

Achieving High Performance Through Middleware Reconfiguration

Shikharesh Majumdar¹, Imran Ahmad², E-Kai Shen³, Istabrak Abdul Fatah³

1. Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada.
2. Mitel Corporation, Kanata, Ontario, Canada.,
3. Nortel Networks, Ottawa, Canada, **Email- majumdar@sce.carleton.ca**

Extended Abstract

CORBA has proven its worth in low speed networks that do not demand high system performance. Although CORBA compliant Commercial-Off-The-Shelf (COTS) middleware products provide inter-operability, most of them often incur a performance penalty. Although It provides a flexible solution for a heterogeneous environment, the overhead incurred in providing inter-operability adversely affects the scalability, bandwidth and latency of the system that are crucial for many embedded and performance sensitive systems. Our research focuses on reconfiguring middleware for achieving high performance. Two projects are currently underway: (1) adaptive client-agent-server architecture and (2) path reconfiguration using “flyovers” for systems with limited heterogeneity. Work in progress in both of these research topics is summarized in this position paper.

Adaptive Middlewrae Architecture

The client-agent-server interaction architecture is observed to have a strong impact on system performance [ABD-98]. These architectures refer to the way a request from the client is routed to the server and a reply from the server is sent back to the client. In a Handle-Driven (H-ORB) architecture when a client wants to request a service it sends the server name to the middleware agent that performs a name to handle mapping and sends an IOR or handle back to the client. The client uses the handle to contact the server and receive the desired service. A number of Commercial-Off-The-Shelf ORB's that include Visibroker (previously known as ORBeline) and Orbix-MT used in this research use such a handle-driven architecture. In a Forwarding (F-ORB) architecture the client sends the entire request for service to the agent that locates the appropriate server and forwards the request to it. The server performs the desired service and sends a response back to the client. In a previous research project each of these architectures was observed to be the best performer for a specific workload condition [ABD-98]. Our current research combines the good attributes of these two architectures into an *Adaptive ORB (A-ORB)* [SHE-99] that is expected to perform well for a broad range of workload parameters. The research results are expected to be most useful in the context of systems in which every time a service is required a client contacts the ORB agent for a handle to an appropriate server object instead of storing and reusing the handle used earlier. Such an approach is adopted when it is desirable: (1) to effectively use the load balancing provided by a number of commercial products that steer multiple requests for a particular service to different instances of the same object, (2) to handle server crashes in a manner transparent to the client by letting the agent route the requests arriving after a crash to a live server instance, (3) to allow the agent to choose a server that is most appropriate for a request (based on the QoS requirement of the request for example). Depending on the system load the interaction architecture switches between a handle driven and a forwarding mode. When a client request arrives at the agent, the request is forwarded to the server if the number of requests waiting to be forwarded is lesser than a threshold value QL; a handle is returned to the client otherwise. The intuitive motivation behind this approach is briefly described. At low system load the forwarding mode of operation may be desirable because it uses a lesser number of messages per method invocation in comparison to the handle driven mode. When system load increases more queuing starts occurring at the agent and the servers. Since a synchronous method of communication is used, returning the handle back to the client can reduce the request waiting time and congestion at the forwarding queue of the agent which in turn improves system performance. A performance prototype for the A-ORB is developed in C++ using Orbix-MT. The measurement environment consists of a network of Sun workstations inter-connected by a 10 Mbps hub and running Solaris 2.6. The important contributions of the research on adaptive ORB's are briefly summarized.

- A significant benefit in performance can be achieved under various workload conditions by using an adaptive hybrid ORB architecture that changes its behavior and adapts to changes in system load. A 100% improvement in

throughput over a purely handle-driven or a purely forwarding ORB is observed for example, under certain workload conditions.

- An important observation is that even without having access to the source code of a COTS middleware product it is possible to use it for devising an adaptive ORB. An even higher system performance can be expected if changes to the ORB source code is possible.

The outstanding research questions include: (a) how to make the dynamic change in architecture totally transparent to the client so that the application programmer does not need to provide the facility for receiving the result of a method invocation from multiple sources (agent or server), (b) how to determine the threshold values to be used for a given system, (c) in addition to the length of the forwarding queue at the agent should other criteria be used for changing the mode of operation (d) whether the adaptive architecture is attractive in the context of other CORBA services such as the trading service.

Path Reconfiguration on Systems with Limited Heterogeneity

Although heterogeneity in distributed systems is natural, most distributed systems are characterized by a limited heterogeneity. For example, a subset of the nodes in every distributed system is likely to be similar. A number of inter-communicating objects in an application is often implemented using the same programming languages on top of the same operating system. The components that are diverse can use the CORBA ORB whereas components that are built using the same programming language and operating systems can use a "flyover" and bypass a number of standard CORBA operations and save in terms of middleware overheads. Results of preliminary research that report on the performance improvements that can accrue from using the flyover between a similar client-server pair are described in [AHM-99]. The important contributions of the research are three performance optimization techniques that perform such bypass operations and are briefly summarized.

- The first technique prevents data conversion between the native data types used by system components and the standard data format (CDR) specified in CORBA.
- The second technique saves communication time by removing the padding bytes used in between two data elements in a CORBA message that is exchanged between the nodes hosting the client and the server.
- The third technique shrinks the message sent by the client by compressing the message header thus reducing the communication overhead.

Note that the performance enhancement techniques are to be incorporated into the middleware and are transparent to the application program. We have used two approaches for incorporating the optimization techniques in an open source COTS product called ORBACUS: (1) modification of source code, (2) interceptor programming. Experimental investigations on a network of Pentium II PC's connected by a 100 Mb/sec Ethernet running under Linux indicate that these performance optimization techniques can lead to a significant improvement in system performance. The applicability of each technique depends on the type and size of data being transferred.

So far we have measured the benefits of the performance optimization techniques in isolation. The outstanding research questions include (a) how to develop a system that can dynamically switch between the two paths: the flyover and the regular CORBA ORB (b) how to select the appropriate technique(s) for a given method invocation (data transfer) (c) to understand the performance implications of the reconfigurable system under higher system load with multiple CORBA method invocations competing for system resources (d) to develop other performance enhancement techniques.

References:

[ABD-98] Abdul-Fatah, I., Majumdar, S., "Performance Comparison of Architectures for Client-Server Interactions in CORBA", *In Proceedings IEEE 18th International Conference on Distributed Computing Systems (ICDCS'98)*, Amsterdam, May 1998, pp. 2-11.

AHM-99] Ahmad, I., Majumdar, S., "Achieving High Performance in CORBA-Based Systems with Limited Heterogeneity", Research Report 99-06, Dept. of Systems & Computer Eng., Carleton University, Ottawa, Canada, November 1999 .

[SHE-99] Shen, E.K.-, Majumdar, S., "Performance of Adaptive Middleware for CORBA-Based Systems", Research Report 99-07, Dept. of Systems & Computer Eng., Carleton University, Ottawa, Canada, November 1999.