Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

An Optimized Search Mechanism for Large Distributed Systems

Herwig Unger[1], Thomas Böhme[2], Markus Wulff[1], Gilbert Babin[3], and Peter Kropf[4]

[1] *Fachbereich Informatik*
Universität Rostock
D-18051 Rostock, Germany
{hunger,mwulff}@informatik.uni-rostock.de
[2] *Mathematisches Institut*
TU Ilmenau
D-98648 Ilmenau, Germany
tboehme@theoinf.tu-ilmenau.de
[3] *Technologies de l'information*
École des Hautes Études Commerciales
Montréal, Canada H3T 2A7
Gilbert.Babin@hec.ca
[4] *Département d'informatique et de recherche opérationnelle*
Université de Montréal
Montréal, Canada H3C 3J7
kropf@iro.umontreal.ca

Abstract. The large number of services and the huge amount of information available in today's large scale distributed systems such as the Internet, make efficient search mechanisms an unconditional issue. Many search methods and engines in such systems have been proposed and are in use. However, most of them are based on some centralized catalogue. Only recently, distributed search methods have become available in the context of Peer-to-Peer networks (e.g. Gnutella-like systems). This paper proposes a distributed search mechanism based on multiple concurrent search message chains. The method has been evaluated with a stochastic simulation model and with a prototype implementation. The results obtained from both experimentations correlate well, thus showing the viability of the approach.

1 Introduction

The dynamic nature and the size of the Web makes it difficult at best to keep track of the many desirable available resources. Because some resources may appear and disappear very quickly, the information about new resources may not become available fast enough to be useful. In addition, the shear quantity of information to manage also proves difficult to handle. Finally, resources may seize to be available without the resource databases being notified. For small networks and clusters, it is possible to keep the information synchronised on multiple servers. This is not viable, however, in large networks, where large quantities of information are available. In spite of the potential

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

problems stated above, some distributed system architectures, such as Globus, Charlotte, and Legion, still centralize information about available resources. Even large information bases, such as Alta Vista[1], keep their distributed information providers in sync. In this particular case, the quantity of useful (and used) information is a small fraction of all the information available on each node. It clearly makes more sense to have the information distributed across the Internet and to provide mechanisms to efficiently access this information on demand.

The work presented in this paper is part of a larger initiative, the Web Operating System (WOS^{TM}) [6]. The goal of the WOS project is to create an infrastructure to support transparent use of distributed resources. At the heart of this infrastructure is the WOS node. Each WOS node operates as a resource server, a resource client, and a resource information provider. A major goal of the WOS system is to provide the most suitable resources available to a user, at the moment of the request. To achieve these goals, it is crucial that these resources be found effectively and efficiently, using as little resources as possible.

In the WOS, information about available resources is highly distributed. In this paper, we empirically evaluate a search method to locate distributed resources in a WOSnet (i.e., a network of WOS nodes). This method aims to minimize the response time for a request while minimizing the network load, which are conflicting objectives.

In the next section, we briefly present the search method along with relevant results from a stochastic analysis performed [10]. In Section 3, we describe the environment in which an empirical evaluation of the search method was conducted. The results from the empirical study are presented and discussed in Section 4. We conclude the paper in Section 5.

2 An optimized search model

A WOSnet may be modelled as a graph, where each node corresponds to a WOS node and the edges correspond to the nodes known by another node. The search method is based on the use of multiple sequential search chains [2, 10]. Each chain performs a serial search, the nodes in the chain being visited in turn, while the different chains are processed in parallel. The search thus consists in sending search lists to a set of nodes which then concurrently perform the search defined in the associated list for that node.

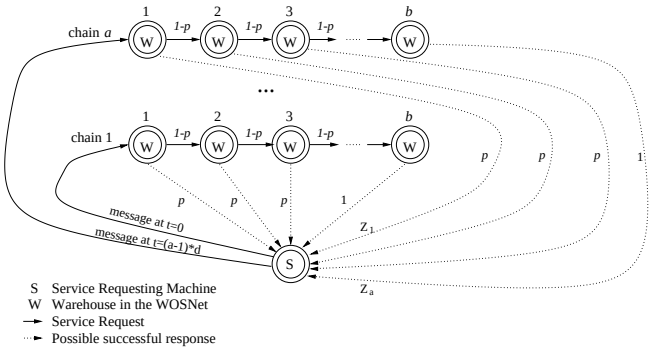
This search method can be generalized as follows: we generate a serial chains, each chain containing the addresses of b nodes. In other to perform an analysis of this new approach, other parameters are needed as well (see Figure 1) :

- t_t, the average transmission time of a message from one node to another;
- t_v, the average processing time of each node, to search whether the information requested is available locally;
- d, the delay between the transmission of two message chains by the originating node;
- p_j, the probability that the requested information is available at the j^{th} node; we suppose that this is a constant for each node and denote this probability by p;

[1] http://www.altavista.com

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

- q, the probability that the service is found through the first search wave (a message chains of b machines). Clearly, $q = 1 - (1-p)^{a \cdot b}$;
- Z_i, the expected response time of the i^{th} chain in a system of a chains of b nodes;
- n_q, the total number of nodes visited (i.e., $n_q = a \cdot b$);
- Z, the expected response time for all chains in a system of a chains of b nodes. Formally, $Z = \min_{i \in [1, n_q]} Z_i$.



Distribution of the answer time Z_i of chain i after its start :

Z_i	$t_v + 2t_t$	...	$jt_v + (j+1)t_t$	...	$bt_v + (b+1)t_t$
Probability	p	...	$p(1-p)^{(j-1)}$	...	$(1-p)^{(b-1)}$

Fig. 1. Sequential Search Chains

Figure 2 (taken from [10]) shows the expected response time Z for different numbers of chains (a) and nodes (n_q) to be visited. It contains a linear correction for the case where too many chains are used and the requesting machine is blocked for a longer time. The graph shows the expected minimum, which is nearly the same for all n_q.

3 Empirical evaluation of the search model

In this section, we present the prototype implementation used to evaluate the search method in a real environment. This prototype was developed within the WOS project. In particular, the search method is used to locate WOS nodes that can provide services to end-users. The module developed for that purpose is the WOS Service Search Unit (SSU). The SSU implements the theoretical search method described above. It is written in Java [5, 4, 7] and is a central part of the WOS prototype. The motivations behind the selection of Java are reliability, simplicity and architecture neutrality. Furthermore, since Java becomes more and more an "Internet Programming Language" [3], support for networking, security, and multi-threaded operations is incorporated and improved. These features predestine Java for the implementation of applications for large distributed and heterogeneous networks like the WOS [1, 8, 9].

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

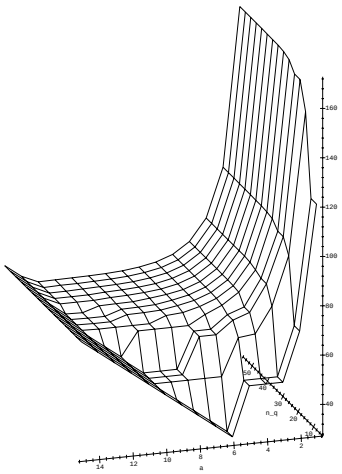


Fig. 2. The response time Z as a function of n_q and a.

3.1 The WOS Protocol (WOSP)

The WOS [6] was developed to provide a user with the possibility to submit a service request without any prior knowledge about the service (where it is available, at what cost, under which constraints) and to have the service request fulfilled within the user's desired parameters (time, cost, quality of service, etc.).

The WOS aims to provide adequate tools that allow the implementation of specific services not initially foreseen. In order to achieve this goal, a generic service protocol (WOSP) allows a WOS node administrator to implement a set of services, called a *service class*, dedicated to the specific needs of users. WOSP is in fact a generic protocol defined through a generic grammar [2]. A specific instance of this generic grammar provides the communication support for a service class of WOS. This specific instance is also referred to as a *version of WOSP*; its semantics depend directly on the service class it supports. In other words, knowing a specific version of WOSP is equivalent to understanding the semantics of the service class supported by that version. Several versions of WOSP can coexist on the same WOS node.

A version of WOSP has been specifically defined to support the multiple sequential search chain method. This version is based on the following assumptions:

- The SSU implements the logic of the multiple sequential search chains method;
- Each WOS node is running a copy of the SSU to manage the search and the response provided;
- The different SSUs communicate using that new version of WOSP.

3.2 Implementing the Service Search Unit

The SSU handles all search requests, as shown in Figure 3. Incoming requests are transmitted by the WOS communication layer to the SSU (step 1). The information request

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

is dispatched to the local Remote Resource Control Unit (RRCU) (step 2), which is the WOS module responsible for the management of remote information requests and remote resource utilisation. In step 3, the RRCU queries its Resource Warehouse (a database of locally known resources) for an appropriate entry. There are three possible results :

1. *The resource is available locally.* In this case, the node sends a positive answer directly to the requesting machine (steps 4,5,6).
2. *The service is "known" but not available locally.* The address of the node offering the service is added to a list of possible nodes to search, if not already present, and added to the service request message (steps 4',5,7). This helps minimize the network load and prevents multiple copies of the same node being sent.
3. *The service is neither available nor "known".*

In every case, the current node removes its address from the list of nodes to be visited and sends the request message to the next address in the list (step 7). If the node is the last node in the chain, a termination message is sent to the requesting machine (step 7').

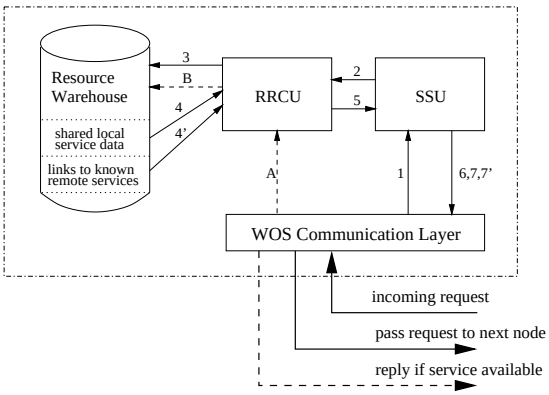


Fig. 3. The flow of a service request inside a node

A reply (step A) is handled directly by the RRCU of the requesting node, which can now update the local resource warehouse with the reply data (step B).

In the specific context of the WOS, this search mechanism is used in two cases:

1. to locate nodes that can provide a special service (search request);
2. to find the "best server," if the service actually should be used (execution request). In this case, "best server" means for instance the machine with the currently lowest workload or with the lowest price, and so on, based on the end-user's preferences.

A search request or an execution request must contain the following information :

– a reference to the original request,
– the name of the required service (i.e., the resources to locate),

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

- possible execution parameters (i.e., the “best server” selection criteria),
- the address of the requesting node,
- the list of nodes (IP addresses and port number) to be visited,
- the list of potential nodes to visit (initially empty).

If a node can offer the required service, it sends a reply message back to the requesting machine. This reply contains :

- the name of the requested service,
- parameters and restrictions for this service.

Parameters and restrictions of a service are all details concerning the service itself (environment, resources, etc.) and statements about the availability of the service (time windows, prices, etc.).

3.3 Tests and data collection

After implementing the SSU, experiments were conducted in order to show that the system behaves as expected with the theoretical model. In order to perform these experiments, the communication times for several cases were measured.

Each outgoing request message contains a time stamp taken at the sending time. By using this time stamp, the running time of a chain or the response time (reply) were calculated.

For this experiment, 40 machines running SUN Solaris, Java and the WOS communication layer were involved. The test environment was heterogeneous, each machine having different computing power; some slow Sparc Classic where used as well as medium fast Sparc Station and fast SUN Ultra workstations. The network connecting the machines is a switched Ethernet (10 to 100 Mbps) consisting of three subnetworks (see Figure 4).

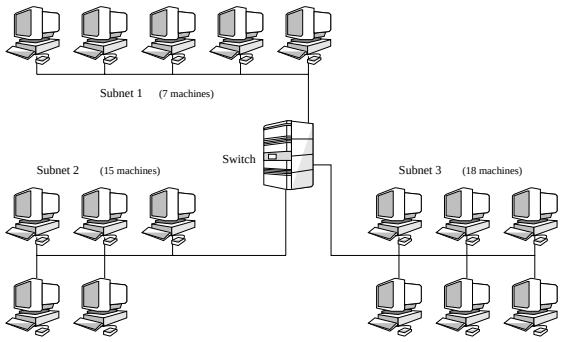


Fig. 4. Topology of the test network.

To establish the response time, we collected the time of transmission of each search chain and the time of arrival of each positive reply. This information allowed us to

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

calculate the smallest response time for a single request. We were also able to calculate the average delay d for preparing and sending the different chains. This last result is used to confirm that d is in fact constant and independent from the number of chains a and the number of nodes to contact n_q.

4 Experimental results

Figure 5 shows the average response time for a request, noted $\bar{Z}$. The average response time is the time it takes for a response to reach the requesting node. The standard deviation of these measures is shown in Figure 6.

In this experiment, the requesting machine is a SUN Sparc classic. The optimal number of chains is between 6 and 12, which corresponds to the simulation results (Figure 2). This result is the same for different values of n_q (15, 20, ..., and 40 machines).

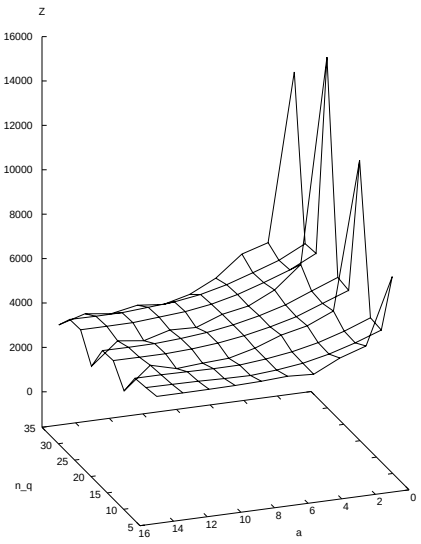


Fig. 5. Average response time $\bar{Z}$ a a function of n_q and a.

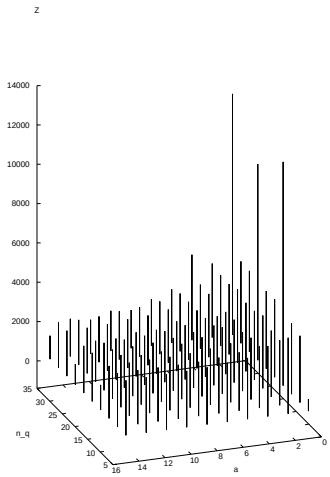


Fig. 6. Standard deviation of the average response time $\bar{Z}$.

In the simulated model and in the empirical study, the probability p that a node finds an entry for the required service in its own list of services is set at 10%. A p different from 10% will cause shorter (>10%) or longer (<10%) response times, but the overall behavior stays the same.

Figure 7 shows the average delay ($\bar{d}$) between the transmission of two chains. These values seem fairly constant, except on the edges. However, as can be seen from Figure 8,

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

the standard deviation is quite large, indicating that, although constant on average, the delay varies greatly.

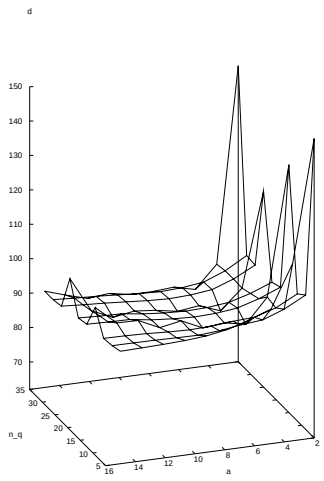


Fig. 7. Average delay $\bar{d}$ a a function of n_q and a.

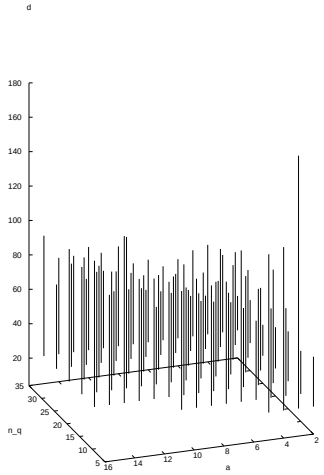


Fig. 8. Standard deviation of the average delay $\bar{d}$.

5 Discussion and conclusion

The empirical data are matching the theoretical model pretty well. There are many parameters influencing the actual empirical data. The optimal number of chains is different for machines and networks with different speeds. When the request is issued by a slow machine (SUN Sparc classic) the best results will be achieved with a small number of chains ($a \in [4, 8]$). On a faster machine, (SUN Sparc Station-10), a larger number of chains ($a \in [15, 20]$) will give the best results. On our fastest machines (SUN Ultra-60), it takes less time to broadcast the request ($b = 1$) that to run search chains.

We experimented with an efficient search method for the service search in the Web Operating System (WOS). However, the approach described can also be used in other systems in an efficient manner. The search system makes use of a compromise between speed and resource consumption. The implementation realized fulfils the requirements of a search engine with no centralized data catalogues. While the prototype implementation proved successful, further mechanisms to improve security and performance must still be discussed and included.

From the results of the empirical measurements, it is clear that the number of chains for a search might be subject to adaption. This should be an additional feature of the

Gilbert Babin, Thomas Böhme, Herwig Unger, and et Peter Kropf. *An Optimized Search Mechanism for Large Distributed Systems*. Preprints no. CS-06-99. Universität Rostock. Rostock, GermanyAllemagne. 1999. ISSN 0944-5900.

SSU. Therefore, an intelligent mechanism is needed which can find out the optimal chain number and length, accounting for machine speed, machine load, network speed, network load, and the number of nodes to be visited.

References

1. A.D. Alexandrov, M. Ibel, K.E. Schauser, and C.J. Scheiman. SuperWeb: Research issues in java-based global computing. In *Workshop on Java for Computational Science and Engineering Workshop*, Syracuse University, December 1996.
2. Gilbert Babin, Peter Kropf, and Herwig Unger. A two-level communication protocol for a Web Operating System (WOS^{TM}). In *IEEE Euromicro Workshop on Network Computing*, pages 939–944, Västerås, Sweden, August 1998.
3. J. Farley. *JAVA Distributed Computing*. O'REILLY, Cambridge, 1998.
4. C.S. Horstmann and G. Cornell. *Core Java 2 : Advanced Features*, volume II. Prentice Hall, 4th edition, 1999.
5. C.S. Horstmann and G. Cornell. *Core Java 2 : Fundamentals*, volume I. Prentice Hall, 5th edition, 2000.
6. Peter Kropf. Overview of the WOS project. In *1999 Advanced Simulation Technologies Conferences (ASTC 1999)*, San Diego, CA, USA, April 1999.
7. P. Niemeyer and J. Peck. *Exploring JAVA, Second Edition*. O'REILLY, Cambridge, 1997.
8. Sun Microsystems Inc. Jini, 1999. http://java.sun.com/products/jini/whitepapers/.
9. Sun Microsystems Inc. Jini Specification, 1999. www.javasoft.com/products/jini/specs.
10. Herwig Unger, Peter Kropf, Gilbert Babin, and Thomas Böhme. Simulation of search and distribution methods for jobs in a Web operating system (WOS^{TM}). In A. Tentner, editor, *High Performance Computing Symposium '98*, Boston, MA, USA, April 1998. SCS International.