Performance Evaluation of Open-Source Multiagent Platforms[*]

Luis Mulet
lmulet@dsic.upv.es

Jose M. Such
jsuch@dsic.upv.es

Juan M. Alberola
jalberola@dsic.upv.es

Department of Information Systems and Computation
Polytechnic University of Valencia
Cami de Vera s/n. 46022, Valencia (Spain)

ABSTRACT

Nowadays, most multiagent platforms are internally designed as middleware and are usually implemented in Java and run on top of an operating system. This kind of design maximizes portability and reduces the development cost; however, it may lead to low performance and scalability. In this context, our research has the long-term goal of integrating into the operating system some key services which are currently supported by middleware platforms. The first step in achieving this goal is to study some well-known, open-source platforms in order to understand to what extent the internal design of a platform influences its performance.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

performance evaluation of agent systems, agent and multiagent architectures

1. INTRODUCTION

The area of Multiagent systems has produced both methodologies and actual frameworks to make the implementation of agent-based systems possible. The support software of such frameworks are often known as multiagent *platforms*. In essence, these platforms provide both a model for developing multiagent systems and an environment for running distributed agent-based applications. Among all the existent platforms, only those whose source code was available were feasible for us, since part of the study required the inspection of specific parts of the code. Three of these platforms were selected: JADE[3], MadKit[4] and AgentScape[1]. They are representative of different design principles and different programming models (from FIPA[2] compliance to multi-language support).

*This work has been partially funded by grant TIN2005-03395 from the Spanish Government

2. EXPERIMENT DESIGN

Services tested in this paper were either common to the three platforms or offered the same function. Moreover, these services are defined by the FIPA standard. The messaging service is the most important service due to the fact that multiagent systems are based on the communication between the agents. The service directory service is also important because an agent platform requires services offered by agents be accessible to other agents.

2.1 Messaging Service

We measured the RTT (round-trip time) of each sent message between a sender agent and a receiver agent, i.e. the time elapsed between when the message is sent to the receiver and when it comes back to the sender. The number of couples was increased in order to observe the platforms scalability. For each agent couple, we calculated an average RTT from a collection of 1000 sent messages. The result of the experiment was the average of the RTT averages from all the couples in the experiment. The experiments were also repeated changing the kernel (container in JADE terminology) and the host where the agents are placed.

The effect of message size on messaging service performance was also tested. For this purpose, we performed an experiment modifying the message size (from 1 byte up to 100000 bytes). The number of agent couples were fixed to 50 and 100.

Finally, in order to observe the platform behaviour when a massive message sending occurs, we designed a test in which many agents tried to communicate with an unique receiver. The number of sender agents was increased from 1 to 250.

2.2 Service Directory Service

We tested the service directory service response with two kinds of experiments. First, registration and deregistration time were measured by increasing the number of agent services already registered. The structures of the platforms were changed from one host to two hosts in order to determinate the influence of agent location. We also estimated the platform response when a service search was requested. As explained above, the number of agent services already registered was increased. The agent location was also changed. We allocated the searcher and registered agents in both the same or different hosts.

3. EXPERIMENT RESULTS

In order to perform the experiments, we used two PC's Intel Pentium 4 @1.5 GHz, 256 MB of RAM memory and Sun JDK 1.5 version.

3.1 Messaging Service

Figure 1 shows the behaviours of the three platforms when the sender and receiver are placed in the same kernel located in one host. AgentScape offers the poorest RTT values. These results may be due to the fact that there is yet no mechanism to make a message blocking reception in this platform. AgentScape proposes a polling mechanism for message reception. According to this, the platform suspends the agent between each iteration. JADE and MadKit have a similar response in low load. In contrast, when load increases, JADE RTT grows with respect to MadKit. This may be due to the fact that MadKit uses a direct copy-write mechanism, while JADE uses a more complex process which is also utilized in distributed communication.

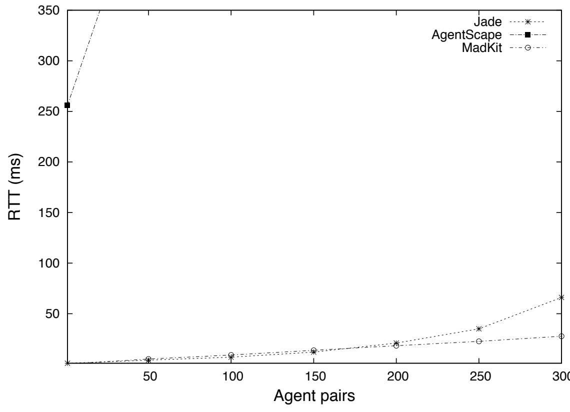

Figure 1: Message Sending, 1 host, 1 Kernel

When the agents are located in different kernels (Figure 2), JADE presents shorter RTT delays in the message sending process than the other two platforms. This is due to how containers are implemented and the fact that they communicate among them using Java-RMI. Madkit distributed messaging introduces an overload due to the set of *Communicator Agents* that are responsible for communicating different kernels through sockets. As a result, several local messages are needed to carry out a distributed communication. AgentScape uses XML-RPC technology to communicate different kernels and requires a thread for each kernel.

We can see the message size experiment in Figure 3, in this case AgentScape indicates the worst RTT values, being less adaptive to size changes than JADE and MadKit. Besides, JADE presents the best response to a multiple sending. It manages the message queues more efficiently than the others when a lot of messages are received. In contrast, AgentScape and Madkit are more influenced by this massive reception.

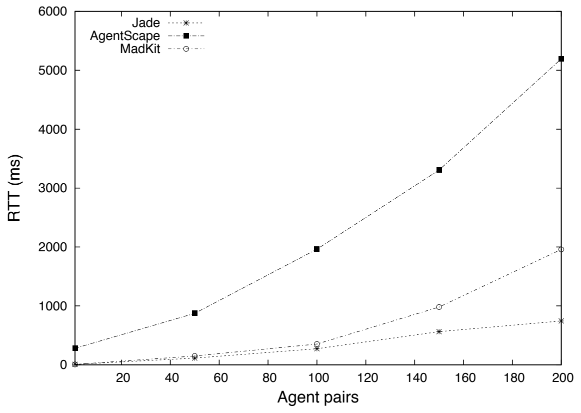

Figure 2: Message Sending, 2 hosts

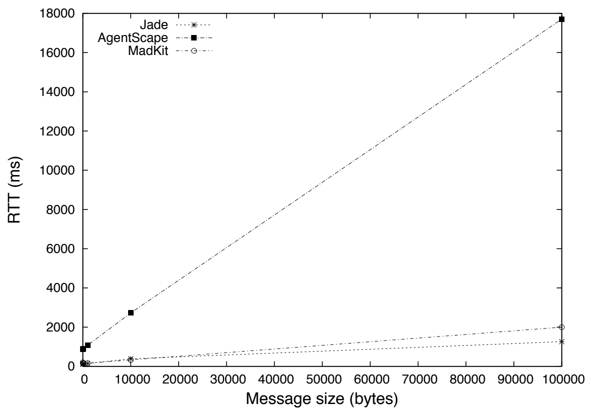

Figure 3: Message Sending 50 pairs

JADE is more scalable than the other platforms. Due to the lack of space, figures showing results for multiple sending are not included.

3.2 Service Directory Service

Figure 4 depict the registration time in 1 host. In these experiments, the differences are reduced. The number of services already registered does not have any influence on a new agent service register time. The MadKit registering service is the fastest one because it is distributed and duplicated among all the kernels. When an agent wants to register a service, it is done automatically by the local kernel. Afterwards, if there is more than one kernel, information replication is needed in the other kernels. AgentScape and JADE do not have this service distributed. In AgentScape, when an agent wants to register a new service, it must contact the *lookupService* of the platform. This platform has a better register time when the agents are in a host that is different from the lookupService. This is due to the fact that lookupService can be placed in a host without any other components of the platform. Hence, there is less overload in this host. JADE offers worse times than MadKit because agents that want to register their services must establish a communication act with the DF agent. This fact increases the time due to ACL message exchanges.

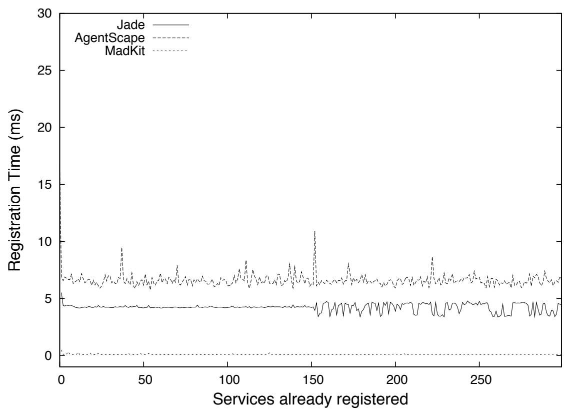

Figure 4: Registration in 1 host

As shown in Figure 5, AgentScape presents a linear increase in search time. This may be due to the fact that a method has not yet been developed in the lifecycle of the AgentScape platform to suspend the agent execution. For this reason, there are more active threads, so the platform load is higher. In contrast, MadKit and Jade offer a constant behaviour. Since JADE implements service directory service processes using communication acts with the DF agent, its result times are not as good as those in MadKit. This is a result of how MadKit implements this service. While JADE implements it as an agent (DF agent), MadKit integrates it in the platform software. In other words, this service is centralized in JADE, whereas it is distributed among the platform kernels in MadKit. Due to the lack of space, the figures of the other search experiments have been not shown, but the conclusion is similar.

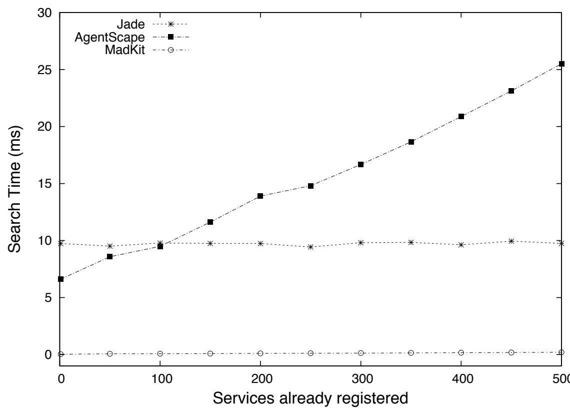

Figure 5: Search in different hosts

4. CONCLUSIONS

The main focus of the experiments was to try to link performance with internal design, that is, to identify the key design decisions that lead to better performance. In addition to the quantitative results about performance presented in the paper, two general conclusions can be drawn.

First, some platforms implement basic services (such as messaging) by means of agents, while others implement these services in their kernels. Offering services by means of agents may increase the platform modularity; however, it also degrades performance. Every time an agent needs the service, it has to communicate with the agent offering the service through an ACL act. This increases both the platform load and the service's response time. This conclusion is clearly proved in the two experiments shown in the paper. JADE implements the messaging service directly in the kernel, whereas MadKit requires an *AgentCommunicator* when agents in different hosts want to communicate. The difference in performance in this case is shown, for example, in Figure 2, in which JADE clearly outperforms MadKit. The service directory service in JADE is provided by means of an agent (the DF), while MadKit offers it directly. Again, Figure 4 shows how the platform that provides a service by means of an agent (like JADE) has a penalty in performance.

Second, centralizing services in a single host in the platform also degrades performance since this host can become a bottleneck in the case of very popular services. The experiments presented in the paper also support this conclusion. For example, Figure 5 shows that the performance for the service directory service in both JADE and AgentScape is much worse than in MadKit since, in these cases, the service is centralized. Therefore, distributing the services among the various hosts in the platform may benefit the response time of these services, however, the coordination among the hosts is an issue to be taken into account.

5. ADDITIONAL AUTHORS

Additional authors: Vicente Botti (`vbotti@dsic.upv.es`), Agustin Espinosa (`aespinos@dsic.upv.es`), Ana Garcia (`agarcia@dsic.upv.es`) and Andres Terrasa (`aterrasa@dsic.upv.es`).

6. REFERENCES

[1] Agentscape. `http://www.iids.org/research/aos/`.
[2] Fipa. `http://www.fipa.org`.
[3] Jade. `http://jade.tilab.com`.
[4] Madkit. `http://www.madkit.org`.
[5] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade a white paper. *EXP*, 3:6–19, 2003.
[6] F.M.T. Brazier, D.G.A. Mobach, B.J. Overeinder, S. van Splunter, M. van Steen, and N.J.E. Wijngaards. Agentscape: Middleware, resource management, and services. In *Proceedings of the 3rd International SANE Conference*, pages 403–404, 2002.
[7] K. Burbeck, D. Garpe, and S. Nadjm-Tehrani. Scale-up and performance studies of three agent platforms. In *IPCCC 2004*.
[8] D. Camacho, R. Aler, C. Castro, and J. M. Molina. Performance evaluation of zeus, jade, and skeletonagent frameworks. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on.*
[9] E. Cortese, F.Quarta, and G. Vitaglione. Scalability and performance of jade message transport system. *EXP*, 3:52–65, 2003.
[10] O. Gutknecht and J. Ferber. The madkit agent platform architecture. *Lecture Notes In Computer Science*, 1887:48–55, 2000.
[11] P. Vrba. Java-based agent platform evaluation. In *Proceedings of the HoloMAS 2003*, pages 47–58, 2003.