A Methodology for Improving the Performance of Agent-Based Applications Through the Identification of the Optimal Number of Mobile Agents

S.Kalogeropoulos, G.Karetsos, A.Anagnostopoulos

National Technical University of Athens, Electrical and Computer Engineering Department

Abstract. The results of the research in the area of Mobile Agent Technology (MAT) have demonstrated the utility of this paradigm in building a wide range of distributed applications and systems. In the past few years, we have witnessed an enthusiastic interest in MAT and plenty of research activities have been carried out. Despite this effort, a number of issues concerning mainly the performance optimisation of agent-based applications are still open. In this paper, we investigate the problem of optimising the number of mobile agents used in an agent-based application in order to improve its performance. As an outcome of our research, we propose a methodology for the decision of the optimum number of mobile agents. To investigate the effect of the number of mobile agents on the performance of agent-based applications and to evaluate the proposed methodology through real measurements, an agent-based distributed application was designed and implemented.

1. Introduction

With the establishment of Internet as the default worldwide infrastructure for communication and information exchange, distributed applications have gained remarkable popularity. One of the most promising approaches for developing such applications is the mobile agent paradigm [1,2].

Several technical and non-technical hurdles must still be addressed before MAT is widely adopted in the distributed applications area. An important aspect for the applicability and survival of an application, especially of one distributed over the Internet, is performance. This paper aims to investigate the problem of optimising the performance of agent-based distributed applications. The performance improvement is expected to be the result of the transformation of a single-agent execution in contrast to multi-agent [3] execution thus by exploring the advantage of the cooperative aspects inherent in MAS systems. With the terms single-agent or multi-agent we refer to applications that use one or several mobile agents respectively for the execution of a single task. The research is focused on discovering the number of agents that leads to the optimum performance. An optimisation methodology is proposed and tested against a real agent-based application, designed and implemented for the needs of this research.

2. Optimizing the number of mobile agents in a multi-agent system

The parameters that influence the performance of a mobile agent are related to the migration and the execution time of the agent at the target machine. Agent-based applications are subject to performance bottlenecks due to insufficient network or computation resources. Those parameters can be summarized as follows:

- Underling Network parameters [4]
- Agent host parameters [5]
- Agent size [6]

All the efforts for improving the performance of mobile agent systems are directed towards improving the performance of each individual agent, by reducing the migration or execution time.

Another way to improve the performance of an agent-based system is by changing the design. The idea is simple: split the task assigned to a single agent to more agents. This idea could be applied to most of the agent-based applications. The performance improvement is the result of the parallel process [7] of the task and of the reduction of the agent's migration time. The first argument is obvious. By splitting the task of the agent into several agents, that act autonomously, the total execution time will, most probably, get reduced. By using more than one agent to carry out a task, each of the cooperative agents has to visit a fixed number of target machines keeping its size small, which makes the migration process of the agent easier. The only disadvantage, which will be investigated within this paper, is that this approach is more bandwidth consuming.

An important question that arises is how many agents to use in order to optimise the overall execution time of the agent-based application. The following parameters influence the performance of an agent-based application.

- The creation time of the agents
- The migration time of the agents
- The execution of the agent on the service machine

If the task of a mobile agent, that has to visit a fixed number of target machines, is split into several agents, then not all the derived agents will visit the same number of target machines. Since all the agents act asynchronously, executing the same task on different target machines, the total execution time of the task must be calculated based on the agent that has the maximum execution time. At this point, we must clarify that the condition of the underlying network and of the target machines are not taken under consideration, for the migration and execution time of the agents. We assume that these conditions are the same for all the agents and do not affect the execution time. Based on this assumption, the agent with the maximum execution time is the one that has to visit more machines than the other agents. The assignment of the target machines to the agents must be performed in a way that minimises the total execution time of the task. This can be achieved only by distributing the target machines to the available agents in a way that minimises the maximum number of target machines, assigned to an agent of the system. Based on this concept, the maximum number of machines, that one agent of a multi-agent application has to visit, can be extracted as follows.

If the total number of target machines for a given task is l, then the maximum number of machines n that an agent of a multi-agent system, consisting of r agents, has to visit is:

$$n = \begin{cases} l \ \textbf{div} \ r & \text{If} \quad l \ \textbf{mod} \ r = 0 \\ (l \ \textbf{div} \ r) + 1 & \text{If} \quad l \ \textbf{mod} \ r \neq 0 \end{cases} \tag{1}$$

Suppose that the number of target machines for a given task equals 10 and the proposed number of agents equals 5. The 10**mod**5=0 and each of the agents have to visit 10**div**5=2 machines to complete the task. If the proposed number of agent equal 3 then 10**div**3=3 and 10**mod**3=1. Hence, each of the 3 agents have to visit 3 machines and one of them must visit one more in order to complete the task. Thus, in the second cases the maximum number of machines that an agent of the multi-agent system has to visit equals 3+1=4.

The total execution time of a given task performed by a multi-agent application can be expressed by the following equation.

$$T(r) = \sum_{i=1}^{n} \{M(i) + E(i)\} + C(r) \tag{2}$$

Where:

T: Is the total execution time of the task
n: Is the maximum number of target machines

M: Is the migration time of the agent
C: Is the creation time of the agents
E: Is the execution time at the target machines
r: Is the number of the agents

The optimum number of agents used in a multi-agent application is the one that minimises the outcome of equation (2).

By increasing the number of the system's agent, according to equation (1) the maximum number of target machines, that one or several agents of the system have to visit, is decreased. This leads to a smaller value for the first component of equation (2), which depends on the number of the target machines. On the other hand, the second component of the equation, which is the creation time of the agents, is increased. The effect of those two components on the total execution time can be investigated only through measurements on a real agent-based application system. For this reason an agent-based distributed application was designed and implemented.

3. Real test experimental results

In order to test the effect of the number of agents on the performance of an agent-based application, an Information Gathering (IG) system was designed and implemented. The system provides Quality of Service (QoS) statistics information for a Linux-based Differentiated Services (Diffserv) [8] Domain.

A testbed with Linux-based DiffServ routers was contracted in order to test and evaluate the behavior of the system, when the number of used agents is changing. The performed tests were not focused only on the execution time of the agent-based application, but also on the network resources consumed for different number of agents. The test-bed's setup is illustrated in figure 1.

The scenario of the experiment is the execution of the application, using a different number of mobile agents for the completion of the task. The experiment starts with one mobile agent and continues by increasing the number of agent by one until, an agent is sent to every target machine. This way, the effect of the number of mobile agents on the execution time can be monitored.

Since the tests are performed on a constructed network of 10 Mbps bandwidth consisting of identical machines, the following assumptions can be made. The average migration and execution time of the agents, from one target machine to another, is constant. The assumption concerning the migration time of the agents is, in this case, valid because the additional data carried by the agent, as it moves from one target machine to another, is only 10 bytes. This is achieved by the aggregation of the collected data. The execution time of the agent on the remote machines depends on the type and the current CPU workload of the machine. Since all the target machines are identical and the CPU workload is close to zero, the average remote execution time of the agent is constant. Based on real measurements the average migration time of the agent from one machine to another is equal to 1450 ms. The remote execution time is equal to 20 ms and the execution time of the master agent controller, after all the agents have returned, equals 80ms.

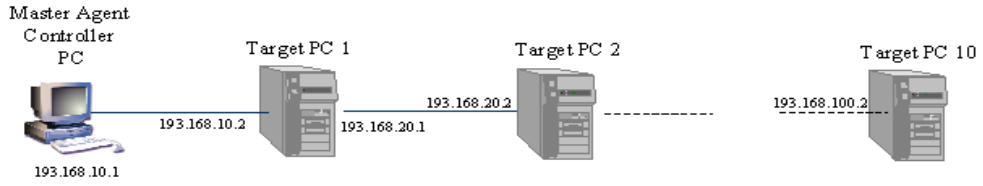

Fig.1. The test-bed of the DiffServ QoS statistics application

The total execution time of the task, based on the number of used agents, is given from equation 2 presented in section 2. The assumptions that the migration and the execution time of the agents are

constant, makes the calculation of the total execution time easier. The only parameter that has to be measured is the average creation time of the agents, which is presented in table 1.

As it was expected, the creation time increases significantly, as the number of agents increases. The average creation time was calculated using a number of measurements contacted on the machine that hosts the master agent controller. For a number of agents higher than five, the standard deviation is close to 250 ms.

The outcome of equation (2), for the task of collecting and reporting the QoS related statistics in a Linux-based DiffServ domain is given in table 1. The Maximum number of target machines is extracted from equation (1), presented in section 2. The total number of target machines equals 10. In this task, all the agents must return and report to the agent controller. This migration time must be added to the total migration time. The execution time of the agent controller, after all the agents have return, must be also added to the execution time.

In order to prove that the proposed methodology provides the total execution time of the task for a different number of mobile agents used, the real execution time is also measured and presented in table 1. Of course, due to the assumptions made there will be a deviation between the real and the calculated values. If this deviation is big then the optimum number of mobile agents for the execution of the task cannot be extracted based on the calculated values.

Table 1: Total execution time

No of Agents	Max No of Targets (n)	Migration Time	Execution time	Creation time	Calculated Total execution Time (T)	Measured Total execution Time
1	10	10x1450 + 1450	10x20 + 80	40	16270	16938
2	5	5x1450 + 1450	5x20 + 80	310	9190	9228
3	4	4x1450 + 1450	4x20 + 80	600	8010	7455
4	3	3x1450 + 1450	3x20 + 80	740	6680	6397
5	2	2x1450 + 1450	2x20 + 80	1200	5670	5504
6	2	2x1450 + 1450	2x20 + 80	1450	5920	5418
7	2	2x1450 + 1450	2x20 + 80	2400	6870	5743
8	2	2x1450 + 1450	2x20 + 80	2900	7370	6467
9	2	2x1450 + 1450	2x20 + 80	3400	7870	6934
10	1	1x1450 + 1450	1x20 + 80	5100	8100	7630

The calculated and the real execution times for different number of agents are illustrated in figure 2.

As it was predicted, there is a small deviation between the real and the calculated values, which is caused by the assumptions made and the standard deviation of the average creation time. A point worth mentioning is that based on the calculated values the number of agents that optimizes the execution time is five and based on the real values this number is six. The real execution times for five and six agents are very close and the difference is within the range of the standard deviation of the average creation time of the agents. Thus, this result is considered as normal and the extraction of the optimum number of agents can be based on the calculated values.

The small deviation between the real and the calculated values, proves that the assumptions made for the migration and the execution of the agents are correct. This conclusion is very important for the applicability of the proposed method. If the agent size does not change significantly as it migrates from one machine to another, then this time depends only on the conditions of the underling network and the average migration time of the agents can be considered as constant. The same consideration can be made for the execution time of the agent on the target machines, which depends on the type of the machine and the current CPU workload. In this case, the measured parameter for the application of the proposed methodology is limited to the creation time of the agents, which makes the calculation process much easier.

Based on the calculated values, the execution time of the task is significantly reduced until the number of agents is less or equal to five, which is the optimum number of agents. From this point, the execution time is increased. This can be easily explained based on the fact that, for a number of agents between 5 and 9, the maximum number of targets assigned to one or several agents of the system is the same and equals 2. Thus, the migration and execution time of the agent with the maximum number of targets is the same, for a number of agents between these two values. On the other hand as the number of used agents increases the creation time of the agents also increases significantly, leading to higher execution times for the task. In case an agent is used for every target machine (10 agents) the migration time of the agent with the maximum target machines is reduced. However, the creation time of the agents is at this point very high. Thus, despite the improvement of the migration time, the total execution time is worst than when the number of agents is between 5 and 9.

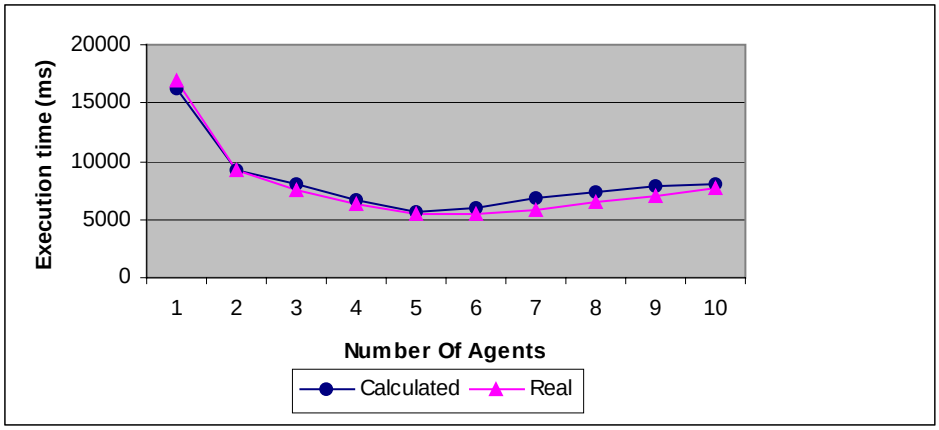

Fig.2. Calculated and Real execution time for different number of agents

Based on the results depicted in figure 2 the performance of the application is considerably increased even when the number of agents is less than the optimum. The performance of the single-agent application is increased almost by 40% even when only two agents are used. Another important parameter in distributed applications is the utilization of the network bandwidth. Thus, without sacrificing much of the performance, a smaller than the optimum number of agents can be used in order to reduce the amount of the generated traffic. But this decision must be made based on the available resources of the network. In figure 3, the generated traffic for different numbers of agents is presented. The additional traffic is the result of the initial size of the agent, which in this case is 6.2 KBytes. If an agent-based application uses a high number of agents with a big initial size this can lead to a considerable consumption of network resources.

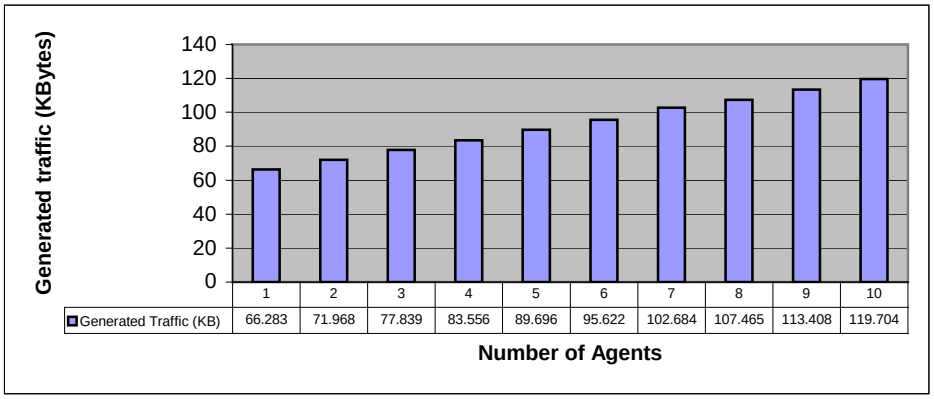

Fig.3. Generated traffic for different number of agents

4. Conclusions and future work

In this paper, the effect of the number of mobile agents on the performance of an agent-based application was investigated. In addition, a methodology for the extraction of the optimum number of mobile agents was proposed and evaluated. The Experimental results show that the use of the optimum number of agents, based on the proposed methodology, could improve the performance of an agent-based application up to 75%.

The assumptions for the migration and execution time were made in order to ease the calculation process and to prove that the proposed methodology can be easily applicable. They do not affect in any way the correctness of the results.

For the calculation of the total execution time the migration time of the agents from one machine to another was considered constant, based on the fact that the agent size does not change significantly. However, there is always the case in which the mobile agent size is increased significantly making the migration harder. In this case, the performance can be further improved only by using more agents. This leads to a small number of maximum target machines, as defined in the methodology, maintaining the size of the mobile agents small. The experimental results revealed that the creation process of the agents is very time-consuming. This has to do with the fact that Grasshopper and most of the MAPs are implemented in Java, which treats every agent as a different process (thread). The execution of multiple threads in the slow interpreted Java environment leads to a very high creation time as the number of agents is increasing. In the future, the interaction of those two parameters on the performance of a multi-mobile agent system will be further investigated. A possible solution could be cloning. Instead of creating all the agents on one machine, the optimum number of agents can be achieved by cloning a set of initial agents on the target machines. This might lead to the reduction of the agent's creation time and further improvement of the performance of the system.

The initial size of the agents causes additional traffic to the network. If the available network resources are limited, this traffic must be reduced. This can be achieved by selecting a number of agents lower than the optimum. Of course, part of the performance will be sacrificed, but as it was noticed, even a small addition to the number of agents leads to high performance improvement.

Based on the results of this work, an easy way to apply the methodology is to calculate the total execution times only for the number of agents, which leads to a different number of maximum target machines, as provided by equation (1). This conclusion is based on the fact that the first component of the total execution time in equation (2) is constant for a given number of target machines. The second component of the equation is increasing as the number of target machines increases, leading to longer execution times. For example, if the number of target machines equals ten, then to define the optimal number of agents the total execution time equation must be applied only for one, two, three, four, five and ten number of agents.

References

1. D. Kotz and R.S. Gray, "Mobile Agents and the Future of the Internet", In ACM Operating Systems Review, pp. 7-13, August 1999.

2. T. Anand, A. Tanvir and K. Neeran, "Experiences and Future Challenges in Mobile Agent Programming", In Microprocessor and Microsystems, pp. 121-129, 2001.

3. E.H. Durfee, V.R. Lesser and D.D Corkill, "Trends in Cooperative Distributed Problem Solving", In IEEE Transactions on Knowledge and Data Engineering, pp. 63-83, March 1989.

4 K. Moizumi, "The mobile agent planning problem", PhD thesis, Thayer School of Engineering, Dartmouth College, November 1998.

5. A. Michalas, T. Kotsilieris, S. Kalogeropoulos, G. Karetsos, M. Sidi and V.Loumos, "Enhancing the Performance of Mobile Agent based Network Management Applications", In ISCC 2001, HAMMET Tunisia, July 3-5 2001.

6. M.G. Rubinstein and O. C. M. B. Duarte, “Analyzing Mobile Agent Scalability in Network Management”, In IEEE Latin American Network Operations and Management Symposium (LANOMS'99), Rio de Janeiro, Brazil, December 1999.

7. M. Muthukrishnan and T. B. Suresh, “A Multi-Agent Approach to Distributed Computing”, In Autonomous Agents, Seattle Washington USA, May 1 1999.

8. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, “An Architecture for Differentiated Services”, IETF RFC2475, December 1998