Using Negotiation to Reduce Redundant Autonomous Mobile Program Movements

Natalia Chechina, Peter King, and Phil Trinder
School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh, UK
{nc75, P.J.B.King, P.W.Trinder}@hw.ac.uk

***Abstract*—Distributed load managers exhibit *thrashing* where tasks are repeatedly moved between locations due to incomplete global load information. This paper shows that systems of Autonomous Mobile Programs (AMPs) exhibit the same behaviour, identifying two types of redundant movement and terming them *greedy effects*. AMPs are unusual in that, in place of some external load management system, each AMP periodically recalculates network and program parameters and may independently move to a better execution environment. Load management emerges from the behaviour of collections of AMPs.**

The paper explores the extent of greedy effects by simulation, and then proposes *negotiating* AMPs (NAMPs) to ameliorate the problem. We present the design of AMPs with a competitive negotiation scheme (cNAMPs), and compare their performance with AMPs by simulation.

***Keywords*-mobile computation; load balancing; autonomous mobile program; greedy effect; mobile agent; scheduling; workflow management.**

I. Introduction

Autonomous mobile programs (AMPs) are mobile agents that improve execution efficiency by managing load; AMPs are aware of their resource needs, sensitive to the execution environment and periodically seek a better location to reduce execution time [1]. AMP behaviour has previously been investigated on local area networks (LANs) using mobile languages like Java Voyager [1] and using simulation [2]. Comparing the simulation results and observations of the real system shows that simulated and real AMPs enter the same *stable states* where no AMP can reduce its execution time by moving. The differences between simulated and real AMPs are minor and readily explained.

A key design aim for AMPs is to provide scalable dynamic load balancing. Experiments with real AMPs showed that AMPs effectively perform decentralised load balancing in a LAN; moreover, AMPs have the potential to scale to very large and dynamic networks. However, analysis of real and simulated AMP movements revealed the presence of greedy effects [1]. Greedy effects are redundant AMP movements during the balancing of loads between locations, and are a result of locally optimal choices made by each AMP. Although AMPs relocate autonomously, the greedy effects we investigate directly correspond to thrashing behaviours observed for programs relocated by conventional distributed load managers, e.g. [3]. The fundamental reason for non-optimal movements is that both systems have locally optimal information.

The aim of the current paper is to examine properties and features of the greedy effects in collections of AMPs and expose their causes. We further aim to reduce the greedy effects, and to estimate the extent of the greedy effects in the modified algorithm using simulation. Detailed discussion can be found in [4].

The paper is organised as follows. We identify two forms of greedy effect (Section II), and examine the AMP greedy effects on initial distribution and rebalancing experiments (Section III). Analysis of types of movements shows that the majority of redundant movements occur because an AMP is unaware of the intentions and movements of other AMPs. Thus, we discuss ways to reduce the greedy effects and propose the concept of negotiating AMPs (NAMPs) that communicate their intentions with the view to reducing redundant moves. While a number of negotiation schemes are possible, we have designed and simulated AMPs with a competitive scheme (cNAMPs) (Section IV).

An analysis of simulated cNAMP results shows that even this simple negotiation significantly decreases both the number of redundant movements and the time to rebalance. For example, cNAMPs make no redundant movements during initial distribution, making initial balancing in the experiments at least three times faster than for AMPs (Section V). A summary of the results and discussion of future work are provided in Section VI.

II. Greedy Effects

An *optimal rebalancing* is a sequence of AMP movements that is the minimum number needed to enter a stable state. AMP *greedy effects* result in a non-optimal AMP rebalancing with redundant movements. Greedy effects are a result of AMPs making locally optimal choices, i.e. AMPs do not possess sufficient and accurate global state information to make the optimal movement decision. There are two types of greedy effect: location thrashing and location blindness. Both types are observed in real [1] and simulated [2] AMP experiments.

Location thrashing results from an AMP's lack of information about other AMPs that intend to move to the same location. That is, two or more AMPs decide to move on the basis of the same information about the target location, which causes further AMP retransmission. Location thrashing occurs in dynamic load balancing systems; other terms are *processor thrashing* [5], *task thrashing* [6], *task dumping* [7], *transmitting dilemma* [8].

Location blindness results from an AMP's lack of information about the remaining execution time of other AMPs. The problem is not with poor runtime predictions, but rather an inability to obtain accurate AMP runtime predictions at distributed locations, i.e. the more accurate information that is required, the more *expensive* it becomes to collect [9]. Location thrashing is the more harmful effect because it increases AMP execution time.

III. AMP Greedy Effect Analysis

A. AMP Distribution Scenarios

To illustrate the greedy effect we first introduce the following AMP scenarios that specify the number of AMPs and locations, and types of locations:

Scenario 1: 25 AMPs on 15 locations with CPU speeds 3193 MHz ($Loc1 - Loc5$), 2167 MHz ($Loc6 - Loc10$) and 1793 MHz ($Loc11 - Loc15$).

Scenario 2: 20 AMPs on 10 locations with CPU speeds 3193 MHz ($Loc1 - Loc5$), 2168 MHz ($Loc6$) and 1793 MHz ($Loc7 - Loc10$).

Scenario 3. 10 AMPs on 3 locations with CPU speeds 3193 MHz.

For all scenarios $Loc1$ is the root location, where all AMPs start. In the experiments we use large and small AMPs which are matrix multiplication programs of 1000×1000, and 500×500 matrices respectively. The same scenarios are used in the experiments with real AMPs [1].

B. AMP Greedy Effect Experiments

The experiments are conducted on homogeneous and heterogeneous networks. A *homogeneous network* is a set of locations with the same available speeds, except the root location, which may be different because of the overheads of initiating the remote processes. A *heterogeneous network* is a set of locations with different available speeds.

The main rule on the basis of which AMPs make a decision to move to a new location is whether execution time on the current location, T_h, exceeds execution time on the new location, T_n, and communication delay, T_{comm}:

$$T_h > T_n + T_{comm}. \quad (1)$$

Each experiment is repeated eleven times. As the experiments with the real system did not investigate the greedy effects, we have not made systematic comparisons with the real experiments, but the results are consistent where they have been compared. The experiments are as follows:

Table I: AMP greedy effect experiment summary

Config.	Initial distribution		Rebalancing after an AMP termination		Large AMP execution time, (sec)	
	Mean No. redun. moves	Mean time, (sec)	Mean No. redun. moves	Mean time, (sec)	Mean	Standard deviation
Scenario 1	64	60.4	6	22.5	173.8	7.66
Scenario 2	43	50.5	11	28.2	182.1	11.5
Scenario 3	13	26.8	6	14.1	232.6	9.91

1) *Initial distribution* experiment investigates the greedy effects when large AMPs distribute themselves over the network from a single location.

2) *Rebalancing after an AMP termination* experiment measures the number of movements and time required for a system to rebalance after an AMP termination.

3) *Large AMP execution time* experiment estimates execution time of large AMPs and measures its variability.

The simulation experiment results in Table I show that an increase of the number of AMPs and/or locations causes an increase in the number of redundant movements per AMP *during initial distribution*. The dependence of the number of redundant movements on the number of locations and the total number of AMPs *after an AMP termination* is less obvious. However, additional experiments where AMPs have larger execution times show that the number of redundant movements is directly proportional to the number of locations and AMPs [4]. Therefore, to make systems of AMPs scale, they must be adapted to minimise redundant movements.

Figure 1 shows the initial AMP distribution between locations in scenario 1 as the system rebalances from initial (unstable) state U1 to stable state S. The arrows show AMP movements. To make the Figure clearer we do not indicate AMP movements from state U1 to state U2 with lines because all AMPs move from $Loc1$ in state U1 to $Loc2 - Loc5$ in state U2. There are 88 movements in total,

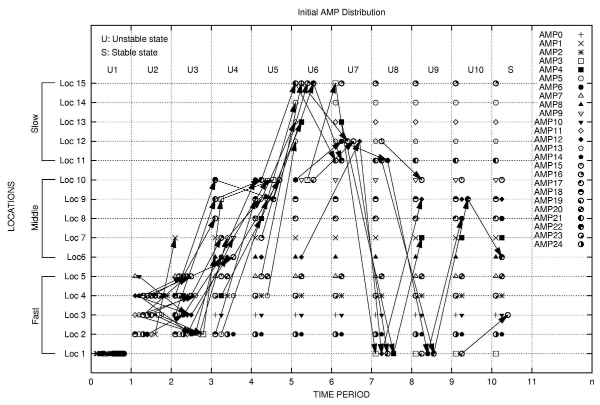

Figure 1: AMP initial distribution (scenario 1)

but if each AMP moved directly to the location it occupies in state S there would only be 24, i.e. there are 64 redundant movements.

The analysis of redundant movements allows them to be classified into three main types: 1) two or more AMPs move from one location to another, and then some of them rebalance; 2) two or more AMPs move from a location, and then some AMPs move back to the location; 3) two or more AMPs move from different locations to one location, and then some of them immediately move again.

Therefore, we conclude that *redundant AMP movements are mainly caused by AMP ignorance of intentions and actions of other AMPs in the network and, hence, lack of information to make a good decision*, i.e. location thrashing.

IV. Negotiating AMPs and cNAMPs

To reduce the greedy effects we propose to use negotiation, because the main reason for poor AMP movement decisions is a lack of communication between AMPs, i.e. an AMP does not request information about other AMPs, does not provide information itself, and does not communicate with other locations to make efficient movement decisions.

AMPs and load servers can provide information in different ways, such as malicious, honest, etc. A malicious strategy would be for a load server to misrepresent the load so that other AMPs were deterred from moving to a location. An honest strategy requires AMPs and load servers to share information to reduce wasted movements. Our belief is honest behaviour is more effective to reduce redundant movements. The modifications to AMP algorithm to provide this behaviour are described in Section IV-A.

A. The Design of cNAMPs

cNAMPs are negotiating AMPs with a competitive scheme, which announce their intentions to move and compete with each other for opportunity to transfer to the new location. Negotiation has many interpretations; the one that is used in terms of cNAMPs is a "coordination among competitive or simply self-interested agents" [10] (more definitions can be found in the same source). cNAMPs do not negotiate directly with each other, but by means of a load server.

cNAMPs are designed only to reduce location thrashing; and hence, eliminate redundant movements during initial distribution and significantly reduce the number of redundant movements during rebalancing (Table II in Section V). Reduction of location blindness requires that cNAMPs and load servers possess more information about locations and other cNAMPs in the network.

Unlike an AMP, a cNAMP first sends a request to the selected target location declaring a wish to move. The request is also an agent which can be seen as a cNAMP representative. On arrival to the target location the request recalculates parameters, informs the target load server if the decision is positive and moves back. Only if the request confirms the movement decision does the cNAMP actually move; otherwise the cNAMP continues execution locally. When a request informs the target load server about a cNAMP movement the load server reports the load as if the transferring cNAMP had already arrived so each load server maintains two values for the load: a) the *actual load* which is the number of executing cNAMPs and is used for local cNAMP calculations; b) the *committed load* which represents the actual load of a location together with the cNAMPs that have received permission to transfer to the location, and is used by remote load servers. cNAMP and load server pseudocodes are presented in [4].

V. Comparative cNAMP and AMP Performance

Table II reports a comparison of the greedy effects for the experiments designed in Section III-B. For each scenario the first and the second rows show results of AMP and cNAMP experiments respectively.

The results in Table II show that *even simple negotiation in cNAMPs significantly reduces the number of movements* (4^{th} *and* 6^{th} *columns) and time to rebalance* (3^{rd} *and* 5^{th} *columns). cNAMPs do not make redundant movements during initial distribution* (4^{th} *column)*, and all scenarios show at least three times faster initial balancing in comparison with AMPs (3^{rd} column), e.g. dropping from 60.4s to 14.7s in Scenario 1.

During rebalancing after an AMP/cNAMP termination, cNAMPs make far fewer redundant movements (6^{th} *column).* The vast majority of experiments in scenarios 1 and 3 show that cNAMPs do not make redundant movements to rebalance, and cNAMP rebalancing takes less then half of the time of AMP rebalancing (5^{th} column)., e.g. to rebalance 19 AMPs take 28.2s, and 19 cNAMPs take 7.8s in Scenario 2.

cNAMPs require less execution time than AMPs (7^{th} *column).* In experiments considered mean cNAMP execution

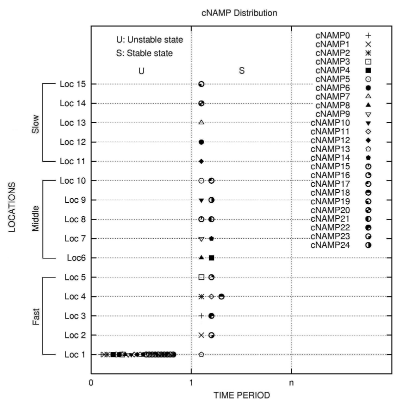

Figure 2: Initial distribution and rebalancing after a cNAMP termination (scenario 1)

Table II: Comparative summary of AMP and cNAMP greedy effects

Configuration and type of experiment		Initial distribution		Rebalancing after an AMP/cNAMP termination		Large AMP/cNAMP execution time, (sec)	
		Time, (sec)	Mean No. redun. moves	Time, (sec)	Mean No. redun. moves	Mean	St. dev.
Scen. 1	AMPs	60.4	64	22.5	6	173.8	7.66
	cNAMPs	14.7	-	5.9	-	104.8	12.9
	Reduction	4.11	64	3.81	6	1.65	
Scen. 2	AMPs	50.5	43	28.2	11	182.1	11.5
	cNAMPs	12.4	-	7.8	1	113.6	9.43
	Reduction	4.07	43	3.62	10	1.6	
Scen. 3	AMPs	26.8	13	14.1	6	232.6	9.91
	cNAMPs	8.5	-	5.6	-	142.2	4.97
	Reduction	3.15	13	2.52	6	1.64	

time is at most 3/5 of the mean AMP execution time, e.g. mean execution time of 10 AMPs is 232.6s, whereas mean execution time of 10 cNAMPs is 142.2s in Scenario 3.

Figure 2 shows initial cNAMP distribution. As before we do not show the cNAMP movements from state U1 to state S1. Figure 2 should be compared with Figure 1 in Section III. More examples of a cNAMP distribution are given in [4]. The results show that cNAMPs only display location blindness (Section II), which does not increase cNAMP execution time.

VI. Conclusion and Future Work

We have shown that like other distributed load managers, collections of AMPs exhibit thrashing, or greedy effects. We identify two types of redundant movements: location thrashing causes additional movements *and* increases AMP execution time; location blindness causes only additional movements (Section II).

We have simulated the greedy effects in an AMP implementation, and shown that each AMP makes on average five redundant movements during execution for the scenarios considered. Although greedy effects have limited impact on networks with a small number of AMPs, few locations, or small AMPs, their effects increase as any of these factors scale. The analysis of the redundant movement types and the reasons they occur have showed that redundant movements are mainly caused by location thrashing (Section III).

To reduce location thrashing we have introduced the concept of negotiating AMPs, described and implemented AMPs that negotiate with a competitive scheme, so called cNAMPs. By negotiation we only mean a coordination among competitive and self-interested agents. The key differences between cNAMPs and AMPs are as follows: before the movement a cNAMP sends a request to the target location; *and* each location has two values for the number of cNAMPs, one that is used by local cNAMPs, and another that is published to other locations (Section IV).

cNAMP simulations exhibit only location blindness. cNAMPs do not make redundant movements during initial distribution, and all scenarios show at least three times faster initial balancing in comparison with AMPs. During rebalancing after an AMP/cNAMP termination, cNAMPs make far fewer redundant movements, and the cNAMP rebalancing takes less then half of the time of AMP rebalancing. cNAMPs require less execution time than AMPs: mean cNAMP execution time is at least 1.6 times less than mean AMP execution time (Section V).

A mathematical analysis of location blindness on homogeneous and heterogeneous networks is in preparation to estimate maximum number, and probability of, redundant movements. In the longer term we plan to investigate cNAMP behaviour on wide area networks.

References

[1] X. Y. Deng, G. J. Michaelson, and P. W. Trinder, "Cost-driven autonomous mobility," *Computer Languages Systems and Structures*, vol. 36, no. 1, pp. 34–59, 2010.

[2] N. Chechina, P. King, R. Pooley, and P. Trinder, "Simulating autonomous mobile programs on networks," in *PGNet'09*. Liverpool, UK: Liverpool John Moores University, 2009, pp. 201–206.

[3] L. M. Ni, C.-W. Xu, and T. B. Gendreau, "A distributed drafting algorithm for load balancing," *IEEE Trans. Softw. Eng.*, vol. 11, no. 10, pp. 1153–1161, 1985.

[4] N. Chechina, P. King, and P. Trinder, "Reducing redundant autonomous mobile program movements by negotiation," Heriot-Watt University, Edinburgh, UK, Tech. Rep. 0072, 2010.

[5] H. Kuolin, "Allocation of processors and files for load balancing in distributed systems," Ph.D. dissertation, University of California at Berkeley, USA, 1985.

[6] A. Ghafoor and I. Ahmad, "An efficient model of dynamic task scheduling for distributed systems," in *COMPSAC '90*. IEEE Computer Society Press, October 1991, pp. 442–447.

[7] A. Ross and B. McMillin, "Experimental comparison of bidding and drafting load sharing protocols," in *DMCC '90*, vol. 2. IEEE Computer Society Press, 1990, pp. 968–974.

[8] M. Livny and M. Melman, "Load balancing in homogeneous broadcast distributed systems," in *Proceedings of the Computer Network Performance Symposium*. New York, NY, USA: ACM, 1982, pp. 47–55.

[9] T. L. Casavant and J. G. Kuhl, "Analysis of three dynamic distributed load-balancing strategies with varying global information requirements," in *DCS '87*. New York, USA: IEEE Press, 1987, pp. 185–192.

[10] G. Weiss, Ed., *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. Massachusetts, USA: The MIT Press, 1999.