Shared Distributed Memory : the Workspace Model

Didier Donsez, Liming Chen, Pascal Faudemay†

Laboratoire HEUDIASYC / UTC
Centre de Recherche de Royallieu - BP 649
60206 Compiègne Cedex, France

{donsez,lchen}@hds.univ-compiegne.fr

†Laboratoire MASI / UPMC
4 Place Jussieu, 75 252 Paris Cedex 05, France

faudemay@masi.ibp.fr

Abstract

Shared Distributed Memory Systems offer uniform access to data which are distributed on servers. The Workspace model is a model of shared distributed memory. It is based on communicating processes which are both clients and servers. It enables to implement hierarchical views of data, to enhance security and it adapts to heterogenous networks.

Keywords: Distributed Systems; Object-Oriented Database Systems; Storage Management.

1. Introduction.

Database systems were first used on mainframes, and remain one of their major utilizations. As workstations display ever growing processing power and memory size, mainframes are less and less used, and their functions are often distributed between minicomputers and workstations. This favours the development of the Client-Server model, in which each client accesses one or more servers.

The server may implement data shipping, or query shipping. Query shipping corresponds to the old access model where a mainframe was accessed by terminals. When using workstations as terminals, a more evolved "presentation layer" is implemented on the workstation, and therefore it unloads the server which execute simpler queries. This model remains quite used with relational DBMS.

In object oriented database systems, the interaction is mainly based on Data Shipping. The server implements a simplified service for archiving, data exchange, and consistency maintainance.

- In these systems, the server cannot execute complex queries. They must be executed by an intermediate client application, which evaluates these queries on behalf of other clients. This introduces more data exchange levels between application processes. However this type of service is needed for secured applications and more generally for other "value added" services (query languages, etc...).
- A second inconvenience is that Client-Server architectures, although well adapted to homogeneous workstations networks, are much less adapted to heterogeneous networks, including wide area networks, minicomputers, and possibly mainframes.

The first series of criticisms has led to the definition of new access mechanisms to servers, which implement "symetric distributed shared memory". An example of such an architecture is SHORE, which is implemented at University of Wisconsin [1]. In this approach, each workstation is associated with a server process, which exchanges distributed data with the server processes of other workstations. Client applications contact their local server in order to import data or to execute Value Added Services.

Ce papier a été accèté au *European Research Seminar on Advances in Distributed Systems* (ERSADS) L'Alpe d'Huez, France - April 3-7, 1995

- However, in this architecture, each server communicates with the destination workstation only by data shipping. Data processing is not secure as the user of this workstation can have any privilege. The communication volumes between servers may be larger than in the case of remote processing.
- Another limitation of this architecture is that it does not easily integrate several hierarchical levels of servers, such as department servers and company servers.
- This architecture remains poorly adapted to the combination of local and distant networks, and therefore to heterogeneous networks including communications between several sites.

This led us to define a new model of system architecture, the Workspace model. This model is more general than the Client-Server architecture, or than the symetric model of SHORE. These two models are only specific instances of the Workspace model.

The Workspace model maps a network of communicating processes onto a graph of nested transactions, which are a generalization of Moss nested transactions [2]. In Moss' model, a parent transaction encapsulates a group of sub-transactions, or nested transactions. The validation of one of the sub-transactions makes their updates visible to the other sub-transactions of the same group, but not outside the group. This model is hierarchical, as a sub-transaction can also be the parent of other sub-transactions. The validation of the root transaction guarantees the durability of database updates. The Workspace model extends this model since a parent transaction decides if updates from its sub-transactions have to be propagated to the "grand-parent" transaction or not at sub-transaction commit time.

In its general form, this model is well adapted to heterogeneous networks, and to new applications such as cooperative work [3,4]. In its various utilization contexts, it enables performance improvements and functionalities extensions vs. other architectures. In section 2, we present the Workspace model. Then in section 3, we present the uses of the model. Section 4 concludes and presents the implementations.

2. The Workspace Model

The Workspace Model is intended to be used to build Distributed Persistent Object Managers. It is based on a generic building block, the Workspace. A Workspace is a process that includes an access kernel for accessing local persistent data. It also includes concurrent activities which share data transactionally [5]. Each activity execute a session of successive transactions.

The Workspace also offers an access to its data space objects as a service, and it can suscribe to services from others Workspaces. A workspace service can export objects to one or more other workspaces, or execute operations on these objects. The first service type corresponds to Data Shipping, while the second one corresponds to Query Shipping.

2.1. The Workspace

The Workspace defines a persistent object space, which is shared by transactions which belong to an application. This object space merges a view on local objects (which are archived in volumes which are directly managed by the Workspace), and views on distant objects which are accessed through the object space of other Workspaces. The resulting view can itself be accessed by other Workspaces. This recursive definition of the objects space is the basis for the building of hierarchical architectures of Persistent Object Managers.

Within a view, objects are accessed through their interface. This is permitted by the use of an object-oriented users interface (see section 4). Object manipulation can be local to the application, or remote through an operation service.

The Workspace includes an application, which is intended to be used by a single user. This application is executed as a set of several successive or simultaneous transactions. These

transactions read and update objects which are visible within the Workspace objects space. They can concurrently access objects, and therefore must usually respect the ACID properties of the transactional model (ACID : Atomicity, Consistency, Isolation, Durability) [5].

The Workspace can operate in passing mode or in retaining mode. In passing mode, updates validated by a transaction are immediately transmitted to the local archive or to the object space of distant Workspaces where they come from (this is a write-through cache operating mode). In the retaining mode, there is a "retaining" transaction which is the parent of the local transactions (in the nested transactions sense), and which keeps the updates visible only to its subtransactions and their descendants. When the retaining transaction commits, its updates become visible from its parent transactions, in other Workspaces.

Communication between Workspaces is based on the concept of Service. A service is a set of operations that can be executed by the server. The server publishes a service by calling a naming entity (this naming entity is replicated in the system). When a Workspace (or a other process) wants to use this service, it contacts the naming entity in order to suscribe to the service. Several servers can propose the same service. In this case, the naming entity chooses the appropriate server according to proximity and load balancing criteria.

2.2. Proposed services.

The Workspace can propose an external access to the objects which are visible in its objects space. This access is proposed as a Service which is published by the Workspace. From the point of view of a server Workspace, a service is an application which dialogues with the outside, according to a specific protocol. This trivializes the service implementation, so any client Workspace can publish services itself .

The services are able either to export objects to other Workspaces, or to directly execute operations on these objects. These two types of services respectively correspond to Data Shipping and to Query Shipping. The Workspace can also propose an intermediate service type, which combines the properties of both of them. We call this service type the Mixed Service.

The Data Shipping service exports to the client an image of the server Workspace objects space. Objects belonging to this image are still accessed through their identifier (or logical address) and through their interface, though these properties are not secured. This service is needed for the recursive definition of the objects space. It is used by the Workspaces which define their objects space by merging their database views with that of the server.

The Data Shipping service exports to the client an image of the server Workspace objects space. This service is needed for the recursive definition of the objects space. Workspaces use it to define their objects space by merging their own databases with database image exported by the server.

When using this service, the client Workspace imports objects through their identifier (or logical address). It only imports the objects which are needed by its transactions. If the client is a passing Workspace, a transaction send back updated objects at commit time. If the client is a retaining Workspace, updates are maintain in the client object space until the client commits globally all updates contained in its objet space. The Data Shipping service implements the call-back of locks [6], when the client caches locks for transaction sequences.

The Query Shipping service executes on the server all operations which are requested by the client. This approach unloads heavy processing from the client when this one is not adapted to data transfers (mobile terminals, "walkstations"). This service can also implement secured operations or operations on restricted data when the client is not reliable and may corrupt the database.

When the application needs secure processing, Query Shipping seems to be the only solution. However, we also propose a mixed solution which can be useful if a client want to use Query Shipping for some functionalities and Data Shipping for other operations, or if has access to some objects in the external view and not to all.

The Mixed service therefore proposes an intermediate approach which authorizes the client to read or update part of the objects which it previously imports (through Data Shipping), but which also asks the server to execute operations on part of the objects (through Query Shipping). These two objects sets may intersect. Therefore the Mixed service checks the consistency of data access by the client and the server. This consistency is implemented by the generic locking mechanisms.

The Mixed service is completed by a protection mechanism, which enables to know the access rights which are granted to the non-secured client. This mechanism differentiates the authorizations which are needed to process objects on a client or on a server.

3. Architectures for Persistent Object Managers.

The Workspace model enables to implement classical Client-Server architectures, symetric distributed virtual memories such as SHORE, and Object Manager architectures which are adapted to heterogeneous networks including LANs and WANs. This section modelizes architectures which are mainly based on the publication of Data Shipping services between clients and servers.

The recursive definition of the object space enables the client of a service to re-publish this service. The introduction of intermediate levels in the service publication enables the information system designer to build multi-levels database architectures, which can adapt both to the software and hardware architectures of the information systems. The graph of service subscription is correct while there is one path only to access a persistent data (by Data-Shipping and by Query-Shipping) : two access paths would be considered as concurrent transactions by the data server.

Client-Server structure

In order to implement the classical Client-Server architecture, we define client Workspaces which do not have local objects, and server Workspaces, which activity is limited to Data Shipping services on their local objects. When a client wants to access all database objects, it suscribes to all these services. A major inconvenient of this architecture is that the transactions two-phase commit is coordinated by the client, which is not secured and possibly less reliable. In the figure 1.a, we present a Workspace configuration which implements the Client-Server architecture.

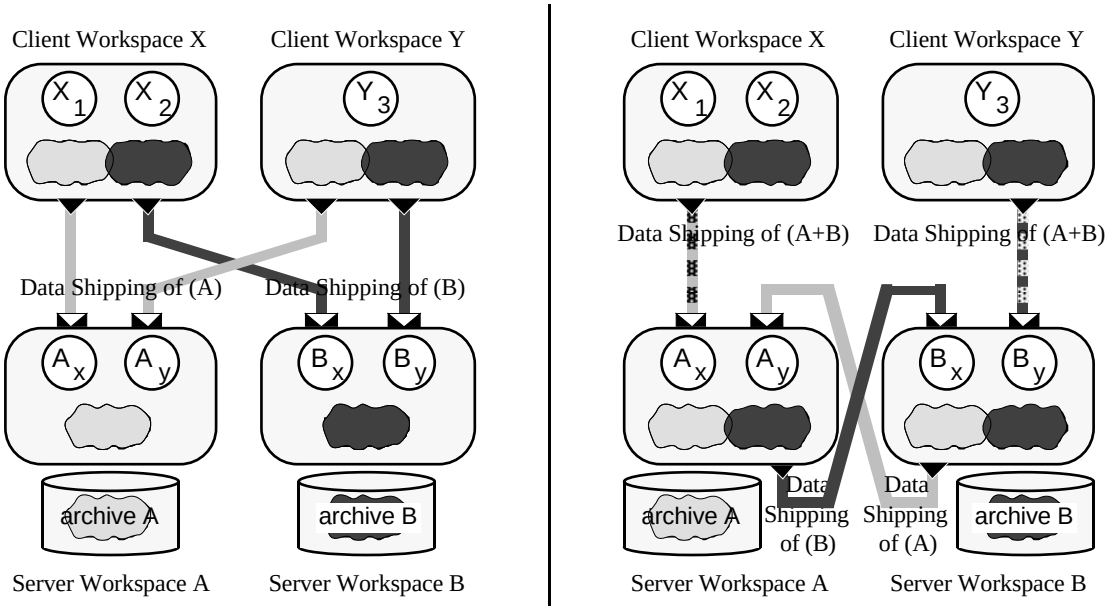
Symetric structure

In a symetric structure such as that of SHORE, each server serves a global view on its local data and on data which are managed by other servers. In the Workspace model, this architecture is implemented by using two types of services. One of them exports local data towards other servers, the other one exports a merge of the various local views towards clients. A client subscribes to the merge service from one server. This server controls the 2 Phase commit of the client transactions. The figure 1.b presents a Workspace configuration which implements a symetric architecture.

The solution based upon the Workspace model is more secure than that of SHORE. In SHORE implementation indeed, the client and the server are two processes on the same machine, which limits the server security. When using the Workspace model, the client and the server can be on two different computers.

Heterogeneous networks

In heterogeneous networks, several LANs are connected by one or several WANs. It is desirable to minimize the volume of communications on the WANs. This can be done in a classical way by using a front-end which concentrates the communications, and implements a gateway between the LAN and the WAN.



(a) Client-Server architecture **(b) Symetric architecture**

figure 1 : Applications of the Workspace Model

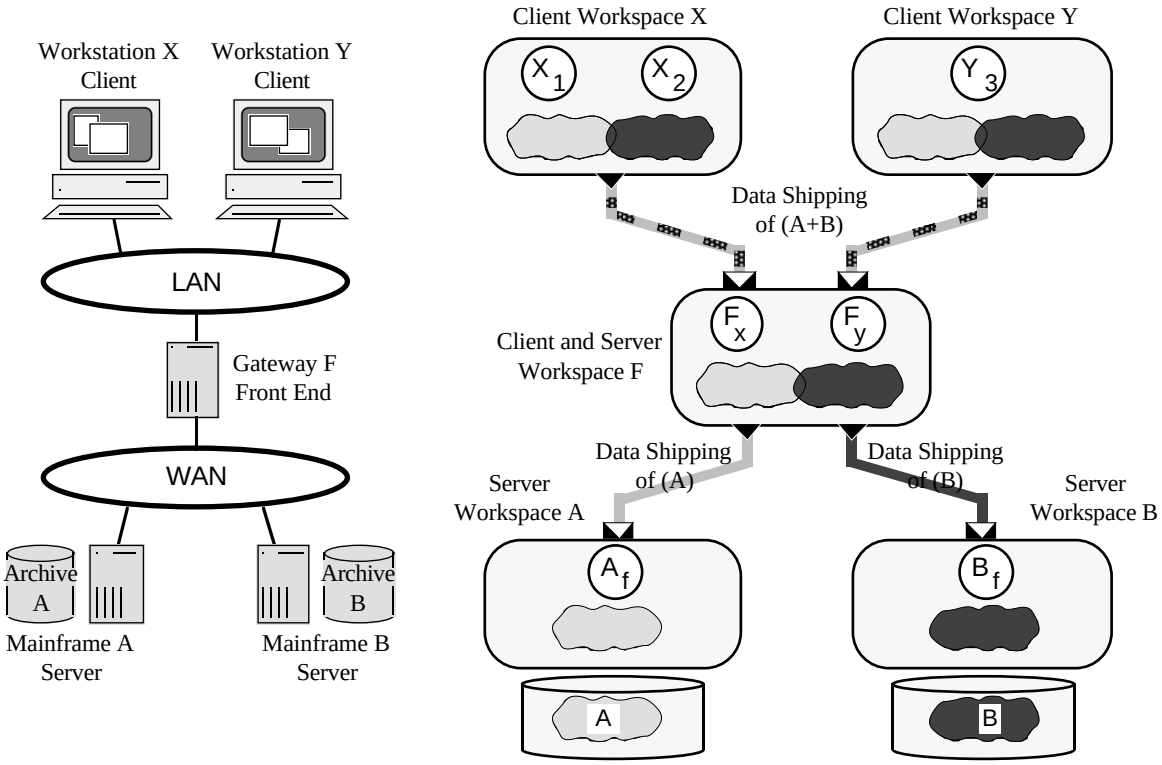
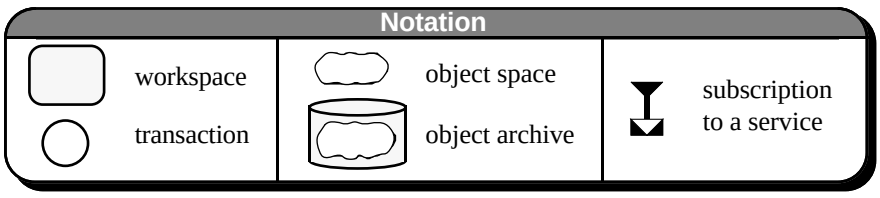


figure 2 : Applications of the Workspace model to heterogenous networks



In the Workspace model, this gateway is implemented by a retaining Workspace. The retaining property of the Workspace transaction enables it to group the updates which are transmitted through the WAN, at the commit time of the retaining transaction. During this transaction, the Workspace caches the objects accessed through the WAN to the benefit of the client transactions.

In figure 2.a, we present the hardware architecture of a heterogeneous network. In this figure, two LANs are connected by a WAN through a gateway F. Figure 2.b presents the corresponding Workspace configuration. Client Workspaces X and Y map on workstations X and Y. Server Workspaces A and B map on mainframes A and B. Workspace F, which is both client and server, is placed on gateway F.

Other applications.

The Workspace model also enables to implement replicated databases and cooperative services. Replicated databases are based on a replication service which is an extension of the Data Shipping service. This service replicates the updated objects on several copies of the local volumes. Cooperative services use versions and notification mechanisms to reduce the isolation of cooperating transactions. Due to space limitations, we shall not present these utilisations in this paper.

4. Conclusion

The Workspace model is the result of a research on Objects Managers which took place in MASI Laboratory and in industry since 1991. The initial idea was proposed by Eric Abécassis in 1993. It was then the subject of many discussions within the RAPID team at MASI Laboratory, and was then the target of two implementations. One of them, called YOODA, is an industrial project which is used as a data repository for Geographical Data Base Systems [7]. The other implementation, called WEA, is a research vehicle for the study of performance issues and of cooperative applications [3,4].

These implementations use present functionalities of modern operating systems. MultiThreading enables to implement asynchronous services, and to use several forms of parallelism which are available on a computer (processing, I/Os, etc...). Memory Mapping is used as a technique to easily access persistent data. The application interface is the C++ language. This interface is used by the developer for transparent management of persistency and concurrency.

References

1. M.J. Carey, et al, "Shoring Up Persistent Applications", Proc. of the 1994 SIGMOD Conf., Minneapolis, Minnesota, May 1994
2. J.E.B. Moss, "Nested Transactions: An Approach to Reliable Distributed Computing", Boston, MIT Press, 1985.
3. D. Donsez, P. Homond, P. Faudemay, "WEA, A Distributed Object Manager based on a Workspace Hierarchy", Proc. of IFIP Conf. on Applications in Parallel and Distributed Computing, Caracas, Venezuela, April 1994, pp247-256.
4. D. Donsez, P. Homond, P. Faudemay, "A Cooperative Database System based on Workspace Hierarchy", Int'l Conf. Codata, Chambéry, septembre 1994
5. J. Gray, "The Transaction Concept : Virtues and Limitations", Proc. of 7th Int'l Conf. on Very Large Data Bases, Cannes, September 1981, 144-154
6. Y. Wang, L.A. Rowe, "Cache Consistency and Concurrency Control in a Client/Server DBMS Architecture", Proc. of the 1991 SIGMOD Conf, pp367-376.
7. E. Abécassis, "YOODA: Handling Distribution Through OODBMS", submitted for publication