PAGE-BASED DISTRIBUTED SHARED MEMORY FOR OSF/DCE*

Jerzy Brzeziński Michał Szychowiak Dariusz Wawrzyniak

Institute of Computing Science
Poznań University of Technology
Piotrowo 3a, 60-965 Poznań, Poland
e-mail: brzezinski@put.poznan.pl
Michal.Szychowiak@cs.put.poznan.pl
darek@cs.put.poznan.pl

Abstract

Distributed shared memory systems strive to overcome the architectural limitations of shared memory computers and to make easier developing parallel programs in distributed environment. As is known, however, in order to meet these goals in practice many specific and difficult problems have to be solved. In this paper fundamentals of DSM systems' construction — including basic design, mechanisms, memory consistency models, and problems — are presented. Then, a general concept and hierarchical structure of page-based DSM system for UNIX and OSF/DCE platforms, has been proposed. Applications of the basic DCE components for improving security, modularity, scalability and portability of the proposed system, in comparison with the existing ones, have been described.

1 Introduction

Generally, *distributed computer systems* are thought of as a collection of loosely coupled *processing units* (*nodes*, *sites*) interconnected by *communication network*. Nodes in a distributed systems are always equipped with a *processor* and a *local memory*, but they do not share any *common memory* or a *clock*, and they may vary significantly in size and functionality. Nodes may include both single-chip computers, RISC workstations, minicomputers, general purpose mainframes as well as supercomputers. On the other hand, communication network may be a typical Ethernet, Token Ring, FDDI, ATM, HIPPI or specialized packet or circuit switching structure. Thus, to distributed systems category fall networks of personal computers, clusters of workstation as well as distributed memory parallel computers such as SP2, T3E, Exemplar, etc. These systems due to their scalability, computational effectiveness, reliability, inherent communication facilities, flexibility and efficiency in resource utilization — are one of the most important and promising trends in computer systems development. However, due to a lack of a

* This work was supported in part by the State Committee for Scientific Research (KBN), Poland, under grant KBN 8T11C05208.

common memory and a clock, their programming, based on message passing only, is extremely tedious, difficult and in some sense not natural. This is because programmers used to commonly apply shared memory programming paradigm. Thus, to take simultaneously advantage of both distributed and shared memory systems one has to implement distributed memory management component with adequate algorithms and protocols which hide an explicit exchange of messages between nodes from user, giving him illusion of possessing shared memory space. Distributed systems equipped with such components are known as *distributed shared memory* (DSM) *systems*.

In essence, a structure and mechanisms of DSM systems are similar to traditional virtual memory ones: when a process refers to a location (page) non-resident in local memory of a node, a trap occurs and the distributed operating system fetches the page from another node over the network and maps it in. Hence, from user viewpoint all communication and synchronization can be done via memory with no message passing visible to the user processes. As a result, DSM systems offer, at least potentially, the following advantageous characteristics: attractive for users shared memory programming model, scalability allowing for practically unlimited and incremental growth, large size of shared memory that may consist of local memories available at all nodes, possibility to run programs written for shared memory multiprocessors. Unfortunately, it turns out that reaching these advantages requires efficient solution of many problems. The fundamental issues concern: keeping track of the current location of remote data, reduction of delays and overhead when accessing data, efficiency and correctness of concurrency control, security and reliability. These issues have been studied intensively for the last decade and as a result many interesting systems have been proposed and implemented till now ([6]], [12]], [15]], [16]]).

To our knowledge the common weaknesses of all these systems comprise: the lack of embedded security mechanisms, sensibility to link and node failures (low resilience to failures), unavailability of different consistency models for concurrently running processes, limitation of applicability in heterogeneous environment.

This paper intends to present a general concept of page-based distributed shared memory system aimed at overcoming the above mentioned disadvantages of existing DSM systems. To meet this goal we have chosen UNIX and OSF/DCE platforms as underlying components. It has been because UNIX is still the most popular multiprocessing system, and because DCE offers many features important in the context of DSM system's construction, including: high level of standardization (each DCE component corresponds with some widely accepted standards, e.g. ISO, X/Open or POSIX), portability of applications, availability of sophisticated security mechanisms, availability of scalable naming (directory) services allowing applications to be distributed over a heterogeneous environment.

The rest of the paper is organized as follows. In Section 2 fundamentals of DSM systems' construction — including basic design, mechanisms, problems and memory consistency models — are presented. Section 3 describes the proposed general concept and hierarchical structure of DSM system for UNIX and DCE platforms. In section 4, application of DCE services (DCE Threads, DCE Directory Services, DCE RPC, DCE Security Services) in the construction of the DSM system is presented and analyzed. Finally, Section 5 concludes the paper and outlines further lines of investigations.

2 DSM fundamentals

2.1 Concepts and mechanisms

Distributed shared memory (DSM) is a single address space shared by several hosts connected via a network communication environment. The DSM is a kind of virtual memory and the job of the DSM system is an automatic mapping of the shared virtual address space into the physical address space of the hosts composing the system.

In order to get access to a shared variable it is necessary to locate the host which contains the addressed variable and then to communicate with it to perform a required operation. The operation can be performed either on the remote host (*remote access*) or locally after fetching the variable to a local cache (*local access*). The local access requires the variable to migrate from the remote host to the local one. In practice the unit of transfer is a page rather than a single variable. This can reduce the access time when the page is multiply accessed on the same host. The DSM systems applying unique pages as basic exchange unit are called *pagebased* .

To allow concurrent access to the same page (especially for reading) the page can be replicated on distinct hosts. It is necessary, however, to synchronize the write access to prevent replicas of a page from being inconsistent.

2.2 Basic problems

Fundamental problem in implementation of a DSM system is an efficient translation of virtual addresses into the physical ones. It is difficult, as pages of DSM can migrate from one host to another or be replicated on different hosts, so the translation of the virtual address encompasses the identification of the hosts which currently contain the corresponding page. Various approaches to the problem are presented in [16]].

To allow migration of a page it is necessary to ensure free space for the migrating page in the local cache of the destination host. This may require one of the pages kept locally to be removed from the local cache. It seems to be less expensive to remove the page which has a copy in some host. Unless the page is replicated, it is necessary to make another migration or store the removed page on the disk. Possible solutions for the replacement problem are described in [13]].

Another problem, resulting from the replication, is a correct and efficient control of concurrent access to the same page on different hosts to maintain the consistency of DSM. The problem concerns especially the write access, because the write operation changes the state of DSM. This is the job of coherence protocol and the way of doing it depends on the consistency model assumed for the DSM.

Besides the essential problems mentioned above, the problem of false sharing and thrashing is frequently pointed out ([16]], [18]], [21]]). False sharing occurs while two or more processes on distinct hosts try to access distinct locations on the same page. When one process gets an exclusive access to a location (e.g. for writing), other processes are not allowed to access locations on the same page, although they attempt to access distinct locations. Thrashing occurs when two processes frequently write to the same location (or distinct locations on the same page), which causes the DSM system to transmit the subsequent page back and forth between the hosts. Thrashing is very often caused just by false sharing and can be reduced by decreasing the page size, which in turn can increase the number of page faults.

2.3 Consistency models

Following [19]], we assume that the DSM system is a finite set of sequential processes P_1, P_2, ..., P_n that can read or write a finite set X of shared locations. A write operation of a value v into the location $x \in X$, issued by the process P_i, defines a new value for this location and is denoted $w_i(x)v$. Similarly a read operation which obtains a value v is denoted $r_i(x)v$. All operations of P_i are executed in the same order as they have been issued, hence they are totally ordered. The total order relation in the set of all operations of P_i is denoted $\rightarrow_i$. The order of execution of operation issued by different processes depends on the consistency criterion for the DSM. In the literature various consistency models have been considered: atomic consistency, sequential consistency, causal consistency, PRAM consistency, processor consistency, release consistency, among others. For details see [1]], [4]], [5]], [6]], [14]], [17]], [19]].

2.4 Coherence protocols

A task for a coherence protocol is to maintain a consistent state of DSM pages replicated on hosts. As data are being changed by write operations, an appropriate synchronization of these operations is a question of great importance. Every protocol has to ensure that a content of each replica will eventually reflect an execution of all write operations.

A possible way to achieve the atomic consistency is to prevent from an execution of two or more competing operations at the same time. It is ensured, for instance, by a *data invalidate protocol* in DSM system with read only replicas [16]]. To exploit advantages of other consistency models it is necessary to use a *data update protocol*. A common method of implementation of a data update protocol in asynchronous message passing system is applying a suitable multicast protocol, i.e. total order multicast protocol for sequential consistency ([7]]), causal order multicast protocol for causal consistency ([4]]) and FIFO order multicast protocol for PRAM consistency. The simplest way to ensure sequential consistency is to allow only one process, called page owner, to write to a page, while the others are allowed only to read it simultaneously. This solution does not require the total order multicast protocol.

3 General concepts of DSM for DCE

3.1 System structure

The proposed concept of DSM system is based on the OSF/DCE - one of the most popular environments for distributed computing. As was mentioned, we have chosen the DCE mainly because it offers: availability of sophisticated security mechanisms, full standardization (each DCE component conforms to a widely accepted standard, e.g. ISO, X/Open or POSIX), portability of applications, availability of scalable directory services allowing applications to be distributed over a heterogeneous environment.

A hierarchy of the proposed system is presented in Figure 1. We describe each layer of the system in the following subsections.

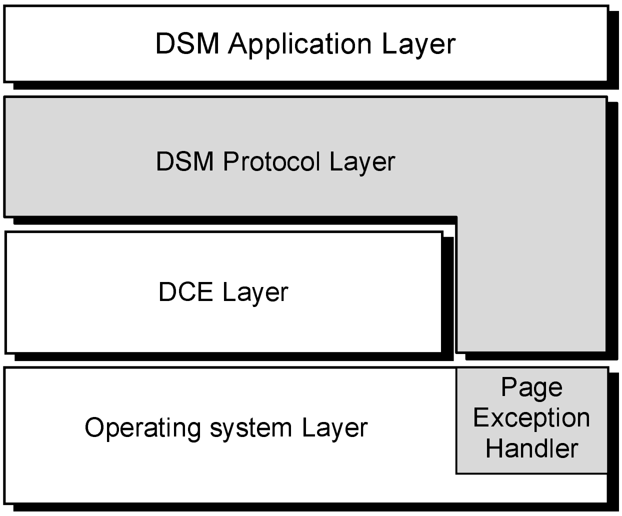

Figure 1. Hierarchy of the proposed DSM system software

3.1.1 Operating System Layer

The Operating System Layer consist of two entities: Unix operating system[1] and Page Exception Handler module (PEH). Unix function is to be a platform for the next layer — DCE Layer, and it signals memory page access exception (page fault) to the PEH. It can be one of many available Unix systems on which the OSF/DCE is portable (see Section 3.1.2 for more details about the DCE). The PEH is a software module responsible for providing access to shared memory pages. It uses Unix *mprotect()* system function and *SIGSEGV* signal to catch a page access operation and redirect control to the DSM Protocol Layer. The PEH uses similar access catching method as TreadMarks [6]] and CVM [12]] systems do.

3.1.2 DCE Layer

Distributed Computing Environment (DCE) is one of the major environment systems for distributed computing. It was conceived as a project of the *Open Software Foundation (OSF)*, with the aim to define a standard middleware platform for distributed applications. This standardization allows the distributed applications to run regardless of the different operating systems and transport services. Today, DCE is available for many Unix systems, including DEC/Ultrix, VAX/VMS, IBM/AIX, SunOs, SCO Unix, and others.

The most important advantage of DCE is the comprehensiveness and interoperability of its components. Next, we briefly describe the basic DCE components, called *fundamental distributed services*.

DCE Threads: this is the lowest layer of DCE services, responsible for realization and management of multithread process execution. This layer is used, e.g. by multithreaded RPC servers and in RPC asynchronous calls.

---

[1] We are now investigating the possibility to use also other operating systems, including Windows NT among others.

DCE RPC: it offers communication services in the form of remote procedure calls. The RPC imposes a client-server model for interprocess communication.

Security Services: they implement symmetric authentication of clients and servers, and offer encryption facilities.

Cell and Global Directory Services: they are responsible for name management in the distributed environment and they are used especially by RPC to locate endpoint entities.

Distributed Time Services: they take care of the synchronization of local clocks of DCE nodes. This feature is often required by many distributed algorithms and other DCE components.

3.1.3 DSM Protocol Layer

The DSM Protocol Layer offers for DSM applications a right to choose a consistency model out of the following ones: atomic consistency, sequential consistency, causal consistency, PRAM consistency.

Each model is implemented by appropriate consistency protocol. In the project it has been assumed that every DSM page has an associated page manager, PAGEman (see Figure 2).

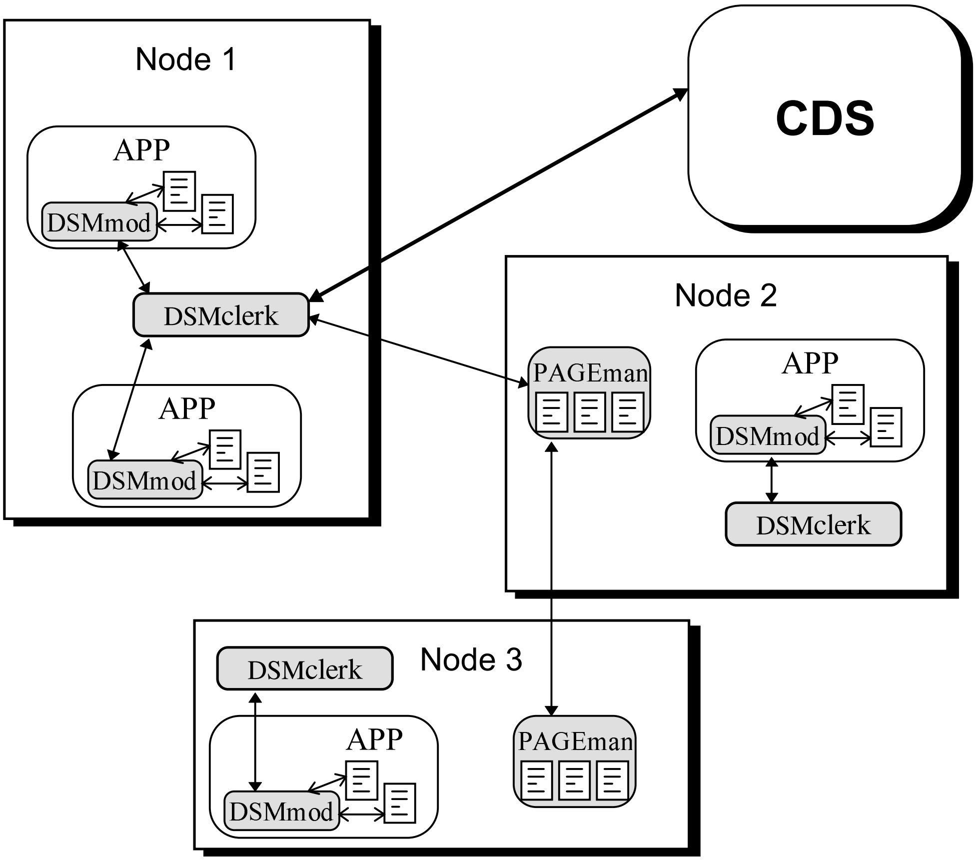

Figure 2. Structure of the DSM system

A page manager can manage one or several pages. In a special case there can only be one page manager for all pages in the system. The DCE Layer allows us to implement in the same way central page manager model and distributed page manager model.

Consistency protocols are implemented by two modules of the DSM Protocol Layer:

- **DSMmod** — a page access control module linked with every DSM application;
- **DSMclerk** — a DSM protocol management server associated with every DCE node.

DSMmod takes care of communication between the Page Exception Handler module and the application. Moreover, communication between the DSMmod and the PAGEman is accomplished by a DSMclerk. A page access is redirected by the PEH to the DSMmod which is responsible for obtaining desired page and support adequate consistency. To provide this consistency, a DSMmod needs to communicate with one or several PAGEmans. For example, when an application wishes to write a content of page X, the DSMmod of this application catches the request of access. If the page is not locally available, the DSMmod contacts its DSMclerk which finds appropriate PAGEman (manager of the page X) and asks for the current location of the page X (the current owner of that page). After the response, the DSMclerk finds desired page X and brings it to the DSMmod of the application. Then the application becomes new owner of the page X.

3.1.4 Application Layer

DSM applications are placed in the Application Layer. Before an application can access shared pages, it has to select desired consistency model. When this is done, the underlying DSM Protocol Layer offers DSM pages attachment and access, and transparently to the application guarantees selected consistency model.

4 Application of DCE services for DSM implementation

In the DCE there is neither notion of shared memory nor any global data structures shared between two DCE applications. The only possible communication model is the client/server one, with the remote procedure call semantics, where data are passed via RPC parameters.

Some of the DCE services can be used to provide the application programmer with the Distributed Shared Memory services. These include DCE RPC as the only communication service, and DCE Threads, DCE CDS and Security Services for the DSM Protocol Layer implementation purpose. In the rest of this section we will discuss possible applications of these services in the DSM Protocol Layer implementation.

4.1 Application of DCE Threads

A DSMmod is an application thread. This allows very efficient realization of process memory control mechanism. A DSMmod is executed in parallel with the main thread of an application, and is responsible for handling both page access requests from the application and update or invalidate requests from DSMclerk. DCE Threads services are also used by DSMclerk and PAGEman servers to allow asynchronous RPC calls in some consistency protocols.

4.2 Application of DCE Directory Services

CDS services are used to accomplish many purposes. First they are necessary to obtain access to correct DSMclerk service on appropriate node. Next, CDS is used to locate the page manager of desired DSM page (i.e. to locate correct PAGEman). It is also possible to use GDS to extend the scope of the DSM to the worldwide context (Internet).

We found some important problems related to the access of shared memory pages. The first one was the selection of the most appropriate DCE service to find a page and possibly to synchronize simultaneous accesses to it. This can be achieved by the concept of page manager, and we have analyzed two following hypothetical its implementations:

- page manager as an object of the CDS;
- page manager as a dedicated DCE/RPC server.

There are some advantages and disadvantages of both propositions.

4.2.1 Page manager as an object of the CDS.

The directory is a collection of information about a set of objects in the DCE global or local environment. Objects of a DCE cell are referenced in so called entries of the Cell Directory Service (CDS) managing that cell. The CDS entry can store such information as the current localization of a DSM page and the current access mode to this page. Moreover, any CDS object can be seen in a global context, with the aid of the Global Directory Service. Thus every DSM page can be located in the entire environment, not only in its local cell.

In this application the CDS has several disadvantages from the viewpoint of the page access synchronization. The CDS is not designed to be a generalized distributed database. It does not have sufficient capabilities to offer update consistency of its replicated or cached objects, nor to provide high level of access time. It does not guarantee atomic transactions. Its services has been intended to provide applications with an information that is relatively rarely changed. Look up (read) operations are expected to be dominant in comparison to update (write) operations, and there is no assumption of instantaneous reflection of performed update (transient inconsistency states are acceptable, [2]]). Finally, we did not found the CDS as a good tool for realization of DSM page manager.

4.2.2 Dedicated DCE server.

Another solution is to implement the page manager as a dedicated DCE/RPC server. We have called this server PAGEman. PAGEman synchronizes access to memory pages issued by various applications (i.e. DSMmod routines) and invoked via DSMclerks. Several PAGEmans compose distributed page manager, thus consistency has to be reached somehow with the aid of internal communication between all PAGEmans. This is done by one of internal consistency protocols of the DSM Protocol Layer.

Every page access issued by an application is overtaken by its DSMmod routines. If the DSMmod needs to obtain this page from adequate page manager, it contacts a DSM node server — DSMclerk. It is DSMclerk's work to find appropriate PAGEman and bring up desired page, respecting chosen consistency model. How can the DSMclerk contact the suitable PAGEman? The best solution to this is to use the CDS object entry fetch. Every DSM page is a CDS object, and every DSMclerk can find wanted page and import the binding information required to perform an RPC call to its PAGEman.

4.3 DCE RPC

Every DSMclerk is registered as an RPC server in the CDS, and every DSMmod can import the binding information required to perform a RPC call to chosen DSMclerk. Sometimes, within update or invalidate protocol, a given DSMclerk has to contact associated DSMmods, e.g. for page invalidation. It is then obvious that also DSMmods have to act as RPC servers and wait for client calls from DSMclerks. DSMclerk can remember any required RPC binding information from previous communication with a DSMmod and use it later as RPC client. All PAGEmans are also RPC servers and clients.

DCE/RPC offers advanced time-out and cancel mechanisms to ensure communication correctness. DCE does not include group RPC communication, useful for some consistency protocols. But it is still possible to implement group RPC with point-to-point RPC mechanisms. For more details the interested reader is referred to [11]].

As is shown in [8]], [9]], [10]] RPC can be effectively applied for coherence protocols implementation.

4.4 DCE Security Services

Executing remote procedures over a computer network imposes serious requirements for security. The DCE RPC includes an interface to the DCE Security Services. It is up to the programmer to select desired level of security mechanisms: client-to-server authentication, server-to-client authentication, authorization of access to server resources and cryptographic protection of communication. Security Services support multiple encryption standards (DES, RSA, MD5) and offer a comfortable encryption negotiation facility. Every page access request is confronted with the Access Control List associated to each DSM page, for the reason to enable or disable access from a given user application.

5 Conclusions

Distributed shared memory systems strive to join advantages of both shared memory multiprocessor and distributed systems offering, at least potentially, attractive for users virtual shared memory as well as system scalability. As it is known, however, in order to meet these goals in practice many specific and difficult problems have to be solved.

In this paper we have analyzed, first, fundamentals and basic problems of DSM systems' construction. Then, the general concept and the hierarchical structure of page-based DSM system for UNIX and DCE platforms have been proposed. In the proposed system, the basic DCE services — such as DCE Threads, DCE Directory Services, DCE RPC, DCE Security Services — have been applied to improve the system security, modularity, scalability and portability in comparison with existing solutions, including IVY, PLUS, DASH, Mirage, TreadMarks, CVM, among others.

The proposed DSM system implementation is in progress. After putting the system into operation, we plan to use it first as a platform for our further research activities concerning mainly performance and reliability analysis of DSM mechanisms. Then, we are going to address the important issues related to the construction of more efficient, secure and failure resilient DSM systems.

6 References

[1] Adve S. V., and K. Gharachorloo: *Shared Memory Consistency Models: A Tutorial*, IEEE Computer, Vol. 29, No. 12, December 1996, Pages 66–76.

[2] *AES/Distributed Computing — Directory Services*, Revision B, Open Software Foundation, Cambridge, October 1994.

[3] *AES/Distributed Computing — Remote Procedure Call*, Revision B, Open Software Foundation, Cambridge, October 1994.

[4] Ahamad M., P. W. Hutto, G. Neiger, J. E. Burns, and P. Kohli: *Causal Memory: Definitions, Implementation and Programming*, Technical Report GIT-CC-93/55, Collage of Computing, Georgia Institute of Technology, Atlanta, September 1993.

[5] Ahamad M., R. A. Bazzi, R. John, P. Kohli, and G. Neiger: *The Power of Processor Consistency*, Technical Report GIT-CC-92/34, Collage of Computing, Georgia Institute of Technology, Atlanta, March 1993.

[6] Amza C., A. L. Cox, S. Dwarkadas, P. Keleher, H. Lu,R. Rajamony, W. Yu, and W. Zwaenepoel: *TreadMarks: Shared Memory Computing on Networks of Workstations*, IEEE Computer, Vol. 29, No. 2, February 1996, Pages 18–28.

[7] Attiya H., and J. L. Welch: *Sequential consistency versus linearizability*, ACM Transactions on Computer Systems, Vol. 12, No. 2, May 1994, Pages 91–122.

[8] Brzeziński J., and Wawrzyniak D.: *Multi-Coherence Protocol for Distributed Shared Memory*, to appear in the proceedings of the 6th IEEE Workshop on Future Trends in Distributed Computing Systems (FTDCS'97), Tunis, Tunisia, October 1997.

[9] Brzeziński J., Szychowiak M., and Wawrzyniak D.: RPC-based Coherence Protocols for Distributed Shared Memory, to appear in the proceedings of the IASTED IC on Parallel and Distributed Computing and Networks (PDCN'97), Singapore, August 1997.

[10] Brzeziński J., Szychowiak M., and Wawrzyniak D.: *Page-based Distributed Shared Memory for OSF/DCE*, Technical Report RA-011/96, Institute of Computing Science, Poznañ University of Technology, December 1996.

[11] Hiltunen M. A., and Schlichting R. D.: *Constructing a Configurable Group RPC Service*, 15th IEEE IC on Distributed Computing System, Vancouver, May 1995, Pages 288–295.

[12] Keleher P.: *The CVM Manual*, University of Maryland, November 1996.

[13] Kermarrec Y., and L. Pautet: *Integrating Page Replacement in a Distributed Shared Virtual Memory*, 14th IEEE IC on Distributed Computing Systems, Poznañ, Poland, June 1994, Pages 355–262

[14] Lamport L.: *How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs*, IEEE Transactions on Computers, vol. C-28, no. 9, September 1979, Pages 690–691.

[15] D., Laudon J., Lenoski Gharachorloo K., Weber W.-D., Gupta A., Hennessy J., Horowitz M., and Lam M. S.: *The Stanford Dash Multiprocessor*, IEEE Computer, vol. 25, no. 3, March 1992, Pages 63–79.

[16] Li K., and Hudak P.: *Memory Coherence in Shared Virtual Memory Systems*, ACM Transactions on Computer Systems, vol. 24, no. 8, November 1989, Pages 321–359.
[17] Lipton R. J., and J. S. Sandberg: *PRAM: a scalable shared memory*, Technical Report CS-TR-180-88, Princeton University, September 1988.
[18] Nitzberg B., and V. Lo: *Distributed Shared Memory: A Survey of Issues and Algorithms*, IEEE Computer, vol. 7, no. 4, August 1991, Pages 52–60.
[19] Reynal M., and A. Schiper: *A Suite of Formal Definitions for Consistency Criteria in Distributed Shard Memories*, Technical Report PI n° 968, IRISA Rennes, November 1995.
[20] Singhal M., and Shivaratri N. G.: *Advanced Concepts in Operating Systems: Distributed, Database, and Multiprocessor Operating Systems*, McGraw-Hill, 1994.
[21] Stumm M., and Zhou S.: *Algorithms Implementing Distributed Shared Memory*, IEEE Computer, vol. 24, no. 5, May 1990, Pages 54–64.
[22] Tanenbaum A. S.: *Distributed Operating Systems*, Prentice-Hall, 1995.