Computing MAS Dynamics Considering the Background Load

Maciej Smołka[1,⋆] and Robert Schaefer[2]

[1] Institute of Computer Science, Jagiellonian University, Kraków, Poland
smolka@ii.uj.edu.pl
[2] Department of Computer Science,
Stanisław Staszic University of Science and Technology, Kraków, Poland
schaefer@agh.edu.pl

Abstract. The paper extends the formal model of a computing Multi Agent System introduced in our previous papers to the case in which the background load coming from operating systems activities as well as other applications is included. Results concerning the existence of the optimal scheduling strategy as well as the characterization of such strategies have been obtained. The theorems partially verify scheduling heuristics (diffusion rules) designed and tested for large scale CAE computations.

1 Introduction

Multi-Agent Systems (MAS) dedicated to the distributed computing have been developed for the last several years (see. e.g. [1]). A formal model of such systems was introduced in our earlier papers [2, 3, 4]. It enables us to formulate a new definition of the optimal scheduling problem which is well suited to the diffusion-type distributed government of computing agents. We extend this model to the case in which the background load coming from operating systems activities as well as other applications is included. Results concerning the existence of the optimal scheduling strategy as well as the characterization of such strategies have been obtained. Such theorems partially verify scheduling heuristics (agent diffusion rules) designed and tested for large scale CAE computations [5, 6].

2 Properties of a Computing MAS

The suggested architecture of the system is composed of: *a computational environment* (MAS platform) and *a computing application* being a collection of mobile agents called *Smart Solid Agents* (SSA). The computational environment is a triple $(\mathbf{N}, B_H, perf)$, where:

$\mathbf{N} = \{P_1, \ldots, P_N\}$, where P_i is a Virtual Computation Node (VCN). Each VCN can maintain more than one agent.

⋆ This author has been supported by the State Committee for Scientific Research of the Republic of Poland under research grants 2 P03A 003 25 and 7 T07A 027 26.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 799–806, 2006.

B_H is the connection topology $B_H = \{\mathcal{N}_1, \ldots, \mathcal{N}_N\}, \mathcal{N}_i \subset \mathbf{N}$ is an immediate neighborhood of P_i (including P_i as well).

$perf = \{perf_1, \ldots, perf_N\}, perf_i : \mathbb{R}_+ \to \mathbb{R}_+$ is a family of functions, which describes relative performance of all VCN's with respect to the total memory request M^i_{total} of all allocated agents. If M^i_{total} on P_i is small, $perf_i$ turns back the constant value, which depends only on the CPU architecture. If M^i_{total} is larger, $perf_i$ decreases due to the intensive swap utilization.

The MAS platform is responsible for maintaining the basic functionalities of the computing agents.

- It delivers the information about the local load concentration $\mathbf{L}_j$ and Q_j (see (2) and (3) below),
- It performs the agent migration between the neighboring VCN's,
- It executes the agent partitioning and destruction,
- It supports the transparent communication among agents.

We shall denote an SSA by A_i where index i stands for an unambiguous agent identifier. Each A_i contains its computational task and all data necessary for its computations. Another component of an agent is a shell which is responsible for the agent logic. An SSA can denominate the pair (E_i, M_i) where E_i is the remaining computation time measured in units common for all agents of an application and M_i is the agent's RAM requirement in bytes. An agent may undertake autonomously the following actions:

- Continue executing its internal task,
- Migrate to a neighboring VCN if it founds better resources there (e.g. the new VCN is significantly less loaded),
- Decide to be partitioned, which results in creating two child agents $\{A_{i_j} = (T_{i_j}, S_{i_j})\}, j = 1, 2$. We assume that in the case of the agent partitioning the following conditions holds: $E_i > E_{i_j}$, $M_i > M_{i_j}$, $j = 1, 2$. The parent SSA dies after such a partition.

A computing application may be characterized by the triple $(\mathbf{A}_t, G_t, Sch_t)$, $t \in [0, +\infty)$ where:

$\mathbf{A}_t$ is the set of application agents, $\mathbf{A}_t = \{A_{\xi_j}\}_{\xi_j \in I_t}$, I_t is the set of indices of agents active at the time t.

G_t is the tree representing agents partitioning at the time t. All agents constitute the set of nodes $\bigcup_{\xi \in \Theta} A_\xi$, $\Theta = \bigcup_{j=0}^{t} I_j$, while G_t edges show the partitioning history. All information on how to rebuild G_t is spread among all agents such that each of them knows only its neighbors in the tree.

$\{Sch_t\}_{t \in [0,+\infty)}$ is the family of functions such that $Sch_t : \mathbf{A}_t \to \mathbf{N}$ is the current schedule of application agents among the MAS platform servers. The function is represented by the sets ω_j of agents' indices allocated on each $P_j \in \mathbf{N}$. Each of ω_j is locally stored and managed by P_j.

Each server $P_j \in \mathbf{N}$ periodically asks all local agents (allocated on P_j) for their requirements and computes the local load concentration

$$L_j = \frac{E^j_{total}}{perf_j(M^j_{total})} \text{ where } E^j_{total} = \sum_{i \in \omega_j} E_i \text{ and } M^j_{total} = \sum_{i \in \omega_j} M_i \tag{1}$$

Then P_j communicates with neighboring servers and establishes

$$\mathbf{L}_j = \{(L_\zeta, E^\zeta_{total}, M^\zeta_{total}, perf_\zeta)\} \text{ where } \zeta \text{ is such that } P_\zeta \in N_j \tag{2}$$

as well as the set of node indices Q_j such that

$$k \in Q_j \iff k \neq j,\ P_k \in N_j,\ L_j - L_k > 0 \tag{3}$$

The detailed description of the architecture and the agent governing strategies mentioned above as well as their current application may be found in our earlier papers (see e.g. [2, 5, 6]). The agent migration and partitioning rules described in these papers are related to the phenomenon of the molecular diffusion in crystals. The agent A_i migrates along the gradient of its binding energy which depends mainly on its remaining computation time E_i and the local load concentration [2].

3 The System State and the Background Load

In this section we recall our formal mathematic model for multi-agent computations. It was already presented in [2] and further developed in [3, 4], so we shall use the notations introduced therein. Denote by $\mathbf{A}$ the set of all agents of an application. Recall the notion of the *vector weight of an agent* which is the mapping $w : \mathbb{N} \times \mathbf{A} \longrightarrow \mathbb{R}^2_+$ whose components are E_i and M_i as above. Assume that the dependency of the total weight of child agents after partition upon their parent's weight before partition is well-known, componentwise and linear, i.e. we know the constants $c_1, c_2 \geq 0$ such that in the case of partition $A \to \{A_1, A_2\}$ we have

$$w^i_{t+1}(A_1) + w^i_{t+1}(A_2) = c_i w^i_t(A)$$

for $i = 1, 2$. Such an assumption seems realistic, in simple cases we can have $c_1 = c_2 = 1$ but in general the constants may be either greater or less than 1.

In order to avoid considering a single agent's dynamics we define a global quantity describing our computing system as a whole. Namely recall the notion of the *total weight of all agents allocated on a virtual node P at any time t*, i.e.

$$W_t(P) = \textstyle\sum_{Sch_t(A)=P} w_t(A).$$

We put 0 if no agent is maintained by P.

In the sequel we shall assume that the number of virtual nodes $\sharp\mathbf{N} = N$ is *fixed*. Thus we can consider W_t as a nonnegative vector in $\mathbb{R}^{2N}$ such that

$$W^j_t = W^1_t(P_j), \quad W^{N+j}_t = W^2_t(P_j)$$

for $j = 1, \ldots, N$. We shall treat W_t as a *state of the computing MAS at a time t*. Considering its dynamics we assume first of all that in an idealized situation when migrations and partitions do not occur W_t evolves according to the following recursive equation

$$W_{t+1} = F(W_t, \xi_t) \tag{4}$$

where ξ_t is a given sequence of random variables (a stochastic process) representing the *background load*. (4) means that W_t is itself a discrete stochastic process. $W_t = 0$ means that our computations are finished so we do not want the system to leave this state. This leads to the assumption that for every t we have

$$F(0, \xi_t) = 0 \tag{5}$$

with probability 1.

As for ξ_t, we assume that it takes the finite number of values in the cube $[0, M]^N$. Its i-th coordinate $(\xi_t)_i$ may be interpreted as an additional load concentration on the i-th VCN, which may slow down computations performed by agents. We shall distinguish the following background load behaviors expressed by the conditions imposed on the sequence $(\xi_t)_{t=0,1,2,\ldots}$.

- $\xi(1)$ The momentary background load changes rapidly many times during a single time period of the model. It may be generated by the basic activities of operating systems and network protocols. We assume that all ξ_t have the same distribution for each time period and they are mutually independent. This case has been considered in [4].
- $\xi(2)$ The variables ξ_t are also mutually independent but have different, well-known probability distributions. The distributions may change periodically after each T time steps of the model. Such a background load may be caused by small tasks which are executed in the period comparable with the time step of the model.
- $\xi(3)$ The sequence $(\xi_t)_{t=0,1,2,\ldots}$ constitutes a stationary Markov chain. This is the case of the system additionally loaded by a set of similar tasks whose execution time is comparable to the several steps of the model. The large number of such tasks in a particular time step increases the probability of the high background load in the next step.
- $\xi(4)$ The variables ξ_t constitute a nonstationary, periodical Markov chain with period T. Such a situation may be related to the case of background tasks with the long execution time (much longer then the time step of the model) that appear according to a well-known trend. Such a situation is considered e.g. in [7].

4 The MAS Dynamics

Now we shall formulate the full equations of evolution of W_t. To this end we need to put into (4) terms representing migrations and partitions. A detailed

description of this process is presented in [4], here we show only the final step. Namely we have

$$\begin{cases} W^i_{t+1} = F^i(\tilde{W}_t, \xi_t) + c_1\ u^1_{ii,t}(W_t)\ W^i_t + \sum_{j\neq i} u^1_{ji,t}(W_t)\ W^j_t \\ W^{N+i}_{t+1} = F^{N+i}(\tilde{W}_t, \xi_t) + c_2\ u^2_{ii,t}(W_t)\ W^{N+i}_t + \sum_{j\neq i} u^2_{ji,t}(W_t)\ W^{N+j}_t \end{cases} \tag{6}$$

for $i = 1, \ldots, N$, where

$$\tilde{W}^i_t = \left(1 - \sum_{k=1}^N u^1_{ik,t}(W_t)\right) W^i_t, \quad \tilde{W}^{N+i}_t = \left(1 - \sum_{k=1}^N u^2_{ik,t}(W_t)\right) W^{N+i}_t$$

with initial conditions

$$W_0 = \hat{W}. \tag{7}$$

$u^m_{jj,t} : \mathbb{R}^N_+ \to [0,1], m = 1,2$ are the proportions of the weight components of splitting agents to the corresponding components of the total weight of all agents at node j at time t and $u^m_{jk,t}$ for $j \neq k$ are the respective proportions of the weight of agents migrating from j to k. Thus the first term on the right-hand side of (6) corresponds to the 'established' evolution of agents which neither migrate nor split, the second is the weight of partitioned agents and the third is the weight gathered as a result of the immigration. It follows that our W_t is a *controlled stochastic process* with a *control strategy*

$$\pi = (u^1_t, u^2_t)_{t\in\mathbb{N}} \tag{8}$$

such that $u^m_t : \mathbb{R}^{2N}_+ \longrightarrow U$. The control set U contains matrices $\alpha \in [0,1]^{N\times N}$ that satisfy at least the following conditions

$$\alpha_{ij} \cdot \alpha_{ji} = 0 \text{ for } i \neq j, \quad \alpha_{i1} + \cdots + \alpha_{iN} \leq 1 \text{ for } i = 1, \ldots, N. \tag{9}$$

The first equation in (9) can be interpreted in the following way: *at a given time migrations between two nodes may happen in only one direction.* The second equality says that *the number of agents leaving a node must not exceed the number of agents present at the node just before the migration.*

In practice it may be appropriate to impose some stronger conditions on a control set (detailed considerations about that can be found in [4]). Denote by $G(W, u, \xi)$ the right hand side of (6) with $u = (u^1, u^2)$. Then we can rewrite our equation of evolution in the following way

$$W_{t+1} = G\left(W_t, u_t(W_t), \xi_t\right). \tag{10}$$

It is easy to see that thanks to (5) we have also that

$$G(0, u(0), \xi_t) = 0 \tag{11}$$

with probability 1 for any admissible u and any $t \in \mathbb{N}$.

In the most general case one could take the whole $\mathbb{R}^{2N}_+$ as the state space S of W_t. But in reality both the available RAM and the maximal estimated time

of computations are bounded and quantized, so it is not unnatural to consider the state space *finite*.

The character of the state dynamics changes for different types of the background load. Detailed considerations on the case $\xi(1)$ are contained in [4]. In short, it can be shown that in this case W_t is a *stationary controlled Markov chain*. The case $\xi(2)$ is similar, but this time W_t is not stationary. In fact when ξ_t is T-periodic, W_t is a *T-periodic Markov chain* (for every u). In both cases (11) means that 0 is *an absorbing state* (cf. [8]) of W_t.

When ξ_t is itself a Markov chain, we can no longer say the same about W_t. We can only show that the latter is a *second-order Markov chain* (i.e. a process with two-step memory). Such a process can be transformed into a Markov chain by means of a simple classical transformation of the state variable (see e.g. [8]. Namely we should put

$$X_t = \begin{bmatrix} W_{t-1} \\ W_t \end{bmatrix}, \quad X_0 = \begin{bmatrix} \hat{W} \\ \hat{W} \end{bmatrix}. \tag{12}$$

It can be shown that the state equations for X_t have the form

$$X_{t+1} = \begin{bmatrix} X_t^2 \\ G(X_t^2, u_t(X_t^2), \xi_t) \end{bmatrix}. \tag{13}$$

In the case $\xi(3)$ such X_t is simply a stationary controlled Markov chain, and, again, in the case $\xi(4)$ it is T-periodic provided ξ_t is T-periodic. From the form of (13) and from (11) it follows that also in these cases 0 is an absorbing state.

Note that when W_t (or X_t) is T-periodic, it can be transformed into a stationary process by introducing a new state $Y_t = [W_{Tt}, \ldots, W_{T(t+1)-1}]$. In the sequel we shall consider only stationary Markov processes, in other words the cases $\xi(1)$ and $\xi(3)$. But when we have a background process of type $\xi(2)$ or $\xi(4)$ which is periodic, we can apply the above transformation and all the following results shall remain true.

5 The Optimal Scheduling Problem

The general form of the cost functional for controlled Markov chains of our type is (cf. [8])

$$V(\pi; \hat{W}) = E\left[\sum_{t=0}^{\infty} k(W_t, u_t(W_t))\right] \tag{14}$$

where π is a control strategy (8) and $\hat{W}$ is the initial state of W_t. Since 0 is an absorbing state we shall always assume that $k(0, \cdot) = 0$, i.e. remaining at 0 has no cost (it guarantees that the overall cost can be finite). In the case $\xi(3)$ we need to rewrite (14) in the following manner.

$$\bar{V}(\pi; X_0) = E\left[\sum_{t=0}^{\infty} k(X_t^2, u_t(X_t^2))\right] \tag{15}$$

Note that (15) inherits good properties (such as putting no cost on remaining at 0 or satisfying assumption A(3) of Prop. 1) from (14), so in the sequel we can safely substitute X_t for W_t and $\bar{V}$ for V and all the results shall remain true.

We define the following set of admissible controls $\mathbf{U} = \{\pi : u_t^m(s) \in U_s, m = 1, 2\}$. where U_s is a *closed* subset of U (see (9)). Now we are ready to formulate the *optimal scheduling problem*. Namely given an initial configuration $\hat{W}$ we search for such control strategy $\pi^* \in \mathbf{U}$ that

$$V(\pi^*; \hat{W}) = \min\{V(\pi; \hat{W}) : \ \pi \in \mathbf{U}, W_t \text{ is a solution of } (10), (7)\}. \tag{16}$$

Consider some cost functionals which seem appropriate for multi-agent computations. The first one is the expected total time of computations

$$V_T(\pi; \hat{W}) = E\big[\inf\{t \geq 0 : W_t = 0\} - 1\big]. \tag{17}$$

We can rewrite it in the form (14) if we put $k(s, \alpha) = 1$ for $s \neq 0$ and any α.

The second example takes into account the mean load balancing over time. It has the following form

$$V_L(\pi; \hat{W}) = E\left[\sum_{t=0}^{\infty} \sum_{i=1}^{N} (L_t^i - \overline{L}_t)^2\right] \tag{18}$$

where $L_t^i = \frac{W_t^i}{perf_i(W_t^{N+i})}$ is the load concentration and $\overline{L}_t = \frac{1}{N} \sum_{i=1}^{N} L_t^i$ is its mean over all computing nodes. This time the form of k is straightforward.

Both the above examples do not contain an explicit dependency on the control. Generalizing it a little allows us to penalize migrations. Namely take $\varphi : S \to \mathbb{R}_+$, $a \geq 0$, $\varrho_{ij}^m \geq 0$ and $\mu_{ij}^m : [0, 1] \to \mathbb{R}_+$ continuous nondecreasing and such that $\mu_{ij}^m(0) = 0$, and put

$$V_M(\pi; \hat{W}) = E\left[\sum_{t=0}^{\infty} \left(\varphi(W_t) + a \sum_{m=1}^{2} \sum_{i \neq j} \varrho_{ij}^m \mu_{ij}^m(u_{ij,t}^m(W_t))\right)\right]. \tag{19}$$

6 Results

Now let us revisit some theoretical results that were proven for our model in [4] and extend them for considered types of the background load. First consider the existence of optimal solutions for problem (16). Let us recall that in [4] we used the notation $R(u)$ for the 'probably not absorbing' part of the transition matrix for our system while applying control u and by $R^n(u)$ the analogous part of n-step transition matrix obtained by applying the stationary control $u_t = u$ (for details on this notion see [4] or [8]). The following proposition is a straightforward consequence of [8, Theorem 4.2].

Proposition 1. *Assume that A(1) and (A(2) or A(3)) hold.*

A(1) $k(s, \cdot)$ *is continuous on* $U_s \times U_s$
A(2) $R^K(u)$ *is a contraction for every* u *such that* $u(s) \in U_s \times U_s$
A(3) $R^n(u)$ *is a contraction for some* $n \geq 1$ *and* u *as above but additionally* $k(s, \alpha) \geq \varepsilon > 0$ *for* $s \neq 0$, $\alpha \in U_s \times U_s$.

Then there exists the unique optimal solution of (16).

This proposition allows us to prove the existence of optimal solutions for the cost functionals V_T, V_L and V_M. In case of V_T we need the assumption A(3), in case of V_L the assumption A(2) and in case of V_M either A(3) or A(2) depending on whether φ is separated from 0 or not (cf. [4]).

Now we shall recall some Bellman-type optimality conditions for (16). Let us denote the elements of the state space S by s_i, $i = 0, \ldots, K$. The following proposition is another consequence of [8, Theorem 4.2] and its proof.

Proposition 2. *Make the same assumptions as in Proposition 1. Then the optimal solution of* (16) *is a stationary strategy* $\pi^* = u^\infty = (u, u, \ldots)$ *and it is the unique solution of the equation*

$$V(\pi^*; s) = \min_{\alpha \in (U_s)^2} \left[\sum_{j=1}^{K} P(G(s_i, \alpha, \xi_0) = s_j) V(\pi^*; s_j) + k(s, \alpha) \right]. \quad (20)$$

The solution of (20) *exists and it is the optimal solution of* (16).

7 Conclusions

The presented MAS technology as well as the diffusion scheduling of agents give the advantageous environment for large-scale distributed computations (see [5]). The presented formal model based on the discrete stochastic field enables us to formulate a new definition of the optimal scheduling in such an environment. It have been shown (Propositions 1 and 2), that there exists an optimal diffusion scheduling strategy in the class of stationary rules that are intensively utilized during tests [5, 6]. This extends the earlier results [4] to the case in which the background load caused by operating systems and different kinds of other applications is taken into account.

References

1. Luque, E., Ripoll, A., Cortés, A., Margalef, T.: A distributed diffusion method for dynamic load balancing on parallel computers. In: Proceedings of EUROMICRO Workshop on Parallel and Distributed Processing, San Remo, Italy, IEEE Computer Society Press (1995) 43–50
2. Smołka, M., Grochowski, M., Uhruski, P., Schaefer, R.: The dynamics of computing agent systems. Lecture Notes in Computer Science **3516** (2005) 727–734
3. Grochowski, M., Smołka, M., Schaefer, R.: Architectural principles and scheduling strategies for computing agent systems. Fundamenta Informaticae (2006) accepted.
4. Smołka, M.: Optimal scheduling problem for computing agent systems. Inteligencia Artificial (2006) accepted.
5. Grochowski, M., Schaefer, R., Uhruski, P.: Diffusion based scheduling in the agent-oriented computing systems. Lecture Notes in Computer Science **3019** (2004) 97–104
6. Momot, J., Kosacki, K., Grochowski, M., Uhruski, P., Schaefer, R.: Multi-agent system for irregular parallel genetic computations. Lecture Notes in Computer Science **3038** (2004) 623–630
7. Lepiarz, M., Onderka, Z.: Agent system for load monitoring of the heterogeneous computer network. Lecture Notes in Computer Science **2328** (2002) 364–368
8. Kushner, H.: Introduction to Stochastic Control. Holt, Rinehart and Winston (1971)