A Monitoring System for Heterogeneous Distributed Environments*

Aleksander Laurentowski, Jakub Szymaszek, Andrzej Uszok, Krzysztof Zieliński

Institute of Computer Science
University of Mining and Metallurgy (AGH)
Al. Mickiewicza 30, Cracow, Poland
tel. +48 12 339406, fax. +48 12 338907
e-mails: {pinio,jasz,uszok,kz}@ics.agh.edu.pl

1 Introduction

Emerging trends in high performance computing tend to exploit massively parallel systems (like SPP 1000) and distributed memory architectures (e.g. SP1, SP2). The most appropriate computational model for such systems is built of a set of autonomous objects, communicating via message-passing mechanism. Advanced applications within this model may be engineered of components implemented in various programming languages. Hence, particular parts of such applications can be executed in those programming environments which inherently express the nature of computation. This allows mixing various programming and execution paradigms (e.g. imperative, functional, object-oriented or logic programming) in one system. Taking advantage of software components reuse and easier implementation process, this technique can speed-up the development of new, modern applications – unfortunately, much increasing their complexity at the same time.

Tools for monitoring, debugging and maintaining of such applications could substantially help in that case. Our Managed Object-based Distributed Monitoring System (MODIMOS) aims at monitoring of large distributed software-heterogeneous applications, organized according to an object wrapping technique. The area of its application covers visualization of the systems' structure and activity, management of their logical configuration, thus enabling appropriate tuning. This influenced characteristic features of MODIMOS, like e.g. configurability of its modules, information management & filtering mechanisms, expandability of monitored environments set.

2 Description

The functionality of MODIMOS is determined by an abstraction we call the Uniform Model of Computation, pertinent to the object-based environments used for wrapping of programming modules. Well-known distributed systems

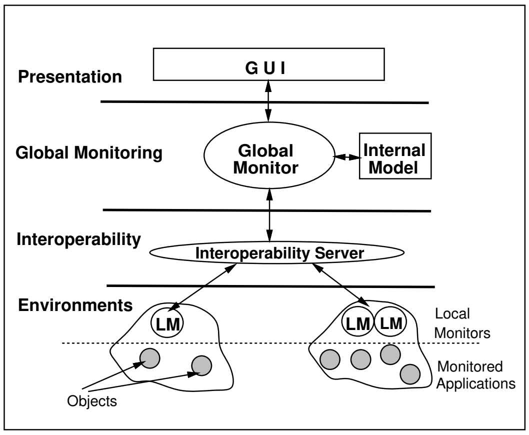
* This work was sponsored by KBN under grant no.: 8 S503 015 06.

used for wrapping technology show some similarity of their computational models. Hence, in our Uniform Model the following levels of abstraction have been recognized: environment, application, container, object, interface, method and thread (which we plan to handle in the near future). The detailed mapping of them into a couple of popular programming environments, e.g. ANSA, ISIS, and a CORBA-compliant (Orbix) is shown in the table below.

Unit of abstraction	**Environment**		
	ANSA	ISIS	Orbix
application	—	makefile	—
container	capsule	—	process
object	object	process	object
interface	interface	process interface	interface
method	service	function	operation
thread	thread	thread	thread

A set of relevant events is associated with each abstraction unit. These events are reported to the monitoring subsystem. Events concern the aspects of the items' behavior, such as initiation, creation, destroy, change of state, operation call/end, etc.

MODIMOS has a multi-layer architecture shown below. Functionality of these layers may be described as follows.



The Environments Layer consists of an expandable set of popular distributed programming environments. Any new object-wrapping environment can join the Environments Layer, provided it fits at least partially the Uniform Model and supports basic mechanisms of communication with the outside world. The Environments Layer consists of Monitored Applications sublayer and Local Monitoring sublayer.

The Monitored Application Sublayer represents original application code, instrumented with notification functions by special preprocessors. The events reported by notification functions are collected in the Local Monitors sublayer. Each local monitor is a managed object written in a language provided by the given environment. It has three interfaces: Monitored Events, Management and Reported Events. Management Interface is used for monitoring policy setting, that determines which events received via Monitored Events Interface are forwarded through Reported Events Interface. Information sent via Reported Events Interface is structured according to the Uniform Model. Therefore, above the first layer only the abstract semantics is recognized. Local monitors that perform the same policy may be grouped into a domain. Hence, domain management functions (similar to those defined in [2]) may be used for local monitors management.

The second layer deals with interoperability aspects. The aim of the Interoperability Layer is to ensure a universal and general platform for operations dispatching between local monitors and Global Monitoring Layer. The dispatching mechanisms used for this purpose are transparent to the local monitors. This layer dispatches invocations concerning reported events notification to the Global Monitoring module and, respectively, the management decisions from the global monitor (i.e. the user) down to local monitors.

Global Monitoring Layer collects reported events in a database called Internal Model (reflecting the architecture and current state of the monitored environment), cooperates with Graphical User Interface in the process of information visualization, implements the selective monitoring policy, and manages domains of local monitors.

3 Results & Discussion

As MODIMOS is itself a software heterogeneous application, the general interoperability mechanisms proposed and employed during its construction can be used in further multi-paradigm systems. We have developed and verified three different variants of environments' connection via the Interoperability Layer: a socket, proxy, and gateway solutions. The appropriate variant is chosen depending on characteristic features of particular environment. CORBA-compliant system is used as a general platform for achieving interoperability in MODIMOS. The CORBA standard is one of the most advanced proposals for distributed integration systems available today, and is going to become a standard *glue* environment of the future. We have tested and proved usefulness and efficiency of Orbix CORBA-compliant system for the needs of Interoperability Layer.

High performance computing imposes substantial requirements on monitoring tools, e.g. ability to cope with large amount of data, high speed and intensity of interactions. In contrast to systems like IPS-2 [3], Paragraph [4], and similar, MODIMOS is not intended to present the whole program execution history. Introduction of filtration and selection mechanisms enables the user to significantly limit the amount of collected data and focus on the the most interesting and "hot" areas of the monitored system's activity. Filtration can take place at Environments Layer, where selection rules can be set up at preprocessing time and through the management interfaces of local monitors, as well as at the Global Monitoring level. Moreover, selection options at GUI enable the user to specify a policy defining group of objects or levels in the Unified Model hierarchy as interested during the particular visualization process.

Equipped in those features, MODIMOS can help in development, tuning and debugging of future software heterogeneous high performance applications.

References

1. M. Mansouri-Samani and M. Sloman, "Monitoring Distributed Systems", Imperial College Research Report No. DOC92/23, London, April 1993.

2. R. Wies, "Policies in Network and Systems Management – Formal Definition and Architecture", *Journal of Network and Systems Management*, Plenum Publishing 1994.
3. B. P. Miller, M. Clark, J. Hollingsworth, S. Kierstead, S. Lim and T. Torzewski, "IPS-2: The Second Generation of a Parallel Program Measurment System", *IEEE Trans. on Parallel and Distributed Systems*, 1,2, April 1990.
4. M. T. Heath and J. A. Etheridge, "Visualizing Performance of Parallel Programs", *IEEE Software*, 8(5), September 1991.