Tool to Evaluate Performance in Distributed Heterogeneous Processing

J.A. McCann, K.J. Manning

HiPeX, Interoperable Systems Research Centre
CITY University, LONDON, EC1V 0HB, ENGLAND

Abstract

The next generation computing architectures will consist of a fusion of heterogeneous systems interoperating. This concept is gaining popularity, yet there are currently few tools available for performance visualisation and management of heterogeneous interoperating systems using middleware.

This paper looks at the design of middleware performance tools, using our current project as an example, focusing on information systems processing. We discuss the issues such as performance metrics (obtained from a user market study), raw performance data extraction, its analysis, and finally performance presentation and visualisation.

1. Interoperable Environments and Middleware

Today's enterprises consist of a decentralised collection of heterogeneous systems. In an attempt to upgrade older systems they have introduced more modern (e.g. Relational or OO) DBMS running on non-mainframe; workstation and PC based architectures. Business critical legacy systems have been a heavy investment and are practically impossible to rewrite. Consequently, there is a fundamental business need for interoperability as a cost effective way to use these systems, or migrate from legacy systems to the more modern.

Michael Brodie's defines interoperability as *the ability to interact effectively to achieve shared goals* [7]. There has been a notable increase in interest in interoperability over the past few years, with a significant number of London financial institutions investing in some form of middleware. This interest will increase as more applications use the internet - aided by implementations of CORBA incorporating internet hooks. Consequently, many researchers and industry counterparts share our vision of Co-operative Information Systems which they deem as being the next generation of information systems, emerging over the next few years [7].

The integration of heterogeneous systems are not new to the research community. Work on Federated DBMS have been around for some time [20]. The issues concerning interoperability are not the focus of this paper, but for clarity the reader may reference the many papers which discuss these issues [7,8]; and some which discuss different interoperability mechanisms: *Object Equivalencing* [14], *Mediation* [15], *KBS Assist* [19], *Orb based Intelligent Co-operating Systems* [16,22,4,1,18] and *Com based object systems* [9].

However, this paper will concentrate on the design and development of performance tools for interoperable systems using our present project as a case study. The Iriss project is a continuing Eureka project which has evolved from a project which focused on distributed parallel database systems simultaneously running decision support and on-line transaction processing processes [23]. The distributed systems in the previous project made use of the middleware framework, Clearinghouse, to provide interoperability. It is for this reason that this paper will focus more on Clearinghouse; a message based framework, rather than CORBA object based technology, to describe the issues concerning the design of performance tools. Having said this, the authors are familiar with CORBA and bear it in mind when drawing conclusions. Therefore, it is our belief that the techniques used provides useful insight into performance tool design for all middleware systems.

In this paper we state the case for middleware performance tools which is backed up by a user market study. We then explain the general issues of tool design, focusing on performance metrics and performance tools embedded in middleware. Finally, we briefly outline implementation issues and discuss the question of performance visualisation.

0-8186-8332-5/98 $10.00 © 1998 IEEE

2. Support for Middleware Performance Tools

Large heterogeneous intercommunicating systems inevitably yield larger and more complex performance data than before. That is, the performance of not only differing architectures and applications has to be reported, but the meaning of performance metrics may change within an application type (e.g DBMS). Middleware, such as implementations of CORBA, already support architectural neutrality using opaque objects. A useful performance tool therefore, must capture this complexity, without making the tool more difficult to use.

It is our contention, therefore, performance tools imbedded in the middleware layer assists performance management of heterogeneous systems on two fronts. The first, and most obvious, lies in the fact that there is performance overhead caused by the middleware itself. Consequently, the areas of poor performance in the middleware must be highlighted. Initial studies of CORBA based middleware have already highlighted three main causes of poor middleware performance. They are, ineffectual memory management, data conversion and data copying [23]. Secondly, as the middleware is neither application or architecture dependent we can produce performance metrics which are specific to heterogeneous system operation, measuring the system as a whole. It could be argued that network performance monitors can provide some of this functionality already. Nevertheless, they are concerned only with communication metrics. Using middleware, we can provide a much richer and generic set of performance metrics which we will demonstrate later.

To examine the state of the art in this area we undertook a study to establish what management services are being provided by current vendors of middleware. On the whole we found that no such utilities excisted for general management of heterogeneous systems, much less performance management. Some systems did provide debugging facilities. For example, Verilog's GEODE is a case development and program generation tool which provides a simulator which can perform formal verification and validation, detecting deadlocks and livelocks [10]. As this system only debugs distributed system design; it has no performance tools. On the other hand, we did find one system which provided some on-line system management. The DOME system; an ORB based distributed object management environment, includes DV; a Debugger and Visualisation tool, which provides an animated view of the traffic through the system, signalling interesting events. While this information is highly important in the management of an interoperable system's functionality, they still do not provide any timing information [12]. Related to this, we have found that less generic frameworks, (those which allow different relational DBMS to communicate) currently provide the best management tools for restricted heterogeneous environments. This is discussed more in the market survey section.

3. Performance Tool Design

When designing a performance tool, to monitor middleware or any other system, there are a number of issues to take into account. These are: the *cost of analysing* the system, which is helped or hindered by how *accurate* and *visible* its output is, and how *intrusive* the tool is. Three stages make up a performance tool: the instrumentation stage, the analysis stage and the presentation stage. For each of these stages there is an important underlying philosophy which must address the issues of portability, scalability and extendibility[3]. For example, if we focus on the issue of scalability and relate it to each of the stages, we can see that:

- Gathering raw performance data should not add significant or disproportionate perturbation while monitoring the system as the system size (e.g. number of nodes or volume of processing) increases.
- Likewise, the analysis stage should not disproportionately bottleneck the performance tool itself as the system size increases.
- Furthermore, mapping performance results in a visual way should be able to effectively represent the growing system. (e.g. if colour is used to represent each node in a system, as soon as that system becomes greater than 10 nodes the colours become less distinctive and that feature becomes less useful.)

To enhance our tool design we carried out a quick user survey. This is described below.

4. Market Survey

With the lack of middleware performance measurement support to base our work on, we carried out a market survey to find out what *real world* users used to report the performance of their current systems, and what they thought was their relative strengths and weaknesses.

The thirty sets of users sampled for this quick study were immediate colleagues in industry and users on the internet. Overall the internet users were not that forthcoming in their replies. However, we did manage to communicate with some users who had just finished carrying out a review of a number of monitoring tools for procurement purposes. The users listed the advantages and disadvantages of tools they used to measure performance throughout their organisation. Those users which aimed to introduce interoperability, then listed the performance metrics which they would ideally require if their systems were integrated with middleware. There was 13 DBMS performance tools mentioned in the study, with two being

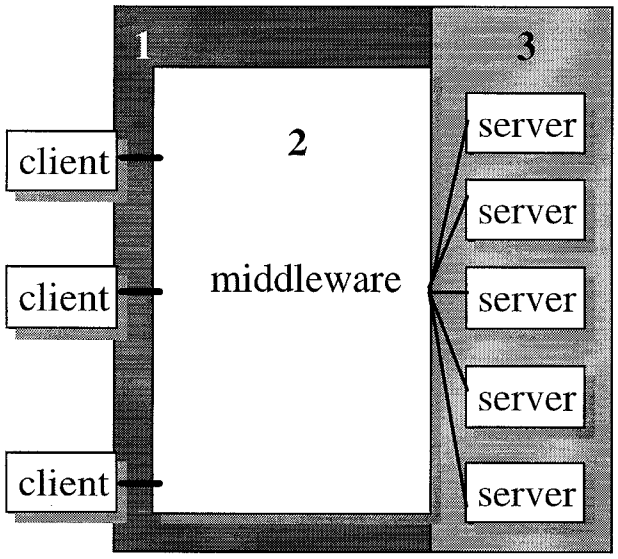


Figure 1. Three levels of Performance Metrics.

favoured by a significant number of users [21,13]. This is mainly because these tools are part of a fixed framework which allows heterogeneous relational database systems to communicate, and consequently these utilities provided more system wide performance metrics. This further demonstrated the need for such global performance information. From these results we have drawn up a list of common desirable functions which the users liked or would have liked to see in the ideal performance tool operating in an even more heterogeneous environment. These are briefly summarised below:

- Ability to simply define rules and trigger events
- Triggered action must send message to appropriate user(s).
- Ability to monitor complete system on a single console.
- Ability to have a number of layers of abstraction.
- Ability to use history information to do post processing analysis.
- Hotspots.
- Deadlock in middleware

5. Middleware Tool Design

Taking the general requirements described in sections 4 and 5 into account, our objective is to design a middleware performance tool which helps to pin-point performance problems in a heterogeneous interoperating system. To do this we view the system at three levels. At the highest level the system is treated as a complete unit, reporting metrics concerning the operation of the system as a whole. At the next level we wish to know how the middleware is effecting the performance of the complete system in detail. Therefore, the second level is middleware specific to, e.g. Clearinghouse or a CORBA implementation. Finally, the lowest level produces statistics of each client application, e.g. DBMS specific performance metrics. This is illustrated in figure 1 and described in more detail below.

At the top level the middleware is effectively a black box. This means that, at this level, the user is unaware and is disinterested in what is going on within the middleware. Likewise, they are disinterested in what is happening in servers. If a user should detect a performance problem in the middleware, this information is found at the second level which is highly dependent on the middleware system being used to provide interoperability. This may describe an implementation of CORBA (e.g. Orbix) or provide performance statistics on Clearinghouse.

If the user wishes to be informed about problems in the servers they view the servers' own performance reporting tool, if available. This has the advantage in that each server has better knowledge of which performance metrics relate to that systems' operation. For example, a DBMS's performance tool will report on the number of pages, IOs and locking statistics which are applicable to that system. It is undesirable that the user simply calls the windows of other tools which may look and feel different from the main tool. It is more useful to build a performance tool which extracts off each servers' performance data and presents it using the standard used throughout the main tool.

5.1 Level One : General System Metrics

This views the system as a whole and therefore only those metrics which are applicable to general interoperating are reported. For clarity we have divided the metrics into timing and quantitative, these are described below:

Timing Metrics

- *Total Response Time*: total elapsed time between the client issuing the enquiry and receiving the results.
- *Client-Middle Response Time:* elapsed time between the client issuing the enquiry and the middleware receiving it.
- *Middle-Server Response Time:* elapsed time between the middleware issuing the enquiry(ies) to the server and the server returning the result.
- *Middleware Response Time:* elapsed time spent in processing the enquiry in the middleware layer

Quantitative Metrics

- *Middleware Throughput:* average number of enquiries serviced per unit of time.
- *Number of Clients:* number of clients active in the system at that time
- *Number of Servers*: number of servers active in the system at that time

- *Messages to Middleware:* average number of messages received by the middleware from the client.
- *Messages to Servers:* average number of messages sent by the middleware to the servers.
- *Server Hotspots:* server which is accessed the most.
- *Fanout:* the average number of servers activated by a single client.
- *Deadlock Count:* number of deadlocks detected in the middleware over time.

5.2 Level Two: Middleware Metrics

These are the performance figures relating to the operation of the middleware itself. This is very specific to both the middleware paradigm and the implementation of that paradigm itself. For example if we were to use CORBA implementation we many wish to report the following performance metrics[5]:

- remote transfer time
- buffer statistics.
- time in presentation layer (marshalling/demarshalling)
- average ORB response time
- average Inter-ORB time
- Number of blocked clients waiting on server
- Average static invocation time, Number of static invocations
- Average dynamic invocation time, Number of dynamic invocations.

In the Iriss project we are using Clearinghouse middleware therefore to continue our discussion we present our second level metrics specific to Clearinghouse only.

6. Clearinghouse and Performance Metrics

The Clearinghouse data management system provides dynamic and intelligent systemwide interoperability[6]. It is the *intermediation* layer in a heterogeneous client-server architecture. The paradigm used is based on message-passing with the added provision of data storage facilities. This multi-functional approach gives three additional major benefits: it forwards updates from any client pro-actively to all other clients, it works bi-directionally supporting intermittent connection, and it provides automatic re-start facilities in the event of failure. Recently, Clearinghouse was used in legacy system migration without disruption to users.

Clearinghouse consists of a number of components, which are illustrated in figure 2. Users access the data sources via a Clearinghouse *Exchange*. Exchanges may be distributed and replicated, and all exchanges are managed

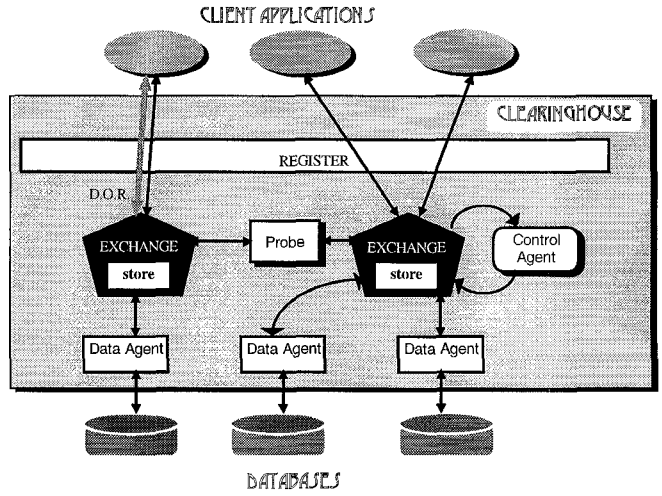


Figure 2. Clearinghouse Components.

by the *Registers*. Transactions are forwarded to the system in the form of *Data Operation Requests (DORs).* A *Control Agent* manages these DORs and queues them to the appropriate *Database Agent* for processing. The exchanges provide a single data model, with complex models requiring multiple exchanges connected by *Probes. Probes* control the passing of data between exchanges. The exchange provides caching facilities for improved performance. Based on this, below we present performance metrics monitoring and highlighting performance problems within the Clearinghouse system.

- number of Clients connected/disconnected to exchange
- number of registered exchanges (exchange IDs + mirrors)
- number of cloned agents (shared agents)
- number of aborts/rollbacks per exchange
- number of commits per exchange
- number of fail to check out errors and protocol errors
- number of DOR's queued to agents
- number of buffers per exchange and their sizes
- agent state (squirting, normal, idle)
- send messages queues
- failed requests
- suspended clients or requests (and length of time)
- physical read/writes
- cache hits/misses
- Triggers (e.g. Hotspots reached trigger point > ***n***%)

7. Three Stage Design

The implementation of this the performance tool can be described in three stages. Briefly, the *extraction* stage obtains the raw performance data. The *analysis* stage

takes this data and processes it, calculating statistics and outputting it in a format which is expected by the *presentation* stage, which in turn produces graphs, gauges, and charts for the user. It is this stage which we discuss below.

7.1 Performance Visualisation

The ability to analyse and understand data, representing more complex systems, relies more heavily on visual processing [1]. Performance visualisation can be defined as the use of graphical display techniques to present an analysis of performance data for an improved understanding of complex performance phenomena [2].

We looked at a number of recent performance tool projects which have a special focus on performance visualisation. PABLO is probably one of the more interesting systems [3]. It supports two types of data presentation: graphic and sonic. Xwindows present the graphical display, supporting Bar Graphs, Bubble Charts, Strip Charts, Contour Plots, Dials amongst many others. Interestingly, PABLO also represents performance data using sonification. That is, they make use of sound (duration, pitch, volume, timbre) and spatial location to present data aurally using MIDI. PABLO's researchers assert that the system could represent performance information 'in the same way a movie sound track conveys information complementary to the imagery'. Initially, we thought that this would be too noisy to work with all day and would appear as just a novelty feature. However, this idea has been exploited in some communications monitors [11]. Moreover, most modern workstations and PCs contain sound devices therefore sonic output, if it were subtle, could be very useful. For instance, PABLO has a set of *earcons* or audio warnings which uses sampled voice, enumeration's and alarms which activate when a threshold is reached. We hope to exploit these techniques further in this project.

8. Conclusions

The increased requirement for distributed interoperating processing has lead to an increased interest in middleware technology. When the middleware systems gain more popularity, the need to manage the resulting larger, intercommunicating systems will become much more apparent. Products based on standards, such as CORBA, are already encouraging the amalgamation of many heterogeneous systems. Subsequently, the timely design and development of management tools to real user requirements, will aid this process, preventing future problems.

A significant amount of the work described in this paper is presently undergoing implementation and has already proved to be useful in the Iriss project. The results of further experimentation will be presented in a follow up paper. In this paper we have aimed to show that a tool which monitors a distributed interoperating system is very useful but can soon become very complex. We have highlighted the many design issues and trade-offs which we came across using our case study with Clearinghouse, and showed how useful it was to include user requirements. Finally, we believe that our approach and the techniques we used can provide useful insight into performance tool design for all mechanisms which provide distributed interoperable system performance.

9. References

1. Universal Networked Objects, ENR Europe Limited, Expersoft Ltd., IBM Corporation, ICI, plc, IONA Technologies, SunSoft Inc., September 1994, OMG TC Document 94.9.32.
2. Malony A., Wilson G., 'Future directions in parallel performance environments', in Workshop on Performance Measurement and Visualisation of Parallel Systems.
3. Reed D., Aydt R., Madhyastha T., Noe R., Shields K., Schwartz B., 'The PABLO performance analysis environment', Department of Computer Science, University of Illinois at Urbana-Champaign, 1992.
4. Orbix Distributed Object Technology. Iona Technologies Ltd., Ireland.
5. The Common Object Request Broker (CORBA): Architecture and Specification. Object Management Group, document number 91.12.1, Revision 1.1, 1992
6. The Clearinghouse Data Management System. Strand Software Technologies Ltd., UK.
7. Brodie M.L., 'The promise of distributed computer and the challenges of legacy information systems'. Proc. IFIP TC2/WG2.6 Conference on Semantics of Interoperable Database Systems, Australia, eds. Hsiao D., Neuhold E.J., Sacks-Davis R., Elsevier North Holland, Amsterdam 1993.
8. Ventrone V., Heiler S., 'Some Practical Advice for Dealing with Semantic Heterogeneity in Federated Database Systems'.
   URL: http://info.gte.com/pub/dom/reports/VENT93.ps
9. Redmond W.A., 'Object Linking and Embedding Version 2 (OLE 2), Programmers Reference', Volumes 1 & 2, 1993.
10. Geode Technical Presentation, VERILOG SA, France, document reference C/GXXX/TA/100/436, 1994.
11. Evision: Ether Vision LAN Monitor. Triticom Ltd., 1994
12. DOME: Distributed Object Management Environment, Object-Oriented Technologies Ltd., Uk.
13. EcoTools: Integrated Systems Management for the Client/Server Data Center. Compuware Corporation, USA, URL: http://www.compuware.com/
14. Manola M., et al, 'Distributed Object Management', International Journal of Intelligent and Cooperative Information Systems, 1992.
15. Barsalou T., Wiederhold G., 'Knowledge-directed mediation between Application Objects and Base data', in Data and Knowledge base Integration, ed. S.M. Deen, 1993.
16. Papazoglou M. P., et al., 'An Organizational Framework for Cooperative Intelligent Information Systems', International

Journal of Intelligent and Cooperative information systems, 1992
17. Qutaishat M.A., et al, 'Association Merging in a Schema Meta-Integration System for a Heterogeneous Object-Oriented Environment', BNCOD10, Springer-Verlag, 1992.
18. Murphy J., Grimson J., 'The Jupiter System : A Prototype for Multidatabase Interoperability', Proc. BNCOD12, Springer-Verlag, 1994.
19. Jeffery (RAL) - Jeffery K.G., 'A Model for Heterogeneous Distributed Database Systems', Proc. BNCOD12, Springer-Verlag, 1994
20. Hammer M, McLeod D., 'On Database Management System Architecture'. Tech Rep. MIT/LCS/TM-141, Massachusetts Institute of Technology, Cambridge, Mass, 1979
21. Patrol. BMC Software Ltd., USA,URL: http://www.bmc.com/cgi-bin/state.cgi?id=/local12/html/
22. Dogac A., Kilic E., Ozhan G., Dengi C., Kesim N., Koksal P., 'Experiences in using CORBA for a Multidatabase Implementation', 6th International Conference on Database and Expert Systems Applications Workshop presentation, London, September 1995.
23. Gokhale A., Schmidt D.C., 'Measuring the Performance of Communication Middleware on High-Speed Networks', SIGCOMM, August,1996.