn, Michael	**Kiong, Tan Kok**	**Radenkovic, Bozidar**
Stanford University, Palo Alto, California,	National University of Singapore,	Faculty of Organizational Sciences, Belgrade,
USA	Singapore	Serbia
Fujii, Hironori	**Kovacevic, Branko**	**Rutledge, Chip**
Fujii Labs, M.I.T., Tokyo,	School of Electrical Engineering, Belgrade,	Purdue Discovery Park, Indiana,
Japan	Serbia	USA
Ganascia, Jean-Luc	**Patricelli, Frédéric**	**Takahashi, Ryuichi**
Paris University, Paris,	ICTEK Worldwide, L'Aquila,	Hiroshima University, Hiroshima,
France	Italy	Japan

A Proposed Hybrid Approach for Patent Modeling

Scekic, Ognjen; Popovic, Djordje; and Milutinovic, Veljko

Abstract—***In an effort to find a general model which could capture the essence of any concept, we chose to narrow down the first part of our research to patents only. Patents can be considered as adequate first-step substitutes for concepts in general, because of their diversity and the precision of their descriptions.***

Our aim is to define a model which could allow detection of individual concepts and relationships among them, both within and across patent boundaries.

The approach is based on a hybrid solution – employing existing conceptual indexing techniques for extraction and hierarchical organization of individual concepts, and RDF/OWL descriptions for application-specific data.

1. Introduction

EVER since the first computers appeared in the 1940s, people started dreaming of "intelligent machines", capable of not only processing given data, but also understanding it. An important branch of computer science – Artificial Intelligence (AI), is dedicated to achieving that goal. Research efforts in the field of AI provided many useful insights into different new techniques for knowledge extraction and manipulation. Many ideas originating from AI research are being further developed today in a number of related fields. Knowledge representation, Data Mining and Semantic Web are only a couple of those research fields, sharing a common goal – taking data manipulation abilities of a computer to a higher level. We must first make a clear distinction between the terms *data* and *information.* In this context, the term *information* can be defined as a set of relations among different pieces of data that together provide some knowledge. Pieces of data deprived of such relations carry no knowledge.

We will also introduce another term here – concept, to denote a piece of information regarding any concrete or abstract category, as humans perceive it.

In order to take a computer's data manipulation abilities to a higher level, we first need to make data representation and data processing more similar to human's perspective. In order to do that, we need a new layer which could act as an intermediary between a human's perception of reality and a computer's internal way of data representation. Similar aspirations in the field of programming languages led from the initial machine binary language to the creation of object-oriented languages. The missing layer should be able to handle concepts at the level of abstraction as close as humans do. The layer should, basically, be able to deliver two things:

- Define a way to model concepts,
- "Translate" the abstract concepts to a lower level, and vice-versa.

The first point is more of a philosophic and logic nature, whereas the second one is more a technical issue. However, the two issues are closely related, since the complexity and breadth of the first point are inevitably narrowed by the limitations of the second point. The research area that explores these issues is called *concept modeling*.

A good concept model should provide:

- A way of identifying new concepts,
- A way of identifying relations among the new and the existing concepts,
- A way of searching and processing existing concepts to create new concepts/knowledge.

There exist many different models [1] devised for many specific applications. However, the ultimate goal is to find a universal model, powerful enough to capture the essence of any concept.

At IPSI Belgrade we decided to pursue this goal, and devised three different approaches. One of them is to be described in the following part of the paper.

2. Problem Statement

In the effort to find a general model which could capture the essence of any concept, we chose to narrow down the first part of our research to patents [5] only. Patents can be considered as adequate first-step substitutes for concepts in general, because of their diversity and the precision of their descriptions. Furthermore, patents have some other very convenient characteristics for this kind of research:

- They are described by a very formal, structured language – claims.

Manuscript received May 30, 2006. This work was supported in part by the Sun Microsystem Company, United States of America.

Scekic Ognjen, Popovic Djordje, and Milutinovic Veljko are with the Faculty of Electrical Engineering, University of Belgrade, Serbia.

- Each patent is a novel concept, defined in terms of existing low-level concepts.
- Each patent is often based on or closely related another one.

Nevertheless, while thinking of a suitable way to model patents we came upon a number of problems:

- How to create a model that has a uniform structure, and can therefore be used for any concept, i.e. to be able to capture the essence of any concept (a patent, in our case)?
- How should these models be linked without creating a myriad of direct links, leading to problems of exponential order when storing and searching such complex structures?
- How could we bridge the gap between natural language and a machine-processable model?

Being a separate problem, we decided to adopt an existing solution to try to solve the last issue – *conceptual indexing* [2, 3, 4] – a method for extracting concepts, word constructs and sentence fragments from any text in natural language, and arranging them in an index.

2.1 Conceptual Indexing

Conceptual indexing is a technique that can improve people's ability to find information in textual materials, using semantic relationships among concepts and natural language processing. This technique is used for indexing and organizing information in structures called conceptual taxonomies that can be used for browsing and information retrieving. The taxonomies represent structured networks of concepts based on conceptual relationships of these concepts.

Conceptual indexing technology can be divided into three major parts that work closely together (Figure 1):

1. Concept extractor – identifies words and phrases to be indexed. It also keeps record of number and places of occurrences of these words.
2. Concept assimilator – analyzes a concept phrase to determine its place in conceptual taxonomy. In other words, it creates the mentioned taxonomy.
3. Conceptual retrieval system – uses conceptual taxonomy to make connections between requested and indexed items. It first uses the concept extractor to identify the requested words and phrases. After that, it uses the Concept assimilator to determine the connections between concept phrases extracted from the query and those placed in taxonomy.

Using the conceptual indexing technique one can relate the terminology of a query to the terminology of some textual information that is placed inside this taxonomy, and conclude with a certain degree of probability that the essence of a text is similar to what is asked in a query. In this way, conceptual indexing can be used to bridge the gap between a natural language and a machine - processable model.

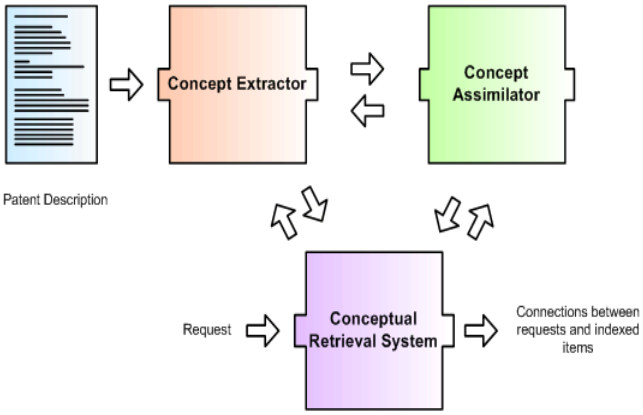


Figure 1: Components of a conceptual indexer

2.2 The Seven Ws

The first step of our research was to talk to ordinary people, who are in no way related to this project. We asked them to try to explain how they create a picture of a concept in their heads? The answers were either a number of other concepts that they associate with the concept they were asked to describe, or in the form of answers to some of the 7 Ws (WHAT, WHO, WHEN, WHERE, WHY, WHICH, HOW). (Giving associations can be thought of as giving answers to the WHAT question, so basically it boils down to the same thing.) However, depending on the type of concept, different Ws were used. For example, when describing birds as a class of animals, they would use only certain W associations - WHAT and HOW mostly. But, when they were to describe a historic event, they would use all the 7 Ws. Our conclusion was that only the WHAT associations provided general facts about *any* concept, and were *always* present. Other Ws provided extremely useful additional information when present, but were not always present.

We decided to use a conceptual indexer, to go through the text we want to model into a concept, and create an index of terms, phrases and sentence fragments (later referred to as *terms*). We will use this "small index" – *descriptive index,* (its size is approximately 1-5% of the analyzed text) as a list of WHAT associations. Other Ws will be used depending on the field of application.

For example, when describing patents:

- WHICH – All the numbers that describe a patent (e.g. application number),
- WHEN – All the dates that describe a patent (e.g. when it was filed),
- WHERE – e.g. addresses of the inventors,
- WHO – e.g. inventors, examiners, attorney,
- etc.

As we can see, each of these Ws can have several sub-categories, which are application-specific. Therefore, we could use RDF/OWL [6, 7, 8, 9] statements to capture all this information

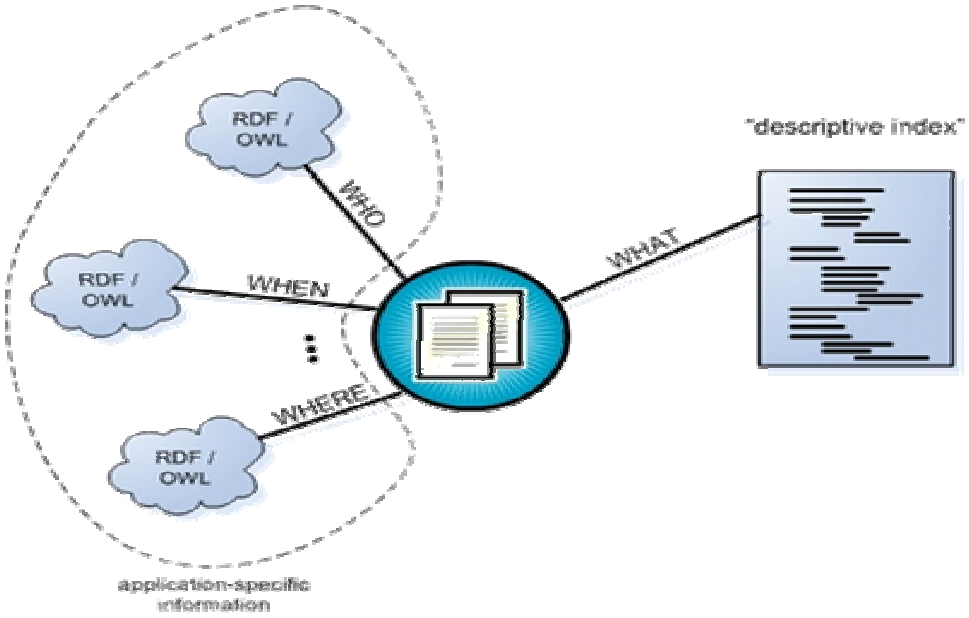


Figure 2: A simplified patent model

properly. This could give us the exactness and the expressiveness like that of ontology, but without driving us into problems of complexity, because we could limit the RDF statements to atomic concepts only. This way, we can easily use automated reasoners to process this data.

3. EXISTING SOLUTION AND THEIR CRITICISM

3.1. Ontology

If we wanted to place concepts in an ontology, we would either have to use a well-established ontology (which might not suit our needs), or create a new one. In the latter case, we could create several different ontology/taxonomies/structures, depending on what information we want to capture, and how we want to capture it. However, if we later want to merge our ontology into a different one, we would have to carefully examine existing relationships, determine potential equivalent classes, create new links, etc. This is because in a regular ontology we have all kinds of links, not just between atomic concepts. Such approach works fine within closed community with specific needs, where already exists a well-defined basic ontology structure, and the community members have a good knowledge of how to model new concepts in terms of the existing ones. In such cases, that approach provides a very information-rich model.

In our case, we think it might be better to use only simple relations between atomic concepts. The result will be a loss of much of the expressiveness, but with the benefit of a reduced complexity. This loss is to be compensated through the use of indices.

We hope the research done by the other groups of our team could prove very helpful in this area.

3.2. Indices

For example, let us take the simplest possible definition, for a bird:

Our index might then contain the following associations: creature, wings, feathers, eggs, fly.

Our approach does not offer the possibility to explicitly state the fact that some birds do not fly, as ontology does, but it does allow us to create a simple model, similar to what humans have on their mind when they think of a bird.

Having enough associations, one can create a model with a considerable degree of accuracy. The more associations we have, the clearer picture we get. It is not a question of making an exact guess, but making a near, or very near guess. This is usually good enough for many applications.

It is also important to keep track of how many times a term is mentioned in the text, because it affects its descriptive power. For example, in the claims section of a patent we used, the most frequent terms were “synthetic grass [10]” and “playing surface [9]”. Clearly, these terms represent the essence of what is being described.

	Conceptual indices	RDF/OWL ontologies
Major advantages:	Linear-complexity structures	Very expressive and precise
	Provide basic subsumption relations	Based on First-Order Logic
	Provide built-in knowledge on low-level concepts	Supported by W3C
Major drawbacks:	Incapability of establishing explicit relations among high-level concepts	Great complexity
	Incapability to create precise models	

Figure 3: Comparison of the two methods

However, this is only because we *know* what "synthetic", "grass" and "surface" are. So, at some level, we need to have some intrinsic, built-in knowledge, so that all the other concepts can then be described in terms of these basic concepts. Conceptual indexing could provide us with that built-in basic language understanding.

Basically, our research is a hybrid approach aiming to use advantages of one technique to eliminate the drawback of the other one.

4. PROPOSED SOLUTION

While the RDF/OWL part of the model is used to link atomic concepts links between more abstract concepts (patents in this case) can be established dynamically, through the use of a joint index. This joint index – *system index*, is created by merging indices of individual concepts, while retaining the links between its terms and their concepts of origin. The merging is done once again by the conceptual indexer, by processing all the descriptive indices to produce the system index. The key advantage here is that adding, removing and searching for concepts is quite easy, and requires little time, because there are no direct, explicit links among them (except among parts that are modeled with RDF/OWL, but those are far less complex issues because we have a predefined, limited structure in such a case).

For example:

When describing two different vaccines we would probably make a frequent use of terms like: vaccine, inactivated antigens, immune response, etc.

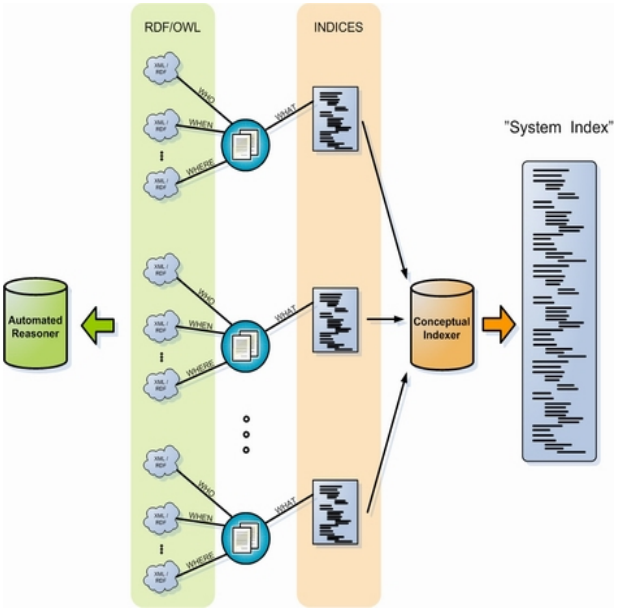


Figure 4: Top-level scheme

By determining overlapping terms we create dynamic, implicit links among similar concepts. Number of such implicit links can be used to express similarity among concepts.

Of course, we need not to have identical terms to establish that there is some level of similarity. Thanks to the existence of basic subsumer-subsumee relations in a conceptual index, we can also infer similarity of patents based on the number of occurrences of similar terms.

The results of the experiment we made on a set of patents referring to communication devices and protocols showed that there is a significant number of overlapping terms. If the number of occurrences of terms is propagated vertically via subsumer-subsumee relations and gradually incremented as we move up the structure, then we get a clear picture of what the patent is actually about. The same experiment also showed that the above-mentioned method does provide a credible way of establishing similarity among patents.

However, the algorithm for establishing similarity can only be tweaked empirically, and its performance may vary according to the field of application. We think the structure of the claims section of a patent document is very appropriate for our approach also because the claims impose a frequent use of key terms.

5. CONCLUSION

Our idea is still in the first stage of development. Its key advantages are its general applicability and reduced complexity, at the price of reduced precision; a consequence of using indices.

Further research is needed to explore the quality and feasibility of the proposed solution. However, we expect that the combination of OWL/RDF structures and indices might produce a satisfactory performance/exactness ratio.

REFERENCES

[1] Omerovic, S., Savic, D., Tomazic, S., "A Survey of Concept Modeling," Faculty of Electrical Engineering, University of Ljubljana, Slovenia (to appear).

[2] Woods, W. A., "Conceptual Indexing: A Better Way to Organize Knowledge," Technical report, *Sun Microsystems Laboratories*, 1998.

[3] Woods, W. A., Bookman, L. A., Houston, A., Kuhns, R. J., Martin, P., Green, S., "Linguistic Knowledge Can Improve Information Retrieval," *Proc. of the Applied Natural Language Processing Conference* (ANLP-2000), Seattle, 2000.

[4] Dobrov, B. V., Loukachevitch, N. V., Yudina, T. N., "Conceptual Indexing Using Thematic Representation of Texts," *Scientific Research Computer Center,* Moscow State University, Moscow, 1998.

[5] http://www.uspto.gov – U.S. Patent Office

[6] Scekic, O., Bojic, P., "An Overview of OWL and its Role in Semantic Web Architecture," *YU-INFO 06,* Kopaonik, Serbia & Montenegro, 2006.

[7] Klyne, G., Carroll, J., "Resource Description Framework (RDF): Concepts and Abstract Syntax," *W3C Recommendation*, http://www.w3.org/TR/rdf-schema, 2004.

[8] Hayes, P., "RDF Semantics," *W3C Recommendation,* http://www.w3.org/TR/rdf-mt, 2004.

[9] Patel-Schneider, P. F., Hayes, P., Horrocks, I., "OWL Web Ontology Language: Semantics and Abstract Syntax," *W3C Recommendation,* http://www.w3.org/TR/owl-semantics/, 2004.

A Reflective Memory System for Personal Computers

Tomasevic, Milo; Protic, Jelica; Savic, Savo; Jovanovic, Milan; Grujic, Aleksandra; and Milutinovic, Veljko

Abstract— ***In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: how to design a cost-effective board that connects a personal computer as a node in an RMS system and how to increase the efficiency of the basic RMS solution. Existing solutions to the problem are briefly surveyed, and their drawbacks are underlined. Then, the essence of the proposed solution is presented: designing a board that interfaces a personal computer to the RM bus. The fifth part defines conditions and assumptions of the analysis to follow. Elements of a simulation study are presented in part six. The conclusion, as a summary of new ideas, acquired during this project is presented in the part seven.***

1. INTRODUCTION

DISTRIBUTED shared memory (DSM) architecture represents a successful hybrid of shared memory multiprocessors (SMP) and distributed systems (DS) that inherits advantages of both architectures [1]. All processors in a DSM system access shared logical address space, which simplifies the programming model, and makes many applications portable from SMP systems. However, underlying architecture is typically a DS, so the logically shared memory is physically distributed, which means that, with an appropriate interconnection network, the system has good potential for scalability.

A DSM system is composed of nodes connected by hypercube, LAN, bus hierarchy, etc). Each node can be a single-processor system, or even an SMP (like in Dash system [2]). In both cases, it contains a portion of local memory mapped to the unique shared address space of DSM. In general, shared data stored in local memories can migrate from one node to another, depending on the access pattern of an application. It can also have multiple copies replicated in local memories of several nodes, which requires that all copies have to be kept up to date according to some consistency rules defined by memory consistency model (MCM).

There is a variety of algorithms for consistency maintenance [3], and the majority of them can be classified as multiple reader single writer (MRSW) and multiple reader multiple writer (MRMW). Consistency related information is stored in system tables or directory that can be centralized or distributed, corresponding to the responsibility for management of DSM system.

The meaning of the word “consistency” has been the most variable aspect of DSM systems from the early efforts in this area, in mid-80 till their today’s maturity. Memory consistency model (MCM) determines its meaning by defining the legal ordering of memory references issued by some processor, as observed by other processors in the system [4, 5]. While the most strong MCMs, sequential and processor consistency do not distinguish between ordinary shared data accesses and synchronization accesses, more sophisticated models like weak release, lazy release, and entry consistency take advantage of the fact that shared data accesses are typically protected by some synchronization operations. Therefore, the points of enter/exit of critical sections and barriers are used as suitable moments when some consistency-related actions should take place. The global trend in most sophisticated MCMs [6, 7] is to minimize the necessary communication between nodes, paid by making the programming model more complex. This trend distances DSM systems from their original goal (simplification of the programming model) but greatly improves their performance by reducing and/or hiding the communication latency.

The mechanism that provides the illusion of shared memory in a physically distributed memory system can be implemented in software, in hardware, or in cooperation of hardware and software, which results in a hybrid implementation. The first DSM systems were implemented on the underlying network of workstations, typically connected by Ethernet [8]. They were implemented in software, as runtime systems and/or modifications of the original operating system. Some of the systems developed more recently also included (or were primary based) on compiler modifications [6]. On the other hand, a class of DSM systems implements the DSM mechanism in hardware, which brings full transparency to all software layers, besides their performance advantages.

Manuscript received May, 2006. This work was supported in part by the Encore, HP.

Tomasevic Milo, Protic Jelica, Savic Savo, Jovanovic Milan, Grujic Aleksandra, and Milutinovic Veljko are with Faculty of Electrical Engineering, University of Belgrade.

Virtually the best potential for further improvements in the field of DSM is in combining software approach that can better explore the characteristics of the application, known by the programmer, with the hardware approach, that helps in performing critical operations more efficiently. Therefore, majority of the most recent DSM systems can be classified as hybrid implementations.

Reflective memory systems (RMS) [9] belong to the class of hardware implemented DSM. Their main characteristic is the non-demand, write-through update consistency mechanism. The majorities of DSM systems generally keep track of valid copies of shared data present in local memory of the node, and if data is not present in the valid state, fetch it from another node(s) on demand. Unlike them, an RMS performs all necessary updates automatically, using broadcast over the RM bus as an appropriate mechanism. Each node maps some segments in its local reflective memory to the shared address space, treating these segments as open "windows." Each write operation to an open window is immediately propagated, and received by all other nodes that also have opened windows containing that particular address. This method makes the write operation costly, but the read operation is not delayed, since the valid data is always present in the local reflective memory.

2. *Problem Statement*

Although the first RM systems [10] appeared about a decade ago, they suddenly gained emerging popularity in the last few years. In the mean time, many variants of demand-based DSM systems were built, proving the validity of the concept, but few of them became commercial products. The reason for that can be found in the high overhead of operating system and the considerable time consumption of the layered protocol software. In addition, communication latency in such a system is hard to predict, which is not tolerable for real-time applications.

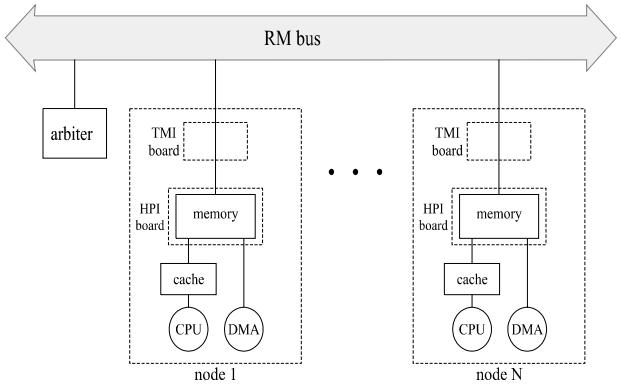


Figure 1: Architecture of the reflective memory system (RMS).
Legend: CPU - central processor unit; DMA - direct memory access unit.

All these problems are avoided in RM systems, with their simple and efficient hardware mechanisms for consistency maintenance. The communication in these systems is usually overlapped with computation, and the shared memory access time is predictable and relatively low.

The earlier RMS systems were usually implemented in minicomputer environment, and used for a wide range of applications, from real-time to on-line transactions processing. However, in early 90's the expansion of DSM concept to a NOW (Network of Workstations) and NOPC (Network of Personal Computers) became one of the most important development trends in this field [11]. Encore was one of the first who understood this trend, and concluded that using a standard and relatively powerful machine such as PC, as a node in the RM system, could lead to a cost-effective system with high configuration flexibility. That was the motivation to start this project, with prospective to build a system which costs less, connects more nodes and supports bigger traffic.

The first technical goal of the project was the design of a fully operational RMS board, which plugs into a PC, and turns it into a node within the bus-based DSM system, using reflective memory approach. The board had to be compatible with the RM bus interconnect (Encore's proprietary bus), and also had to provide a larger amount of local memory.

Besides the strictly technical goal of designing the board, Encore was also interested in some theoretical research, which would go beyond the currently operational systems and result in the development of the know-how, which could create new break-through in the near future. The requests for this part of the project were not so strict: while the management stuff of our research sponsor insisted on the solution for 64 or 128-node RM system, engineering stuff was skeptic and suggested the generation and elaboration of several ideas that would probably improve the performances of the system, and their detailed simulation analysis. The essence of this strategy was to achieve a relatively large global performance improvement through a series of small incremental improvements. This was probably the only way to go, since the RM concept existed for years, and a considerable advance could be obtained only through a superposition of relatively small advances in different directions.

3. *Existing Solutions*

There are multiple types of systems based on non-demand write-through update, some of them referred to as mirror memory systems [12] or replicated memory systems [13]. However, the term reflective memory is the most frequently used. This class of systems includes some commercial products (Modcomp [14], Systran

[15], and VMIC [16]) targeted to the real-time market, as well as ATC's ring-based approach [17] and DEC's Memory Channel [18]. The SHRIMP project of the Princeton University [19] also belongs to this class. An almost exhaustive survey [20] covers the majority of RMS systems. The first RM system was made and patented by Gould Electronics in 1985. Encore acquired Gould in 1989, and updated the system, which later evolved into a number of RMS variants. The latest developments resulted in passing the technology from Encore to DEC corporation, and its promotion of DEC Memory Channel, which is now Compaq. This paper focuses on Encore's development of RMS.

In the RMS of [21] each node connected to the RM bus consists of a processor with cache, local memory, DMA device, and the RM board with memory and interface to the RM bus (see Figure 1). Bus arbitration is centralized, using a round-robin synchronous arbitration algorithm. Processor (responsible for word transfers) is connected to the RM board via the local processor/memory and/or the host system bus. The DMA unit (responsible for block transfers) is attached via the host system bus. Transmit/receive mapping tables are used to configure local memory pages as reflective (shared) or private (non-shared), as well as for address mapping, since copies of a shared page could be at different addresses on different nodes. On each write to the local RM memory, a check is made in the transmit mapping table if the address belongs to an open window, and, if so, data word together with the translated address in reflected shared space is put into the transmit FIFO buffer. The node with a pending write requests the RM bus from the RM bus controller. When the node obtains the bus grant, bus write operation (address + data) is initiated on the RM bus. All other nodes check if the address falls into a local open receive window. In that case, the data and translated receive address are written into the receive FIFO. Finally, data from the receive FIFO are forwarded to the local RM memory.

The state-of-the-art among RM systems is RM/MC (Reflective Memory/Memory Channel) which supports blocked updates. The RM/MC system is based on the multiplexed synchronous 64 bits wide bus, connecting up to eight processing nodes, with the 15000 feet maximal distance (for a fiber optic cable). RM/MC supports both single word and block update transfers. Each node in the system is composed of two boards: TMI (Transition Module Interface) for the RM/MC bus and HPI (Host Port Interface) for the host system bus. There are implementations for the VME bus, the EISA bus, and the PCI bus.

Since RMS and similar systems are based on asynchronous, non-instantaneous communication, they typically can not guarantee strict consistency. However, if all writes are seen by all processors in the system in the same order (which is true for centralized arbitration and no bypassing in transmit/receive FIFO buffers) sequential consistency can be achieved. More relaxed consistency models can be implemented if we allow some bypassing and/or deleting in transmit/receive FIFO buffers. However, the latest developments in this area introduced some hybrid solutions [22]. Encore proposed LAM - a software-based consistency maintenance for RM/MC underlying hardware. The LAM uses library routines to manage allocation of the RM space and to synchronize the accesses to this space, providing the programmer with the means of entry consistency-like MCM.

4. Proposed Solution and Its Advances

The first goal of the project was to design a board that interfaces a personal computer to the RM bus. Since the interface between the local RM memory and the RM bus is similar for different computers, it is moved into a separate hardware module called TMI (Transition Module Interface). The TMI is connected with the host RM board by a special cable with multiplexed address and data lines.

Host board (Figure 2) contains the system bus interface, local RM memory, and memory arbiter and control logic. The memory on the host board and all data paths are 32-bits wide. There are no FIFO buffers between the memory and the system bus, since the on-board memory cycle is synchronized with the system bus cycle. The system bus burst cycles are also supported. The board is designed to have 16MB or 64MB of DRAM organized in four standard 72-pin 36-bit SIMMs. The system base address of the memory is programmable in 16MB/64MB quantum. Memory arbiter gives the highest priority to the DRAM refresh logic, then to the system bus interface, and at last to the receive FIFO (data coming from the RM bus). The priority can be changed when the receive FIFO is almost full, or on the system bus time-out.

The host board also contains transmit translation SRAM. This mapping table keeps the shared/private indication bit for each 8K-segment of the memory, and a global RMS address where a block should be mapped if it is shared. From the memory side, address and data FIFOs are accessed by separate buses, while on the host-TMI cable side address and data lines are multiplexed to reduce the bulk of cabling. The TMI demultiplexes the address and data lines, puts the address and data into separate FIFOs, and connects them to the RM bus. The receive translation SRAM is placed on the TMI to keep the information about mapping from the RMS address space to the local RM memory. If an address belongs to a block (which is reflected),

data and the translated address are written into the receive FIFOs on the TMI board, then multiplexed, sent to the FIFOs on the host board, and finally written to the memory.

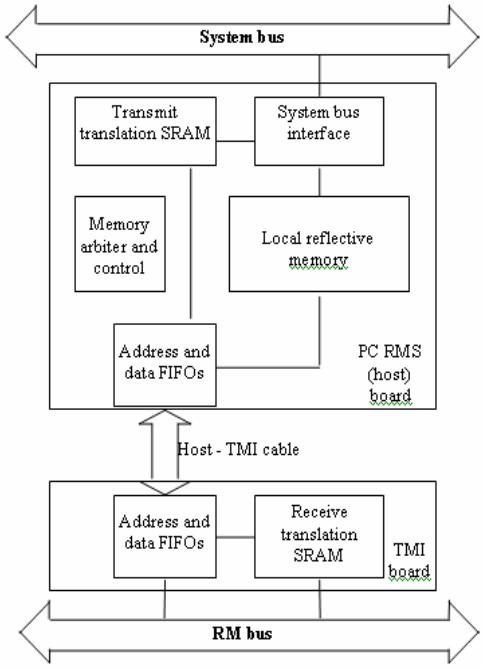


Figure 2: Implementation scheme of the PC-RMS. Legend: TMI - Transition Module Interface.

5. Conditions and Assumptions of the Research to Follow

After the design of the PC-RMS board was finished, it was carefully analyzed, in order to notice the drawbacks of the RM concept and its implementation, and then, based on its criticism, to propose and elaborate several improvements [23]. Each proposed solution was separately simulated, in order to examine its potential effects, and evaluate its impact in comparison with other proposals. After that, a specific decision-making methodology (modified for DSM) [e.g., 24] was applied, in order to choose a subset of proposals that will give the solution with a relatively large global performance improvement, achieved through a series of small incremental improvements. However, the final decision about the incorporation of proposed improvements into the final product was left to the research sponsor; our task was only to give recommendations, having in mind results of particular simulations. Therefore, simulation of the overall system, implementing all proposed improvements was not done, having in mind time limitations, and the fact that overall effects will be measurable on the implemented system.

6. Simulational Analysis

Here we present four suggested improvements that we found the most interesting:

Improvement #1: Caching of shared RM regions - in the existing RM systems, the shared reflected regions of the RM memory are not cached. RM board appears as a memory slave on the host system bus, and can not initiate any action on this bus when a write from the RM bus occurs. Since the effects of caching to the overall system performance are well known [25], we suggested some changes that make caching possible in RM systems. Each block in the RM memory should be associated one bit, which denotes its caching status (C-bit). When a block is cached, its C-bit should be set. After that, when a write from the RM bus to a shared location comes, the C-bit will be checked, and if it is set, invalidation will be sent to that cache, and the C-bit will be reset. Only one additional bit per block, together with the very simple comparator logic represents reasonable complexity paid for significant performance gain (Figure 3a). The simulation shows that caching of the shared memory segments results in the improvement of about 10% for the CPU utilization. The improvement is better for smaller node counts, since for the larger node counts, the contention on the RM bus caused by multiple invalidations becomes significant. The benefits of caching should have been even more pronounced if the system follows write-back policy.

Improvement #2: Double RM bus - since the shared bus represents notorious bottleneck in bus-based systems, the natural solution for the restricted scalability is to employ multiple busses. We have analyzed the solution with a double RM bus. When the need for accessing the RM bus arises, the bus request is issued on both buses at the same time, yet only one bus will be granted that transfer (whichever grant comes first). The proposed solution can be efficient both in cases of transient and permanent bus overloads, in terms of traffic of writes to the shared memory regions. The overall effect is that the average time to obtain the bus is decreased, which results in the higher system throughput. This also offers a potential for increased system reliability. However, the cost of this approach is significant because of the introduction of the second bus, and the hardware complexity of the bus interface logic is increased. This complexity increase can be justified only by a satisfactory performance improvement (Figure 3b). While in the conditions of saturated bus traffic in the single-bus RM system average waiting time for the bus grant linearly increases, in the double bus system this time is insignificant and practically insensitive to the system size.

Improvement #3: Improved RM bus arbitration The RM bus controller assumes equal priority for all requests coming from each node. However, the overload of the nodes can be different, so some FIFOs can be full, while FIFOs in other nodes are empty. The solution for this problem is to give a higher priority to the overloaded node, from the moment it reaches the “almost full” condition. When the number of items held in the transmit FIFO buffer reaches certain threshold,

the "urgent need" for the RM bus will be signaled. Then, the RM bus arbiter temporarily departs from the round-robin policy, and gives the upper hand to that node. This priority will be assigned temporarily (only for one transfer on the RM bus). When no "urgent need" signals are asserted, the arbiter is back to the round-robin arbitration. For the purpose of announcing the priority request, bus lines reserved for the node ID can be used, since they are free during the RM bus transfer. This approach introduces no additional bus lines, but requires very careful timing. The simulation shows that in conditions of the increased bus traffic, this method improves processor utilization, but only for a lower number of nodes, while in the hypothetical systems with 12 or 16 nodes it has no advantages (see Figure 3c).

*Improvement #4:*Write Filtering - the essence of this improvement is the filtering of subsequent write requests issued to the same address by the same node, while these requests are sitting in the transmit FIFO of the writer. This method helps in reducing the traffic on the RM bus. According to the number of previous writes in the FIFO that should be compared to the issued write operation, we have proposed three solutions: (a) variant “all” - comparison with the entire contents of the transmit FIFO, (b) variant “last 4” - comparison with the last four writes, and (c) variant “last 1” - comparison with the previous write only. Performance improvements for all these variants are given in Figure 3d.

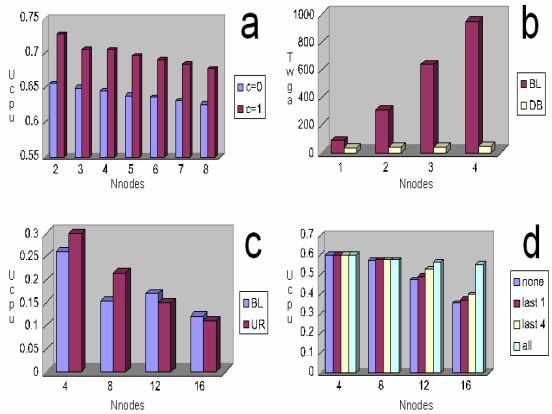


Figure 3: Performance results for four proposed improvements.

Legend: Ucpu - processor utilization; Nnodes - number of the RM nodes; Twga - average waiting time for the RM bus grant; C=0 - without caching; C=1 - with caching; BL - baseline system with single bus and round-robin policy; DB - double bus system; UR - urgent request arbitration.

Description: Figure 3a shows the comparison of processor utilization with and without caching. Figure 3b shows the comparison of average waiting time for the bus grant in a single (BL) and double (DB) bus RM systems. Figure 3c shows comparison of processor utilization for a system with the round-robin policy (BL) and the dynamic priority policy with urgent request arbitration (UR). Figure 3d shows the impact of four write-filtering variants on the processor utilization.

The best effects of this solution can be expected for applications with high processor locality of memory references (long write runs). When applying this improvement, one must be careful about the consistency model assumed by the programmer, since the loss of some write operations is allowed only for relaxed consistency model.

Simulations were performed for no write filtering, and for variants “last 1”, “last 4” and “all”. Processor utilization is almost the same for all four variants in the systems with less then 8 nodes; however, in systems with larger node counts, the difference becomes significant, in favor of write filtering applied to the longer sequences in the transmit FIFO buffer.

7. CONCLUSION

Fortunately, hard work always leads to results. While exploring the possible enhancements of RM concept, in order to achieve the theoretical goal of the project, after the incubation period has passed, we come up with some general ideas, applicable in the wider area of DSM systems, as well as some solutions in the field of cache consistency in multiprocessor systems. Even two years after the project was finalized, these ideas were applicable in other projects, and resulted in several thesis and research papers, summarized in [26]. The most important new concept was the STS idea, which suggests the separation of data that expresses predominantly spatial locality from the data that expresses predominantly temporal locality, and different treatment of these two data spaces [26]. The continued efforts inspired by the write filtering idea presented here resulted in even more sophisticated improvements of the RM system [27], while the application of some concepts found in RM to the cached data in multiprocessor systems resulted in a new solution for lazy prefetching [28].

Finally, was it worth the effort? Instead of the research paper published in a respected journal (which is probably what every researcher dreams of when starting a new project), we ended up with a working board (to the best of our knowledge, the first implemented multiprocessor system with reflective memory DSM mechanism, using the PC-based computer nodes) and a lot of experience and accumulated knowledge, that resulted in a mature view of the field, presented in [MILU2000]. The detailed survey of DSM concepts and systems that we did, inspired by this project, and the great collection of papers we read, did not finish in our archive; instead, we also published a tutorial book with the state-of-the art works in the field of DSM [29].

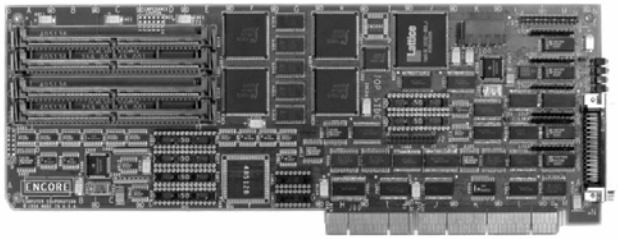


Figure 4: The photo of the PC-RMS board

This project was a pioneer effort from two points of view: it was organized in cooperation between a university and an industrial company located on two distant continents before the concept of globalization and virtual project teams became commonplace, and it resulted in the first commercial DSM system based on the network of PCs. Projects like this bring great benefits to the industry, allowing efficient outsourcing of some research and development activities to the appropriate research centers. Thanks to the information superhighway, geographical limitations no longer apply when choosing the most suitable partners. On the other hand, university researchers get the opportunity to cope with the real world problems, and to incorporate their knowledge into the technology development. However, the obstacles to the process of globalization and sharing the results with the research community are industrial secrets and proprietary limitations, which slow down the free flow of information. The avenue we opened with this project was proved to be the prospective one: in the years after we have completed our research, several other approaches in the field of DSM for PC were initiated and proved to be successful.

REFERENCES

[1] Protić, J., Tomašević, M., Milutinović, V., "Distributed Shared Memory: Concepts and Systems," *IEEE Parallel and Distributed Technology*, Summer 1996, pp. 63-79.

[2] Lenoski, D., Laudon, J., et al., "The Stanford DASH Multiprocessor," *IEEE Computer*, Vol. 25, No. 3, March 1992, pp. 63-79.

[3] Stumm, M., Zhou, S., "Algorithms Implementing Distributed Shared Memory," *IEEE Computer*, Vol. 23, No. 5, May 1990, pp. 54-64.

[4] Gharachorloo, K., et al., " Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors," *Proceedings of the 17th International Symposium on Computer Architecture*, 1990, pp. 15-26.

[5] Protić, J., Tartalja, I., Tomašević, M., "Memory Consistency Models for Shared Memory Multiprocessors and DSM Systems," *Proceedings of the 8th Mediterranean Electrotechnical Conference Melecon '96,* Bari, Italy, May 1996, pp. 1112-1115.

[6] Bershad, B., N, Zekauskas M., J., Sawdon, W., A., "The Midway Distributed Shared Memory System," *Proceedings of the OMPCON 93,* February 1993, pp. 528-537.

[7] Keleher, P., Cox, A., L., Dwarkadas, S., Zwaenepoel, W., "TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems," *Proceedings of the USENIX Winter 1994 Conference,* January 1994, pp. 115-132.

[8] Fleisch, B., Popek, G., "Mirage: A Coherent Distributed Shared Memory Design," *Proceedings of the 14th ACM Symposium on Operating System Principles,* ACM, New York, 1989, pp. 211-223.

[9] Lucci, S., Gertner, I., Gupta, A., Hedge, U., "Reflective-Memory Multiprocessor," *Proceedings of the 28th IEEE/ACM Hawaii International Conference on System Sciences*, Maui, Hawaii, USA, January 1995, pp. 85-94.

[10] "RMS Functional Specification," *Encore Computer Corporation*, Fort Lauderdale, Florida, USA, 1990.

[11] Li, K., "Development Trends in Distributed Shared Memory," *Panel of the 22th International Symposium on Computer Architecture*, Santa Margherita Ligure, Italy, June 1995.

[12] Maples, C., Wittie, L., "Merlin: A Superglue for Multicomputer Systems," *COMPCON'90*, March 1990, pp.73-81.

[13] Oguchi, M., Aida, H., Saito T., "A Proposal for a DSM Architecture Suitable for a Widely Distributed Environment and its Evaluation," *Proceeding of the Fourth IEEE International Symposium on High Performance Distributed Computing* (HPDC-4), Washington, D.C., USA, August 1995, pp.32-39.

[14] Furht, B., "Architecture and Performance Evaluation of the MMM," *IEEE TCCA Newsletter*, March 1997, pp. 66-75.

[15] "SCRAMNet+ Protocol Description," *SYSTRAN Corporation*, Dayton, Ohio, USA, 1995.

[16] "VMIC's Reflective Memory Network," *VME Microsystems International Corporation,* Huntsville, Alabama, USA, 1995.

[17] Ramanjuan, R., Bonney, J., Thurber K., "Network Shared Memory: A New Approach for Clustering Workstations for Parallel Processing," *Proceedingsof the Fourth IEEE International Symposium on High Performance Distributed Computing* (HPDC-4), Washington, D.C., USA, August 1995, pp. 48-56.

[18] Gillett, R., "Memory Channel Network for PCI," *IEEE Micro*, February 1996, Vol. 16, No. 1, pp. 12-18.

[19] Blumrich, M., et al., "Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer," *Procedings of the 21th Annual International Symposium on Computer Architecture*, IEEE Computer Society Press, Los Alamitos, California, USA, 1994, pp. 142-153.

[20] Jovanović, M., Milutinović, V., "An Overview of Reflective Memory Systems," *IEEE Concurrency*, Vol. 7, No. 2, Spring 1999.

[21] Gertner, I., "The Reflective Memory / Memory Channel System Overview*,* " Technical Report, *Encore Computer Systems,* Fort Lauderdale, Florida, USA, 1993.

[22] Denton, R., Johnson, T., "Distributed Shared Memory Using Reflective Memory: The LAM System, " Technical Report (TR96-021), *University of Florida*, Gainesville, Florida, USA, July1996.

[23] Savić, S., Tomašević, M., Milutinović, V., "Improved RMS for the PC Environment," *Microprocessor Systems*, Vol. 19, No. 10, December 1995, pp. 609-619.

[24] Hoevel, L., Milutinović, V., "Terminology Risks with the RISC Concept in the Risky RISC Arena," *IEEE Computer*, December 1991, pp. 136.

[25] Tomašević, M., Milutinović, V., (editors) "Cache Coherence Problem in Shared Memory Multiprocessors: Hardware Solutions, " *IEEE Computer Society Press*, Los Alamitos, California, USA,1993.

[26] Milutinović, V., "Some Solutions for Critical Problems in The Theory and Practice of DSM," *IEEE TCCA Newsletter*, September 1996, pp. 7-12.

[27] Jovanović, M., Tomašević, M., Milutinović, V., "A Simulation-Based Comparison of Two Reflective Memory Approaches," *Proceedings of the 28th Hawaii International Conference on System Sciences (HICSS-95),* IEEE Computer Society Press, Los Alamitos, California, USA, 1995, pp. 140-149.

[28] Milenković, A., Milutinović, V., "Lazy Prefetching," *Proceedings of the HICSS-98*, Mauna Lani, Hawaii, USA, January 1998.

[29] Protić, J., Tomašević, M., Milutinović, V., (editors) "Distributed Shared Memory: Concepts and Systems," *IEEE Computer Society Press*, Los Alamitos, California, 1998.

Assembly Language in Modern Technologies still Faster than HLL: Myth or Reality

Micic, Milos; Etinski, Maja; and Milutinovic, Veljko

Abstract— ***When technologies change, realities change, as well. Many believe that advances in the technology of optimizing compilers cause the programming in a high-level language to create a faster code (compared to assembly language programming), due to the fact that individual programmers can never optimize their hand-written code as successfully as the optimizing compilers do it. This paper demonstrates that it is not always the case, and clarifies the issue for the case of a selected algorithm of importance for a number of modern applications. Architectural changes demonstrate how powerful can be the usage of advanced assembly instructions [1].***

1. Introduction

1.1 Development and Variety of C Compiler Optimizers

THE development of compilers is focused on optimizations. There are many different goals. The first of them is to speed up execution of generated code. The diminishing of code size is also one of the main goals. So, there are many compiler options that determine whether optimization is needed and which kind of it should be used. Compiling time with optimization takes more than without it. That issue isn't relevant for this paper. Our main aim is to observe the time needed for the execution. For that purpose we choose the newest versions, MS Visual Studio 2005 and older version of GCC compiler – version 3.4.4. These versions were chosen in order to see achievements in the area of code optimization.

1.2 Concrete Platform

Compilers also tend to use advantages of a concrete platform so the generated code could get the maximum from it. Our work is based on Intel Pentium III processor. This processor uses 16 KB of data L1 cache, 16 KB instruction L1 cache and 256 KB of L2 cache. L1 cache is synchronous write-back. Pentium III supports following technologies: MMX (Multimedia extensions) adds instructions for fast execution of typical primitives in multimedia processing such as vector manipulations, matrix manipulations and big block moves; SSE (Streaming SIMD Extensions) expands the SIMD (Single Instruction, Multiple Data) execution model introduced with Intel MMX technology by providing a new set of 128-bit registers and the ability to perform SIMD operations on packed single precision floating point values.

1.3 Fast Fourier Transform

In this work we used an FFT (Fast Fourier Transform) algorithm as an example. It is a fast algorithm (in the sense of algorithm complexity) for computing discrete Fourier transform dividing the problem into two of smaller dimensions.

2. Problem Statement

Because of long lasting efforts to develop better optimizers the speed of generated code becomes better. That arise a question about using the assembly code in order to gain on execution speed. Assembly programmers could be expensive resource comparing to the achievement of code performance that they make. Code written in some HLL could be faster than the one written in assembly language. According to this, some aspects of assembly programming must be carefully considered. Sometimes compilers don't use hardware specific instructions that trained programmer is able to implement. This could be explained by the fact that compilers don't have the capability for intelligent abstract methodologies that could solve the problem in global manner. In such situation inline assembly code could be a good solution. We should observe that on the FFT algorithm example written in C high level language.

3. Existing Techniques

There are different ways for faster implementation of some problem. One way is to find better algorithm. In the case of FFT there are different algorithms developed in last few decades. A lot of effort was given to their comparison and analyzes. The other aspect is implementation of a concrete algorithm.

Manuscript received Jun 1, 2006.

Micic Milos and Milutinovic Veljko are with Faculty of Electrical Engineering, University of Belgrade, Serbia.

Etinski Maja is with the Faculty of Mathematics, University of Belgrade, Serbia.

Performances of implementation depend on hardware and software used for development.

3.1 Categories of Compilers Optimization

Techniques in optimization can be divided into two groups depending on the scope of optimization: global and local. Local techniques are much more common and easier to develop. The main ideas for optimization are based on next tasks: avoiding redundancy, making less code, jump reducing, code locality, extracting information from code etc. They are all based on making a faster code, making less code or both using existing advantages of modern computer technologies such as pipeline and cache. GCC and MS Visual Studio 2005 compiler have both options for enabling different types of these optimizations.

3.2 DSP

The other way to speed up FFT implementation is to use a specific architecture. Digital signal processors (DSP) have important role in a great number of computationally intensive signal processing applications in fields such as communications. Many of today's DSP architectures employ a deep pipeline that supports high-frequency operation and a very long instruction word (VLIW) that enables the execution of multiple instructions in parallel. This kind of architecture is suitable for FFT usage.

4. PROPOSED ANALYSIS

4.1 Different Compilers

Using different compilers to compile the same code of an FFT algorithm we could compare execution time of programs that compilers made. As we mentioned, observed compilers are GCC 3.4.4 and Visual Studio 2005 C compiler. Usually default option is code generation without optimization. It is reasonable because an optimization takes time, makes debugging more difficult and very often isn't necessary. Optimizer efficiency could be discussed if we compare execution speeds without optimization and with compiler optimization.

4.2 Modifications Using Assembler

We are also interested in a rate of improvement gained by the usage of an assembly inline code. Testing compiler optimized code and the new one with inline assembly will give as a result the difference between execution time of C code without and C code with inline assembly. The smaller difference means the better optimizer.

4.3 Possible Optimizations Using Assembler

There is a variety of possible optimization techniques in the context of inline assembly programming. Depending on algorithm structure different ones could be suitable. Sometimes it is possible to reduce size of a part of code in order to be possible putting it in instruction cache at once. The other technique is to gain more from pipeline mechanism changing order of jump instructions and by reducing their number. Of course the code semantics must remain the same. The inline assembly optimization used in this paper is based on arithmetic instructions that provide one kind of parallelism. The used set of instructions is SSE set of MMX instructions. It is used in the most inner loop so, small amount of gained time, multiplied by many times should significantly reduce the execution time.

4.4 Variations of Input Size

Input size variations are included. The size of data cache is limited so at one point data cache won't be enough large to store all program data. The input is generated randomly. The time for the input generation isn't included in execution time. The input presents some values of function whose discrete Fourier transform should be calculated. Sizes of inputs that will be used are from the range: 1KB - 16MB. Value of arithmetical mean carried out from many measurements for the same input size will be presented.

5. EXPERIMENTAL DETAILS

Figure 1 show that GCC compiler is better for compiling pure C code without inline assembler. This achievement is not as significant as the next one shown on figure 2 when using inline assembly.

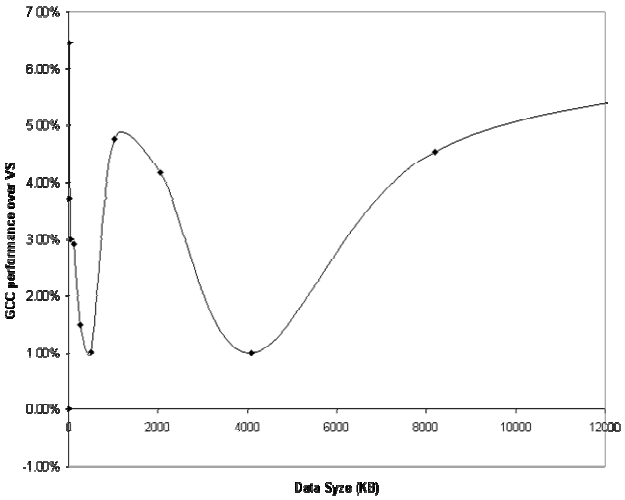


Figure 1: GCC performance over MS Visual Studio 2005 compiler using speed optimization

On the figure 2 we have results for execution times for both GCC and VS compilers before and after inline assembly implementation. Input data sizes are small enough to fit into cache. Figure 3 shows us the same characteristics but for input data that can not fit into cache.

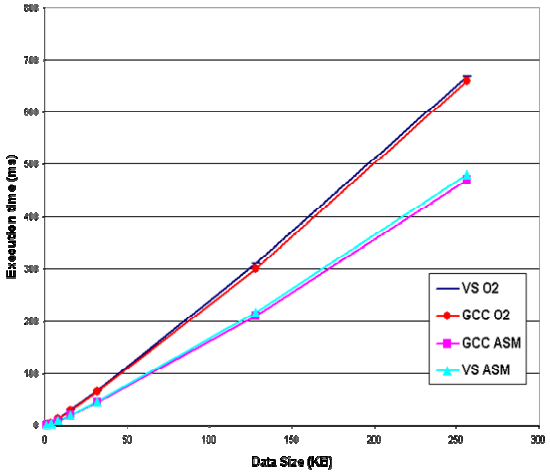


Figure 2: Execution time analysis for cache hit input data sizes

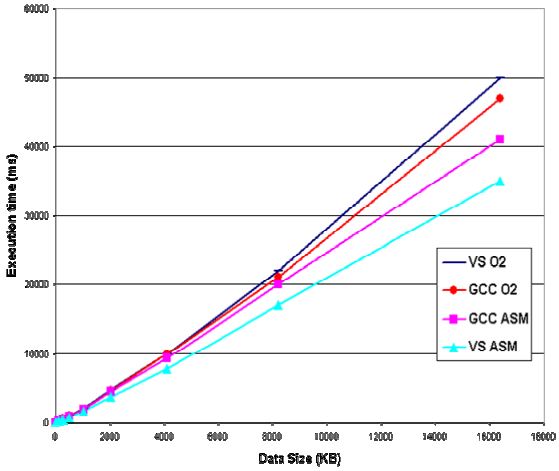


Figure 3: Execution time analysis for cache miss input data sizes

It can be seen that performances are better when using inline assembly as shown on figure 4 and 5. Performance achievement is shown comparing C code without inline assembly. 100 % performance is for pure C code.

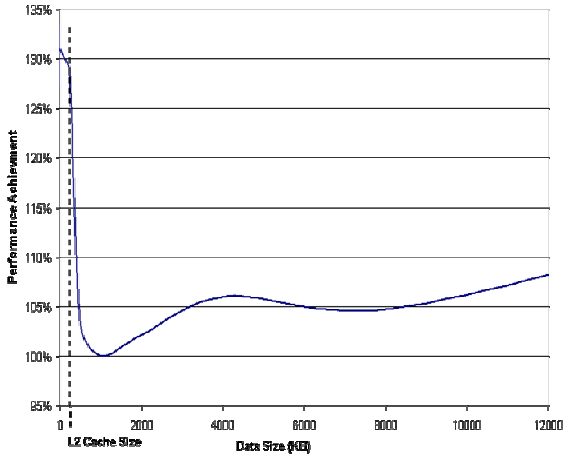


Figure 4: GCC 3.4.4 performance achievement using inline assembly code

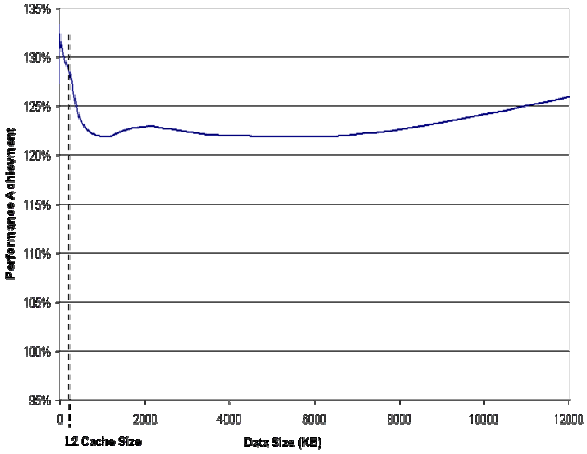


Figure 5: MS Visual Studio 2005 performance achievement using inline assembly code

It can be seen that MS VS 2005 achieves better performance for cache miss data sizes. The maximum of achieved performance is 30 % above the pure C code performance in both cases.

6. Conclusion

6.1 FFT Algorithm Based Conclusion

Due to the nature of FFT algorithm it was possible to accomplish its performance by replacing a part of code from the inner loop. This replacement was based on set of instructions which are independent from each other. That is suitable for parallel implementation with SSE Pentium set of instructions.

6.2 Discussing the Myth

The important question about assembly language today is whether there is a need for its usage at all. In this paper we have showed that there are some situations where assembly inline code could make obvious improvement despite the great set of various compiler optimizers' techniques.

The code replacement that we made isn't possible without reasoning. Simulation of reasoning could be a good development direction for compilers. For now, inline assembly is still reality for achieving greater performance for specific problems on specific platforms.

Appendix with Math Related Details

Evaluating a discrete Fourier transform would take O (n^2) arithmetical operations. Fast Fourier Transform (FFT) is an algorithm which reduces the number of computations from O (n^2) to O (n log n). The idea is to break problem into two smaller using the Danielson Lanczo's lemma (1942.):

$$\sum_{n=0}^{N-1} a_n e^{-2\pi i n k / N} = \sum_{n=0}^{N/2-1} a_{2n} e^{-2\pi i (2n) k / N} + \sum_{n=0}^{N-1} a_{2n} e^{-2\pi i (2n+1) k / N}$$

$$= \sum_{n=0}^{N/2-1} a_n^{even} e^{-2\pi i n k / (N/2)} + e^{-2\pi i k / N} \sum_{n=0}^{N/2-1} a_n^{odd} e^{-2\pi i n k / (N/2)}$$

This is break up of transform of dimension n into transforms of dimension n/2 and that is radix-2 case. It is possible to divide problem of size n=n1n2 into smaller DFTs of sizes n1 and n2 what is called mixed-radix case.

FFT was popularized by a publication of J. W. Cooley and J. W. Tukey in 1965. But similar idea was known to Carl Friedrich Gauss what was discovered later.

There are many FFT algorithms. Some of them are PFA, Rader-Brenner algorithm and Bruun's algorithm.

REFERENCES

[1] McNeley, K., Milutinovic, V., "Emulating a CISC with a RISC," *IEEE Micro*, Vol. 7, No. 1, February 1987, pp. 60-72.

[2] Chung, C., Xiangdong, F., "Achieving Better Code Optimization in DSP Design," 30.12.2005. www.us.design-reuse.com/articles/article8123.html.

[3] Hiroyuki, S., Yutaka, M., "A Type System for Optimization Verifying Compilers," 30.12.2005. www.is.titech.ac.jp/ppl2004/proceedings/p007.pdf.

[4] Milutinović, V., "Surviving The Design of Microprocessor and Multimicroprocessor Systems," *A Wiley Interscience*, New York, USA, 2000.

[5] Rokić, J., "Implementation of the fastest FFT algorithm," Graduate paper, *Faculty of Electrical Engineering, Belgrade*, Serbia, 2004.

[6] "IA-32 Intel Architecture Software Developer's Manual," 30.12.2005, www.intel.com/design/pentium4/manuals/index_new.htm .

[7] Ganapathiraju, A., et. al., "Contemporary View of FFT Algorithms," 30.12.2005, ardra.hpcl.cis.uab.edu/publications/documents/PDSP/ganapath_etal_1998-iasted.pdf.

[8] Balducci, M., et. al., "Benchmarking of FFT Algorithms," *Proceedings of the IEEE*, Blacksburg, VA, USA, April 1997.

[9] Cipra, B., A., "Faster than a Speeding Algorithm," 30.12.2005, www.siam.org/siamnews/11-99/speed.pdf.

[10] Allen, R., Kennedy, K., "Optimizing Compilers for Modern Architectures," *Morgan Kaufmann Publishers Inc*, San Francisco, CA, USA, 2001.

[11] Benjamin, C., "Types and Programming Languages the Next Generation," 30.12.2005, http://www.cis.upenn.edu/~bcpierce/papers/tng-lics2003-slides.pdf.

One Approach to Efficient Management of Zillion Signatures

Rudan, Sasa; Kovacevic, Aleksandra; Babovic, Zoran; Jovic, Darko; Milutinovic, Veljko; and Milligan, Charles

Abstract—*In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: How to manage efficiently billion file signatures from a specially introduced new file signature management layer; this problem is important because it enables the files signatures to be handled in a fast way. Existing solutions to the problem are briefly surveyed in the third part, and their drawbacks are underlined; each surveyed piece of research is analysed according to the same template. Then, the essence of the proposed solution is presented: Efficient storage of 1billion 20-byte digital signatures, their fast lookup, insert and delete, fast rebuild of the storage digital signature index; it also includes primitives that can be directly ported into hash functions and other appropriate mechanisms used for management of file signatures; this idea has several versions. The fifth part defines conditions and assumptions of the analysis to follow. Analytical analysis of all above is presented next. Elements of a simulation study, to compare performance, are presented in part seven. The conclusion is presented in the part eight.*

1. INTRODUCTION

NEW storage systems deal with billion of files. These files are usually stored on classical hard disk based storages. This environment is characterized with inefficiency of hierarchical file systems.

New concepts of raw file systems use digital signatures as Uniform Resource Identifier's (URI) for stored files. This implies a need for extremely fast digital signature management, i.e. searching, storage, and deleting of digital signatures in the domain of $2^{20*8}=2^{160}\approx 10^{48}$ values.

One of the basic feature of digital signatures is sharp miscorrelation between two digital signatures that come (are calculated) from two extremely correlated documents (even if documents differ only in one bit). It implies an approximately uniform division of digital signature values created from a finite number of documents.

Manuscript received January 30, 2006. This work was supported in part by the StorageTech.

Sasa Rudan, Aleksandra Kovacevic, Zoran Babovic, Darko Jovic, and Veljko Milutinovic are with Faculty of Electrical Engineering, University of Belgrade, Serbia.

Charles Milligan is with StorageTech.

2. PROBLEM STATEMENT

The Digital Signature Index Mechanism represents a service used by the archive platform. This service provides fast and reliable lookup/insert functionality. The archive system is designed to handle up to 1 billion unique data files. Each file is identified by a unique digital signature, calculated using at least two different algorithms.

The archive system has to be able to quickly determine if a data file is either a unique data file or a duplicate of an existing file. This is accomplished by generating the digest on the file, then querying the existing digital signatures looking for a match.

The Digital Signature Index Mechanism has to be durable to system failures, redundantly stored, and secured from unauthorized manipulation.

The primary challenges are:

1. Efficient storage of 1 Billion 20-Byte digital signatures.
2. Fast lookup, insert and delete of a digital signatures.
3. Fast rebuild of the storage digital signature index.

The digital signature storage mechanism should be used as a lookup system only. It does not act as a system of records for digital signatures. The index lookup returns either a duplicate status, or a unique status. If the status is unique, the signature is stored in the index.

The archive system executes lookups/inserts in two distinct ways: real time and batch mode. Batch mode provides log files with the results of each lookup/insert request.

The signature index should allow multiple readers, and a single writer.

Upon system restart, the signature index needs to recover to a state that is consistent with the reset of the archive system. The signature index should be able to be rebuilt quickly and efficiently.

3. Existing Solutions and Their Criticism

This chapter will present two solutions which are currently in use for management of the storage systems.

The first of the proposed solutions is a log-structured file system, which is presented at the Berkeley, University of California by Mendel Rosenblum and John K. Ousterhout [1]. The essence of this solution is based on the assumption that files are cached in main memory and that increasing of memory size will make a cache more and more effective. A log structured file system writes all modifications to disk sequentially in a log-like structure, thereby speeding up both file writing and crash recovery. The log is the only structure on disk; it contains indexing information so that files can be read back from the log efficiently. In order to maintain large free areas on disk for fast writing, the log is divided into segments and uses a segment cleaner to compress the live information from heavily fragmented segments.

Another solution of the storage management is based on the use of specialized storage devices known as "Appliances." This solution is developed at the University of Wisconsin-Madison by John Bent and his associates [2]. The appliances are designed solely to serve files to clients. These devices usually have the ability to adapt to the characteristics of the underlying hardware and operating system. The appliance developed by this team is called NeST, the open-source, user-level, software-only storage appliance.

4. Proposed Solution

Our proposed solution has several versions; all of them are developed in order to improve all of requests and drawbacks of other solutions. The main advantage of our model is its simplicity. It does not require additional memory and hardware.

4.1 Linear Area of Region to Bit Mapping

This represents a simple, but powerful solution for our problem. It is presented by a simple bit array structure. For each equivalent width region from the whole digital-signature-value domain there is one corresponding bit. In this way, we have some kind of or-hash function inside the specified regions. Figure 1 specifies the concept.

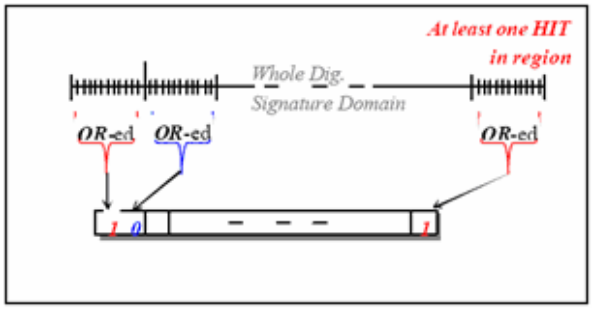


Figure 1: Bit Array Structure

4.1.1 Client-System Communication Protocol

Client-System communication is quiet simple. Client sends the request about specific digital signatures. Our system component checks in local structures and responds with an appropriate answer. Answer is one of the following:

NO – stands for absolutely no existence of the specified Digital Signature (DS) in the whole system.

YES – stands for an absolutely positive statement that the specified DS exists somewhere in the system.

Also in the subcomponents of the system (i.e., volatile storage memory subcomponent) we can get the following answers:

POSSIBLE_NO – This means that system subcomponents "believes" that DS does not exist in the system.

POSSIBLE_YES – This means that system subcomponents "believes" that DS exists in the system.

Now we can explain more precisely the client-system communication. When our DSM (DS Management) system component receives request from the client, it proceeds DS to the "volatile" system subcomponent which retrieves answer (NO or POSSIBLE_YES) from "region to bit" array. If the answer is NO, it can be sent back to client. If the answer is POSSIBLE_YES, we have to find absolute answers. Therefore, we access to the *Disk Regions* structure which resides on disk and is explained later in this chapter. There will be only one disk access since DSs are arranged on appropriate way.

4.1.2 Required Disk Organization

Our proposal is to use a dedicated hard disk which will only serve for managing digital signatures. This will result in Disk Regions structure that holds digital signatures organized by regions in which they falls. The main idea is to organize storing of DSs to provide direct access to disk sector that contains requested DS. It has to involve gaps that can be populated in later system growth. Since each region is covered with 0.125 digital signatures it is quite reasonable (because of uniform division) to allocate 0.25 DSs per region. We have control over the hard disk so we can choose that each sector (common size of 512 bytes per sector is assumed in this calculation) holds RegionsPerSector = 512 / 20 / 0.25 = 102 regions that are "mixed" inside the sector but ordered between sectors.

This organization also gives us space of 12 bytes for (global) specific fields and for a pointer (for the case of overloading). This will results in about 40 GB of disk consumption.

4.1.3 Process of Structure Creation

Initial structure creation is quite simple. We have to go through each digital signature stored in our system and to determine to which region it belongs to.

This determining process could be done easily if each region is mapped into a specific bit of digital signatures. In that case we just have to use these bits to access the appropriate bit in the

Bit Area Structure. When we find a specific bit we just set its content to 1.

4.1.4 Process of Adding Signatures

Adding of new digital signature to system is just iteration in the process of creation of the array structure. It implies modification (setting to 1) of a single bit, responsible for the region in which value of digital signature falls.

As one can see, this is not a time consuming process. However, we have to store this change to non-volatile memory before system is powered off. This storing can be done in the process of shutting down the system. Nevertheless, we have to be prepared for the system failures, also. To create a failure-resistant system, we have to store all the changes of our structure to the non-volatile memory (i.e., hard disk) before giving announcement to client about completion of storing operation. This would unconditionally imply at least one access to the hard disk and this is not allowed in our system (this presumption is one of the most important requirements in the software). All what we can do is to allocate a small amount of memory *NewlyAddedDS* for storing newly added digital signatures (Figure 2):

Sporadically, we withdraw collected digital signatures to disk. With this action we enter to the new consistent state.

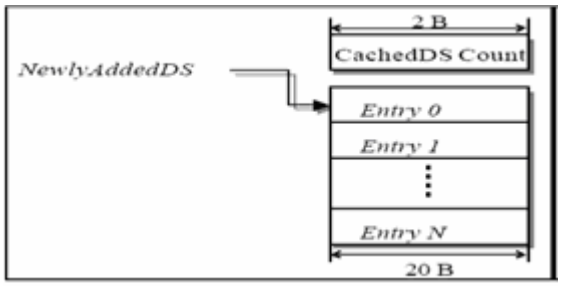


Figure 2: Storing of New Added Digital Signature

4.1.5 Process of Removing Signatures

Process of removing Digital Signatures is a bit tricky. The main problem lies in the nature of hash function; it is not a one-to-one function. It means that we can not decide whether we have to reset the state of bit that presents region to which the specified digital signature belongs, or we should leave it on 1 (meaning that the removed digital signature is not the only one that exists in the specified region). However, process of removing digital signatures is a rear process in archiving large systems, which is the case in our problem.

The simplest way of handling the removal of digital signatures would be ignoring this activity. What we would lose with this solution will be the possibility of one region that "arrogantly" announces Hits.

Another solution is to allocate another small amount of memory *RemovedDS* for storing digital signatures removed from the system. Sporadically, we remove collected digital signatures from disk. With this action we enter to the new consistent state. Also we have to recalculate values of conditioned bits in the array structure. This is not an exhaustive process since we have all digital signatures organized by regions, which they belong.

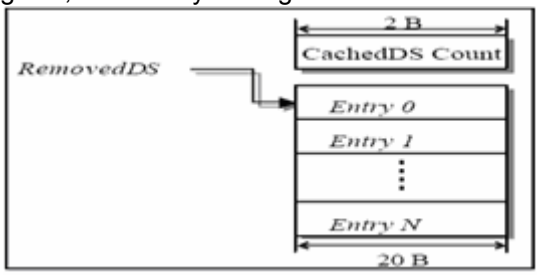


Figure 3: Removing of Digital Signature

4.1.6 Process of Recovering Structure

Recovering of the bit array structure is not an extensive process, since the bit array structure is a secondary data structure and it can be easily recovered from the primary data structure; in this case from *DiskRegions* structure (disk sectors that hold digital signatures from specific regions).

4.2 Client-System Communication Protocol

This solution covers another part of our problem; case for YES answers. It is presented with an array structure for each part of the digital signature. Figure 4 specifies this concept:

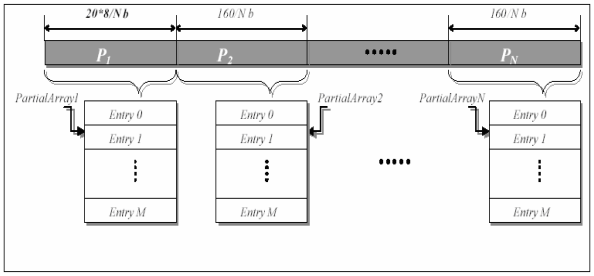


Figure 4: Case for YES Answers

4.2.1 Required Disk Organization

Required disk usage is N times larger than in the previous algorithm. Our decision is not to use additional disk structures which are specific for this model.

4.2.2 Process of Structure Creation

Initial structure creation is quite simple. We have to go through each digital signature stored in our system and to determine the value of each digital signature partition. Each value is used as entry point to an adequate Partial Array. Each entry is a bit that should be set. This determining process requires simple shifting, concatenating, and masking operations.

4.2.3 Process of Adding Signatures

Adding of new digital signature to system is just iteration in the process of creation of array structure. It implies modification (setting to 1) of N bits responsible for value of each digital signature partition. As one can see, it is not time consuming process. However, we have to store this change to non-volatile memory before system is powered off. This storing can be done in the process of shutting down system. Nevertheless, we have also to be prepared for system failures. As in the previous algorithm we

have no possibility for creation of a failure-resistant system. All what we can do is to allocate small amount of memory *NewlyAddedDS* for storing newly added digital signatures. Sporadically, we withdraw collected digital signatures to disk. With this action we enter to the new consistent state.

4.2.4 Process of Removing Signatures

Removing Digital Signatures is a bit tricky. The main problem lies in the nature of hash function; it is not a one-to-one function. It means that we can not decide whether we have to reset the state of bit that presents the region to which the specified digital signature belongs, or we should leave it on 1 (meaning that removed digital signature is not the only one that exists in the specified region). However, process of removing digital signatures is a rear process.

The simplest way of handling the removing of digital signatures would be ignoring this activity. What we would lose with this solution will be the possibility of one region that "arrogantly" announces Hits. Another solution is to allocate another small amount of memory *RemovedDS* for storing digital signatures removed from the system. This is a pretty exhaustive process since we can not arrange digital signatures by partitions in which they belongs (we will need N times more disk space than for the previous algorithm). Sporadically, we remove collected digital signatures from disk. With this action we enter to the new consistent state.

4.2.5 Process of Recovering Structure

Recovering of the bit array structure is extensive process, since the bit array structure is a primary data structure and it can be recovered only sequentially from the whole collection of digital signatures.

5. Conditions and Assumptions of the Research

The archive system can handle several streams of data simultaneously. The number of streams, speed of the streams and size of the files will determine the average number of lookup/inserts per second. Formula for the number of lookup/inserts per second is:

(AvgSize / Throughput per second)*NumStreams = Ops per Second

Assumptions:

- Average File Size = 10 MB,
- Throughput = 300 MB/sec,
- Number of Streams = 8,
- Operations per second (for entire system) = 2400,
- For a fully configured system, there are 8 computing nodes,
- Each node handles 300 lookups/inserts per second.

The goal of 300 lookups/inserts per second per node has been set.

6. Analytical Analysis

Lets have a digital signature DS1 that falls in a region R1. Additionally, let's assume that request for digital signature DS2 arises, and DS2 also falls in the region R1. In this case we will respond to the client with POSSIBLE_YES answer. However, adequate answer should be NO. This phenomenon is called groundless Hit.

Main advance of Linear Area of Region to Bit Mapping concept is in its simplicity. Also it gives 100% precisely answers on each request that SHOULD result in NO. On the other hand, this model suffers from groundless Hits.

Digital Signatures Partially Caching concept still suffers from groundless Hits, but in less measure than Linear Area of Region to Bit Mapping; however, the amount of disk utilities is N times larger then in the first algorithm. It could be improved to almost completely avoid groundless Hits, but, as one can see, there is no algorithm and data organization that will give just valid Hits answers. The reason is that the amount of RAM storage needed for storing whole data (20 * 1 Billion = 20 Billion = 20 GB) and because of uniform division of digital signature values.

In the case of uniform division impact, our software allows maximum each third check that results in disk access. If we include human and real-system-utilization impact we will get slightly different results. For example: a user "A" is storing collection of Mozart's compositions that user "B" have just stored on the system. In this specific case, request for each document's digital signature will result in justified Hit. The result will be 100% disk access!

7. Performance Analysis

In our specific case we have:

. DS_COUNT - a 1 billion of digital signatures,
. MEM_SIZE - a 1 GB of volatile memory (RAM),
. DISK_SIZE - almost unlimited disk space (we will limit it to 256 GB).

Let us first consider Linear Area of Region to Bit Mapping concept. With 1 GB RAM we can create 8Gb large Bit Array Structure (BAS). Thereby, we can divide DS value domain on 8G regions and finally we can calculate BAScoverage, the percent of coverage (coverage with existing digital signatures) for each region:

BAScoverage = DS_COUNT / BAS_SIZE= DS_COUNT /8*MEM_SIZE = 1/8 ˜ 12.5%

This result is telling us that in case of the first model, each eight check (client request to our system component) will result in groundless Hits (and need for disk access).

In the case of Digital Signatures Partially Caching concept we have to find on how many parts (N) digital signatures should be split to fit

into the specified amount of RAM memory (1 GB). We can find N from the following equitation:

MEM_SIZE=MEM_USAGE=N*2(20*8/N)/8

At first we have to see if it is satisfied with N=1. In this case we have MEM_SIZE = 5KB => equitation is satisfied. This equitation is a transcendental equitation and it can only be calculated by probing (we have to found the smallest N that satisfies the equitation). One can see that N = 6. Since 20*8/6 is not an integral number, we will get the following partitions:

26 + 26 + 26 + 26 + 28 + 28 = 160b

In this case:

MEM_USAGE = (4*2(26) 2*2(28))/8 B = 96 MB

8. CONCLUSION

In conclusion, the proposed approach proves to more efficient compared the existing solutions, in conditions of interest for this research. The proposed solution is of interest to those who a faced with the management of extremely large volumes of files.

REFERENCES

[1] Rosenblum, M., Ousterhout, J., "The Design and Implementation of a Log-Structured File System," *ACM Transactions on Computer Systems*, 1992.

[2] Bent, J., Venkataramani, V., LeRoy, N., Roy, A., Stanley, J., Arpaci-Dusseau, A., Arpaci-Dusseau, R., Livny, M., "Flexibility, Manageability, and Performance in a Grid Storage Appliance," *Proceedings of the 11th IEEE Symposium on High Performance Distributed Computing* (HPDC-11), July 24-26, 2002.

Cache Clearing System

Babovic, Zoran; Jovic, Darko; Cakarevic, Vladimir; Milosavljevic, Ivan; Stevanovic, Marija; Minic, Predrag; and Milutinovic, Veljko

Abstract—*In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: How to implement a system which clears the cached content from the production web sites. The essence of the proposed solution is presented in the part three: The main part of the system, CacheBot, consists of two multithreaded applications, called Hard Daemon and Soft Daemon, which are in charge of actually deleting or updating the contents on the web servers. The fourth part defines advantages that we made compared to an existing solution, as well as the conclusion.*

1. INTRODUCTION

THE Dow Jones Cache Clearing System is responsible for clearing the cached content associated with pages, sections and collections on the production sites (WSJ.com, Barrons.com). Content is cached in order to improve response times and resource utilization when serving pages requested by browsers. However, as a news site, it is important that the content is as current as possible. In order to achieve the optimal trade off on how often to clear cached content so as to maximize response times and conserve resources without unduly compromising the currency of information on the pages, complex business rules have been developed and implemented to dictate when, how often and in what manner to clear cached content.

2. PROBLEM STATEMENT

This part describes requirements and expectations established for this project. Required features were to make Hard Clear Cache – part of the system that is in charge for removing the files immediately; Soft Clear Cache – part of the system that replaces file with a newly generated file; Flexible configuration of URLs for soft and hard clear cache; guaranteed delivery of Cache Clearing. Any cache clearing requests should never become lost. The system had to be able to resubmit failed requests based on configurable parameters or properly log any persistent problems. System also has to be able to work in a number of abnormal conditions, to eliminate all duplicate cache-clearing requests from the message queue (as we have many of duplicate requests).

All of these requirements had to be designed in a simple robust way that emphasizes reliability, easy maintenance and high performance over flexibility and abundance of features. [1]

3. PROPOSED SOLUTION

In order to clear cached content, the Cache Clearing System must be able to locate the exact content to be cleared. Content is identified by its content ID which is then resolved to one or more URL paths and file names. It is this URL path and file name that are cached on disk and used to serve all subsequent requests from user browser sessions. Once the Cache Clearing System determines that a particular cached content item is to be cleared (i.e., an editor has published an updated version of the content), it will either:

-Delete the cached URL path(s) and file name(s) associated with the content item from the disk. The next time a user requests this content item via a browser session, the absence of that URL on the disk will cause the Cache Clearing System to retrieve a fresh copy of the content from the one of the production rails, and generate a new URL path and file name on the production rail on which the content was originally requested. This method is referred to as Hard Cache Clearing.

-Proactively obtain a fresh copy of the content from the one of the production rails, generate a new URL path and file name and distribute that URL to all production rails. This method is referred to as Soft Cache Clearing.

In Figure 1 we can see the architecture of the Cache Clearing System. CacheBot is the main part of the system, and consists of two multithreaded applications, [2] called Hard Daemon and Soft Daemon, which are in charge of actually deleting or updating the contents on the web servers. Requests for Hard Clearing are inserted into CLEAR_CACHE_QUEUE table by CacheManager and read periodically by the Hard Daemon, while requests for soft clearing are defined in SOFT_CLEAR_CACHE table and read periodically by the Soft Daemon.

Web servers are grouped to serve requests from browsers faster. These groups are called *Elementary Targets* (or just *Targets*), and each web server in the target is called *Rail*. Every rail is

Manuscript received July, 2006. This work was supported in part by the Dow Jones Company.

Babovic Zoran, Jovic Darko, Cakarevic Vladimir, Milosavljevic Ivan, Stevanovic Marija, Minic Predrag, and Milutinovic Veljko are with Faculty of Electrical Engineering, University of Belgrade, Serbia.

defined by the unique name and the URL. Elementary targets can also be grouped to form higher-level targets.

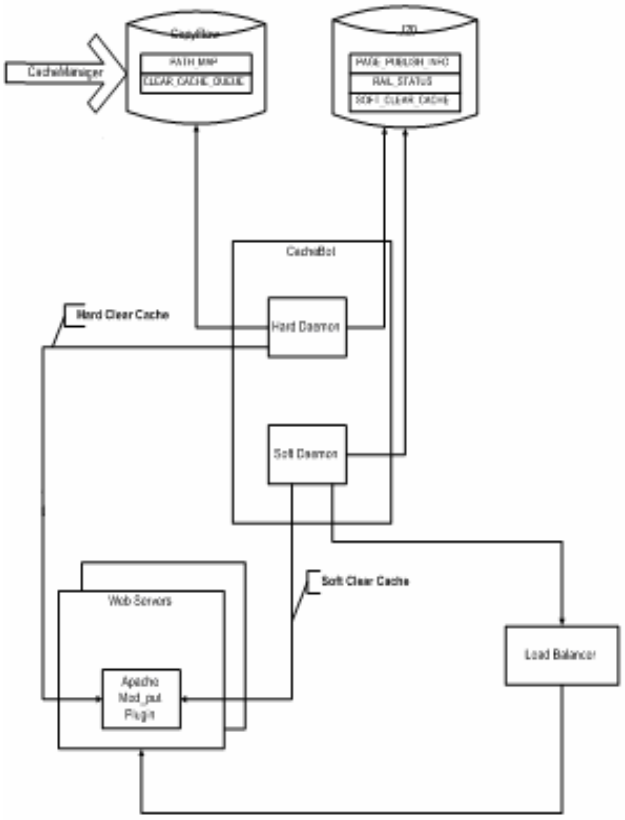


Figure 1: Cache Clearing System Architecture

Every request for cache clearing has corresponding target, which means that it needs to be executed on all rails belonging to the specified target. There are exceptions to this rule: if request has a target of higher level assigned and one or more elementary targets which could be soft cleared.

Every Apache web server contains mod_put plug-in, which makes possible to delete or update web content from the server with HTTP PUT and DELETE requests. Hard Daemon uses only HTTP DELETE requests for clearing the content, while Soft Daemon uses both PUT and DELETE requests for updating the content on the web server.

In Figure 1 is also shown a load balancer, which Soft Daemon targets to generate a new content that will be delivered to required rails.

3.1 Hard Cache Clearing

The Hard Clear Cache daemon provides the following functionality:

1. Fetches requests for Hard Cache Clearing from CLEAR_CACHE_QUEUE table,
2. Eliminates duplicate requests if they occur,
3. Checks the syntax of requests to ensure that they are well formed and valid,
4. Resolves partial URLs from requests by rules defined in PATH_MAP table,
5. Eliminates duplicate URLs which may occur during resolution phase,
6. Deletes contents from the appropriate rails by sending HTTP DELETE requests,
7. Deletes processed requests from CLEAR_CACHE_QUEUE table.

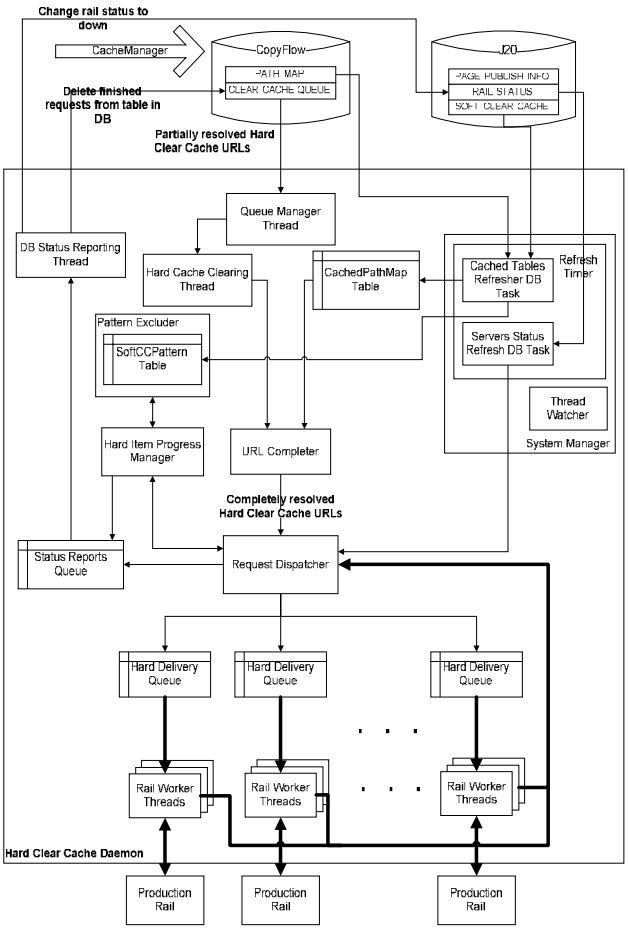


Figure 2: Hard Clearing Daemon Architecture

Hard Daemon fetches requests from the CLEAR_CACHE_QUEUE table periodically, with the configurable time interval. This is done by the Queue Manager Thread. If some requests are fetched from the table, Hard Daemon does not wait for the next time interval, but reads immediately from the database table in case that there are new requests available. If there are no requests in the table, Hard Daemon waits for the next time interval to check the CLEAR_CACHE_QUEUE for new requests. Duplicate requests (requests which have the same content id, content type and target) will not be fetched, but will be deleted on original request completion. For each fetched request, Hard Cache Clearing Thread creates object called Hard Item, which encapsulates request data (URL, target…). After requests are fetched, partial URLs are resolved based on the request's content type and target, which are paired with path and extension fields from the PATH_MAP table. One partial URL can be resolved to several resolved URLs, which are generated by concatenating path, content id and extension. In this phase it is possible that some partial URLs

are resolved to same URLs. These duplicates are also eliminated by the CacheBot. URL resolving is done by the URL Completer component.

After URL completing, Hard Item with completed URLs is passed to Request Dispatcher. It uses Hard Item Progress Manager for determining on which rails request should be executed and excluding URLs which are configured for Soft Cache Clearing (Pattern Excluder). Hard Item Progress Manager is also responsible for tracking of request execution and handling of duplicates of currently processing requests.

After Item processing in the Hard Item Progress Manager, for every resolved URL an object called URL Request is created and assigned to Hard Item. Request Dispatcher takes list of URL Requests from the Item and dispatches them to appropriate rails (i.e. inserts them in corresponding delivery queues).

There is a configurable number of Rail Workers (running in separate threads) dedicated to each rail. Those workers take URL Requests from the Delivery Queue and execute them on the appropriate rails. Rail Workers can handle uploading (used in Soft Cache Clearing) and deleting of contents from the rails by using HTTP PUT and DELETE requests. In case of Hard Cache Clearing only deleting is performed. Depending on the success of request execution, Rail Workers notify Request Dispatcher and Hard Item Progress Manager of the execution status.

If rail worker detects that rail is down, it notifies Request Dispatcher which creates rail status report for updating RAIL_STATUS table.

Status of the rail is kept in the RAIL_STATUS table, and there are two possible states of the rail: UP and DOWN. Internally, CacheBot has two more states which are not stored in the RAIL_STATUS table, but only logged into a log file. These states are: SLOW and DELAYED. When the rail is DOWN, requests are not sent to it. Rail can be set to DOWN state by either CacheBot or administrator. CacheBot sets rail state to DOWN when it determines that server is not accessible (gets timeout during connection attempt), while administrator just changes the value in the RAIL_STATUS table for particular rail. On the other hand, rail can be set to UP only by administrator, by changing the state to UP in the RAIL_STATUS table. Rail statuses are periodically read by the Server Status Refresher DB Task which then updates the rail statuses in the CacheBot governed by the following rules:

- The state of the rail will be changed to DOWN if the rail state is not already DOWN and the time of last change (recorded in the CacheBot) is before than the change time stored in the table.

- The state of the rail will be changed to UP if the state is DOWN and the stored time of the last change is before the change time from the table.

SLOW state represents the situation when the average execution time of the configurable number of last requests on particular rail is greater than the configurable threshold.

DELAYED state represents the situation when number of requests waiting for execution in the queue for particular rail is greater by the configurable threshold than number of requests in the queue for the rail of the same elementary target which has the minimal number of requests in it.

All database reports are inserted into the Status Reports Queue and there is a DB Status Reporting Thread which will take tasks from the queue and execute them.

3.2 Soft Cache Clearing

The Soft Cache Clearing Daemon provides the following functionality:

1. Fetches requests for Soft Cache Clearing from SOFT_CLEAR_CACHE table, but only if they have been updated since the time of the last fetching. This occurs at regular time intervals,
2. Checks the syntax of requests to ensure that they are well formed and valid,
3. Communicates with the Apache Web server (via HTTP routed through the Load Balancer) in order to force the generation of an updated content file on a single production rail. New content is retrieved by inserting timestamp in the URL for which new content is needed and sending HTTP request with the new URL to the load balancer,
4. Uploads updated content on the appropriate rails via HTTP PUT requests,
5. Deletes generated page (with a timestamp added) from the rails via HTTP DELETE request.

In Figure 3 we can see the internal architecture of the Soft Daemon. The architecture is slightly different from the Hard Daemon architecture. Rail Worker Threads, Request Dispatcher, DB Status Reporting Thread, Rail Status Refresher Task, and System Manager, function in the same way as described in Hard Cache Clearing section. New elements are Soft Cache Clearing Thread, Soft Queue Manager Thread, and Soft Item Progress Manager. Soft Daemon fetches URLs for Soft Clearing from the SOFT_CLEAR_CACHE table by Soft Queue Manager Thread, but only if they have been updated since the time of the last fetching. This is checked by including PAGE_PUBLISH_INFO table in the fetch query.

The data needed for page update is stored in an object called Soft Item. Fetching is done periodically with the configurable time interval. Soft Queue Manager Thread behaves in the same way as in Hard Daemon, but reads cache clearing requests from different database tables.

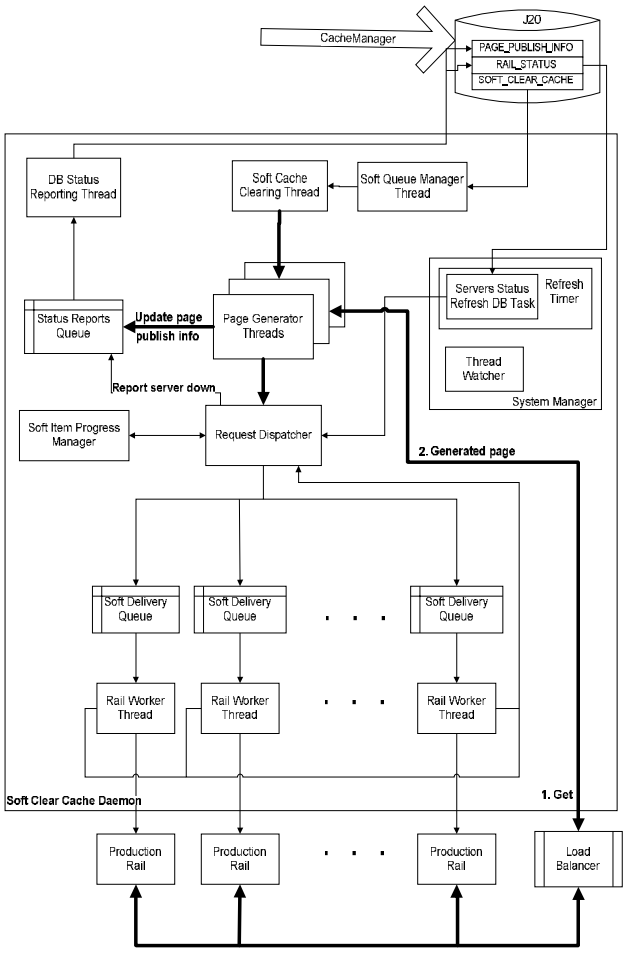


Figure 1: Soft Clearing Daemon Architecture

Page Generation Task is created for each Soft Item. This task executes GET HTTP request through Load Balancer and saves received page. It calls Request Dispatcher to dispatch generated page to rails. If an error occurs during page generation, the corresponding PAGE_PUBLISH_INFO table record will be updated so the page updating will be tried in the next iteration.

In order to get a new content for specific URL, Page Generation Task inserts a timestamp into the URL and targets Load Balancer which will recognize this type of URL and will generate a new page. Timestamp is in a form yyyyMMddHHmmssSSS and is inserted before the file extension.

After the page has been created, it is stored in the Soft Item object and passed to Request Dispatcher for dispatching to required rails. Request Dispatcher then calls Soft Item Progress Manager to eliminate possible duplicates and inserts page update request to all rails. Request is considered to be a duplicate if the request with the same URL and target is currently waiting to be processed on one or more rails. In case that duplicate has been detected, content of the original request is replaced with the new one, so rail will receive the latest version of the generated page and the old content will be deleted. Soft Daemon can store generated page content in a file or in a memory structure.

Each Rail Worker Thread will update page on its rail via HTTP PUT request. When some page is updated, Rail Worker deletes temporary generated page(s) from the rail, if there is any, via HTTP DELETE request with a URL which contains -*.* (delete extension) at the end. This will delete the generated page with a timestamp in a URL.

Request Dispatcher behaves in the same way as in Hard Daemon. This means that rail states and their behavior is the same as described in Hard Cache Clearing section.

All database writes are done through Status Reports Queue by DB Status Reporting Thread as in Hard Clearing.

The System Manager works in the same manner as in Hard Daemon.

4. CONCLUSION

In this part we will summarize facts that we gained in this project and also, we will make a comparison with one existing solution, but currently used system will not be described now.

Main problem with the current solution is that each thread has to wait as long as the last working server finishes with certain page. In our solution this problem is resolved by independent threads. This means that servers do not wait each other, which, obviously speed up the whole system. Weakness of this solution is bigger memory consumption, which is being recompensed with the fact that there is no possibility of overhead situations (overhead in the current solution is the thread information on which server it has to be executed).

In the current solution of the problem, pages are being updated all at the time, with a certain time stamp. The page is updated only if it is changed since the last updated time. This means that the superfluously updating is being eliminated.

Page tracking is optimized; primarily by decreasing the data base utilization (one table is even excluded from the data base used for our solution).

Servers are being set to UP and DOWN state only by the administrator, which implies that pages are being sent even to the DOWN rails, in case that administrator did not notice and change server state changes. Our solution gives much more reliable system. Beside administrator, which works in the same way, the server itself sends the message to the data base so it can be updated with the new server state. This change is now known through the whole system and the pages will not be sent to these servers.

Having in mind those advantages that we gained in this project, we think that all the requirements specified in the problem statement are satisfied, but also, some extra advantages are scored.

REFERENCES

[1] Horton, I., “Beginning Java 2,” *Wrox Press Ltd*, Birmingham, 2000.

[2] Oaks, S., Wong, H., “Java Threads 2nd Edition,” *O’Relly & Associates*, January, 1999.

Data Assurance in a Conventional File System

Rudan, Sasa; Kovacevic, Aleksandra; Milligan, Charles; and Milutinovic, Veljko

Abstract— ***In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: How to find a mechanism that guarantees that a file stored in a conventional file system, on disk, has not been modified; this problem is important because it enables security of the imperceptible changing of data, which is the main problem with digital data acquisition. Existing solutions to the problem are briefly surveyed in the third part, and their drawbacks are underlined; each surveyed piece of research is analysed according to the same template. Then, the essence of the proposed solution is presented: Our proposal is a smart card based Digital Sealed File System (DSFS). The conclusion is from the point of view of performance/complexity ratio.***

1. Introduction

Cryptographic techniques play an important role in data protection, since accessing data can be limited to those who hold the proper key.

There are several basic security services which cryptography may provide: confidentiality, data integrity, authentication, authorization, and non-repudiation. In many applications, the combination of cryptographic services is desired [2].

Common security systems imply a kind of Write Once Read Many (WORM) storage systems. This is not what we need, since almost all WORM systems relay on the specific format of data storage/archiving [4]. As soon as we transfer the data of interest to another medium/system, security of the data is broken.

2. Problem Statement

The goal of our research is not to protect document content, but only to protect unnoticed modification of the document. The idea is to allow any person to check if stored document or data is consistent and not modified. On the other hand, no one is allowed to be an authorized person and to do imperceptible change of the document content. The medium dependency has to be avoided and also any specific operating or file system organization.

At first, one can think that our system is much easier to implement then a classical security system. What we need to achieve is to prevent any unnoticed modification of the document, and not to protect content of the document against disclosure. In the classical encryption system there is a document owner; thereby this person has absolute rights over the document and its security. On the contrary, in our case there is no person who we can trust. What is more interesting, we have to be at most careful with the owner of the document (i.e. in the case of financial books, government wants to protect itself from any malicious modification by the accountant; Government wants to put a kind of digital seal over the document that is inspected).

3. Existing Solutions and Their Criticism

The data security systems are based on the classical cryptography principles [2]; two of them are symmetric and asymmetric.

Symmetric-key cryptography principles are presented in the "*Applied Cryptography,*" by B. Schneier [1]. It is a mechanism where the same key is used for both encrypting and decrypting. It is intuitive principle because of its similarity with, for example, using the same key for locking and unlocking a door.

Principle of asymmetric cryptography is presented in "*New Directions in Cryptography*," by D. Whitfield and M. Hellman [2]. Encryption-decryption process involves key pair - one for encrypting, and the other for decrypting. Given a key pair, data encrypted with the public-key can only be decrypted with its private-key and conversely, data encrypted with the private-key can only be decrypted with its public-key. The public key is distributed widely and freely. The private key is never distributed and must be a secret.

The solution for our problem using these techniques requires following participants: the client, the server, and the third party. Client is a user who wants to check the document authenticity. Server is an interface to the document and the document itself. Third party is an auxiliary service that provides verification of server (i.e. "Document is not modified").

The scenario for this solution can be described as follows: client is accessing the server with a certain request (typically, to check if the document is modified). Upon receiving the request, server is answering (typically, with claim

Manuscript received October, 2004. This work was supported in part by the StorageTech.

Sasa Rudan, Aleksandra Kovacevic, and Veljko Milutinovic are with Faculty of Electrical Engineering, University of Belgrade.
Charles Milligan is with StorageTech.

"Document is (not) modified"). Client refers to the third party, in order to verify correctness of the answer. It is necessary that both, client and server have a trust in the third party (Fig. 1):

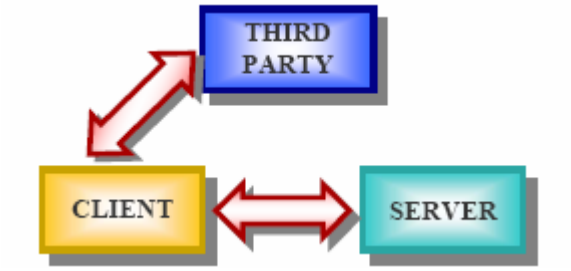


Figure 1: Possible solution scheme

However, requirements for our system imply that it is not allowed to have an intermediary authoritative body (third party).

4. Proposed Solution and why It Is Expected to Be Better

In our solution, it is used the public key encryption technology and the digital signing. Cryptography algorithms are public available and confidentiality of cryptographic methods relies on the confidentiality of the private key. Therefore, the most important point of every public-private key system is to secure the location where the private key is stored. The solution for that problem is using the specialized hardware components that have enough memory to store all relevant cryptographic data and enough processor power to perform basic cryptographic operations independently of operating system and software applications. It has been used the characteristic of the smart card: the resistance of reading their contents; upon attempt of opening smart card, it will destroy itself [11].

4.1 System Environment

The server in our system contains trusted documents, created by some users which are used by the other users. There are three actions over the documents that have to be provided: access to the documents, verifications of the documents, and document signing.

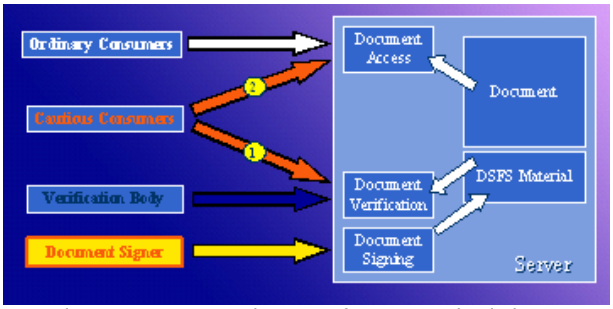


Figure 2: System environment for proposed solution

It can be identified four types of the document users or use cases (Fig. 2):

Ordinary consumers - they use document, not bothering with document integrity, so they only do "document access" action.

Cautious consumers - before using the document they want to verify document integrity, so they do both actions: verification and access.

Verification bodies – they are charged for document integrity verification and not interested in the document content.

Document signer - signer is a person (equipment) to whom is granted trusted period for signing the document.

4.2 System Components

The following components exist in the proposed solution (fig. 3):

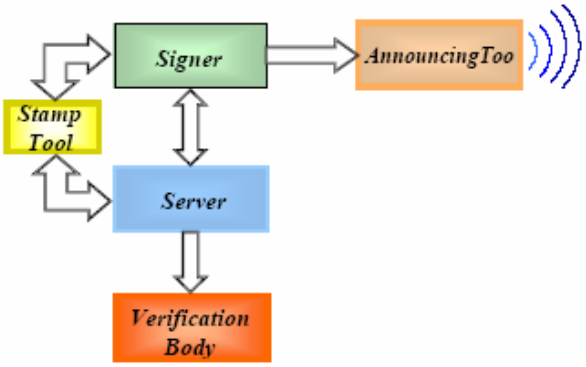


Figure 3: Block diagram of DSFS architecture

The central "actor" is a classical file server, or more generally, document server. It stores and publishes documents, but also contains DSFS material (document stamps, public keys, etc). DSFS material on server is secured by the classical file - system security policy. It means that it is liable to classical file-system attack. This is the reason why server is used just as secondary DSFS-material container. Also, there is a signer of the document, which is an authorized person that could be later interested in verification of the document authenticity. The signer uses stamp tool (or signing tool) for document signing (sealing). In our case it is the smart card based tool. After process of document signing, DSFS material has to be safely published. This is performed by announcing tool. Verification body verifies document authenticity. This is usually document signing initiator. Finally, there is user, the most active member of the system. He or she uses (none) signed documents.

4.2.1 Scenario of System Usage

User uses the system, almost in a same way as in a non-signed system. If the document is signed, only reading of the sealed document or data is allowed to the user. Writing activity may be allowed by the system, if the document is not sealed, otherwise user will influence the consistency of the system (i.e. the seal will be broken).

4.2.2 Server Specification

We can imagine server as a standalone or network file server. It provides the following functions:

Reading of the document – activities for signed document do not differ from the unsigned ones. If the sign of the document is broken, user should be warned about it, and the system should

provide an opportunity to give up from reading activity.

Writing of the documents – if the document is signed, it has to be forbidden to user to proceed with writing operation.

Associating stamp to the document – upon creating the sign for a given document, signer will forward it (using special protocol) to the server.

Reading sign associated to the document – reading the sign and forwarding to the verification body.

4.3 Document Signing

Signing algorithm has the following phases:

Phase 1 (system –> signing tool/stamp tool):

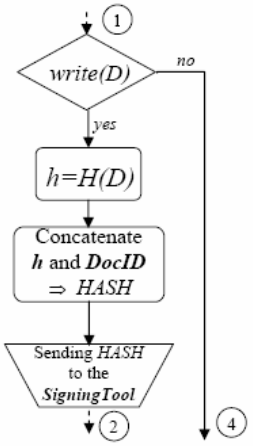


Figure 4: Phase 1 signing algorithm

First, server has to check the type of the incoming request: document writing request, associating stamp to the document, or document read request. In the case of document read request whole signing process is skipped. In the case of seal creation or writing request server initiates signing process; hash *h* of the document *D* has to be created [7]. Next step is concatenation of the document identifier (*DocID*), the unique document identification (it is necessary to have an evidence of all signed documents) with calculated hash *h,* creating document *HASH* and sends it to the stamp tool (Fig. 4). The following components exist in the data, which signing tool expects from the system: *DocID*, *HASH* – hash value of the document that is requested.

Phase 2 (signing tool):

After receiving *HASH*, stamp tool concatenates *HASH* with the time stamp, creating *THASH* object [6]. After that, stamp tool checks if it has already received this *THASH* before. If the answer is yes, error is reported, because *THASH* uniquely identifies specific document. Additional *THASH* insertion would imply the possibility of the document resigning, which is not allowed. If the answer is no, stamp tool can finally generate private/public key pair KP_U, KP_R [13]. The next step is *THASH* encryption with generated private key, producing digital signature *DS* [5]. When DS is generated there is no need for private key existence, so it can be destroyed (Fig. 5).

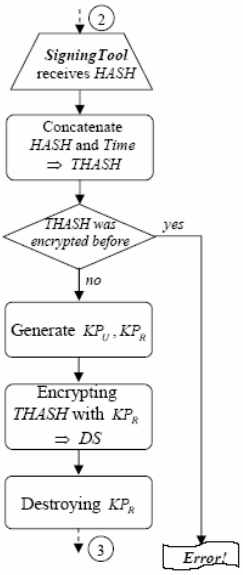


Figure 5: Phase 2 signing algorithm

The following components exist in the signed data stored on server: *Digital Sealed Document* – consists of DocID, document itself, Time Stamp and Encrypted HASH, *Publishing Material* – consists of DocID (identical with the previous one), Time Stamp (also identical with the previous one) and generated public key corresponding to the destroyed private key.

Phase 3 (signing tool –> system):

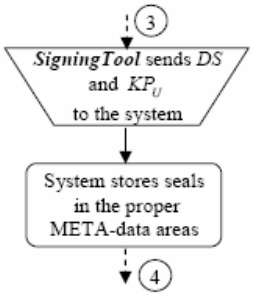


Figure 6: Phase 3 signing algorithm

Signing tool forwards seals to the system together with the public key (Fig. 6). When server receives DSFS material from stamp tool, it stores material inside the DSFS META-data area. DSFS META-data area should not be specifically implemented (an ordinary XML document is quite acceptable).

These are the following components of the data, which signing tool gives as output: the first one is *Digital Seal* – consisting of encrypted *DocID*, *Time Stamp* and *HASH*. DocID is identical with the DocID mentioned above. Time Stamp is generated by signing tool, using internal real time clock to provide information of signing time. HASH is encrypted with already destroyed private key. Another component is *Publishing Material* which consists of DocID (identical with the previous one), Time Stamp (also identical with the previous one) and generated public key,

corresponding to the destroyed private key. Public key is the main reason for the existence of the publishing material. If there is possibility that someone can replace pubic key, than he or she can modify document, calculate new hash, choose new key pair, encrypt time and hash with new private key, store them in DSFS META-data area, and replace old public key with the new one.

In general, all three phases can be represented as followed:

$$X: h = H(D)$$
$$HASH = Concatenate[h, DocID]$$
$$X \rightarrow Y: HASH$$
$$THASH = Concatenate[HASH, Time]$$
$$Y: DS = E_{KP_R}[THASH]$$
$$Y \rightarrow X: DS, KP_U$$

X - System,
Y - SigningTool
DS - Digital Seal

4.4 Publishing Scenario

The next step is publishing DSFS material to the real publishing storage. A type of publishing storage depends on what publishing policy is being chosen. Here are presented three techniques of public key publishing.

4.4.1 Publishing on a Public Site

In this scenario, anybody who is interested in document integrity (verification body), could access to one of those public sites, retrieve DSFS material, and check document integrity. However, this policy is liable to attack, since we use classical server technologies for publishing. It is required to make all public sites resistant to afterwards public key publishing of the existing document. This approach is acceptable for some level of security. It is not difficult to create an adequate service with append-only and random-read features. Also, some kind of secure communication between publishing tool and public sites is required.

We can improve publishing security using distributed and non-correlated sites. In this case Verification Body should access to as many as possible public sites and retrieve from each of them DSFS material for the specific document. If all of them are not equal it means that there is possibility that someone tried to obstruct document integrity.

4.4.2 Publishing with the Smart Card

In this scenario, it has to be used two different smart cards: primary (signing tool smart card) and secondary (publishing tool smart card). We use secondary smart card for storing public keys and distributing them to each user of DSFS. On a document verification request, associated corresponding DSFS material should be returned. The most secure solution is when signing tool smart card publishes directly to the publishing tool smart cards. It has to be defined some secure broadcast protocol which will transfer DSFS material created on signing tool to publishing tool [11]. Number of publishing tool smart cards must be equal or greater comparing with the number of parties interested in verification of a document.

Problem with this solution is requirement that all secondary smart cards have to be updated. Also, there is possibility that someone can replace a publishing smart card with identical (from the interface point of view) smart card, with different DSFS material.

4.4.3 Publishing with the Signing Tool

The most secure process would be storing DSFS material inside the signing tool. This means that signing tool is also the publishing tool. This is the most secure solution, since there is no need for open communication between signing tool and publishing tool. Also, in this case we can easily check existence of the same HASH in the process of document signing. However, in this case the problem that can appear is limited storage space.

4.5 Verification Scenario

Depending on the technique applied in public key publishing, there are various verification methods.

Figure 7 shows simplified process of making signed hash.

First HASH value of the current document has to be calculated as HASH'. Than, DSFS material has to be used to extract original HASH using adequate public key which is also stored in DSFS material. If the original HASH is the same as the calculated HASH', that means the document is not modified after it was signed. Otherwise it is modified, and error should be reported.

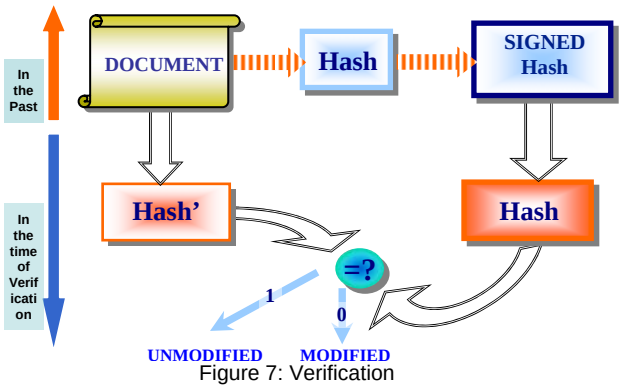


Figure 7: Verification

The first verification model requires from system to calculate the HASH of the current state of the document and to compare it with the HASH stored in META-data area.

The second one is based on forwarding necessary publishing material (retrieved from publishing tool) to the system and system calculates new values. This happens when verification body suspects in reliability of DSFS META-data area, and not in the system reliability.

The most secure case is when verification body requests download whole document from the server in order to personally check the

document integrity using DSFS material obtained from publishing tool.

4.6 Potential Problems

In this part it would be underlined some of the potential problems of this system.

4.6.1 Communication Snooping

Smart card – server communication is basically a serial communication which means that there is no real danger of snooping activities. There is only possibility of hooking in somewhere in operating system.

Snooping smart card – publishing tool communication is similar with the previous one. In the case of second policy we have serial communication, so there is no danger of snooping activities. In the last publishing scenario signing tool and publishing tool are the same. It means that there is no possibility of snooping.

4.6.2 Distribution

One of the possible problems is replacing publishing smart card with a new one. Protection against this is time stamp, which is generated inside the smart card. We can use unique smart card ID to detect smart card replacement.

4.6.3 Intra Snooping

First possibility of *copying content* of current smart card is technically impossible. Another possibility is *scanning content* of smart card. We can scan electro-magnetic field to find out what is the values of stored bits. To scan more precisely, we can use liquid oxygen to slow down changes in smart card. Also there are *snoop activities* inside the smart card, using statistic methodology. On the other hand, technology progress almost eliminated any possibility of those kind of scanning and snooping, so we can presume that we are completely safe using smart cards.

5. CONCLUSION

One of the advantages of our system is that it does not require any special equipment. It can be implemented on conventional servers/desktops that are extended only with classical smart card reader.

In conclusion, the proposed approach provides needed security of the document, if it is assumed that the security of the system components (i.e. smart card) is guaranteed. The proposed solution is of interest to those who have to prevent imperceptible changing of data.

REFERENCES

[1] Schneier, B., “Applied Cryptography,” 2nd edition, *Wiley*, 1996.

[2] Diffie, W., Hellman, M.,”New Directions in Cryptography,” *IEEE Transactions on Information Theory*, 1976.

[3] Apvrille, A., Hughes, J., Girier, V., “Streamed or Detached Triple Integrity for a Time Stamped Secure Storage System,”

[4] Wang, Y., Zheng, Y., “Fast and Secure Magnetic WORM Storage Systems,” www.verisign.com/resources/wp/enterprise/management/management.pdf

[5] Haber, S., Stornetta, S., “How to Time-Stamp a Digital Document,” *Journal of Cryptology*, 1991, Vol. 3, No. 2, pp. 99-111.

[6] Apvrille, A., Hughes, J., “A Time Stamped Virtual WORM System,”

[7] Menezes, A., Oorschot, P., Vanstone, S., “Handbook of Applied Cryptography,” *CRC Press*, ISBN 0-8493-8523-7, October 1996.

[8] National Institute of Standards and Technologies, NIST FIPS PUB 140-2, “Security Requirements for Cryptographic Modules,“ *U.S. Department of Commerce*, August 17, 2001.

[9] Adams, C., Cain, P., Pinkas, D., Zuccherato, R., “Internet X.509 Public Key Infrastructure Time Stamp Protocol (TSP),” *Network Working Group*, RFC 3161, August 2001.

[10] HP Automated Storage Division: “Safeguarding Data with WORM: Technologies, Processes, Legalities and Standards,” *Technical report*, Hewlett Packard, 1999.

[11] StorageTek: “Encryption Key Management System using multiple Smart Cards,”

[12] The ISO 7816 Smart Card Standard, http://www.cardweb.com.tw/card/iso/ISO7816.htm

[13] Blaze, M., “Key Management in an Encrypting File System,” AT&T Bell Laboratories, www.crypto.com/papers/cfskey.pdf

[14] VerSign, “Enterprise Key Management,” *VerSign White Paper,* www.verisign.com/resources/wp/enterprise/management/management.pdf

Data Mining: A Brief Overview and Recent IPSI Research

Radivojevic, Zaharije; Cvetanovic, Milos; and Milutinovic, Veljko

Abstract—*In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: Information extraction from large amount of data. Efficient usage of data mining models and algorithms will be presented using the recent IPSI research as a case study. Existing solutions to the problem are briefly surveyed in the third part, and their drawbacks are underlined; each surveyed piece of research is analysed according to the same template. Then, the essence of the proposed solution is presented: A novel algorithm based on the K-nearest neighbor model has been developed. The algorithm is general enough to be considered as a common solution for a group of similar problems. On the other side, it is a good example on how data mining techniques could be efficiently used for reverse engineering problem types. The conclusion is from the point of view of performance/complexity ratio.*

1. Introduction

INFORMATION extraction from large amount of data is a multidimensional problem addressed in various research fields. The need to uncover the hidden knowledge buried in the raw data faces us with a huge *NP* complete problem domain. Therefore, existence of partial solutions that cover specific problem areas is expected. Instead of that, a completely new methodology has been developed. The newest solution of the problem is data mining. It can be defined as automated extraction of predictive information from different data sources. It is a powerful technology with great potential to help users focus on the most important information. Data mining can answer questions that were too time consuming to resolve in the past. It can also help us predict future trends and behaviors, allowing us to make proactive, knowledge-driven decisions. On the other side, it does not eliminate the need to know your problem as well as to understand your data or analytical methods [15]. Just like any other novelty term, “data mining” is often misused and applied in wrong contexts. For example, it is often frequently identified with OLAP (On-Line Analytical Processing). OLAP is used for verification of the existing hypothesis; Data mining tries to generate such hypothesis by uncovering the hidden patterns. It is essentially an inductive process. It means that OLAP and data mining should be used together, in order to complement each other. OLAP can be used to verify the results of the data mining process.

In other words, data mining should be comprehended as an assistant that can help you classify your data, predict later behavior, extract association rules or detect sequences.

2. Problem Statement

Before we can use data mining models and algorithms we have to find the most suitable strategy. In order to do that, we have to detect the problem type. Usually, data mining project involves a combination of different problem types, which together solve the problem [1].

At the lower end of the scale of the data mining problems is ‘data description’ and ‘summarization’. It aims at the concise description of characteristics of the data, typically in elementary and aggregated form. This gives us an overview of the structure of the data.

The next data mining problem type is problem of ‘segmentation’. It aims at the separation of data into interesting and meaningful subgroups or classes. All member of a subgroup contain common characteristics.

Misunderstanding of term segmentation is caused by it’s relation with ‘classification’, which is another data mining problem type. Classification assumes that there is a set of objects that belong to different classes, where some attributes or features characterize each class. The objective is to build classification model which assign correct class label to previously unseen and unlabeled objects (so called predictive modeling).

Classification and segmentation may introduce new type of data mining problems; it is a ‘concept description problem’. It aims an understandable description of concepts or classes. The purpose is not to develop complete model with high prediction accuracy, but to gain insights.

Another important problem type that occurs in a wide range of application is ‘prediction’. The aim of prediction is to find the numerical value of the target attribute for unseen objects.

Manuscript received September, 2004. This work was supported in part by the Plurix, Germany.

Radivojevic Zaharije, Cvetanovic Milos, and Milutinovic Veljko are with the Faculty of Electrical Engineering, University of Belgrade, Serbia.

In close connection to the prediction is another problem type, so called 'dependency analyses'. It consists of finding a model that describes significant dependencies and associations between data items or events.

3. Existing Solutions and Their Criticism

Many models were designed in order to conduct a sequence of steps that will lead to adequate results.

One of the possible approaches is modeling based upon 'neural networks' [7]. Many years of experience in this area lead to efficient modeling of large and complex problems. This model can be easily applied for characterization of processed data with single numeric value. Therefore it is usually used for prediction problem types. On the other side this model has numerous drawbacks. It often requires huge amount of data to be preprocessed and beside that, it mostly depend on time spent for training of neurons.

Another popular model is based upon 'decision trees'. It is a way of representing a series of rules that iteratively splits data into discrete groups maximizing distance between them at each split. This model is very flexible when we take growth and stopping rules into consideration.

Similar modeling approach is called 'rules induction'. It creates a set of rules that are unlikely to form a tree. It differs from the previous model because here rules may not cover all possible situations and rules may sometimes conflict in a prediction.

Beside many other existing models we are going to name one more. It is a 'K-nearest neighbor' and 'memory based reasoning model'. It is based on usage of knowledge of previously solved similar problems for solving the new problem. When a new data comes it will be classified as a class member of a nearest existing group. It is very important to find a suitable measure for distance between groups.

4. Proposed Solution

Methodology and algorithms described in this paper arose as a result from the projects that were conducted on the Electrical Engineering Faculty of Belgrade University in cooperation with IPSI Belgrade. The projects goal was developing of GeForce graphic card drivers for custom designed operating system, which in the following text will be referenced as the GF project. At that moment the project was started, there was no documentation or source code available, though, retargeting of the existing drivers was not an option, especially if we consider the fact that the particular operating system is based on a completely different paradigm in the contrary to the traditional operating systems. It is actually distributed operating system, which relies on transactions as atomic operations.

The transaction management follows the optimistic assumptions with no looking introduced at all, which results in very fast and simple algorithms in the kernel, but it implies additional complexity for the driver programmers. This is actually a case of dynamic reverse engineering research project. In cases like this static analysis is useless because there was no source code to be analyzed. Decompiling of existing drivers would not bring anything useful. The reason for this is dynamic driver loading and run time dependency, so it is impossible to reduce dynamic processes to the static foundations.

Dynamic analysis used during GF project involved following steps: gathering of raw data, data processing, and visualization of extracted information. In the following sections universal methodology for the row data gathering will be explained in detail, and novel algorithm for information resolving will be presented, while the last step, visualization of the results, will not be further considered [2], [13].

The dynamic data gathering is a problem for it self, but once it is solved, it produces mass of unstructured data [12]. This is where data mining algorithm shows it's full strength.

4.1 Dynamic Data Gathering Methodology

Understanding of the processor architecture is necessary precondition for comprehension of the presented methodology. Especially important parts of the architecture are those related to debugging and performance measurement capabilities. Before the project started some decisions had to be made. Those decisions have influence on the implementation and not on the methodology itself:

The first decision was selecting operating system. Windows platform was chosen, because it is widespread over personal computers in not only medium sized but also in big enterprises.

The second reason for choosing this operating system would be demystification of the Windows secrets [11].

The last, but not least important reason is personal satisfaction of the team members. This technique is based on a non-invasive dynamic data gathering as the safest way to show the precise characteristics of the analyzed system. This is true not only for applicative software but also for analyzing of the every possible working mode of the operating system. Those advantages were used during GF project [4, 14].

For the purpose of the GF project it was necessary to gather data about all accesses to the graphic card resources that are done in the run-time on behalf of the operating system. Having in mind that all used resources are memory mapped, it is easy to conclude that presented methodology could be generally used in cases when it is needed to record access to any memory location. The capabilities of the Intel processors (usage of DR0-DR3 registers) were not satisfying because access to numerous memory locations had to be evidenced [8].

First step of the algorithm is to determine page number (segmented paged virtual memory) that correspondents to each of the memory mapped

location that we wish to observe [9, 10]. The second step is to reset the *present bit* of each page that was selected in the previous step (present bit that is placed in the *page descriptor*). In this particular project we had a situation to deal with the non cacheable pages (memory that is physically located on the graphic card is memory mapped). If the currently executing instruction tries to access observed memory location, *page fault interrupt* will be raised (this is because present bit in the page descriptor was reset). Interrupt handler will do the instruction parsing and record all important data (instruction address, instruction code, access type write/read, data length, data, and data address). In the case of read access we have to wait till end of the instruction to be able to record the data that had been read from the location. In order to obtain all mentioned data we have set *trap flag* in the *program status word* (PSW). This will result in that after the instruction finishes the *trap interrupt* will be raised. Trap handler will read the data (in the case of read access) and reset present bit (in both read and write access) that corresponds to the page that has just been accessed. Before the trap handler completes, trap flag has to be reset. It is obvious that the described algorithm requires that interrupt handlers to be hooked and new interrupt routines to be put on the beginning of interrupt handler lists [3]. Operations that are referred in the algorithm (interrupt handler hooking, present bit resetting, etc.) could be executed only inside of the program with high level of privileges (kernel level). From that reason during the GF project the SoftICE symbolic debugger had been used. The SoftICE is working on the kernel level and it has one very important feature, the possibility to be extended. This algorithm was implemented in the extension had been written in the NASM assembly language.

4.2 Data Processing Phase – Data Mining

Data gathering methodology described in the previous section had been used for collecting raw data produced in the series of tests conducted on g different GF2 graphic cards. All graphic cards had been tested on the same hardware configuration with generic driver installed. Mappings of memory and I/O address space could vary between different graphic cards, but relative order of registers inside of the particular memory apertures is constant. Data processing described here involves all type of problems described in one of the introductory section, and therefore algorithm presented here could be used as a common solution for a group of similar problems. There are differences between some parts of the gathered data for different graphic cards, and those are implicated only by individual card or operation characteristics. Those parts are referenced as noise. So, data accesses, that include address, order, and data that are card specific for some operations, are also to be treated as noise. On the other side, there are parts of the gathered data that are common for all cards, and they are used for extraction of important memory locations and resources.

4.2.1 Mathematical Model

Mathematical model that had been used for information resolving from gathered data maps problem space into 5D vector space [6]. More precisely, we have to deal with g independent 4D vector spaces, designated by V(t, a, rw, d). In the V(t, a, rw, d), t stands for time axis, a stands for memory addressing axis, rw for memory access type axis (read/write), and d for data axis. Construction of a subspace u of the vector space V is defined with the following formula:

$$\forall u(t_u, a_u, rw_u, d_u), V(t, a, rw, d)(u \subseteq V \Leftrightarrow t_u \subseteq t \wedge a_u \subseteq a)$$

The purpose of the formula is to compact the 4D problem space by bounding on the two coordinates (time axis and memory addressing axis). The remaining two coordinates (rw axis and data axis) serve for similarity function calculation which will be introduced in the following section.

Purpose of the mathematical model is to describe formal method for extracting set of important locations that we denote as set I. It is obvious that set I, can only be used for discovering the important locations and not for giving a meaning to each one of them. We also define I_o as a set of locations important for an operation o2O, where O denotes set of all operations available on the GF chip. We could say that the following formula is:

$$I^o \subseteq I \wedge \bigcup_{o \in O}^{I^o = I|}$$

4.2.2 Similarity Function

Heart of the presented mathematical model is 'similarity function' $S(u_1, u_2, C)$, where u_1, u_2 represent 4D vector subspaces, u_1, u_2 (- V(t, a, rw, d), and C is restriction criterion. Purpose of the similarity function is to compare two subspaces and to establish level of conformance. Similarity function is between [1, 2] and that is the first characteristic of the S function. It could be defined as:

$$\forall(u_1, u_2, C)(0 \leqslant S(u_1, u_2, C) \leqslant 1)$$

The second characteristic of the similarity function is:

$$\forall(u_1, u_2, C)(u_1 = u_2 \Rightarrow S(u_1, u_2, C) = 1)$$

$$\forall(u_1, u_2, C)(u_1 = \bar{u}_2 \Rightarrow S(u_1, u_2, C) = 0)$$

The third characteristic is:

$$S(u_1, u_2 + u_3, C) = S(u_1, u_2, C) + S(u_1, u_3, C)$$

$$u_2 \cap u_3 = \{\}$$

The fourth characteristic is:

$$S(u_1, u_2, C) = S(u_2, u_1, C)$$

Similarity function defined in this manner is equivalent to the definition of probability function. That implies opportunity to use results gained by theory of information, like information quantity and entropy of the system V. What we are actually trying to explain is that similarity function S(u_1, u_2,C) aims to maximize pairing of two subspaces u_1 and u_2. Maximal pairing means finding maximal set of coinciding vectors from u_1 and u_2. Coinciding of given vectors depends on the restriction criterion C. In the other words, we say that each vector from subspace u_1 coincides to the particular vector from subspace u_2 if it is inside of C-environment of that particular vector. Formal definition tells that vector x(t_x, a_x, rw_x, d_x) is in C-environment of the vector y(t_y, a_y, rw_y, d_y) if and only if:

$$rw_x = rw_y \text{ and } |d_x - d_y| < C_d \text{ and} |a_x - a_y| < C_a$$

Previous definition represents a suitable distance measure between vectors [5]. Distance measure for coordinate d is assumed to be the Hemming's distance, while distance measuring for coordinates a and t are 88 measured relatively from the beginning of the vector subspace. Note that restriction criterion C is vector C(C_t,C_a,C_{rw},C_d), and it enables different levels of filtering.

4.2.3 Algorithm

The goal of the algorithm is to discover set I and sets I_o for the given set O, and to minimize noise (noise minimization is equivalent to information discovery). This could be accomplished in three phases. In all phases we iteratively use similarity function. Each iteration is conducted with different level of filtering (level of filtering is expressed through restriction criterion C). We emphasize that outputs of each phase are used as inputs for the following phases. Results of the first phase are sets I_j , that represent sets of important locations (memory locations) obtained for different restriction criterion *C* used with similarity function. The second phase of iterations inducts $I_{o\,j}$ for each o (- O, where j is interpreted according to the results of the previous phase. The purpose of the third phase is to eliminate noise caused by individual card or operation characteristics. This algorithm is based upon K-nearest neighbor model, with such modification that we have to deal with a lack of information about previous similar problems. This deficiency is surmounted by numerous iterations which purpose is to build the knowledge base. The knowledge is formed upon maximization of similarity function applied to the 4D vector subspace of each test card. This process will be conducted on each possible pair of 4D vector subspaces, which results in algorithm complexity of:

$$O\left(\binom{g}{2} \cdot \|V\|^2\right),$$

where ||V|| stands for a norm of each 4D subspace, actually it is a number of possible subspaces. Now we have to explain the essence of the mathematical model; we can step into further consideration of each algorithm phase.

The first phase of iterations results in sets of approximations of target set I. Those sets are gained under different criterion C, and are denoted as I_j sets. This could be expressed with:

$$I_j = \bigcap_{k \in g} I_{jk}$$

where I_{jk} are sets of important locations for a particular graphic card k. Each one of the I_{jk} sets is obtained as a important locations set for the particular vector space V_k under criterion C_j . The V_k are instances of 4D vector space V formed upon gathered data for graphic card k.

The purpose of the second phase is to narrow the results of the previous phase by producing I_{oj} sets. It is accomplished by looking at only those important locations that are involved in the execution of the particular operation o. Operation set O is predefined and unique among all test cards. For each operation o from set O we know point in time t_0 before which operation did not start. This will result in additional compaction for each vector space V_k. So, initial vector space V_k*(t, a, rw, d)* is projected to vector subspace denoted as: V_k*'(t', a', rw', d')* where $t' > t_0$, a' (- I_j .

The final, third phase, tries to resolve noise caused by different card or operation characteristics. In this phase entire problem space (vector space) is narrowed to the level of bit patterns inside of a written or read data from the I_{oj} sets. Comparison of the bit patterns results in identification of common characteristics for all test graphic cards. On the other side, bit patterns and sets of important locations that are constant among all test cards (similarity function S = 1) in this phase do not contribute to the overall quantity of information. The way we defined similarity function S, is equivalent to the definition of probability function and therefore from the theory of information we can conclude that system is sustainable with entropy equals to 0 (H = 0, I = 0). This means that we are not able to extract the meaning of particular bit patterns, but still they can not be omitted, and they have to be used as they are. These kinds of results are presented in the form of a fix functionality tables.

4.3 Project Verification

In order to verify the given methodology, described in the previous section, part of the gathered data will be presented and explained in details. The process of information extraction and visualization will not be presented since it would require complete understanding of the context in which the data were collected. The following

listing (Figure 1) contains the data gathered during graphic card initialization and graphic mode setting for resolution of 1024*768 in 32 bit color palette with 60Hz of vertical refresh rate [3]. Address range from D8000000(hex) to D8FFFFFF(hex) refer to memory mapped graphic card input/output address space, while range from D0000000(hex) to D7FFFFFF(hex) refer to frame buffer range. The observed graphic card is based on GeForce MX400 chip with 64 MB RAM on board. The format of the shown listing is as follows. The first column is an address of the instruction that access to observed memory range. The second column tells us the access type (read or write). The next column presents access data length (byte/word/dword), and it is followed with the column that holds the actual data. The last column presents the address of the memory location (the accessed location).

Instr. Address	Access Type	Length	Data	Data Address
0xbfc96bce	Write	DWORD	0x0000000	0xd8000140
0x80455b74	Read	DWORD	0x11000b2	0xd8000000
0x80455b5c	Read	BYTE	0x55	0xd8700000
0x80455b5c	Read	BYTE	0xff	0xd870c400
0x80455b74	Read	DWORD	0x011000b2	0xd8000000
0xbfbccac2	Read	BYTE	0xff	0xd8300000
0xbfbca696	Read	DWORD	0x011010de	0xd8001800
0xbfbcad5f	Read	DWORD	0x08c10110	0xd8100200
0x80455c00	Write	DWORD	0x00000000	0xd8001830
0x80455bd8	Write	BYTE	0x1f	0xd86013d4
0x80455b74	Read	DWORD	0x02005748	0xd8001084
0x80455b74	Read	DWORD	0x011010de	0xd8001800
0x80455b74	Read	DWORD	0x801d4547	0xd8101000
0x80455b74	Read	DWORD	0x02b00007	0xd8001804
0x80455b74	Read	DWORD	0x03110111	0xd8000200
0x80455c00	Write	DWORD	0x1f000102	0xd800184c
0x80455b74	Read	DWORD	0x00004603	0xd8680504
0x80455b74	Read	DWORD	0x000009ff	0xd8100228
0x80455b74	Read	DWORD	0x0003be0c	0xd8680508
0x80455b74	Read	DWORD	0x11000b2	0xd8000000
0x80455b5c	Read	BYTE	0xeb	0xd8700003
0x80455b74	Read	DWORD	0x00000000	0xd8009200
0x80455b74	Read	DWORD	0x04000000	0xd810020c
0x80455c00	Write	DWORD	0x03020100	0xd8002210
0x80455c00	Write	DWORD	0x00000000	0xd8710000
0x80455c00	Write	DWORD	0x00000000	0xd8711000
0x80455c00	Write	DWORD	0x00000000	0xd8712000
0x80455c00	Write	DWORD	0x00000000	0xd8713000
0x80455c00	Write	DWORD	0x00000000	0xd8714000
0x80455c00	Write	DWORD	0x00000000	0xd8715000
0x80455c00	Write	DWORD	0x00000000	0xd8716000
0x80455c00	Write	DWORD	0x00000000	0xd8717000

Figure 1: Instruction Address Access Type Length Data Address

Explanation and information extraction is not the topic of this paper, but for the illustration purposes will take a look at the instruction on 80455B74(hex). This instruction makes access to frame buffer range mapped with offset 00100000(hex) from the start of the memory mapped input/output graphic card address space (which start on the D8000000(hex) in this particular case). The address of the accessed register is 0000020C(hex) from the start of the frame buffer. The name of this register is NV PFB BOOT 0 RAM AMOUNT, and it holds the amount of onboard RAM memory. The information about available memory is coded with bit range 27- 20. Meaning of those 8 bits is interpreted as follows: 02(hex) (2MB), 04(hex) (4MB), 08(hex) (8MB), 10(hex) (16MB), 20(hex) (32MB), 40(hex) (64MB), and 80 (hex) (128MB). It is obvious that in this particular case we had a graphic card with 64 MB RAM onboard, which is verified with the value that was read (04000000(hex). From the presented example it is not difficult to understand the overall complexity of this project. The project in which it was necessary to analyze millions of bits and to find the meaning for each one of them, as it was defined by one of our project members.

5. CONCLUSION

Usage of data mining techniques for purpose of dynamic reverse engineering is still to be explored, but methodology and algorithm defined here is a platform independent and therefore could be used as a base for all future researches in this domain. Generality of the presented algorithms suggests that they could be used with minimal adjustments for solving a wide range of different problems. Further development of the algorithms in this domain would significantly decrease software engineering expanses and consequently increase total quality of software products.

REFERENCES

[1] Bruha, I., “Data Mining, KDD and Knowledge Integration: Methodology and A case Study,” SSGRR 2000.

[2] Chikofsky, I., Cross, J., H., “Reverse Engineerig and Design Recover: A Taxonomy,” *IEEE Software*, vol.7, no.1, Jan. 1990.

[3] Cvetanovic, M., Radivojevic, Z., “Dinamicki reverzni inzenjering: Metodologija za prikupljanje podataka,” *YUInfo*, Kopaonik, Serbia and Montenegro, March 2003.

[4] Fayyad, U., Shapiro, P., Smyth, P., Uthurusamy, R., “Advances in Knowledge Discovery and Data Mining,” *MIT Press*, 1996.

[5] Glumour, C., Maddigan, D., Pregibon, D., Smyth, P., “Statistical Themes nad Lessons for Data Mining,” *Data Mining and Knowledge Discovery* 1, 11-28, 1997.

[6] Hecht-Nilsen, R., “Neurocomputing,” *Addison-Wesley*, 1990.

[7] Erdos, K., Sneed, M., “Partial comprehension of complex programs (enough to perform maintenance),” *Proc. 6th International Workshop on Program comprehension*, 1998.

[8] Intel Corporation, “Intel Architecture Software Developer’s Manual. Volume 1: Basic Architecture,” *Intel Corporation*, Santa Clara, CA, 1999.

[9] Intel Corporation, “Intel Architecture Software Developer’s Manual. Volume 2: Instruction Set Reference,” *Intel Corporation*, Santa Clara, CA, 1999.

[10] Intel Corporation, “Intel Architecture Software Developer’s Manual. Volume 3: System Programming,” *Intel Corporation*, Santa Clara, CA, 1999.

[11] Microsoft Corporation, “Microsoft Windows 2000 Driver Development Kit,” *Microsoft Corporation*, Redmond, WA, 2000.

[12] Pyle, D., “Data Preparation for Data Mining,” *Morgan Kaufman*, 1999.

[13] Tilley, S.R., Paul, S., and Smith, D.B., “Towards a Framework for Program Comprehension,” *The 4th Workshop on Program comprehension*, 1996.

[14] Woods, S., Yang, Q., “The program understanding problem: analysis and a heuristic approach,” *Proc. 18th International Conference on Software Engineering*, 1996.
[15] Jovanovic, N., Milenkovic, V., and Milutinovic, V., “Issues in Data Mining Infrastructure,” on-line tutorial at http://galeb.etf.bg.ac.yu/~vm/tutorial/tutorial.html, December, 2003.

Internet Application Testing

Prijic, Aleksandar; Jovic, Darko; and Milutinovic, Veljko

Abstract — ***In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: How to find a proper testing solution that will guarantee debugging errors, which will give us reliable software product; this problem is important because it enables us to recognize some errors, which we are not conscious of. Existing solutions to the problem are briefly surveyed in the third part, and their drawbacks are underlined; each surveyed piece of research is analysed according to the same template. Then, the essence of the proposed solution is presented: As our proposal, we have developed a specific test framework, which as a result, brings significant improvement and speedup of the process of online game system functional testing. The main idea is to run tests from test environment JUnit. It is explained, on a concrete situation, which tests to perform, so we can be sure that the system works properly. The conclusion is from the point of view of performance/complexity ratio.***

1. Introduction

BEFORE we start to discuss about concrete situations, as a primary thing, we have to define the term *testing*, itself. Definitions which most precisely describe this term are the following: "… process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component..." [1], "... involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. " [2], "... process of executing a program or system with the intent of finding errors." [3] Considering that, testing does not provide the software without errors; the main goal of the testing is to locate them. On the other hand, it has to secure that the system works properly, even if there are errors. It is necessary that the person that executes testing properly simulates interaction between software and its environment, which includes identification and simulation of the interface used by the system and all the inputs that can appear.

Manuscript received January 30, 2006. This work was supported in part by the Finsoft Company, London, United Kingdom.

Prijic, Aleksandar, Jovic, Darko, and Milutinovic, Veljko are with the Faculty of Electrical Engineering, University of Belgrade, Serbia.

All intellectual property mentioned in this paper is the property of Finsoft, UK.

In the case of the internet applications, the first step is to recognize the iterative nature of the cycle, which would make the testing itself a lot easier. However, this usually brings changing demands during process, resulting in many problems (e.g. testing functionalities that have been changed or that do not exist at all). „The most dangerous errors that can appear are systematic. They are caused because of wrong assumptions that author can predict." [4]. Also, one of the main factors for successful testing is knowing internet explorer, operating systems, data bases, script languages, web servers, and transaction protocols. As it is already mentioned, it is required to set the appropriate environment, so this system will look alike the environment in which the application really works. [2] The most crucial factor is good planning. First it has to be made the detailed plan, which has to be broken in the functional components. After this step, these components are being prioritized and tested according its priorities.

2. Problem Statement

At this very moment, Internet is having its explosion, and the process of testing has to follow it up. One solution for this problem is to increase number of people which are executing testing, while the other is to make that process an automatic. This solution is becoming the strategic need and the critical factor for the process of software evolution. During the test development strategy, it is necessary to minimize the influence of changes in applications and, also, in the tools used for testing. It is necessary to implement the unique test environment, which will improve and develop itself according each new application applied (e.g. acclimatization for different versions of software).

So we can even think about unique test environment, we have to consider a few facts: [3]

-Automation has to be the central task, not the secondary problem. Test design environment and writing code require plenty of time and effort. This is not a project that someone can do it in his free time. This implies a complete software project, which has to be documented, inspected and tested.

-Test environment and test design are separate parts. Test environment is executable environment for automatic tests, while test design shows how to test the particular function or characteristic of the application.

-Test environment has to be separated from the application. Although applications are quite simple, components of which they consist of are

not, and that is the reason why environment for test automation has to satisfy these components. Using this fact, all critical components of the application are separated from the application itself, so the environment can be used for other applications as well as for that one. Data which are characteristic for the application itself are being sent to test environment using variables.

-Test environment has to be easy for evolution, as well as for maintenance. System should be modular; each part should be independent, so it can be deployed particularly, which results in simple maintenance. Also, system should be properly documented and defined (standards and templates for documentation).

-Test strategy vocabulary and test design should not depend on test environment. Test strategy should define low level vocabulary, which will be used for all applications and should be independent from test environment. On the other hand, test design for a certain application should define high level vocabulary, which is specific only for that application and, also, independent from test environment. This fact should improve manually tests, as well as automatic. Vocabulary should be understandable for an ordinary person.

-Test strategy and test design should hide its environment complicacy from the persons which execute testing.

Having these facts on our mind, we tried to evolve the optimal solution.

3. EXISTING SOLUTION AND THEIR CRITICISM

There are a great number of commercially available environments for test automation, which are based on key words (*commercial keyword driven frameworks*). They are mostly used as bridge between applications, which are being tested, and tools for the automatic tests, which the company is developing. These kinds of products require some modifications so they can respond to the test demands. Some of the known tools are:

Puffin Automated Framework – environment for test automation, completely implemented on the programme language Phyton. This project has open code. At the beginning it was used for testing the internet applications, but it was timely evolved, so now it can also be used for supervising the system and for automatic data transmission. [6] Whole environment is based on four types of objects:

- Action – executable object, which can receive one or more inputs and product one or more outputs. Its results can be checked according to the expected standards of the certain type.
- Action token – represents the value used as an input for some action, or the value generated as an output after executing an action. These objects can be associated with data system or data base. They can be used for execution control or condition.
- Task – group of actions which all have to manage completely so the task succeeds. It can be arranged that some task can be executed only if the other task is completely performed, or that failure of the task determines the whole project.
- Plan – structured execution scheme, which consists of set of tasks. In the plan it can be defined which tasks execute and do they execute depending of the certain condition or until the task is complete.

Apache JMeter – tool designed for functional testing of applications and for measuring performances. [7, 8] It is completely implemented in Java. It was first designed for testing internet applications, but timely spread with other functions. This tool can test performance of static and dynamic resources. It can be used to simulate server, network or object load. Using this property, we can test persistence of the system, or analyze performances in the different environments. The interface consists of two elements:

- TestPlan – contains tests which have to be executed.
- WorkBench – implements the work space where the tests are being created. This part can be broken in a as many sections and subsections as needed.

4. PROPOSED SOLUTION

Games can be divided in following types:

- *"One click" games (FOSC - Fixed Odds Single Click)* – user bets on particular selection. Game random generates number and depending on the result, user is winning or losing.
- *High-Low games* –session is performing; user bets money and plays the game. Session enables transactions execute after the end of the game.
- *Lottery games* – there are a several players at the same time. When game starts all players raise their bets. System plays an important role; it has to place values and make all transactions, quickly and correctly.

Although each game has its specialties, there is a certain element collection, common for all games.

First we have to explain the system architecture, for which we proposed test framework. After system architecture, we will introduce you to our solution for online game testing problem.

Elementary structure of the system (Figure 1) consists of:

Games Room Server (GRS) is the central segment of the system. It is realized in the programme language Java, apropos Java Servlets and Java Server Pages technologies. Used data base is the Oracle. The games communicate with server using XML messages. All games are created in the Flash technology.

They can be animated from some of the useful explorers.

GamesWeb (GW) is an internet application used as an interface between Flash games and Games Room Server. This application enables users to actuate games from internet explorers.

GamesManager (GM) serves as backing up for games**.** Backing of the game comprises watching the game and the whole system, configuring prices, generating reports for game analysis, and coordinating in case system fails. GM is an internet application, which is available only for the client. Accessing this application is only by combination of username and password, so as with IP address license.

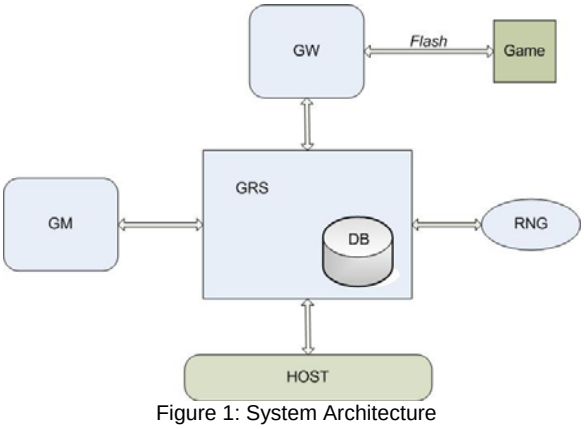


Figure 1: System Architecture

Random Numbers Generator (RNG) is elementary segment of the system. It is very important that this segment works properly. Quality of this part has direct influence on quality of the system, and its speed influences on maximum number of concurrent users. It can be implemented software (Java) as well as hardware.

Host system is already known system client, which has to enable financial transactions. Actually, it is a connection between client and electronic financial business system. It has to enable operations with client bills, registering clients, canceling transactions, etc. All client bills, including all necessary information, are stored in the system.

4.1 System Description on Concrete Example

Game testing system is realized on programming language Java. Thereat, backbone for running tests is test environment JUnit.

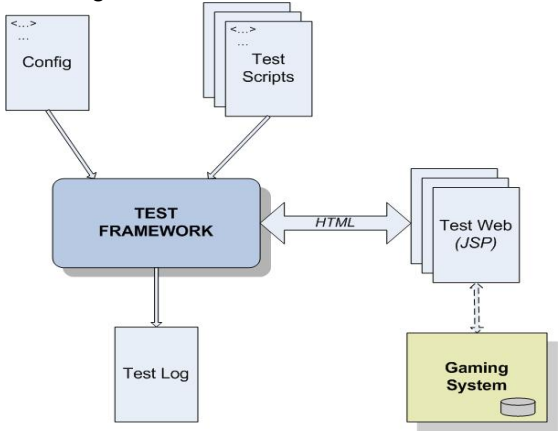


Figure 2: Test Framework

Figure 2 represents proposed components for test environment, which is being used for online game system (Figure 1) testing. It consists of main application – *Test Framework,* which inputs are *Config Files* and *Test Scripts*, and the output is *Test Log*, which holds information about the communication between *Test Framework* and gaming system, as well as the errors which occurred while the testing is being performed.

Communication between game and server is performed through XML messages. All tests are, also, written in a simple language based on XML, which has been developed for the purposes of system testing. Every XML tag has a corresponding Java class. During test execution, test framework reads a test and, according to decrypted objects, takes certain actions, calls adequate classes, thus bringing the system to adequate state.

Basic segments of tests are processes, which represent concrete actions that have to be executed. Each process establishes a connection to appropriate part of the system, after which specific action is being performed.

Processes can be grouped in three segments:

- *<initialization/>* - serve for system initialization prior to tests execution.
- *<destroy/>* - performed at the end of test execution and they do some final actions (e.g. releasing resources).
- *<tests/>* - perform concrete test actions; simulate game playing and check parameters.

System upholds control structures (e.g. exists adequate XML operator for *while* and *if* loops). That enables writing of powerful scenarios of tests for simulating game playing and testing game characteristics. Also, it is important to mention that there is a support for simulating concurrent playing of a number of players (XML marker exists, which enables execution of several tests at the same time). This is of great importance during performance testing.

Test Web (JSP) serves as an interface between test application and a system which is being tested. These pages receive HTTP requests and perform corresponding actions (logging in the system, getting session, setting parameters, setting random numbers, playing game…).

In order to speed up a process of testing, an application has been developed, which enables one to choose all automatic or performance tests that need to be run, and with just one click send them all into processing queue. During test execution it is possible to see number of tests performed, number of tests still waiting in the queue, number of errors etc. Also, every test has a corresponding log file in which one can see complete process of test execution, for the analyzing purposes.

Automatic Tests simulate playing the game. They should, during the game running, cover all bet options within the game and all values of the bets. This is not always possible, but should be

tried. Thereat, during the play, some of the important parameters should be checked inside of tests (bill account, bet, profit, paid sum, etc.).

Performance Tests are used for testing game performance. The main idea is not to check game parameters, but to simulate playing a great number of parties, by the great number of players (i.e. game is tested with 50 thousands played parties - 25 players, where each player played 2000 parties.).

Beside automatic and performance tests, it is important to mention existence of *manual tests*. Their main purpose is not to check game parameters, but to bring system into desired state, after which tester performs manual inspection of specific game functionality.

4.2 JUnit

JUnit is an environment for test writting, based on programme language Java. It is an instance of an architecture *XUnit* for test environment. It enables tester to write a set of automated tests which initialize, execute and inspect the code being tested. The tests are mutually independent and their sequence is not important. As a result, a programmer gets a report with total number of successfully and unsuccessfully performed tests.

4.2.1 Design

Environment design is based on two patterns, *Command* and *Composite*.

Command object is *TestCase*. Each class that comprehends test methods should be subclass of class TestCase. This class can define number of public testXXX() methods. If someone wants to check expected or current test result, he should call method assert(), or any of its modifications.

Subclasses of class TestCase which consist of more testXXX() methods can use setUp() and tearDown() methods for initialization and exemption each object which is being tested. These objects constitute test fixture. Each test works in context of its own fixture calling methods setUp() and tearDown(), before and after, respectively, each test method. With this action, it is prohibited side effects between test executions.

TestCase instances can be associated in *TestSuite* hierarchy, which automatically call all testXXX() methods from each TestCase instance. TestSuite is a combination of other tests, whether TestCase or TestSuite instances. This attribute enables to collect all tests, so they can be uniformly and automatically executed (there is only one result – success or failure).

4.2.2 Test Execution

Running tests is available either individually or within TestSuite object. In case of running TestSuite, automatically it runs all minor TestCase or TestSuite instances. Running TestCase object, it automatically runs all its public testXXX() methods.

JUnit provides either text or graphic user interface. Both intefaces show how many tests are running, errors, if they occured, and status report. The main characteristic of both interfaces is their simplicity.

For running tests using text interface, a command is used:

⇒ java junit.textui.TestRunner
ShoppingCartTest.

If all tests pass, text interface represent the message “OK”. Otherwise, an error message appears.

For running tests using graphic interface, it is used command:

⇒ java junit.swingui.TestRunner
hoppingCartTest.

Graphic interface shows *Swing* window with green line, if all tests pass, or with red line, in case some test fail.

4.3 Functional Testing Game “Fortune 36”

In this chapter, it is described the process of functional testing using concrete example, game “Fortune 36”. Appearance of a game is shown on the picture (Figure 3).

During functional testing, following phases have to be passed by:

1. Defining and introducing a set of test cases and writing adequate document (*test case document*).
2. Writing test cases according the document (preceding step).
3. Running and executing tests.
4. Writing test reports.

First two phases require most effort and time, and these phases have direct influence on correct and rapid passing through the rest of the testing. At the beginning, it is necessary to introduce the game itself, its characteristics and available options. This step is very important, because according this, test cases are noticed properly. Test cases should cover all game characteristics, with an accent on the limited situations, where an error can occur. That is why test cases should be carefully picked, which, sometimes, represents problem.

For testing the game “Fortune 36”, the following test cases have been noticed:

- *User Interface Test* – checks the buttons in the game; do all the buttons work properly. It does not matter if an account of the user is wrong calculated, as long as all the buttons do their job correctly.
- *Sound Test* – checks do music and sound effects, at the time they should appear, work correctly.
- *Game Play – Single Win* – a scenario of playing game in case of one winning. Special attend is on correct update of client’s bill (sum of entered and wined money). It checks weather these data are properly registered in application GameManager.

- *Game Play – Single Loss* – this scenario is similar to the previous one, except this scenario is for loosing money.

Figure 3: Game "Fortune 36"

- *Game Play – Several Bets, Only One Win + History Test* – tests situation when more dices are entered on more numbers (option where only one bet wins). Special attend is on correct update of client's bill (sum of entered and wined money). Also, it checks are the previous played games remembered within option History.
- *Game Play – Several Bets, Only One Loss* - this scenario is similar to the previous one, except this scenario is for situation where only one bet loses.
- *Game Play with Balance 1 Lost* – this is the situation when a client's bill account is 1, and client plays game with bet 1, and loses. Game, after this situation, has to notify the player that he has no more money on the account, and to forbid him to play any more.
- *Game Play with Bet Out of Balance* – tests the situation where player tries to bet the dice, which has bigger value than he has on his account. Game should obstruct this activity and send player adequate message.
- *Balance 0* – scenario where a player tries to run the game, even if his account is 0. Game should notify the player and forbid him to play.
- *Manual Payout* – for each game, for security reasons, should be defined limit for winning, after which the payment should be manually. This test tests does the game recognize this situation and, also, does it inhibit direct payment which is bigger that the limit is.
- *Changed Balance* – this is the situation when player bets a certain sum and before he confirm it someone aside changes his account (the account is now lower than the betted one). Game must recognize this situation and must not accept this bet.
- *Min/Max Stake* – for each game it is possible to set minimum and maximum bet value. This test tests situation when player tries to bet a sum, which is lower that the minimum value or bigger than the maximum value.
- *Games Manager Test* – tests work of application GamesManager; does the setting of a certain parameters inside this application influence the game itself.
- *Fun Mode Test* – each game can be played because of fun, not because of money. This scenario checks performing the game within this regime.
- *Automatic Tests* – these tests simulate playing the game.
- *Performance Profiling Test* – tests game performance (game "Fortune 36" is tested with 50 thousands played parties; i.e. 25 players, where each player plays 2000 parties).

After the document is over, for each test case, the test is being written. Structure of the tests is described in chapter 4.1. *Apache Ant* is being used for their execution. During the execution of the tests, it generates certain messages, which help to follow the flow of execution. If the test recognizes an error, it is being noticed in the report.

5. CONCLUSION

Number and complicacy of internet applications are being exponentially growing, and demands for their quality is being bigger each day. This paper represents process of software testing, with an accent on internet applications. Paper tries to explain basic aspects of application testing in the conditions, such as internet environment and privilege of the automatic testing. After theory of the automatic tests, it is given description of tools, which enables one to see how the process works, and it can also help choosing tools and evolution of new systems.

Testing on line games system represents good environment example, which is being created developing existing environments and their acclimatization on our demands and application specialties.

Software testing, particularly internet application testing, is extremely important process, which has to be attended special caution in case of serious and concurrent software product.

REFERENCES

[1] ANSI/IEEE Standard 729, 1983.
[2] Hetzel, W., "The Complete Guide to Software Testing," *QED Information Sciences Inc*., 1984.
[3] Myers, G. J., "The Art of Software Testing," *Wiley*, 1979.
[4] Frederick Brooks Jr. "The Mythical Man-Month,"
[5] David Janzen, Hossein Saiedian: "Test-Driven Development: Concepts, Taxonomy, and Future Direction," IEEE Computer, September 2005.
[6] Weissingerk, K., "Web Application testing with Puffin: Puffin testing framework, Part 1," http://www-128.ibm.com/developerworks/library/os-puffin.html
[7] http://jakarta.apache.org/
[8] Hansen, K., "Load Testing your Applications with Apache JMeter," http://javaboutique.internet.com/tutorials/JMeter/

Patent Maps: A Simplier Way to Search Patents in the Light of Prior Art

Micanovic, Mina; Milutinovic, Veljko; and Milligan, Charles

Abstract— ***A patent represents an exclusive right to make, use and sell an invention in a country. In order to make it valid, it is necessary to go through a series of verifications among which there is a search to establish if there is any public document which fully or partly describes the invention which is being patented. If such a document exists, it is called prior art. The search, whose purpose is finding of prior art for the patent in question, is not in the least simple. Patent Maps, presented in this work, can significantly facilitate the process of search for prior art among the patent archive.***

1. INTRODUCTION

A patent represents an exclusive right to make, use and sell an invention in a country. It is necessary that the potential patent goes through a very long and complex process of verifications during which is determined whether the invention is new in order to be accepted by Patent Office. If it would be accepted as a patent even it was not new, that would mean that the researcher got the award for the work which brought the public the things they were already aware of. If, during the verification, it is determined that the potential patent is not new, it is rejected.

In order to determine the validity of the potential patent, the first step is to determine which documents can be considered as a prior art against the patent's claims. These documents certainly fully or partly describe the invention and they fulfil the condition that they have been issued before the filling date of the application. However, the national laws concerning the patents are different from country to country, so public discussions do not have the characteristic of prior art in all countries.

So, the question of which documents can be considered as prior art is being raised? In the first place, these can be all the documents which are public, whether as publications (books, magazines, newspapers, Internet, PhD's, etc.), public discussions, accepted patents or some other public forms. We will not be looking into this problem any longer because this work is dealing with already accepted patents as prior arts.

Let us look at the Mind Maps now. Researchers claim that the Mind Maps save up to 50% in the human process of thinking and creating, they decrease the time spent for understanding of information for 90% and enable more comprehensive observing. Human brain is capable of accepting the coloured, multileveled Mind Maps, which are built on connections, much faster than the simple text. A machine certainly is not capable to accept information in this coloured way or speeding up its process of 'thinking', but the comprehensive graph which gives the Mind Map is possible to use.

The machine Mind Maps are better known by the name of Concept Maps.

2. CONCEPT MAPS

Concept Maps represent graphic structures used for organizing and presenting of knowledge. As such, until now, they have been mainly used for the advancement of learning process (better known as MindMap), making decisions and conclusions. The structure of such a map is shown in figure 1.

Judging by their name, Concept Maps are consisted of concepts which are framed with geometric structures, rectangles or circles, and the relation between them is represented by joining links. Every one of these links can be characterized by a word or a phrase which describes the type of a connection between concepts.

There are two types of links: role-oriented and class-oriented. Role-oriented links are all the links which have the semantic meaning of a predicate in a sentence. Their characteristic is that if the role disappears, i.e. the predicate is lost, the identities of objects do not change. Unlike the role-oriented links, class-oriented links actually show that the subordinated object belongs to the superordinated one and as such, the subordinated object cannot break the relation without losing its identity. We should, also, bear in mind that there is a larger number of role-oriented links in a Concept Map and that the

Manuscript received August 1, 2006. This work was supported in part by the Sun Microsystems.

Micanovic Mina and Milutinovic Veljko are with Faculty of Electrical Engineering, University of Belgrade.

Charles Milligan is with Sun Microsystems, USA.

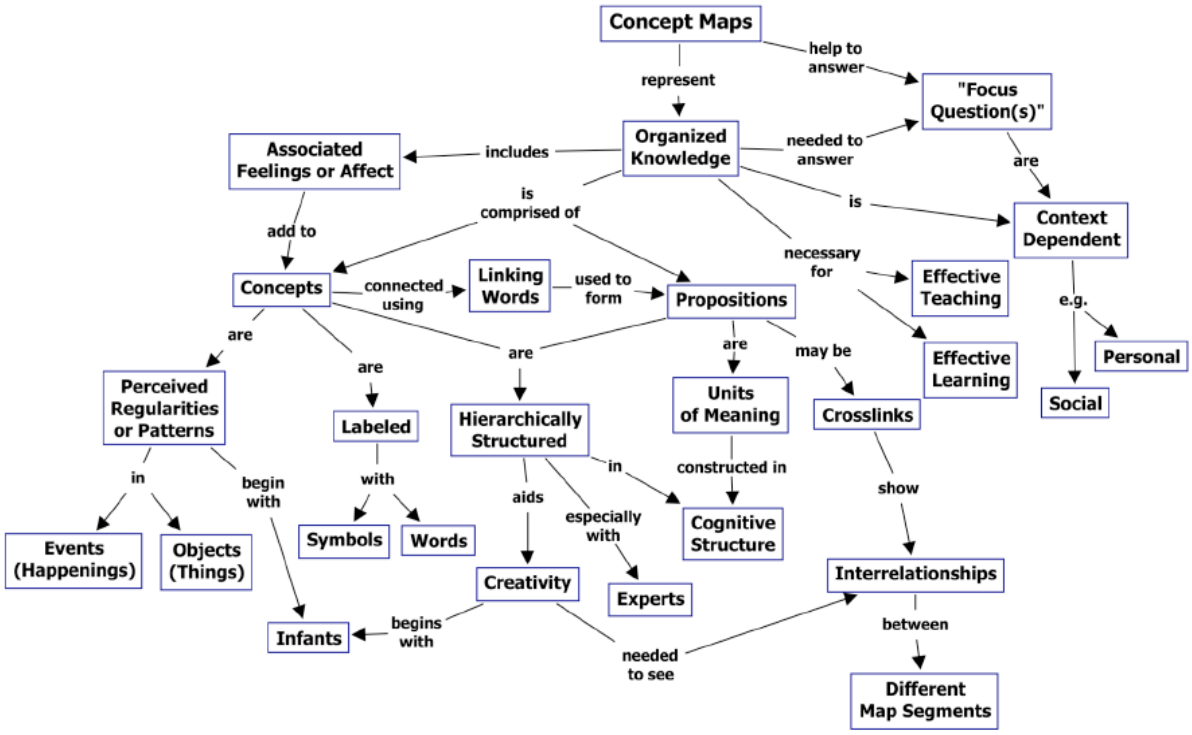


Figure 1: Concept Map Structure

probability that the object will lose its identity when breaking the relation is extremely small.

The fact that a Concept Map is hierarchically organized with the basic concepts in the root(centre) of the map is its very important characteristic.

Going through the map from its centre to its ends(branches), we can find more and more specific concepts. Each of these specific concepts can be the central concept in another Concept Map. This is the reason why new types of links are introduced, the links which play the role of the indicators of the main concept. These links are oriented from the instance of concept to the main concept and they can be thought as a kind of class-oriented links of a reverse direction with a difference that the instance can exist even when the Concept Map of the main concept does not exist.

In order to keep the records about the position of the concept inside a Patent Concept Map, the levels of layers are introduced. The basic concept which represents the root of a Patent Concept map is situated on a level zero. All the other concepts are situated on levels 1 till N depending on the distance from the central concept. The distance of the concept from the root of the Patent Concept Map is calculated as the number of links which can be found on the shortest direct relation between the central concept and the concept in question.

Cross-links are another type of links in Concept Maps which are used today. Cross-links are links which join concepts in different segments or domains of a Concept Map. These links are, also, role-oriented links. However, their characteristic is that they cannot be detected automatically(by using a machine), but for this action, the intervention of a man is necessary and its effect depends on ability, knowledge and creativity of a person who intervenes. In our case, these links do not have any value when being used because they are intended for facilitating of memorizing in learning process which is not the issue discussed here.

So, the main reason for using Concept Maps when making a patent is the possibility of showing the complete content of the patent. From this moment on, we cannot call these maps Concept Maps owing to their original meaning. We will call them Patent Concept Maps, instead. We should, also, bear in mind that one patent is consisted of larger number of Patent Concept Maps, so that the whole group can be called a Patent Map.

3. *Automatic Generation of Patent Concept Map*

Automatic generation of Patent Concept Maps represents a problem in a way. We must not forget that the Mind Maps are created by a man himself based on his mental activities and they are adjusted to the individual who creates them and they are the result of a very subjective process. In order to avoid the subjectivity which is a complete failure looking from a scientific point of view, during the process of forming of maps, it is necessary to do that according to the universal principles which are void of subjectivity in any way. In the case of a patent, the individual-the creator of a map is a machine and through history we have witnessed many useless attempts of implementation of human behavior into machines. However, we must not forget the

bright examples when the attempts resulted in real sensation, which are the motive for this research. The potential solution of this problem lies in the usage of Natural Language Processing (NLP) software, i.e. The Stanford Parser as a representative.

The Stanford Parser is an NLP tool for automatic grammatical text analysis. This package is a Java implementation of probabilistic natural language parsers, both highly optimized PCFG and dependency parsers, and a lexicalized PCFG parser. At the beginning the sentences are directed toward the end where their grammatical position is given as well as mutual grammatical dependance of words in the sentence which is being examined. Its functionality is not completely ideal but the percentage of errors is very small. More details about parser can be found in 2 and 3.

So, in order to create the Patent Concept Map of a claim, it is necessary to process claim through parser. Based on the given result, the Patent Concept Map is formed like this:

The subject represents the central concept; the level of layer 0.

The object represents the lower-level concept; the level of layer 1-N.

The predicate represents a role-oriented link which joins the subject and the object which that predicate refers to.

The adjective represents a characteristic (type) of concept it refers to.

The adverb represents a characteristic of the role-oriented link whose label is the verb which that adverb refers to.

The articles are neglected.

The generation of the Patent Concept Map according to the rules given above is performed by a software, so called Patent Map Generator, expert system made in Java which is in the making. If it is necessary, it is possible to do the revision after the generation of the map.

This part can be broken in a as many sections and subsections as needed.

“Proposed instructions for cache injection support are hand-inserted into synchronization kernels and parallel applications. A detailed simulator of memory subsystem is developed. For each application, we compare the performance of base system and one or more systems that include cache injection. Experiments varied different memory subsystem parameters, in order to explore their influence on cache injection.”

4. PATENT MAP STORAGE

There is no point in placing Patent Maps in graphic shape. The graphic pictures make it easier for a man to have a complete insight into a revision process, but they are not adequate means of keeping the data .

Apart from the graphic way, these maps can be very simply and efficiently showed and implemented by means of RDFS, the W3C standard for annotation of web resources with metadata. Since the data in RDF are presented in triplets, this way of showing the maps is completely adequate, taking into consideration the subject-predicate-object structure which can be found in the Patent Concept Maps. The additional advantage represents the RDFS type system which has the possibility of showing the types of roles (role-oriented) as well as specific types-characteristics of concepts. You can read more about this in 1.

However, we have not solved the problem of placing the maps by defining the shape in which the Patent Maps are placed. Considering great number of data, it is necessary to put similar maps on the same clasters which has the purpose of improving the efficiency of search and getting the information retrieval. There are many criteria based on which it is possible to do the classification:

The classification of maps based on the criterion of similarity of central concept;

The classification of maps based on the criterion of the group of authors;this criterion is a consequence of the fact that today certain research teams are dealing with researches in specifically defined areas with a small percentange of exceptions all over the world;

The classification of maps based on the criterion of scientific area which the patent belongs to;nowadays, this kind of belonging is sometimes very hard to determine considering the encreasing interweaving of different scientific areas.

We would certainly get the best result with the use of the hybrid criterion of classification with the emphasis on the classification based on the criterion of similarity of central concept. But, whichever criterion we choose, the hierarchical structure of patents and their references will not be harmed which is extremely important.

5. CONCLUSION

There is no doubt that showing of patents by means of Patent Concept Maps needs an extremely large space and a large number of processors which will perform parallel search, but this problem can be solved by simple usage of Grid Storage which can be found everywhere nowadays. The main request which should be fulfilled is the quality projected architecture because it guarantees efficiency.

The Patent Maps showed in RDFS are very useful for searching of patents with the purpose of determining whether any of them is a prior art against the patent in question. Further research will go in the direction of the construction of algorithms used for recognizing the structures inside the Patent Maps with the purpose of

advancing the search technique.

We can make a remark here that in order to use this kind of approach, all the patents which are placed in the patent archive must go through the process mentioned above so as to be available for the search. First, every patent must pass through a parser which would extract the relevant parts of a patent which would be transformed into the Patent Concept Maps further on.

REFERENCES

[1] Allert, H., Dolog, P., Nejdl, W., Siberski, W., Steimann, F., "Role-Oriented Models for Hipermedia Construction – Conceptual Modeling for the Semantic Web," *Technical Report*, February 2003.

[2] Klein, Dan, and Manning, D., Christopher, "Fast Exact Inference with a Factored Model for Natural Language Parsing," *In Advances in Neural Information Processing Systems 15* (NIPS 2002), December 2002.

[3] Klein, Dan, and Manning, D., Christopher, "Unlexicalized Parsing," *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 2003.

[4] Novak, J., D., and Cañas, A., J., "The Theory Underlying Concept Maps and How to Construct Them," *Technical Report IHMC CmapTools 2006-01*, Florida Institute for Human and Machine Cognition, 2006

[5] Lassila, Ora, Swick, R., Ralph, "Resource Description Framework (RDF) Model and Syntax Specification," *W3C Recommendation*, 22 February 1999.

[6] Berners-Lee, T., Fielding, R., Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax," *Internet Engineering Task Force*, RFC 2396, August 1998.

[7] Patents Factsheet 1: The description, DDU 4263 - 04/00

[8] Ius Mentis http://www.iusmentis.com/patents/priorart/

[9] United States Patent and Trademark Office http://www.uspto.gov

SwanLink: Mobile P2P Environment for Graphical Content Management System

Popovic, Jovan; Bosnjakovic, Andrija; Minic, Predrag; Korolija, Nenad; and Milutinovic, Veljko

Abstract—*This document describes major trends in the present, and what authors reckon as the future of distributed computing and online collaboration. It considers topology and communication protocols which will be used as standard in future distributed application. Analysis of everything that has spawned from Fred B. Holt and Virgil Bourassa original idea concerning methods for online collaboration based on regular graph topology is presented. After introductory part on current implementation and techniques related in this field, our solution is presented through the project which is a practical implementation of the ideas presented in this paper. Facts and dogma about modern computer infrastructure, communication protocols, and its reliability and scalability are discussed here. Authors' point of view and vision of future work in this field of distributed application is also included.*

Index Terms—*calm technology, context-awareness, pervasiveness, ubiquitous computing*

1. Introduction

THIS document is about humans' all increasable need to communicate with other regardless to space distance that divides them. Increasable grow of telecommunication technology and internet give the people opportunity to communicate with each other in most various ways. Classical communication methods such as e-mail messages, seems to be just satisfying in first era of communication technology. In the first generation of communication technology sending messages to friends and waiting for them to answer was enough for most of the users. In those kinds of systems simple client-server architectures, where one entity has just send request to server (i.e. e-mail message), waiting them to fulfill his need was developed, was quite stable and has done its purpose very well. But increasable desires, such as real time communication with complex multi-medial content, bring many problems to the distributed system architecture. Problems which need to be solved in new architecture were:

- Bandwidth,
- Scalability,
- Stability,
- Mobility.

Those kinds of problems can not be solved using traditional client server approach. Client server philosophy is based on assumption that we have one powerful computer (i.e. mainframe) which serves tens or hundreds small clients. Increasable number of personal computers, with even increasable power, low cost, and availability to ordinary man, opens the new world of opportunity for techniques and protocols which can be used to interconnect them. One of the most promising infrastructures which indicate that will be standard used in most of the future application is peer to peer (P2P). P2P is philosophy where lots of independent computers collaborate with each other, regardless their power and location. In following sections we will describe how P2P architecture can be used to solve a problem of cache consistency in distributed applications.

2. Problem Statement

Theme of our research was development of distributed application which enables complex graphical context exchange between users regardless their distance and number. Application should be able to support large number of participants that can concurrently modify common unique content.

Graphical applications are the most complex type of content management systems regarding the various type and size of the elements, which need to be exchanged with other participant. Due all participants have content cached in local, one of the biggest problem was how to maintain cache consistency between them asynchronies without forcing them to ask for refreshing their local content.

Philosophy we use here is a little bit different from classical push-pull methodology which is used in traditional internet approach, which is currently used on the internet, where the user explicitly demands for update their local content.

Our problem was how to develop fast, scalable and reliable infrastructure and protocols which can be used for maintaining local views for all participants, without their interference in synchronization process.

Manuscript received April, 2006. This work was supported in part by the Panthesis, Inc., USA.

Popovic Jovan, Bosnjakovic Andrija, Minic Predrag, Korolija Nenad, and Milutinovic Veljko are with the Faculty of Electrical Engineering, University of Belgrade, Serbia.

In order to do that, the next problems had to be solved:

- managing user joining and leaving,
- keeping consistent cache state,
- deleting objects,
- sending messages.

3. *Existing Solutions and Their Criticism*

In the history we can find many attempts to implement online collaboration between distance participants. Distributed architecture as we know it today, had several major trends in the past several decades regarding relationship between computer(s) and person(s) using it (them).

Client-server solution appeared as a very first solution, obviously ideal for the set of problems that existed in the very beginning of the network. In the very beginning of computing, only mainframe computers existed, and we have bunch of small terminals using their services to fulfill their needs. This approach was predecessor of Client-server architecture that currently being used.

Collaboration between clients has been established via central server using hub and spoke methodology - client sends message to server and server broadcasts the message to all other clients. In this kind of architecture clients were responsible for handling the messages received.

Those kind solutions were appropriate for small and compact communities that are localized on geographically small area, and it is inappropriate for highly reconfigurable and mobile internet connected communities, which currently exist. Possibilities for real time interaction between clients in client server middleware can be accomplished using so called Hub and spoke message-passing techniques, which is showed on Figure 1. In Hub and spoke system, one client send the message to the server which is responsible for establishing the communication between clients, that server broadcasts the message received to all other participants.

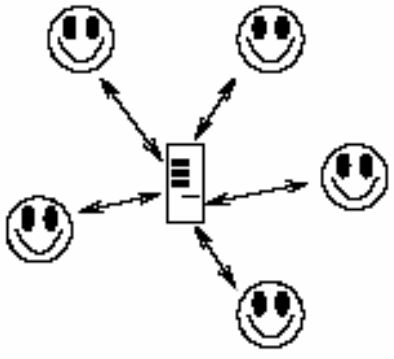


Figure 1: Hub and spoke techniques used in client server middleware for real time participant interaction

The major problem with client server (or hub and spoke) architecture in area of real time online collaboration lies in the fact that this solution is absolutely not scalable. Namely, in communities with thousands participants who collaborate mutually via one server, we can see that server represent the bottleneck of the system. Regarding the issue that in client server system we have N clients on one server, complexity of such kind of system grows as N^2. Due to large number of clients, often server cannot dispatch all received messages from one client to all others, which is unacceptable in modern real time collaboration environments.

Additional problem lies in the fact that client server architecture is mentioned for relatively static communities. Although clients can be placed on various locations, server is (in most cases) placed on static location which is good known to all users. That kind solution is unacceptable in mobile communities where clients can migrate across the world without affecting the session.

Due to mentioned characteristics, client server architecture wasn't so popular in real time application where large number of participants needs to be supported. Client server architecture is effectively used in simple request-response systems, where number of clients can update his local copy of document, post it to the server when is done and retrieve the newest version on demand.

One of the solutions for improving scalability in client server middleware, which is often used, is server farm implementation. Server farm is set of (more or less) equal server instances that cooperate mutually so the client's request can be processed more effectively. Basic idea is that server can mutually redistribute responsibilities for processing client's requests. In contrast to pure client-server solution, where one server cooperates with N clients, in server farm environment we have K server, which mutually collaborate and serve participants request. In server farm system number of participant is partitioned in K groups where one group is under responsibility of one server in farm. Scalability is little bit improved but still it is not satisfying for our solution.

Although server farm has many disadvantage in scalability and supporting the large number of clients, it has found purpose in the effectively online gaming support local network with medium number of participants. Lot of online gaming system has to be implemented using this technology because it supports clients which do not permanently come and go, so scalability issue does not come in the first plan. In our solution, where large number of participants constantly comes and goes, server farm technique can not be applied.

Only solution, which currently exists and can satisfy requests for scalable, real time collaboration infrastructure, which can be easy

reconfigured, is Peer-to-Peer solution (P2P). P2P environment contain large number of collaborator with equal rights. That kind of solution is completely applicable for most of our needs, because it promises that it will support large communities.

4. PROPOSED SOLUTION

4.1. Proposed Solution in General

SwanLink represents revolution solution in area of distributed mobile P2P application that provides users with capability to exchange complex graphical elements with each other. It is a multi-user environment which enables users to concurrently create and modify common graphical content. Graphical content represents set of graphical elements such as lines, pictures and text which can be managed through the system interface. Graphical content is shared among all participants who are currently involved in document managing session, using the local copy stored at each participant's computer. Contents of those copies need to be consistent although they are cashed in physically different locations, without forcing the user to explicitly require synchronization with other participants. Due to such kind requirements system itself needs to take care about maintaining participants' local cache consistency instantly whenever anyone makes some change. Whenever some of the participants alters content on his local view, message that carries information about change that is made, is broadcasted to all others participants. All other participants update theirs local content instantly when they receive notification about modification.

Graphical content is not explicitly owned by any of the participants who are involved in session. Everybody has the equal rights regarding the content management. Newcomers who join the session afterwards will fetch valid copy of current graphical content and be able to work on it just like old ones.

This feature is the key for establishing mobile and secure community for graphical content management. Due to issue that content is saved on EACH participant's computer, chance that it will be lost or destroyed is possible just in case that all participants simultaneously fail down. As long at least one participant "lives" he will preserve valid copy of session's content which can be shared with other future participants.

Regarding the issues mentioned above we can see three key requirements which need to be provided with SwanLink:

1. Ability to exchange and share graphical context regardless the participants which are currently in session.
2. Ability to background maintain graphical content.
3. Ability to preserve common content until at least one participant is present.

Regarding the issues mentioned above the most important part of system is network tier which can support those features. Network tier needs to provide easy reconfigurable community with ability to safely handle leaving from community and acceptance of new participants without possibility to whole community departure. Also network tier needs to provide fast message passing. In following section issues regarding the sufficient architecture of network tier are discussed.

4.2. Proposed Solution for Concrete Situation

Even we have selected P2P architecture as the best one for our solution; our job was still not done. Between various P2P implementation which are currently implemented we needed to find one that will completely satisfy our requests.

One of the frequently implemented solutions for P2P communities is all-to-all architecture. In all-to-all architecture each participant is directly connected with each other as shown in Figure 2.

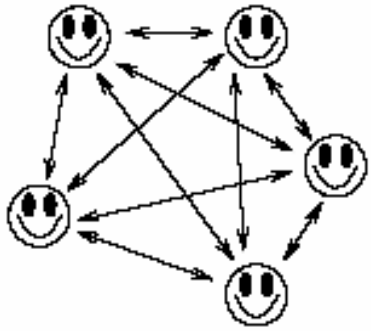


Figure 2: Peer to peer network implemented as all to all community

Advantage of this solution lies in the fact that all messages are immediately delivered to all other participants, so the latency is minimized. Disadvantage of all-to-all solution is that it is very hard to maintain those kinds of communities. Reason for this is the fact that newcomer, which arrives in community, needs to find ALL existing participants so he can join the group. Another problem is frequently connection changing that needs to be established with other participants. This solution is satisfying for small communities where we have not frequently architecture change.

Another extreme approach is daisy-chain implementation showed in Figure 3. In daisy-chain implementation number of connection which is established between participants is minimized, prejudice message delivery time.

In daisy chain, message is sent from one participant to its nearest neighbor, and the message is forwarded to next level of peers. Obviously, problem in this kind of solution is unacceptable latency of message. Another reason why this architecture is not applied, is reliability of that kind of P2P community. Namely, if one of the participants fail down entire community is partitioned in two groups, without possibility to heal them self.

Solution which is probably the most acceptable for underlying communication infrastructure may make use of a multicast network protocol, or a graph of point to- point network protocols, or a combination of the two, making some kind of balanced spanning tree.

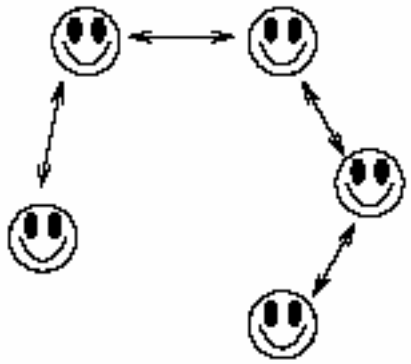


Figure 3: Peer to peer network implemented daisy chain community.

That kind solution provides logarithmic message delivery latency as number of participant grows. The infrastructure in commercial use today, in products such as IBM's Sametime, Data Connection's DC-Share, and Microsoft's NetMeeting, is the T.120 Internet standard.[swan_white_paper].

T.120 Internet Standard: When first connecting to a T.120 communication session on a given host computer, the local application spawns both a proxy process (the MCU) and a daemon process. The daemon process is a resident process which instantiates the MCU and listens for additional requests. The MCU forms a direct connection to the MCU of another host designated by the application user, or is designated as the root of the session. The requesting process and all additional processes on the host wishing to join the session, form a direct connection to the MCU process on that host. To share information, a process sends a message to its MCU, which is sent up the tree of MCUs to the root, then down the tree of MCUs and disseminated among their attached processes.

Although T.120 gives many benefits in distributed on-line collaboration, many disadvantages can be seen regarding its healing and reliability issues. The major problem in spanning tree implementation lies in the fact that administration network topology lies in end user responsibility. When one of the nodes in tree departure, it is responsible for interconnecting all his children to highest level so they will not be departure from parent nodes. In case of disgraceful node fall down, all child nodes will be lost.

Solution that provide logarithmic message latency and reliable infrastructure which is required in our project needs to be based on regular graph topology. Among various graph topology we have choose regular graph topology. Regular R-connected graph represents topology where each vertex is connected with fixed number of neighbors. For regular graph we say that is r-connecter, where r is the number of his neighbors. Example of 4-connected regular graph is showed on figure 4.

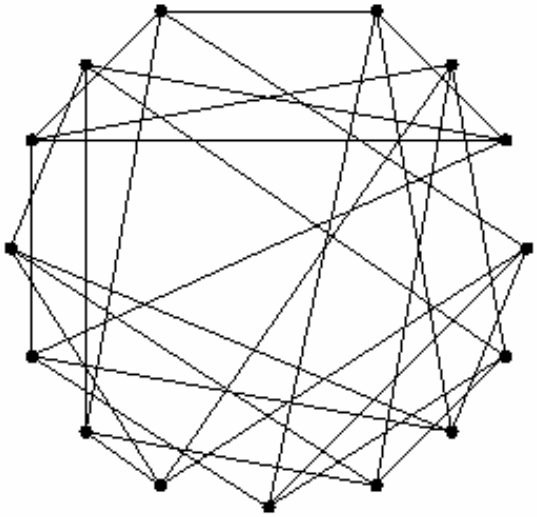


Figure 4: Regular graph. In current example each vertex is connected with exactly 4 nodes.

One of the biggest problems in distributed communities is preventing the cascade departures of nodes when the node that was represent the connection between two group of nodes in community (i.e. in tree topology when parent node fail all the children will be departures from community because the failing one was the only connection).

The key benefit of regular graph community, that exceeds this problem, lies in the fact that every node at each time is connected with exactly r-nodes which give whole community reliability against departures issue. In case that one of the node fails his neighbors will be still strongly connected with community via other three neighbors. Also in case of node failure four nodes which lost one neighbor (the failing one) have ability to interconnect mutually and again establish regular graph infrastructure. Hence, we can find remarkable self-healing opportunity in communities organized as regular graphs. In worst case scenario if four nodes fail which surrounds one neighbor simultaneously fail down as side effect we will have only one node departure from community. Regarding the low probability that the right combination of failing four nodes occurs, we can say that kind of collateral damages will be very rare.

5. *ANALYTICAL ANALYSIS*

Swan (Small-world Wide Area Networking) is a sophisticated implementation of a revolutionary theory of distributed logical networking developed by Virgil Bourassa and Fred B. Holt. Swan was developed to transparently and interactively

communicate, via ad hoc sessions, across intranets and the Internet between hundreds or even thousands of computers (as true peers) while maintaining minimal latency and high security. Features of Swan include very high reliability via adaptive self-healing, minimal latency via active self-optimization, high security via strong authentication and encryption, and virtually no administration or maintenance overhead.

We have used Swan technology because it is [4]:

1. Economical because distributed intelligence eliminates the need for expensive centralized servers to manage data or control flow.
2. Simple to use because it is transparent to every participant with virtually no administration required.
3. Easy to implement. Network communication is transparent to the application; the API is quick and simple to use.
4. Real-time responsive and eliminates the bottleneck of centralized servers and self-optimizes network data flow to achieve the most efficient organization available.
5. Reliable because it is self-healing and requires no hardware or infrastructure planning.
6. Secure because incorporates strong authentication and encryption.

This section is supposed to introduce the reader with the advantages of the above described network architecture.

First, here is the legend of this analysis:

n - number of nodes participating in the architecture,

p - probability that the connection between two nodes will fail (only in case of equal probabilities),

tindex - time needed for node index to respond,

tnodei – time needed for node i to respond.

The analysis is done in the following way. First we define three suitable network architectures which are basically different. Than we compare the main three network architectures with Swan considering time needed for communication, and than considering reliability, and then we try to use one more real model, which is based on the fact that the nodes are not equal considering speed and reliability. Finally, we compare the result to see what the basic differences between given architectures are.

1. Client-server architecture - in this architecture we consider that all of the user nodes are equal, and that they communicate with each other indirectly, by sending messages to the server.

2. Ring architecture - in this architecture we consider equal nodes that are forming a ring, which means that each node is connected only with his two neighbors. Communication is done by sending a message from the starting node to the next one, and so on, until it reaches everyone.

3. P2P all-to-all architecture - in this architecture we consider equal nodes that are forming network in which every node is connected to each other node.

4. Swan - in this architecture we consider equal nodes that form network where every node is connected to four other nodes randomly c

Time needed for finishing communication process:

Client server: n*(tnode1+tserver)

Ring: n * (tnode1+tnode2)

P2P: n * (tnode1+tnode2)

Swan - 4 * log n * (tnode1+tnode2)

What we can see is that advantage of Swan grows as the network grows. If the network consists of nodes that are not equal, it is obvious that all three architectures would have problem with waiting for it, and so much worse result considering time, while Swan wouldn't be affected, because the message will be passed by "the fastest node".

Reliability of the architecture:

Client server: if one node fails, only he will be disconnected, but if the server fails, all of the nodes would became disconnected. As the server is the most used, it would be normal to expect it first to fails. That leads to the fact that this architecture is not usable when we need system that "must not fail".

Ring: p*p*(n-1)*(n-3)+p*p*(1/n)*(1/n) - probability that one node will became disconnected from some other node is equal to probability that any two nodes fail, except if they are two neighbors plus the probability that it will fail itself.

P2P: If one node fails, only it is disconnected.

Swan: In theory, if four nodes connected to a node fail, that node would became disconnected (probability p^4). In practice, we have different situation: As mentioned above, when one connection breaks, nodes that were forming this connection immediately are starting to search for other nodes that have less than four connections. This way, even if more than half connection break, it would be possible to have all of the rest nodes connected to each other, having still four connections per node. As we can see, the Swan architecture is both the fastest in the set given above and the most reliable.

6. CONCLUSION

Having in mind everything mentioned in this paper, we can say with assurance that computer architecture defined here deserves to be one of the leading computer network architectures in the future. The main goal of obtaining both reliable and fast message passing led to success.

REFERENCES

[1] Milutinovic, V., "Some Solutions for Critical Problems in the Theory and Practice of Distributed Shared Memory,"

[2] http://www.computer.org/tab/tcca/news/sept96/sept96.htm

[3] Milutinovic, V., Stenstrom, P, "Improvements of Distributed Shared Memory: The Wholastic Approach," *Proceedings of the IEEE*.

[4] http://www.panthesis.com/features.php

New Modifications of Selection Operator in Genetic Algorithms for the Traveling Salesman Problem

Radovic, Marija; and Milutinovic, Veljko

Abstract—***One of the algorithms used for solving Traveling Salesman Problem is the genetic algorithm. It consists of three important parts: Selection, Crossover, and Mutation. In this paper some of the important concepts and methods of Selection are described. The paper is divided in two sections. In the first one, some of the most popular selection methods are described and in the second one, some new ideas about improving selection methods using the Internet knowledge are presented.***

Index Terms—***Genetic Algorithms, Selection, Traveling Salesman Problem, Semantic Web, Data mining***

1. Introduction

SELECTION is one of the main operators used in evolutionary computing. The primary objective of the selection operator is to emphasize better solutions in a population.

The identification of good or bad solutions in a population is usually accomplished according to a solution's fitness.

Selection can be used in different stages of evolutionary algorithms. Some algorithms (specifically genetic algorithms and genetic programming) usually apply the selection operator first to select good solutions and then apply the recombination and mutation operators on these good solutions, to create a hopefully better set of solutions. Other algorithms (evolution strategies and evolutionary programming) prefer using recombination and mutation operator first to create a set of solutions and then use the selection operator to choose a good set of solutions.

There are two purposes of this report. One is to give a systematic overview of existing approaches. And the other one is to introduce new approaches based on the usage of the Internet.

Manuscript received May, 2006. This work was supported in part by the Faculty of Uppsala, Sweden.

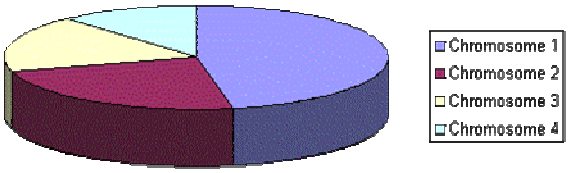
Radovic Marija is with the Faculty of Mathematics, University of Belgrade, Serbia.

Miltinovic Veljko is with the Faculty of Electrical Engineering, University of Belgrade, Serbia.

2. Existing Solutions and Their Criticism

2.1 Roulette Wheel Selection

Individuals are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where all the chromosomes in the population are placed. The size of the section in the roulette wheel is proportional to the value of the fitness function of every individual - the bigger the value is, the larger the section is.



A marble is thrown in the roulette wheel and the chromosome where it stops is selected. Clearly, the chromosomes with bigger fitness value will be selected more times.

Algorithm for roulette wheel selection reads:

Input: population a

Output: population after selection a'

```
begin
  s_o:=O
  /* forming the roulette wheel */
  for i:= 1 to n do
    s_i:=s_{i-1}+f_i/n
  end
   /* simulation of throwing the marble */
  for i:= 1 to n do
      r:= random([0, s_n])
      a_i ' = a_k  for such k to be accomplished s_{k-1} <r<s_k
   end
  return a'
end
```

Example:

6 random numbers: 0.81, 0.32, 0.96, 0.01, 0.65, 0.42.

Figure 1 shows the selection process of the individuals together with the sample trials.

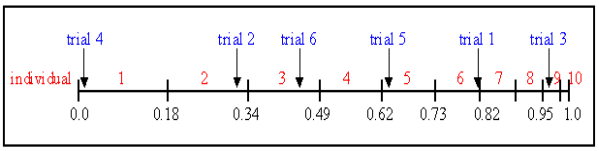


Figure 1: Roulette-wheel selection

After selection the mating population consists of the individuals: 1, 2, 3, 5, 6, 9.

2.2 Rank Selection

The previous type of selection will have problems when there are big differences between the fitness values. For example, if the best chromosome fitness is 90% of the sum of all fitness's then the other chromosomes will have very few chances to be selected.

Rank selection ranks the population first and then every chromosome receives fitness value determined by this ranking. The worst will have the fitness 1, the second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

You can see in following picture, how the situation changes after changing fitness to the numbers determined by the ranking.

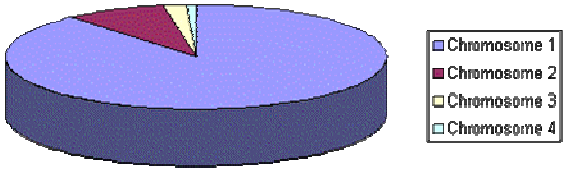


Figure 2: Situation before ranking (graph of fitness's)

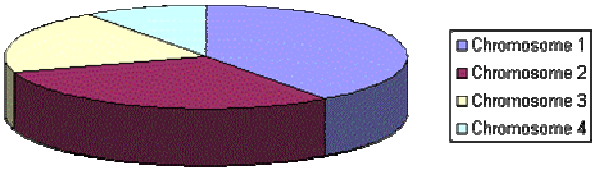


Figure 3: Situation after ranking (graph of order numbers)

Now all the chromosomes have a chance to be selected. However this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

There are two types of ranking selection – linear ranking and exponential ranking.

Linear ranking assigns a selection probability to each individual that is proportional to the individual's rank (where the rank of the least fit is defined to be zero and rank of the most fit is defined to be m-1, given a population of size m). This method has one parameter: the degree of reproduction of the least fit individual - r.

Algorithm for linear ranking:

Input: population a, the degree of reproduction of the least fit individual r in interval [0, l]

Output: population after selection a'

```
begjn
  a' :=  population a sorted in ascending order by fitness value
   /* forming a roulette wheel  */
   s_o:=O
   for i := 1 to n do
   s_i := s_i-1 + p_s(a_i)
   /* value p_s(a_i) is selection probability */
   end
   /* simulation of throwing the marble */
   for i: = 1 to n do
     r := random( [0,s_n])
     a_i ' := a_k'  for such k to be accomplished s_k-1<r<s_k
   end
  return a'
end
```

Exponential ranking is different from the linear ranking because an exponential function is used for determination of selection probability.

2.3 Stochastic Universal Sampling Selection

The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line as many as there are individuals to be selected. Consider NPointer the number of individuals to be selected, then the distance between the pointers are 1/NPointer and the position of the first pointer is given by a randomly generated number in the range [0, 1/NPointer].

For 6 individuals to be selected, the distance between the pointers is 1/6=0.167. Figure 4 shows the selection for the following example.

1 random number in the range [0, 0.167]: 0.1.

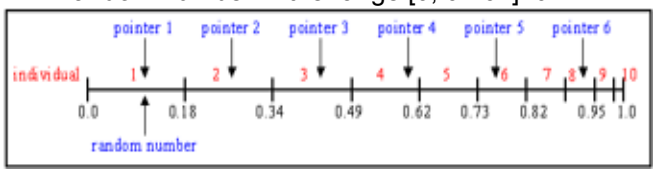


Figure 4: Stochastic universal sampling

After selection the mating population consists of the individuals: 1, 2, 3, 4, 6, 8.

Algorithm for stochastic universal sampling:

Input: population a, the degree of reproduction of the least fit individual R in interval [0,l]

Output: population after selection a'

```
begin
  sum := 0
   j := 1
  ptr := random([0,1])
  for i := 1 to n do
    sum := sum + R_i
    /* R_i - the degree of reproduction of the individual a_i */
    while (sum > ptr) do
      a_j':= a_i
     j:=j+1
     ptr: = ptr + 1
     end
   end
  return a'
end
```

For more information about the algorithm, see [3].

Stochastic universal sampling ensures a selection of offspring which is closer to what is deserved then roulette wheel selection.

2.4 Tournament Selection

Tournament selection is a rank-based method. The probability that an individual will be selected is based only on the rank of that individual in the population ordered by fitness, and not on the size of the fitness.

In tournament selection, element of population is chosen for passing into next generation if it is better (has better fitness value) than several randomly selected opponents. Tournament size N_{tour} is selection parameter, n is a size of the population and population after selection is a'. Running time for this algorithm is $O(n*N_{tour})$.

Algorithm for tournament selection:

Input: Population a (size of a is n), tournament size N_{tour},, $N_{tour} \in N$

Output: Population after selection a' (size of a' is n)

```
begin
  for i := 1 to n do
  ai' := best fitted item among Ntour elements randomly selected from population;
  return a';
end;
```

An advantage of tournament selection is that it is very easy to implement, and it works very well in a parallel implementation where different individuals are on different processors.

2.5 Fine Grained Tournament Selection

This is a variation of tournament selection. Instead of integer parameter N_{tour} (which represents tournament size), new operator allows real valued parameter F_{tour} – wanted average tournament size. This parameter governs selection procedure, so average tournament size in population should be as close as possible to it.

Algorithm for fine grained tournament selection:

Input: Population a (size of a is n), wished average tournament size F_{tour}, $F_{tour} \in R$

Output: Population after selection a' (size of a' is n)

```
begin
  Ftour- := trunc( Ftour )
  Ftour+ := trunc( Ftour ) + 1
  n- := trunc ( n * ( 1 - ( t - trunc ( Ftour ) ) ) )
  n+ := n - trunc ( n * ( 1 - ( t - trunc ( Ftour ) ) ) )
  /* tournaments with size Ftour- */
  for i := 1 to n- do
    ai' := best fitted item among Ftour- elements randomly selected from population;
    /* tournaments with size Ftour+ */
    for i := n-+1 to n do
      ai' := best fitted item among Ftour+ elements randomly selected from population;
    return a'
end;
```

For more information about this method, see [2].

2.6 Local Selection

In local selection every individual resides inside a constrained environment called the local neighborhood. (In the other selection methods the whole population or subpopulation is the selection pool or neighborhood.) Individuals interact only with individuals inside this region. The neighborhood is defined by the structure in which the population is distributed. The neighborhood can be seen as the group of potential mating partners.

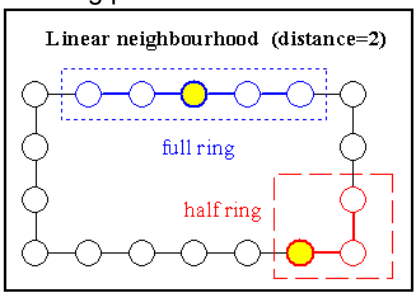


Figure 5: Linear neighborhood: full and half ring

The first step is the selection of the first half of the mating population uniform at random (or using one of the other mentioned selection algorithms, for example, stochastic universal sampling or truncation selection). Now a local neighborhood is defined for every selected individual. Inside this neighborhood the mating partner is selected (best, fitness proportional, or uniform at random).

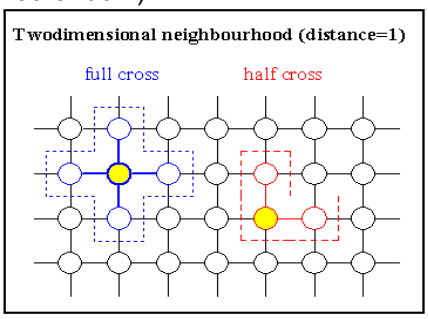


Figure 6: Two-dimensional neighborhood: full and half cross

The structure of the neighborhood can be:

- linear
- full ring, half ring (see figure 5)
- two-dimensional
- full cross, half cross (see figure 6)
- full star, half star (see figure 7)
- three-dimensional and more complex with any combination of the above structures.

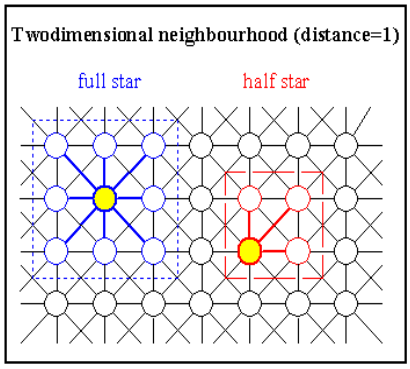


Figure 7: Two-dimensional neighborhood: full and half star

The distance between possible neighbors together with the structure determines the size of the neighborhood. Table 1 gives examples for the size of the neighborhood for the given structures and different distance values.

Between individuals of a population' isolation by distance' exists. The smaller the neighborhood, the bigger the isolation distances. However, because of overlapping neighborhoods, propagation of new variants takes place. This assures the exchange of information between all individuals.

structure	distance	
	1	2
full ring	2	4
half ring	1	2
full cross	4	8 (12)
half cross	2	4 (5)
full star	8	24
half star	3	8

Table 1: Number of neighbors for local selection

The size of the neighborhood determines the speed of propagation of information between the individuals of a population, thus deciding between rapid propagation or maintenance of a high diversity/variability in the population. A higher variability is often desired, thus preventing problems such as premature convergence to a local minimum. Local selection in a small neighborhood performed better than local selection in a bigger neighborhood. Nevertheless, the interconnection of the whole population must still be provided.

Two-dimensional neighborhood with structure half star using a distance of 1 is recommended for local selection. However, if the population is bigger (>100 individuals) a greater distance and/or another two-dimensional neighborhood should be used.

Algorithm for local selection:
Input: Population a
Output: Population after selection a'

```
begin
  a':= population after local selection
  return a';
end;
```

Before we can use data mining models and algorithms we have to find the most suitable strategy. In order to do that, we have to detect the problem type. Usually, data mining project involves a combination of different problem types, which together solve the problem [1].

At the lower end of the scale of the data mining problems is 'data description' and 'summarization'. It aims at the concise description of characteristics of the data, typically in elementary and aggregated form. This gives us an overview of the structure of the data.

The next data mining problem type is problem of 'segmentation'. It aims at the separation of data into interesting and meaningful subgroups or classes. All member of a subgroup contain common characteristics.

Misunderstanding of term segmentation is caused by it's relation with 'classification', which is another data mining problem type. Classification assumes that there is a set of objects that belong to different classes, where some attributes or features characterize each class. The objective is to build classification model which assign correct class label to previously unseen and unlabeled objects (so called predictive modeling).

Classification and segmentation may introduce new type of data mining problems; it is a 'concept description problem'. It aims at an understandable description of concepts or classes. The purpose is not to develop complete model with high prediction accuracy, but to gain insights.

Another important problem type that occurs in a wide range of application is 'prediction'. The aim of prediction is to find the numerical value of the target attribute for unseen objects.

In close connection to the prediction is another problem type, so called 'dependency analyses'. It consists of finding a model that describes significant dependencies and associations between data items or events.

3. PROPOSED SOLUTION

There are two types of information that need to be acquired from the Internet.

The first type is information concerning the future. In the process of selection, one more parameter must be acknowledged – what is the weather forecast for the route of the boat (truck) at the moment of his transition. If the weather forecast is bad (rain, storms) than the chance for the boat (truck) to go that way should be very small.

The second type is the past – in what condition is the boat (truck) that is being used. In case of an old boat (truck) chosen routes must be safer (deeper sea for boats or new roads for trucks) than the routes that would be chosen for a new boat (truck). There are some other things that could be considered for the past, such as status of the companies or the maintenance of the boats (trucks).

3.1 Internet Improved Roulette Wheel Selection

First ranking is done using the knowledge from the Internet. After this ranking, we can start forming the roulette wheel and perform the classical method of roulette wheel selection.

Improved algorithm:

Input: Population a

Output: population after selection a'

```
begin
  s_o:=O
   a:=population a after ranking using the knowledge from the Internet
   /* forming the roulette wheel */
   for i:= 1 to n do
     s_i:=s_i-1+f_i/n
   end
    I* simulation of throwing the marble */
   for i:= 1 to n do
       r:= random([0, s_n])
       a_i ' = a_k  for such k to be accomplished s_k - 1 <r<s_k
   end
  return a'
end
```

3.2 Internet Improved Rank Selection

Since rank selection is in many ways similar to roulette wheel selection, for improvement of this method we also propose that the knowledge from the Internet is used to rank the individuals before performing the classical rank selection algorithm.

Improved algorithm:

Input: population a, the degree of reproduction of the least fit individual r in interval [0,l]

Output: population after selection a"

```
begin
    a:= population a ranked using the knowledge from the Internet
    a' :=  population a sorted in ascending order by  fitness value
    /* forming a roulette wheel  */
    s_o:=O
    for i := 1 to n do
    s_i := s_i-1 + p_s(a_i')  /* value p_s(a_i') is selection probability */
    end
    /* simulation of throwing the marble */
    for i: = 1 to n do
      r := random( [0,s_n])
     a_i ":= a_k '  for such k to be accomplished s_k - 1<r<s_k
    end
   return a"
end
```

3.3 Internet Improved Stochastic Universal Sampling Selection

This method is also similar to roulette wheel so the individuals will be ranked and sorted with the usage of the Internet first and than the classical algorithm will be performed on that new population.

Improved algorithm:

Input: population a, the degree of reproduction of the least fit individual R in interval [0, l]

Output: population after selection a'

```
begin
   a:= population a after being ranked with the knowledge from the Internet
   sum := 0
   j := 1
   ptr := random([0,1])
   for i := 1 to n do
     sum := sum + R_i
    /* R_i - the degree of reproduction of the individual a_i */
    while (sum > ptr) do
      a_j':= a_i
     j:=j+1
    ptr: = ptr + 1
    end
  end
  return a'
end
```

3.4 Internet Improved Tournament Selection

This approach enables Internet to be utilized during the process of selecting the individuals.

In every step, after selecting N_{tour} random individuals we can sort them by using the Internet knowledge and then choose the winner among these individuals.

Improved algorithm:

Input: Population a (size of a is n), tournament size N_{tour},, $N_{tour} \in N$

Output: Population after selection a' (size of a' is n)

```
begin
  for i := 1 to n do
    randomly choose N_tour individuals from a population a
    sort these individuals using the Internet knowledge
    a_i':=best fitted individual among N_tour sorted elements
  end
  return a'
end
```

This can also apply to fine grained tournament selection:

Input: Population a (size of a is n), wished average tournament size F_{tour}, $F_{tour} \in R$

Output: Population after selection a' (size of a' is n)

```
begin
  F_tour^- := trunc( F_tour )
  F_tour^+ := trunc( F_tour ) + 1
  n^- := trunc ( n * ( 1 - ( t - trunc ( F_tour ) ) ) )
  n^+ := n - trunc ( n * ( 1 - ( t - trunc ( F_tour ) ) ) )
```

/* tournaments with size F_{tour}^- */
for i := 1 to n^- do
randomly choose F_{tour}^- individuals from population a
sort these individuals using the Internet knowledge
a_i' := best fitted item among F_{tour}^- sorted elements
end
/* tournaments with size F_{tour}^+ */
for i := n^-+1 to n do
randomly choose F_{tour}^+ individuals from population a
sort these individuals using the Internet knowledge
a_i' := best fitted item among F_{tour}^+ sorted elements
end
return a'
end;

3.5 Other Improvements

We described how Internet knowledge can be used before (roulette wheel, rank, stochastic universal sampling) and during (tournament and fine grained tournament) the classical algorithm. In some cases it can be used after the selection algorithm. That can be done using the fitness function that would contain parameters about future and past knowledge (weather forecast, conditions of the boats or trucks, status of the companies, the history of some routes, the maintenance of the tankers, etc.). But how to define that fitness function is yet to be discovered.

4. Problems Concerning the Internet

To gather knowledge from the Internet is another problem we came upon. Information about weather forecast or the condition of the boats (trucks) is not necessarily formatted as text. It can be formatted as pictures or some applications. These kinds of information can not be used in genetic algorithms.

What we can use in genetic algorithms are just numbers that represent parameters that depend on this information. So, we need to have optimal association of numerical values to different semantic entities.

First, we need to have all the information in a form of text. Then we could associate a certain value to every word (storm, rain, snow, old boat, etc.).

But, like we mentioned above, to find these information on the Internet in a form of text is not so common.

The best ways of gathering information would be Semantic Web and Data Mining.

Semantic Web is a concept that enables better machine processing of information on the Web, by structuring documents written for the Web in such a way that they become understandable by machines. More about Semantic web can be found in [8].

Data Mining can be defined as an automated extraction of predictive information from different data sources. It is a powerful technology with great potential to help users focus on the most important information. More information about data mining can be found in [9].

5. Open Problems For Research

How to assign numerical values in interval [0.1] based on the Internet knowledge that is non-numeric, but symbolic or semantic?

What should be considered to create a fitness function for the past knowledge (status of the companies, conditions of the tankers, the history of some routes, the maintenance of the tankers, etc.)?

What should be considered to create a fitness function for the future knowledge (weather forecast, etc.)?

How should we define the fitness function? Some examples are:

Rnew = Rold * Kp * Kf (Rold is some known fitness function, for example Jaccard's Score, Kp is a parameter that depends of the past, Kf is the parameter that depends on the future).

Rnew = Rold * F(Kp) * F(Kf) (F(Kp) and F(Kf) are some predefined functions of the arguments Kp and Kf).

Rnew = Rold o1 F(Kp) o2 F(Kf) (o1 and o2 are predefined operators)

6. Conclusion

We need to find a way to make this information from the Internet understandable to the algorithm we are using (definition of the fitness function and numerical values for the Internet knowledge). This is still a part of our research.

References

[1] Gremlich, R., Hamfelt, A., Valkovsky, V., "Prediction of the optimal decision distribution for the traveling salesman problem,"

[2] Filipovic, V., Kratica, J., Tošić, D., Ljubić, I., "Fine Grained Tournament Selection for the Simple Plant Location Problem," *Proceedings of the 5th Online World Conference on Soft Computing in Industrial Applications WSC5*, pp. 152-158, September 2000.

[3] Filipovic, V., "Predlog poboljsanja operatora turnirske selekcije kod genetskih algoritama,"

[4] Wright, A., "Evolutionary computation, selection methods,"

[5] Sushil, J., L., "Modified GAs for TSPs,"

[6] Obitko, M., "Introduction to Genetic Algorithms,"

[7] Pohlheim, H., "Evolutionary Algorithms: Overview, Methods and Operators," Documentation for: Genetic and Evolutionary Algorithm Toolbox for use with Matlab (November 2001).

[8] Vujovic, Neuhold, E., Fankhauser, P., Nierderee, C., Milutinovic, V., "Semantic Web: A Brief Overview and IPSI Belgrade Research," *Annals Of Mathematics, Computing & Teleinformartics*, Vol 1, No 1, 2003, PP 65-70

[9] Radivojevic, Z., Cvetanovic, M., Milutinovic, V., "Data Mining: A Brief Overview and IPSI Belgrade Research," *Annals Of Mathematics, Computing & Teleinformatics*, Vol 1, No 1, 2003, PP 84-90

Designing of an XPath Engine for P2P XML Store

Jovic, Darko; and Milutinovic, Veljko

Abstract— ***In the introductory part, this paper defines the general environment for this research and defines the terms of interest. Then we define the research problem: How to perform distributed XPath query execution in a P2P environment. Existing solutions to the problem are briefly surveyed, and their drawbacks are underlined; each surveyed piece of research is analyzed according to the same template. Then, the essence of the proposed solution is presented: implementation of an XPath engine which works in a P2P environment and performs distributed query execution. The conclusion is from the point of view of performance/complexity ratio.***

1. Introduction

XPATH is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer [1][4]. XPath expressions are used to query XML data, express transformations and reference elements in remote documents.

XPath is specification language designed by the World Wide Web Consortium (W3C) [6]. It is an expression language for addressing parts of an XML document, used by various XML technologies such as XSL Transformation (XSLT) and XML Pointer Language (XPointer). As the name implies XPath uses path notation for navigating through the hierarchical structure of an XML document. Like Document Object Model (DOM), XPath views an XML document as a tree structure consisting of nodes of different type, each representing the entities of a document.

2. Problem Statement

Our problem was to design an XPath engine which performs distributed XPath query execution on a distributed data storage system. System stores XML documents in a P2P community (Figure 1) by spreading parts of the document among peers in the community. It is based on JxTa P2P framework, implementation of distributed hash tables and implementation of Document object model (DOM) level 2 interfaces [2][3].

Manuscript received January 30, 2006. This work was supported in part by the Fraunhofer IPSI.

Darko Jovic and Veljko Milutinovic are with the Faculty of Electrical Engineering, University of Belgrade.

We can use XPath engine which works with generic DOM and use it on top of P2P DOM layer. This way we get full XPath functionality, but query execution is performed locally. All required node sets are pulled from the community and nodes matched against query to get the result. Getting node sets from the community in case of large XML documents can cause increased communication among local peer and other peers which contain node sets for specific documents. High communication traffic can be avoided by distributing XPath query execution.

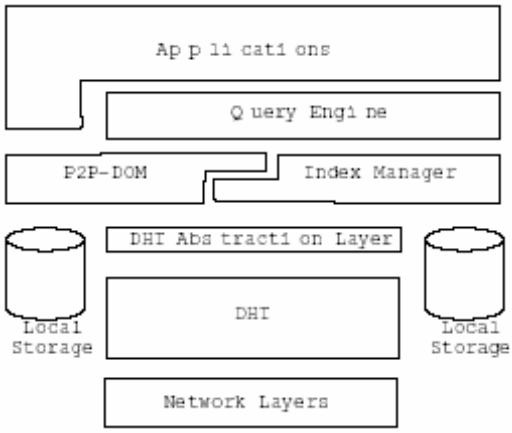


Figure 1: Peer-to-Peer XML Storage System

3. Existing Solutions

XML Store [7] is a distributed and scalable storage system for XML documents that builds on the Chord P2P protocol developed at MIT. It has adaptation of XPath which supports very reduced set of XPath 1.0 functionality. Only one axis is supported – *child* axis, and only one kind of predicate – *position()* operation. What makes a challenge is to build an XPath engine which works in P2P environment and supports full XPath 1.0 specification.

4. Proposed Solution

High communication traffic and memory consumption can be avoided by implementing an XPath engine, which performs distributed XPath query execution

Since there are several open source engines for Java, we decided not to write the XPath engine from scratch, but to modify the existing one which best fits our needs and which will take less time to adapt to work in a P2P environment using P2P DOM layer. Since P2P DOM layer implements DOM interfaces we should choose

engine which works with generic DOM interfaces, and not with specific implementations of DOM.

4.1 Existing XPath Engines

Most of these open source XPath engines are distributed as part of XSLT processor. We will give a short description of few of them with the focus on what kind of XML object model they support.

SAXON

SAXON package is a collection of tools for processing XML documents. It includes:

- XSLT processor, which implements the Version 1.0 XSLT and XPath Recommendations from the World Wide Web Consortium, with a number of powerful extensions.
- A Java library, which supports a similar processing model to XSL, but allows full programming capability, which you need if you want to perform complex processing of the data or to access external services such as a relational database
- A slightly improved version of the Ælfred parser from Microstar. (But you can use SAXON with any SAX-compliant XML parser if you prefer).

Saxon requires a custom DOM that has been annotated with the information it needs. You can't just pass in a Xerces DOM or a Crimson DOM, but you must use JAXP to build Saxon Document object with Saxon's implementation of DocumentBuilderFactory interface.

Xalan – J

The Xalan-J XSLT processor from the Apache XML Project also includes an XPath API that's useful for navigation in DOM programs. Xalan-Java (named after a rare musical instrument) fully implements XSL Transformations (XSLT) Version 1.0 and the XML Path Language (XPath) Version 1.0.

Xalan-J works with generic DOM implementations, but it has a little complicated architecture, so it is not ideal for modifications.

Jaxen

The Jaxen Java XPath Engine is an open source cross-API (DOM, JDOM, dom4j, and ElectricXML) XPath library for Java. Whereas DOM3 XPath [5] attempts to be a cross-implementation, cross-language XPath API for the Document Object Model alone, Jaxen attempts to be a cross-model API for its own XPath engine. It is a Java class library that can operate on various XML object models using a standard engine rather than an API that can be offered by many different engines for one model. It allows you to pass DOM, JDOM, dom4j, and ElectricXML objects directly to XPath functions like position() and translate(). Furthermore, whereas DOM 3 XPath offers fairly rudimentary interfaces for evaluating XPath expressions in a particular document against a context node, Jaxen is a more complete object model for XPath expressions.

Jaxen supports different XML object models through different implementations of org.jaxen.Navigator interface. By implementing this interface we can use Jaxen on any XML object model. But still it is not enough to adapt it to work in a P2P environment. Jaxen packages have clear hierarchy so it makes it less difficult to apply modifications to its classes. For this reasons we chose this XPath engine as a basis for our XPath engine which performs distributed XPath query processing.

4.2 Distributed Processing of XPath Queries

The primary syntactic construct in XPath is the location path, which is a sequence of location steps separated by a slash (/). Each step in turn selects a set of nodes relative to the context node and each node in that set is used as a context node for the following step. The syntax of a location step consists of an axis, a node test and zero or more predicates.

For example:

/child::weather/child::day[attribute::date='2001-12-12']/child::readings/child::reading

In a P2P XML storage XML document can be split into several node sets, which are stored on different peers. P2P DOM layer implements DOM interfaces so XPath engine can use these interfaces to evaluate expressions against XML documents, without needing to know underlying architecture of the P2P DOM layer. Node sets which contain nodes requested by the XPath engine will be pulled from the community to the local peer, so evaluation of the XPath expression can be performed. As we can see we can use any XPath engine which works with generic DOM level 2 interfaces to evaluate XPath expressions against some XML document stored in the P2P community. But this approach has certain drawbacks like memory consumption and time consumption. DOM representation of the XML document takes quite a memory, especially if the document is large. For every node in the document a corresponding object must be created in memory. Each object has links – keys to its owning document, parent node, child nodes, siblings, and also has namespace URI, name, etc. For evaluating XPath expression entire document tree needs to be in memory, which can sometimes lead to lack of memory problems if the document is extremely large. In order to evaluate XPath expression XPath engine must visit larger part of the document tree and get most of the node sets for the document from the community. Gathering these node sets from the community and creating DOM constructs takes some time, so the performance of evaluating XPath expression in this way would not be acceptable in everyday usage.

To overcome time and memory consumption during XPath query processing, we need to distribute the query processing so that peers who owns the node sets for one document take place in evaluation process. Each peer should evaluate part of the XPath query and return the result to the requesting peer, thus contributing in a chain of XPath query evaluation process.

XML Store [7] is a distributed and scalable storage system for XML documents that builds on the Chord P2P protocol developed at MIT. It has adaptation of XPath which supports very reduced set of XPath 1.0 functionality. Only one axis is supported – child axis, and only one kind of predicate – position() operation. What makes a challenge is to build an XPath engine which works in P2P environment and supports full XPath 1.0 specification.

4.2.1 XPath Query Evaluation and Migration

Query evaluation starts from the local peer by evaluating the first location step of the expression. The XPath engine iterates over nodes along the specified axis and when next node along the axis is not stored locally then query is migrated to the peer who owns the next node. When we say that the node is placed locally, it is assumed that the node is located in the persistent XML storage of the peer or in the cache buffer. Expression migration includes several steps:

- Extracting part of the XPath expression which is not processed and converting it to new expression suitable for migration. This part of the expression includes current location step which is not fully processed, and all following location steps in the expression.
- Determining next node along the axis which is not stored at the local peer. This node is new context node for the converted expression;
- Extracting current XPath context from the engine. Context includes namespace mappings and variables;
- Creation of message which contains above mentioned data: converted XPath expression, reference of the new context node and XPath context. This message is sent to the peer which owns the referenced node.

After message is sent to destination peer, current query processing is stopped until the result is received. Remote peer receives the message, recreates XPath context from the data contained in the message, evaluates expression against selected context node and returns the result. After result is received from the remote peer, nodes which are processed before migrating expression are joined to form the unique set of nodes, which form the result of evaluating the expression for the current context node. As we said before, nodes selected by one location step are context nodes for the following location step in the expression. During the expression evaluation we can get to some location step for which we can't perform full evaluation because not all nodes are stored locally. At this moment we have context nodes which are selected from the previous location step. We iterate through each context node and try to evaluate the result of the current location step. If migration is necessary then not only current location step is evaluated at remote peers, but the entire rest of the expression. Evaluating the rest of the location steps for current context node is finished and we can go to next context node.

Peer who receives the message for evaluating XPath expression, during evaluation process can migrate the evaluation process to other peers, thus forming the chain of XPath evaluation process. Each peer waits the result from the peer to which message was sent, and upon receiving an answer it forms the final result and sends it to the requesting peer.

4.2.2 XPath Expression Conversion

In order to successfully migrate the XPath expression conversion must be performed. The reason for this is that the context node must be changed during the migration. When expression needs to be migrated to another peer, current context node is the node which is stored at the local peer. This node does not need to be stored at the remote peer, and because of that it can not be context node for the expression which is sent to remote peer. New context node is the node which needs to be visited next and is located at the remote peer. Let's take for example child axis. If we have location step child::item and the current context node is Items node. If for example third and fourth node are located at different peer then the other item nodes, then when evaluation gets to the point where third item node needs to be visited expression must be migrated with the third item node as new context node. With that in mind we can not get appropriate results if we do not change axis. If we use the same expression *child::item* on the third item node we will not visit the fourth item node because it is the sibling of the third item node and not the child node. For that reason expression needs to be converted to *following-sibling::item*. Also we must include the new context node and thus get the final form of the expression: *self::item* | *following-sibling::item*, where '|' is union operator.

Here is the list of each axis and corresponding conversion (rest_of_expr represents current locations step without axis and following location steps in the expression):

- **Child** axis is converted to **self::*rest_of_expr* | following-sibling::*rest_of_expr***
- **Parent** axis – **self::*node_test***

• **Descendant** axis – **following-sibling::*rest_of_expr* | descendant-or-self::*rest_of_expr***

• **Descendant-or-self** axis - **following-sibling::*rest_of_expr* | descendant-or-self::*rest_of_expr***

• **Ancestor** axis – **ancestor-or-self::*rest_of_expr***

• **Ancestor-or-self** axis – **no conversion**

• **Following** axis – **self::*rest_of_expr* | following::*rest_of_expr***

• **Following-sibling** axis – **self::*rest_of_expr* | following-sibling::*rest_of_expr***

• **Preceding** axis - **self::*rest_of_expr* | preceding::*rest_of_expr***

• **Preceding-sibling** axis - **self::*rest_of_expr*** | **preceding-sibling::*rest_of_expr***

• **Self** axis – **no conversion**.

4.4 Modifications of P2P DOM Layer and Jaxen XPath Engine

In order to support distributed XPath query processing, in our system (Figure 1), P2P DOM layer needed to go through certain changes and modifications.

Link between P2P DOM layer and XPath engine needed to be created so that queries could be passed from P2P DOM layer to XPath engine and enable XPath engine to get necessary data from P2P DOM layer.

New type of message needed to be supported. This message is used to transfer necessary XPath data to remote peer. Also, response message needed to be expanded to support result of the XPath query evaluation.

Constructs for supporting XPath context– namespaces, variables and functions, so that context data can be sent to remote peer and there reconstructed so that query can be processed.

Jaxen was modified to enable communication with lower P2P DOM layer, and to migrate queries to remote peers. Algorithm of query evaluation was adapted to inspect if some node is stored locally, and if this is not the case to request migration of part of the XPath expression.

5. CONCLUSION

We showed that it is possible to adapt existing XPath engine to process XPath expression in a distributed manner in a P2P environment. Expression is evaluated locally until there is condition for that. The condition is that nodes are stored at the local peer, either in the cache buffer, or in the persistent PDOM storage. If this condition is not met, unevaluated part of the expression is migrated to remote peer, where its evaluation continues. There is one exception to this rule, that location step does not have a predicate which consist entirely of position operation. In this case the location step is evaluated locally.

The modified XPath engine was briefly tested with common XPath expressions.

REFERENCES

[1] Simpson, E., John, "XPath and XPointer," *O'Reilly*, August 2002

[2] Knežević, P., Risse, T., Neuhold, E., J., Fankhauser, P., "Designing a P2P XML Database – Applications and Open Issues"

[3] Jović, D., "P2P Store for Static P2P Communities, Sofware Architecture Document"

[4] "XML Path Language (XPath)," Version 1.0, *W3C Recommendation*, 16. Nov. 1999., http://www.w3.org/TR/1999/REC-xpath-19991116

[5] "Document Object Model (DOM) Level 3 XPath Specification," Version 1.0, *W3C Working Group Not*, 26. Feb. 2004., http://www.w3.org/TR/2004/NOTE-DOM-Level-3-XPath-20040226

[6] World Wide Web Consortium Web Site, http://www.w3.org

[7] Thorn, T., Baumann, A., Fennestad, M., "A distributed, value-oriented XML Store", Master's Thesis, *IT University of Copenhgen,* August 2002, http://www.itu.dk/xmlstore/XMLStore.pdf

Cumulative List of Reviewers by Countries

Australia
Abramov, Vyacheslav; Monash University
Banks, A., David; University of South Australia
Begg, Rezaul; Victoria University
Bem, Derek; University of Western Sydney
Betts, Christopher; Pegacat Computing Pty. Ltd.
Brown, Simon; University of Tasmania
Buyya, Rajkumar; The University of Melbourne
Campbell, A., Marilyn; University of Queensland
Chapman, Judith; Australian University Limited
Chen, Yi-Ping Phoebe; Deakin University
Currie, Carolyn V.; University of Technology Sydney
Hammond, Mark; Enfold Systems
Israel, Mark; Flinders University
O'Dwyer, Mark F.; University of Wollongong
Palmisano, Stephen; University of Wollongong
Pease, Wayne; University of Southern Queensland
Pereloma, Elena; Monash University
Pinto, Dale; Curtin University
Ristic, Branko; Science and Technology Organisation
Sajjanhar, Atul; Deakin University
Sidhu, Amandeep; University of Technology, Sydney
Soar, Jeffrey; University of Southern Queensland
Sudweeks, Fay; Murdoch University
Sweller, John; University of New South Wales
Unnithan, Chandana R.; Deakin University

Austria
Baumgartner, Robert; Vienna University of Technology
Derntl, Michael; University of Vienna
Geser, Guntram; Salzburg Research
Haas, Werner; Institute of Information Systems & Information Management
Hug, Theo; University of Innsbruck
Loidl, Susanne; Johannes Kepler University Linz
Mulrenin, Andrea M.; Salzburg Research
Polleres, Alex; University of Innsbruck
Stockinger, Heinz; University of Vienna
Sutter, Matthias; University of Innsbruck
Walko, Zoltan
Wöß, Wolfram; Johannes Kepler University Linz

Argentina
Olsina, Luis; National University of La Pampa
Rossi, Gustavo; UNLP

Belgium
Daele, Peter Van; IMEC-Ghent University
Huang, Ping; European Commission
Tianrui, Li; The Belgium Nuclear Research Centre

Brazil
Fernandes, Marcia Aparecida; University of Uberlandia
Traina, Agma; University of Sao Paulo
Traina, Caetano; University of Sao Paulo
Verdi, Fábio Luciano; School of Electrical and Computer Engineering
Vicari, Rosa; Federal University of Rio Grande
Wazlawick, Raul Sidnei; UFSC

Canada
Ahmed, Mohamed Ali; National Research Council
Bahli, Bouchaib; Concordia University
Bokhari, Asghar; McMaster University
En-Nouaary, Abdeslam; Concordia University
Feng, Franc; University of British Columbia
Fung, Benjamin; Simon Fraser University
Grayson, Paul; York University
Hansebout, Robert R.; McMaster University
Holroyd, Carin; Asia Pacific Foundation of Canada
Hung, Patrick C. K.; University of Ontario (UOIT)
Jirgens, Karl E.; University of Windsor
Jung, Benjamin; University of Victoria
Khedr, Mohamed Essam; University of Ottawa
L'Ecuyer, Pierre; Université de Montréal
McLeod, Bob
Memmi, Daniel; UQAM
Neti, Sangeeta; University of Victoria
Palmer, Robert J.; Bishop's University
Paulin, Michele; Concordia University
Peters, Dennis K.; University of Newfoundland
Plaisent, Michel; University of Quebec
Reid, Keith; Ontario Ministry of Agriculture
Rogers, Todd
Tremblay, Diane-Gabrielle; University of Quebec
Vaillancourt, Remi; University of Ottawa
Vandenberghe, Christian; HEC Montreal
Wesolowsky, George O.; McMaster University
Wolfe, Dalton L.; Lawson Health Research Institute

Chile
Bedini, Alejandro; Universidad Federico S. Maria

China
Cao, Bing-Yang; Tsinghua University
Chung, Walter W.C; Hong Kong Polytechnic University
Guo, Lei; University of Electronic Science
Guo, Zeng-Yuan; Tsinghua University
Hu, Chuan-Gan; Nankai University
Lam, Ho Cheong; The University of Hong Kong
Lau, Henry C. W.; Hong Kong Polytechnic University
Lee, Raymond;Hong Kong Polytechnic University
Li, Lemin; University of Electronic Science
Li, Zhiwu; Xidian University
Luo, Yingwei; Peking University
Sesay, Samba; Huazhong University
Snell, Robin Stanley; Lingnan University
Tao, Yufei; City University of Hong Kong
Tsang, Albert H.C.; Hong Kong Polytechnic University
Yin, TANG Chak; Hong Kong Polytechnic University
Zhou, Haofeng; Fudan University
Zhuoqun, Xu; Peking University

Colombia
Restrepo Rivas, Luis Guillermo; Universidad Pontificia Bolivariana

Croatia
Jovic, Franjo; University of J.J. Strossmayer

Cyprus
Kyriacou, Efthyvoulos; University of Cyprus

Czech Republic
Brandejsky, Tomas; Czech Technical University
Kaluza, Jindrich; Technical University of Ostrava
Korab, Vojtech; Brno University of Technology
Lhotska, Lenka; Czech Technical University
Merunka, Vojtech; Czech University of Agriculture

Denmark
Bang, Joergen; Aarhus University
Edwards, Kasper; Technical University Denmark
Melody, William H.; Technical University of Denmark
Orngreen, Rikke; Copenhagen Business School

Estonia
Kull, Katrin; Tallinn University of Technology
Reintam, Endla; Estonian Agricultural University

Finland
Anttiroiko, Ari-Veikko; University of Tampere
Heimburger, Anneli; Tampere University
Kajala, Jyrki; Savonia Polytechnic
Kajava, Jorma; University of Oulu
Laihonen, Tero; University of Turku

Mattila, Minna Marita; Tampere Technical University
Rautava, Jouko; BOFIT/Bank of Finland
Repo, Petteri; National Consumer Research Centre
Reti, Tommo; Helsinki Institute
Sirkemaa, Seppo; Turku School of Economics
Tico, Marius; Nokia Research Center

France

Cardey, Sylviane; University of Franche-Comte
Klinger, Evelyne; LTCI - ENST, Paris
Lepage, Alain; University of Technology of Compiègne
Mickael, Gardoni; INPG / GILCO laboratory
Roche, Christophe; University of Savoie
Rossi, Fabrice; INRIA Rocquencourt
Valette, Robert; LAAS-CNRS

Germany

Baum, Dieter; University of Trier
Baumeister, Hans-Peter; Reutlingen University
Beer, Doris; GEMI-Beer GbR
Brunkhorst, Hauke; Flensburg University
Buss, Claudia; University of Trier
Fetzer, Anita; University of Lueneburg
Hirzinger, Gerd; Institute of Robotics and Mechatronics
Jungmayr, Stefan; Teradyne Diagnostic Solutions
Kaib, Michael; Booz Allen Hamilton
Köhler, Stefan; Infosim GmbH & Co. KG
Korf, Roman; University of Potsdam
Lengauer, Christian; University of Passau
Morschett, Dirk; Saarland University
Murschetz, Paul; University of Cologne
Nicolescu, Valentin; Technische Universitat Munchen
Nordmann, Alfred; Darmstadt Technical University
Ragia, Lemonia; Technische Hochschule Aachen
Rennings, Klaus; ZEW Mannheim
Schmidt, Carsten; University of Mannheim
Steffen, Peter; University of Bielefeld
Stepping, Michael; Avinotec GmbH
Tschofenig, Hannes; SIEMENS Corporate Technology
Unland, Rainer; University of Duisburg-Essen
Weiß, Christian; University of Würzburg
Wilhelmy, Jeannette; University of Cologne
Zentel, Peter;Knowledge Media Research Center

Greece

Kosmidis, Mary H.; Aristotle University of Thessaloniki
Moulianitis, Vassilis C.; University of Patras

Hungary

Jakabfi, Anna; Eötvös Loránd University
Lorincz, Andras; Eötvös Loránd University
Nagy, Zoltan; Miklos Zrinyi National Defense University
Szanto, Richard; Corvinus University of Budapest

India

Chatterjee, Arnab; Saha Institute of Nuclear Physics
Maheshwari, Neeru; Birla Insititute of Technology
Pareek, Deepak; Technology4Development
Roy, Debanik; Bhabha Atomic Research Centre
Scaria, Vinod; Institute of Integrative Biology
Shah, Mugdha; Mansukhlal Svayam
Sreeram, Pazynur Agneswaran; Bose National Centre

Ireland

Eisenberg, Jacob; University College Dublin

Israel

Feintuch, Uri; Hadassah-Hebrew University
Fund, Zvia; Bar-Ilan University
Haberman, Bruria; Holon Institute of Technology
Koren, Shira; Bar Ilan University
Levine, Adina; Bar-Ilan University
Satran, Julian; IBM Research Laboratory
Shani, Gad; Ben Gurion University

Italy

Aiello, Marco; University of Trento
Appice, Annalisa; University of Bari
Ardissono, Liliana; University of Torino
Bettini, Claudio; University of Milan
Chella, Antonio; University of Palermo
Chella, Antonio; University of Palermo
Dagnino, Giovanni Battista; University of Catania
De Marsico, Maria; University of Rome
Diamantini, Claudia; Polytechnic University of Marche
Ferraresi, Pier Marco; CERP
Fogli, Daniela; University of Brescia
Frankova, Ganna; University of Trento
Furcolo, Pierluigi; University of Salerno
Gherardi, Silvia; University of Trento
Jacobsen, Eric; Università degli Studi di Bologna
Kenett, Ron S.; University of Torino
Lazovik, Alexander; University of Trento
Mello, Paola; University of Bologna
Papini, Pier Luigi; Università di Bologna
Pierattoni, Davide; University of Udine
Porretta, Sebastiano; Food Preservino Industry
Robustelli, Anton Luca; University of Salerno
Salomoni, Paola; Universita' di Bologna
Santucci, Fortunato; Università de L'Aquila
Sirocchi, Fabio; CINECA
Vantaggi, Barbara; Universita' La Sapienza, Roma

Japan

De Marco, Giuseppe; Fukuoka Institute of Technology
Fujimoto, Isao; Toyota Technological Institute
Inoue, Katsuro; Osaka University
Matsuo, Hiroshi; Nagoya Institute of Technology
Okamoto,Toshio;University of Electro-Communications
Robb, Thomas N.; Kyoto Sangyo University
Sasakura, Mariko; Okayama University
Tada, Yasunori; Osaka University
Yamasaki, Susumu; Okayama University
Yokomori, Reishi; Nanzan University

Jordan

Salhieh, Sa'Ed M.; The University of Jordan

Kuwait

Safar, Maytham; Kuwait University

Lithuania

Burinskiene, Marija; Technical University

Malaysia

Lee, Goi Chai; Curtin University of Technology
Yeap, Boon Han; Monash University

Mauricius

Santally, Mohammad; University of Mauritius

México

Domínguez, Humberto Ochoa; Universidad Autónoma
Morado, Raymundo; University of Mexico
Mortera-Gutierrez, Fernando; ITESM-CCM

Netherlands

Bregt, Arnold; Wageningen University
Cowan, Robin; University of Maastricht
Eldin, Amr Ali; Delft University of Technology
Kaymak, Uzay; Erasmus University Rotterdam
Kenny, Steve; Ponemon Institute Europe
Kleinjung, Jens; Vrije Universiteit Amsterdam
Leemans, Rik; Wageningen University and Research
Linnartz, Jean-Paul; Philips Research
Soest, Arthur; RAND and Tilburg University
Vermetten, Eric; University Medical Center

New Zealand

Elley, Raina; University of Otago
Huff, Sid; Victoria University of Wellington
Scornavacca, Eusebio;Victoria University of Wellington

Norway

Hoogensen, Gunhild; University of Tromsø
Skogseid, Ingjerd; Western Norway Research Institute

Pakistan
Khan Sial, Sultan Muhammad; Lahore University
Philippines
Castolo, Carmencita; University of Philippines
Poland
Butkiewicz, Bohdan; Warsaw University
Lenko-Szymanska, Agnieszka; Warsaw University
Pakienkiewicz, Julek; Warsaw University
Pakszys, Elzbieta; Adam Mickiewicz University
Portugal
Silva, Miguel Mira; Instituto Superior Tecnico
Qatar
Kassim, Norizan Mohd; University of Qatar
Romania
Mitrovic, Slobodan; Otorhinolaryngology Clinic
Moga, Liliana; "Dunarea de Jos" University of Serbia
Russia
Oleg, Skljarov; Saint-Petersburg Research Institute
Singapore
Bressan, Stephane; National University of Singapore
Li, Yingjiu;Singapore Management University
Yau, Wei-Yun; Institute for Infocomm Research
South Africa
Blignaut, Pieter; University of the Free State
South Korea
Kim, Cheol-Han; Daejon University
Kwon, Wook Hyun; Seoul National University
Spain
Ceneteno, Clara; Institute for Prospective Technological Studies
Guimaraes, Rafael; Polytechnic University of Catalonia
Maghiros, Ioannis; Institution European Commission
Pablos Heredero, Carmen; Rey Juan Carlos University
Pazos, Juan; Technical University of Madrid
Quero, Soledad; Jaume I University, Castellon
Torrens, Francisco; Universitat de Valencia
Sweden
Barse, Emilie Lundin; AerotechTelub AB
Stenfors-Hayes, Terese; Karolinska Institutet
Switzerland
Bonfadelli, Heinz; University of Zurich
Camponovo, Giovanni; University of Lausanne,
Dogan, Turgut; University of St. Gallen
Fischbach, Nicolas; COLT Telecom
Fischbacher, Urs; University of Zurich
Goyette, Stéphane; University of Fribourg
Iseli, Christian; Ludwig Institute for Cancer Research
Meier, Stephan; University of Zurich
Mutti, Carlo; Datamars SA
Oppliger, Rolf; eSECURITY Technologies
Papadimitratos, Panagiotis; EPFL
Spaccapietra, Stefano; EPFL
Studer, Thomas; University of Bern
Zielinski, Chris; World Health Organization
Taiwan
Chang, Yang-Lang; National Taipei University
Chen, Hengsheng; Chinese Culture University
Hsieh, Ching-Tang; Tamkang University
Shyu, Yuh-Huei; Tamkang University
Sue, Chuan-Ching; National Cheng Kung University
Turkey
Cetin, Enis; Bilkent University
Eksioglu, Kamil; Murat Oksijen
Gucuyener, Ismet; Uludag University
United Kingdom
Ali, Maaruf; Oxford Brookes University
Ashton, Philippa; Staffordshire University
Attwood, Greg; Repton School
Bailey, Stephen J.; Glasgow Caledonian University
Barnes, Stuart J.; University of East Anglia
Bentley, Peter J.; University College London
Brewster, Stephen; University of Glasgow
Coleman, Shirley; University of Newcastle
Conole, Grainne; University of Southampton
Coolen, Frank; Durham University
Coolen-Schrijner, Pauline; Durham University
Earl, Michael J.; University of Oxford
Emes, Michael; Mullard Space Science Laboratory
Engelmann, Dirk; University of London
Eze, Emmanuel; Univeristy of Hull
Gill, John; Royal National Institute of the Blind
Gold, Jeff; Leeds Business School
Graham, Margaret; Northumbria University
Hunston, Susan; University of Birmingham
Jiang, Jianmin; University of Bradford
Johnson, Colin; University of Kent at Canterbury
Jones, Matt; Swansea University
Jones, Paul; University of Glamorgan
Kelbert, Mark; University of Wales-Swansea
Langdon, W. B.; University College
Leka, Stavroula; University of Nottingham
MacDonald, Theodore; London University
MacDonald, Theodore; London University
Markert, Katja; University of Leeds
Marturano, Antonio; University of Exeter
McAndrew, Patrick; The Open University
Ng, Kia; ICSRiM - University of Leeds
Parker, David; Cranfield University
Pinter, Ranko; The TETRA Association
Rees, Philip Howell; University of Leeds
Robins, C., Paul; Aston University Business School
Saggion, Horacio; University of Sheffield
Sharp, Helen; The Open University
Small, Adrian; University of Northumbria
Smith, Peter; University of Sunderland
Soltani, Ebrahim; University of Kent
Stokes, Christopher; University of Sheffield
Teubert, Wolfgang; University of Birmingham
Thompson, Diana; University of Wolverhampton
Turner, Mark; Keele University
Walker, Stuart D.; University of Essex
Westland, Stephen; University of Leeds
Whyte, William Stewart; University of Leeds
Wilson, Andrew; Lancaster University
Wing, Alan; University of Birmingham
Wright, Patricia; Cardiff University
Xie, Changwen; Wicks and Wilson Limited
USA
Adelman, Leonard; George Mason University
Akerson, Valarie; Indiana University
Allen, Peter; Rhode Island College
Ammer, Josephine; University of California, Berkeley
Ashford, Nicholas; Massachusetts Institute
Assavapokee, Tiravat; University of Houston
Auluck, Nitin; Quincy University
Backer, Larry Catá; Pennsylvania State University
Baum, Christopher; Boston College
Beaton, Catherine; Rochester Institute of Technology
Beck, Kent; Three Rivers Institute
Bellows, Peter; Rockwell Collins, Inc.
Benotsch, Eric G.; University of Colorado
Bernstein, Ira; The University of Texas at Arlington
Bhatti, Rafae; Purdue University
Bloomfield, Peter;North Carolina State University
Bowen, Roger; Association of Professors
Boyce, Jeanann;
Briggs, Alice; Franklin Pierce Law Center, Concord
Brill, Michael; Datacolor
Bryant, Carol; University of South Florida

Cassady, Jerrell; Ball State University
Champion, Jane Dimmitt; The University of Texas
Chang, Jeffrey; Duke University
Chang, T., Jeffrey Duke University
Chen, Shu-Ching; Florida International University
Chester, Daniel Leon; University of Delaware
Chircu, Alina; University of Texas at Austin
Choma, John; University of Southern California
Christie, William G.; Vanderbilt University
Chun, Mark W.S.; Pepperdine University
Chun, Soon Ae; City University of New York
Coffin, Zachary; Coffin Capital
Cohen, Sol; UCLA
Coladarci, Theodore; University of Maine
Cole, David N.; Leopold Wilderness Research Institute
Conn, Cynthia Anne; Northern Arizona University
Coppola, Jean F.; Pace University
Dai, Qizhi; Drexel University
Daniels, Peter
Deng, Yuanjian; Texas Southern University
Derzon, Jim; Econometrica Inc.
Derzon, Jim; Pacific Institute for Research
Dewey, Dan; University of Pittsburgh
Dholakia, Nikhilesh; University of Rhode Island
Didier, Crestani; L.I.R.M.M.University
Dinev, Tamara; Florida Atlantic University
Dunn, Mary; University of Texas
Dunn, Phillip C.; University of South Carolina
Embley, David W.; Brigham Young University
Erdogan, Sevki; University of Hawaii
Eskicioglu, Ahmet; CUNY Brooklyn College
Foxman, Ellen; Bentley College, Waltham,
Freedman, Immanuel
Fries, P., Terrence; Coastal Carolina University
Fukuda, Munehiro; University of Washington
Gangopadhyay, Aryya; University of Maryland (UMBC)
Gefen, David; Drexel University
Gill, Sam; San Francisco State University
Giunipero, Larry C.; Florida State University
Glantz, Kalman
Goel, Sanjay; University of Minnesota Duluth
Goldstone, Jack A.; George Mason University
Gorski, Paul C.; Hamline University
Grant, Mary; University of Wisconsin-Extension
Grassian, Esther; UCLA College Library
Grossman, Jerrold; Oakland University
Guduru, Pradeep; Brown University
Haddadin, A., Fadi; The University of Chicago
Han, Jiawei; University of Illinois at Urbana-Champaign
Han, Ju; Lawrence Berkeley National Lab
Hansen, Bill
Harris, Harold; University of Missouri
Hasselmo, Michael Erik; Boston University
Hazeltine, Barrett; Brown University
Hong, Sangjin; Stony Brook University - SUNY
Hungwe, Kedmon; Michigan University
Hunt, Anthony; University of California
Illing, Lucas; Duke University
Kelley, Catherine; Fairleigh Dickinson University
Kilburn, Kaye; USC Keck School of Medicine
King, Kathleen; Idaho State University
Kitts, Christopher; Santa Clara University
Kumar, Vipin; University of Minnesota
Lambert, Linda; Taylor University
Lastowka, Greg; Rutgers University School of Law
Lazarevic, Aleksandar; United Tech. Research Center
Lee, Chin C.; University of California
Lee, James; The George Washington University
Lehman, Tobin; IBM Almaden Research Center
Lesser, Eric; IBM Corporation
Levy, John; John Levy Consulting
Lewis, A., Barbara; Lamar University
Li, Qi; Li Creative Technologies
Li, Tao; Azusa Pacific University
Light, Marc; University of Iowa
Lin, Zhibin; University of Virginia
Lindner, James R.; Texas A&M University
Littman, Michael; Rutgers University
Lopez, Elena; Harvard Family Research Project
Lorenzi, Nancy M.; Vanderbilt University
Loyall, Joseph P.; BBN Technologies
Luo, Xin; Mississippi State University
MacLachlan, Douglas L.; University of Washington
Maher, Julie; Oregon Department of Human Services
Malmendier, Ulrike; Stanford University
Martelli, Alex; Google, Inc
Matthews, Harry R.; University of California
Mazumdar, Subhasish; New Mexico Tech
McKenney, Paul E.; IBM Linux Technology Center
Mead, Nancy; Carnegie Mellon University
Mehlenbacher, Brad; NC State University
Melody, Alexander W.; Ball State University
Mertz, David; Gnosis Software, Inc.
Middleton, Blackford; Harvard Medical School
Mikitiuk, Artur; University of Texas at Tyler
Miller, Robin Lin; Michigan State University
Moore, Michael Grahame; Pennsylvania University
Morell, Jonathan A.; The Altarum Institute
Motter, Adilson E.; Northwestern University
Mueller, John Paul; DataCon Services
Nash, Claudia Peralta; Boise State University, Idaho
Neuberger, John W.; University of North Texas
Newman, Peter ; Colorado State University
Nichols, Randall; The George Washington University
Niemi, Richard; University of Rochester
Nolle, Daniel E.; Comptroller of the Currency
Norman, Kent; University of Maryland
North, Michael J.; Argonne National Laboratory
Ohn, Jonathan; Wagner College
O'Leary, Daniel; University of Southern California
Osterman, Karen F.; Hofstra University
Othayoth, Raghunath; Hewlett-Packard Company
Ott, Margaret G.; Tyler Junior College
Oxenham, William; North Carolina State University
Pankanti, Sharath; IBM T. J. Watson Research Center
Parashar, Manish; University of New Jersey
Patil, Sameer; University of California
Peha, Jon M.; Carnegie Mellon University
Peirce, William; Community College
Pendergrast, Mark; Independent Scholar
Penrod, James I.; University of Memphis
Picciano, Anthony; City University of New York
Pierce, Benjamin C.; University of Pennsylvania
Plass, Jan L.; New York University
Podichetty, Vinod; Cleveland Clinic Florida
Poltrock, Steven;The Boeing Company
Prabhakar, Salil; Digital Persona Inc.
Prato, Tony; University of Missouri-Columbia
Raghunathan, Anand; NEC Labs America
Raje, Rajeev; Indiana University
Reeve, Trevor; Federal Reserve System
Reisner, John; Air Force Institute of Technology
Reynard, Ruth; Trevecca Nazarene University
Rhodes, Philip; University of Mississippi
Richmond, Elizabeth; University of Wisconsin
Roiger, Richard J.; Minnesota State University,
Roozendaal, Benno; University of California
Rosen, Lauren; University of Wisconsin
Rossman, Mark; Capella University
Roth, Elliot; Rehabilitation Institute of Chicago

Rothfork, John; Northern Arizona University
Roundh, Shadrach; LV Sensors, Inc.
Russell, Robert; College of Charleston
Sabharwal, Ashutosh; Rice University
Salovey, Peter; Yale University
Sands, Daniel Z.;Harvard Medical School
Sapci, Hasan; University of Michigan Health System
Schott, Marshall; University of Houston
Schwabacher, Mark; NASA Ames Research Center
Sears, Andrew; UMBC
Serwatka, Judy Ann; Purdue University
Shah, Biren N.; University of Louisiana
Shamim, Tariq; University of Michigan-Dearborn
Shapiro, Linda;
Sharma, Gaurav; University of Rochester
Sheikholeslami, Gholamhosein; Cisco Systems
Sheldon, Frederick; Oak Ridge Laboratory
Shelley, Louise
Sheng, Yongwei; SUNY College
Shiavi, Richard; Vanderbilt University
Sichitiu, Mihail; NC State University
Silbey, Robyn; Montgomery Public Schools
Sittig, Dean F.; Kaiser Permanente, Northwest
Smith, Daniel T.; Purdue University
Sojda, Richard; Rocky Mountain Science Center
Spender, JC; Open University Business School
Srinath, Mandyam; Southern Methodist University
Stavrovski, Boris; City University of New York
Strayer, Timothy; BBN Technologies
Suzuki, Junichi; University of Massachusetts
Takayama, Leila; Stanford University
Tang, Yuchun; Georgia State University
Tarjan, Robert E.;Princeton University
Thompson, Dale R.; University of Arkansas
Thompson, Lori Foster; North Carolina University
Troutt, Marvin D.;Kent State University
Tsitsiklis, John; MIT
Turner, Douglas E.; State University of West Georgia
Uhlik, Felix; Georgia Institute of Technology
University of Maryland University College
Usher, John M.; Mississippi State University
Veanes, Margus;
Vucetic, Jelena; Alpha Mission, Inc.
Wade, Ted; National Jewish Medical Center
Wagstaff, Samuel S.; Purdue University
Wang, Ke; Columbia University
Wang, Li; University of Southern California
Wang, Xinwei; The University of Nebraska-Lincoln
Wang, Yujun; University of California
Wayman, James L.; San Jose State University
Weeks, Michael; Georgia State University
Whitworth, Brian; New Jersey Institute
Wiener, Josh; Oklahoma State University
Williams, Christine; Bentley College
Wright, Dianne; Florida Atlantic University
Xie, Qing; University of Maryland
Zhang, Yanqing; Georgia State University
Zhou, Qiru; Bell Labs

IPSI Team

Advisor for Mathematics-Related Problems:
Jelena Miletic and Maja Etinski
Advisors for IPSI Tutorials:
Tanja Kovacevic and Tanja Petrovic
Advisors for IPSI Research:
Milos Kovacevic and Milos Milovanovic
Advisors for IPSI Developments:
Zoran Babovic and Darko Jovic

Welcome to IPSI BgD Conferences and Journals!

http://www.internetconferences.net
http://www.internetjournals.net

IPSI-2006 SLOVENIA
Hotel Toplice,
Bled Slovenia
Nov 30 to Dec 03, 2006.

IPSI-2007 VENICE SPRING
Venice, Italy
March 19 to 22, 2007.

IPSI-2007 AMALFI
Hotel Santa Caterina
Amalfi, Italy
Finland
March 22 to 25, 2007.

IPSI-2007 LAPLAND
Rovaniemi,
Santa Claus World Capital,
Finland
March 29 to April 01, 2007.

IPSI-2007 TOKYO
M. I. T., Tokyo,
Japan
May 31 to June 03, 2007.

IPSI-2007 OPATIJA CROATIA
Opatija/Abazzia
Vila Ariston
June 07 to 10, 2007.

IPSI-2007 ROVINJ CROATIA
Rovinj/Rovigo
Vila Angelo d'Oro
June 10 to 13, 2007.

IPSI-2007 MONTREAL
Campus of UQAM,
Montreal, Quebec, Canada
Vila Angelo d'Oro
June 29 to July 02, 2007.

IPSI-2007 ITALY
Rome and
Hotel Castello Chiola,
Loreto Aprutino in Abruzzo,
Relatively near Rome, Italy
August 20 to 23. 2007.

IPSI-2007 FLORENCE
Grand Hotel & La Pace in
Montecatini Terme, near
Florence, Piza, Siena, and
Lucca, Italy
August 23 to 26. 2007.

IPSI-2007 MACEDONIA
Lake Ohrid, Macedonia
September 06 to 09, 2007.

IPSI-2007 BELGRADE
Belgrade, Serbia
September 20 to 23, 2007.

IPSI-2007 MONTENEGRO MOUNTAINS
Vila Bianca,
Kolasin, Montenegro
September 23 to 26, 2007.

IPSI-2007 MONTENEGRO
Hotel Sveti Stefan,
Montenegro
September 26 to 29, 2007.

IPSI-2007 VENICE FALL
Venice, Italy
October 11 to 14, 2007.

IPSI-2007 PORTOFINO
Grand Hotel Miramare,
Santa MargheritaLigure,
Italy (near Genova)
October 14 to 17, 2007.

IPSI-2007 SLOVENIA
Lake Bled in Alps,
Slovenia
Nov 29 to Dec 02, 2007.

The publication of this journal
is supported in part by
the Ministry of Science of Serbia

CIP – Katalogizacija u publikaciji Narodna biblioteka Srbije, Beograd

ISSN 1820 – 4503 =
The IPSI BGD Transactions on Internet Research
COBISS.SR - ID 119127052