

A Model of Location Transparency

Katrina E.K. Falkner

Distributed High Performance Computing Group,
Department of Computer Science, University of Adelaide,
Adelaide, SA, 5005, Australia
14th February 2000

Email: katrina@cs.adelaide.edu.au

Abstract

Location transparency removes the need for client objects to explicitly know or define the location of a server object when communicating. If a server object is capable of migration, relocation transparency maintains reference validity throughout the migration.

Several models for providing location transparency exist, including the home location, forwarding locations and Stub-Scion Pair chain models. This paper proposes a model that uses a distributed registry system and dynamic reference updating to provide location and relocation transparency. A comparison of the models is presented.

1 Introduction

A mobile object is an object that is capable of moving, or being moved, readily from one place to another. Mobility within a distributed object system allows mobile enabled objects to move freely between nodes within the distributed system, supporting load balancing (by moving objects from heavily loaded nodes to lightly loaded nodes), fault tolerance (by moving objects to new nodes in response to node partial failure), locality of data access (by moving objects to the vicinity of the data they require) and scheduling and monitoring of the distributed system (by allowing monitoring objects to move about the distributed system assessing load and scheduling requirements).

One of the most important issues when dealing with mobility is that of locality and reference management [MDP⁺99, Sal78]. Objects within a distributed object system may have references to other objects within the system; when these referenced objects move, it is important that any references still be valid and that any referencing object always be able to contact the mobile object regardless of its location.

An object reference that does not expose the location of the referenced object is said to be location transparent; a location transparent reference can access a remote object in the same manner regardless of the remote object's current location or state. Location transparent references have been implemented in several distributed object systems [Hen98, Her99, KT95] and allow distributed communication to be programmed without exposing or requiring location specification. Transparency decreases complexity in distributed programming but additionally incurs communications overhead as references must be resolved or evaluated to a direct reference to the remote object.

An object that can change location without other objects noticing is termed relocation transparent. In a relocation transparent system, a client does not have to explicitly update its references in any way when its referenced object moves; all relocation is performed transparently.

This paper describes a new model for location and relocation transparency that utilises a distributed registry system and discusses the benefits of this model in comparison to other, commonly used models for location and relocation transparency.

In Section 2 a brief description of commonly used models for location and relocation transparency is provided. Section 3 describes the proposed model, while in Section 4, a comparison of the new model to existing models is provided. Section 5 closes this paper with a summary and outline of current work on implementing the model.

2 Existing Models

Several solutions to the problem of maintaining location transparency within a distributed object system have been suggested. Mechanisms have been proposed based on the concept of a home location; this home location is updated whenever an object moves and serves as the well known contact point. The home location model has been used to support location transparency in [BW89], [DO91] and [CRS96].

One common extension of the home location idea is to use a mobile home location [LH89]. This results in a chain of homes or forwarding locations pointing from the original home location to the current location of the mobile object. A client reference may point to any link in the chain. The forwarding location model has been used in [AF89], [AP95] and [Her99].

A fragmented object [MGNS91] model is often chosen as the distribution model for a distributed object system. A fragmented object is one with several components that may be distributed over nodes within a distributed system, while appearing to be a single entity to external clients. A fragmented object consists of several types of components, including client interface fragments, communications enabling fragments and group interface fragments, which provide a common interface between fragments.

A further alternative used to manage location transparency, and an example of a fragmented object model, is that of Stub-Scion Pair (SSP) chains [SDP92]. A stub is maintained as the object reference on the client side and a scion is maintained for each stub on the server side; a SSP is produced for each object reference created. When an object reference is moved, a new SSP is created, resulting in a chain of SSPs through which the correct reference can be obtained.

What makes the SSP chain technique differ from that of forwarding locations or a home location is that a stub within a stub-scion system may have multiple references to the referenced object. A stub will contain a strong locator reference, which will always lead to the destination object, and potentially multiple weak locator references that are not guaranteed to complete but contain shorter paths to the destination object. A weak locator may be returned as part of an invocation through a strong locator reference as a direct path for future use. SSP chains have been used to support reference management in [SDP92] and [Bag99].

A central registry has also been proposed by systems such as Gardens [RS99] which treats a distributed system as a closely linked parallel system. Each reference is treated as if part of a shared memory system where offsets are managed as part of reference access; this causes additional overhead but takes away any required reference updating and also any naming requirement. However, this type of system is only suitable for a restricted set of distributed object systems. V-System [MMTC85] introduces a search based mechanism where references are constructed of (logical host id, local index). When an object moves, the logical host is duplicated and then set to the new physical host address. As existing references become invalid, their reference in the mapping cache becomes invalid and a request is broadcast to the network for the new logical host id.

3 A New Model

A distributed Object Request Broker (ORB) system has been developed to support the Distributed Information Systems Control World (DISCWorld) Metacomputing Environment [HJPV98]. An ORB system is a form of distributed object system that utilises an intermediate well known broker object to provide and manage references on a small scale. This ORB integrates a sophisticated naming model and distributed directory service with support for mobile services. The DISCWorld ORB system provides mechanisms for clients within the DISCWorld system to register services, obtain references to services and perform optimised communications.

Location and relocation transparency are provided through the use of a distributed naming service. A fragmented object model is used, with some location dependent information kept within the client interface fragment as cached data only. When this data becomes invalid, new location information can be obtained from the distributed naming service.

Three types of reference are recognised within the DISCWorld ORB system:

- Connected: references given to a client and currently being actively used.
- Unconnected: references given to clients but currently unconnected to a service.
- Unknown: references yet to be handed out.

Stub-scion chains incur the same cost as the forwarding location model for the first invocation, however a shorter path can be returned or piggy-backed with the invocation response to produce a direct reference (consisting of a single stub-scion pair) for future usage.

The model proposed within this paper makes use of a fragmented object model with migration fragments existing only for connected references. These migration fragments do not act as a chain, a single migration fragment is used as a direct reference; migration fragments also allow updated location message to be forwarded to the client. Unconnected references can update their cached location hints by contacting the registry system. It is expected that the time between the provision of a reference and its connection will be small (minimising relocation possibility) and that this case will be uncommon.

Where relocation is common but client access is rare, the home location model does not suffer from bottleneck issues. When the scale of distribution is small, the home location model is ideal as the effect of tromboning is also removed.

In the case where relocation is uncommon but client access is frequent, the forwarding location model and more so the stub-scion chain model can be efficient as the length of the chain in each case will be small.

The proposed model is more suited to the case where relocation is common and client access is frequent. References can be updated transparently for connected clients without an increasing chain of forwarding locations and due to the distributed nature of the registry system and direct referencing, bottleneck issues are reduced. All models benefit from a smaller scale of distribution due to the reduced latency costs and reduction of any potential trombone effect.

5 Summary

Location management using a home location or forwarding location chains are commonly used methods in both the areas of mobile object systems and mobile host systems. These methods can be inefficient as an extensive chain of forwarding locations can develop for a forwarding method; and additional latency can be introduced into communications when a client has to contact a distant home location to obtain a reference to a potentially close server object.

This paper proposes a model for location transparency, which relies on a distributed registry system and a fragmented object model to allow relocation of unconnected references; connected references are updated by a client interface fragment which stores piggy backed relocation information. This model has been implemented as part of the DISCWorld ORB system, along with implementations of the commonly used models described in this paper.

The use of a distributed registry system provides location and relocation transparency without the need for a well known home location or residual code objects in previous hosts.

6 Acknowledgments

This work was carried out under the Distributed High Performance Computing Infrastructure Project (DHPC-I) of the On-Line Data Archives Program (OLDA) of the Advanced Computational Systems (AC-Sys) Cooperative Research Centre (CRC) and funded by the Research Data Networks (RDN) CRC. ACSys and RDN are funded by the Australian Commonwealth Government CRC Program. Thanks to K.A. Hawick and M.J. Oudshoorn for useful discussion on the work presented here.

References

- [AF89] Y. Artsy and R. Finkel. Designing a Process Migration Facility: The Charlotte Experience. *Computer*, 22(9):47–56, September 1989.
- [AP95] Baruch Awerbuch and David Peleg. Online Tracking of Mobile Users. *Journal of the ACM*, 42(5):1021–1058, September 1995.
- [Bag99] Aline Baggio. Adaptable and Mobile-Aware Distributed Objects. PhD thesis, Université Pierre et Marie Curie, Paris, June 1999.
- [BW89] A. Barak and R. Wheeler. MOSIX: An Integrated Multiprocessor UNIX. In *Proceedings of the USENIX Winter 1989 Technical Conference*, pages 101–112, February 1989.

- [CRS96] K. Mani Chandy, Adam Rifkin, and Paolo A.G. Sivilotti. A World-Wide Distributed System Using Java and the Internet. In Proc. Fifth IEEE International Symposium on High Performance Distributed Computing, August 1996.
- [DO91] F. Douglass and J. Ousterhout. Transparent Process Migration: Design Alternatives and the Sprite Implementation. *Software - Practice and Experience*, 21(8):757–785, August 1991.
- [Hen98] Michi Henning. Binding, Migration, and Scalability in CORBA. *Communications of the ACM*, 41(10):62–71, October 1998.
- [Her99] Maurice Herlihy. The Aleph Toolkit: Support for Scalable Distributed Shared Objects. In Proc. CANPC'99, pages 137–149, 1999.
- [HJPV98] K.A. Hawick, H.A. James, C.J. Patten, and F.A. Vaughan. DISCWorld: A Distributed High Performance Computing Environment. In Proceedings of High Performance Computing and Networking (HPCN) Europe '98, Amsterdam, pages 598–606, April 1998.
- [KT95] Micheal M. Kong and David Truong. DCE Directory Services. *Herlett-Packard Journal*, pages 23–27, December 1995.
- [LH89] Kai Li and Paul Hudak. Memory Coherence in Shared Virtual Memory Systems. *ACM Transaction on Computer Systems*, 7(4):321–359, November 1989.
- [MDP⁺99] Dejan Milojević, Fred Douglass, Yves Paindaveine, Richard Wheeler, and Songnian Zhou. Process Migration. Technical Report HPL-1999-21, Computer Systems Laboratory, HP Laboratories Palo Alto, February 1999.
- [MGNS91] Mesaac Makpangou, Yvon Gourhant, Jean-Pierre Le Narzul, and Marc Shapiro. Readings in Distributed Computing Systems, chapter Fragmented Objects for Distributed Abstractions, pages 170–186. IEEE Computer Society Press, 1991.
- [MMTC85] Keith A. Lantz Marvin M. Theimer and David R. Cheriton. Preemptable Remote Execution Facilities for the V-System. In Proceedings of the 10th ACM Symposium on Operating System Principles, pages 2–12, December 1985.
- [RMR99] Karunaharan Ratnam, Ibrahim Matta, and Sampath Rangarajan. A Fully Distributed Location Management Scheme for Large PCS. Technical Report BU-CS-99-010, Boston University, August 1999.
- [RS99] P. Roe and C. Szyperski. Transplanting in Gardens: Efficient Heterogeneous Task Migration for Fully Inverted Software Architectures. In Proceedings of the 4th Australasian Computer Architecture Conference (ACAC'99), Auckland, New Zealand, January 1999.
- [Sal78] J.H. Saltzer. *Operating Systems - An Advanced Course*, chapter Naming and Binding of Objects, pages 99–208. Springer-Verlag, 1978. Also in *Lecture Notes in Computer Science*, volume 60.
- [SDP92] Marc Shapiro, Peter Dickman, and David PlainFossé. SSP Chains: Robust, Distributed References Supporting Acyclic Garbage Collection. Technical Report 1799, INRIA, Rocquencourt, France, November 1992.