Towards an Artificial Hormone System for Self-Organizing Real-Time Task Allocation

Uwe Brinkschulte, Mathias Pacher, and Alexander von Renteln

Institute for Process Control, Automation, and Robotics
University of Karlsruhe (TH), Germany
{brinks, pacher, renteln}@ira.uka.de

Abstract. This article presents the concept of an artificial hormone system for a completely decentralized realization of self-organizing task allocation. We show that tight upper bounds for the real-time behavior of self-configuration can be given. We also show two simulation results using the artificial hormone system demonstrating the operation of the artificial hormone system under different workloads.

1 Introduction

Today's computational systems are growing increasingly complex. They are build from large numbers of heterogeneous processing elements with highly dynamic interaction. Middleware is a common layer in such distributed systems, which manages the cooperation of tasks on the processing elements and hides distribution to the application. It is responsible for seamless task interaction on distributed hardware. As shown in figure 1, all tasks are interconnected by the middleware layer and are able to operate beyond processing element (PE) boundaries like if they would reside on a single hardware platform. To handle the complexity of today's and even more tomorrow's distributed systems, self-organization techniques are necessary. Such a system should be able to find a suitable initial configuration by itself, to adapt or optimize itself to changing environmental and internal conditions, to heal itself in case of system failures or to protect itself against attacks. Middleware is a good place to realize such self-X features (self-configuration, self-optimization, self-healing) by autonomously controlling and adapting task allocation. Especially for self-healing, it is important that task allocation is decentralized to avoid single points of failure.

This work presents an artificial hormone system for task allocation to heterogeneous processing elements. In the following, we will present our approach in detail and we will discuss several properties considering real-time aspects induced by the hormone system.

2 Using an Artificial Hormone System to obtain Self-X-Properties

For task allocation, three types of hormones are used:

Eager value: This hormone determines, how well a PE can execute a task. As higher the hormonal value, as better the task executes on the PE.

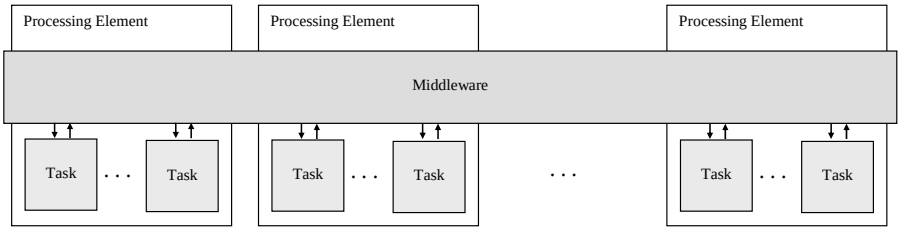

Fig. 1. Middleware in a distributed system

Suppressor: A suppressor represses the execution of a task on a PE. Suppressors are subtracted from eager values. Suppressors are e.g. used to prevent duplicate task allocation or to indicate a detoriating PE state.

Accelerator: An accelerator favors the execution of a task on a PE. Accelerators are added to eager values. The accelerators can be used to cluster cooperating tasks in the neigborhood or to indicate an improved PE state.

The following figure 2 sketches the basic control loop used to assign a task T_i to a processing element. This closed control loop is executed for every task on every processing element. It determines based on the level of the three hormone types, if a task T_i is executed on a processing element PE_γ or not. The local static eager value $E_{i\gamma}$

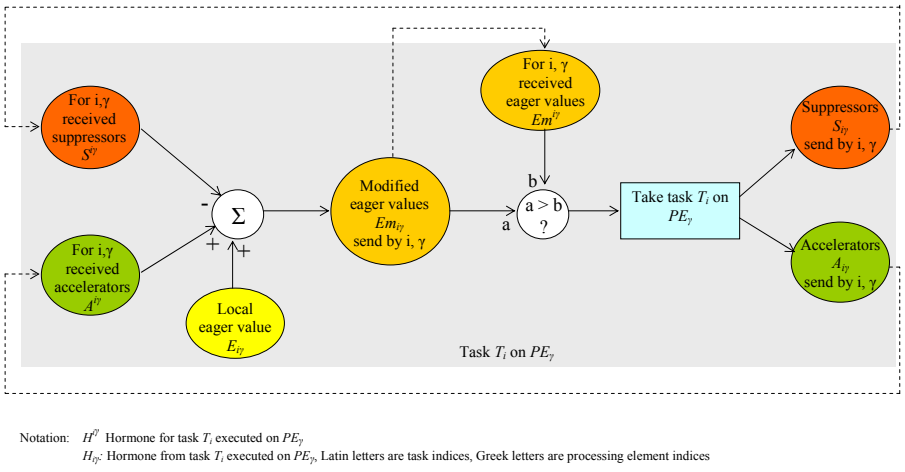

Fig. 2. Hormon based control loop

indicates how well task T_i executes on PE_γ. From this value, all suppressors $S^{i\gamma}$ received for task T_i on PE_γ are subtracted and all accelerators received for task T_i on PE_γ are added. The result of this calculation is a modified eager value $Em_{i\gamma}$ for task

T_i on PE_γ. The modified eager value is sent by the middleware to all other PEs in the system and compared to the modified eager values $Em^{i\gamma}$ received from all other PEs for this task. Is $Em_{i\gamma}$ greater than all received eager values $Em^{i\gamma}$, task T_i will be taken by PE_γ (in case of equality a second criterion, e.g. the position of a PE in the grid, is used to get an unambiguous decision). Now, task T_i on PE_γ sends suppressors $S_{i\gamma}$ to all other PEs to prevent duplicate task allocation. Accelerators $A_{i\gamma}$ are sent to neighbored PEs to favor the clustering of cooperating tasks. This procedure is repeated periodically.

It should be emphasized in this point that the strength of the different types of hormones is initially set by the applicants who want to influence the task allocation. In section 3.1 we show the task allocation process based on the hormone values in detail.

The described approach is completely decentralized, each PE is responsible for its own tasks, the communication to other PEs is realized by a unified hormone concept. Furthermore, it realizes several self-X properties:

- The approach is **self-organizing**, because no external influence controls the task allocation.
- It is **self-configuring**, an initial task allocation is found by exchanging hormones. The self-configuration is finished as soon as all modified eager values become zero meaning no more tasks wants to be taken. This is done by sending suppressors which have to be chosen strong enough to inhibit an infinite task assignment.
- The **self-optimization** is done by offering tasks again for re-allocation. The point of time for such an offer is determined by the task respectively the PE itself. It can be done periodically or at a point of time where the task or the PE is idle.
  In this context it is simple to handle the entrance of a new task in the system: At first, all processing elements have to be informed about their hormone values for the new task. Then, the task is allocated as described for self-optimization.
- The approach is **self-healing**, in case of a task or PE failure all related hormones are no longer sent, especially the suppressors. This results in an automatic reassignment of the task to the same PE (if it is still active) or another PE.

In addition, the self-configuration is **real-time** capable. There are tight upper time bounds for self-configuration which we will present in the next sections.

3 Dynamics of the Artificial Hormone System

In the this section, the dynamics of the artificial hormone system and the conditions and rules for its correct working will be presented. Figure 3 shows the cyclic sequence of sending the hormones followed by the task allocation. The sequence starts with "send hormones" (S) to create the knowledge base for the first decision. At least the eager values need to be available. After sending the hormones and waiting the time t_{SE}, a decision (E) on the task allocation, based on the received hormones, is taken. This process is repeated after a waiting time of t_{ES}.

3.1 Dynamics of Task Allocation

Let PE_γ be a processing element willing to run a task T_i. We need to distinguish three cases:

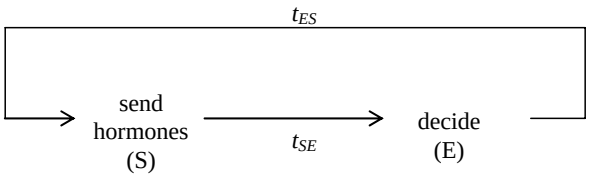

Fig. 3. Hormon cycle

Case 1: All eager values of all processing elements for task T_i are constant and spread over the whole system. Thus, the system is in a steady state and all PEs make their decisions based on up-to-date and constant values. Then, PE_γ can allocate a task if it has the highest eager value, respectively, with equal eager values, a higher priority.

Case 2: The eager value of processing element PE_γ for task T_i decreases (e.g. by suppressor influence). In this case, PE_γ may allocate the task T_i if the decreased eager value is still sufficient. All the other PEs will not allocate the task, as they know either the old or the new eager value of PE_γ which wins with both values.

Case 3: The eager value of the processing element PE_γ for task T_i increases (e.g. by accelerator influence). This case is critical if PE_γ becomes the winner by the increased eager value, because other PEs might not yet know it and therefore decide wrongly. Thus, PE_γ may only allocate the task T_i after the new eager value has successfully been submitted to all PEs and until PE_γ itself has possibly received a suppressor from another PE_δ ($\gamma \neq \delta$), which allocated the task T_i based on the old, lower eager value of PE_γ.

The question, however, is how long the waiting times should be chosen? Figure 4 shows the worst-case scenario in which PE_δ allocated the task T_i just before it has received the new eager value from PE_γ. PE_γ must not come to a decision

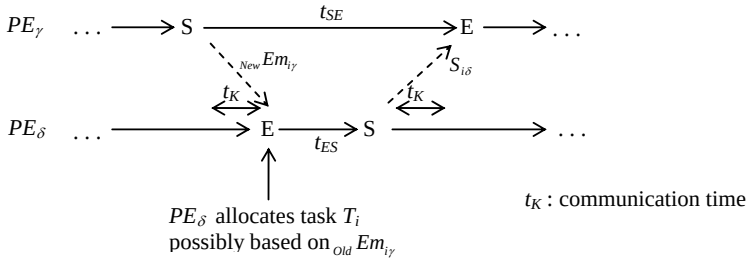

Fig. 4. Worst-case timing scenario of the hormone exchange with the task allocation

until it has received the possibly incoming suppressor from PE_δ. Therefore, the communication time t_K needed by a hormone to be spread over the whole system

is very important to be known. Knowing t_K we present a rule for the task allocation with increased eager values as well as conditions for the times t_{ES} and t_{SE}.

Rule: If a processing element PE_γ gets able to allocate a task T_i only based on an increased eager value, then it has to delay it's decision to the next communication cycle to ensure the transmission of the increased eager value and to wait for possible suppressors for the same task from other PEs. This comes true if (follows directly from figure 4):

$$t_{SE} \geq t_{ES} + 2t_K$$

The cycle time t_{Cycle} defines as follows:

$$t_{Cycle} = t_{SE} + t_{ES}$$

Of course the cycle time should be minimized, thus:

1) t_{ES} should be as small as possible, ideally 0.
2) $t_{SE} \geq t_{ES} + 2t_K$, ideally with $t_{ES} = 0$: $t_{SE} \geq 2t_K$

3.2 Self Configuration: Worst Case Timing Behavior

Figure 5 shows the precise cycle of the hormone distribution and interpretation based on figure 3. First of all, the hormones (eager values, suppressors and accelerators) for all the tasks which PE_γ is interested in are emitted by PE_γ.

After waiting the time t_{SE}, the decision for a task T_i (PE_γ is interested in) will be taken. Afterwards i is incremented and the next cycle starts ($t_{ES} = 0$). In this way the hormones for all the relevant tasks are emitted in each cycle and the decision for exactly one task will be taken. This allows the hormones to take effect. If the allocation decisions for all tasks would take place in parallel, the emitted accelerators would not have any impact (as all tasks would already be allocated in the first cycle).

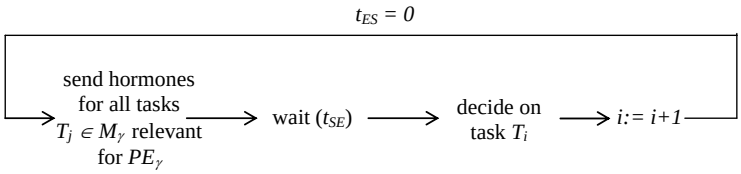

Fig. 5. Cycle of the hormone distribution and the decision making for a PE_γ

Let's assume that all m tasks have to be distributed on all PEs and all PEs are interested in all tasks.

We introduce a further assumption to simplify the scenario: Let all eager values be constant, i.e. there are no accelerators and suppressors. Then all tasks have been inspected and allocated after m cycles and it follows:

$$\text{Worst-case time behavior } = m \text{ cycles}$$

In the following we remove the simplifying assumption of constant eager values, i.e. we allow accelerators and suppressors. Now some tasks might not have been allocated after m cycles. This can be caused by accelerators and suppressors, see fig. 6. Let's assume three PEs checking one after another the possibility to allocate the task T_i. While PE_γ and PE_δ are checking PE_ϵ is the winner. After PE_δ has checked, it increases its eager value caused by a received accelerator. If afterwards PE_ϵ checks its status, PE_δ is the winner now. However, PE_δ has already checked its status regarding

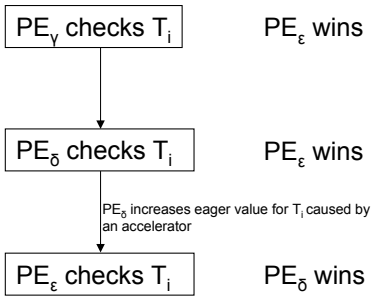

Fig. 6. Accelerator caused delay of the task allocation

task T_i and will not repeat this check within the the next m cycles.

So at worst case task T_i will not be checked again until a complete cycle of all other tasks, thus after m cycles. Afterwards the same scenario could occur again.

However, the maximal number of cycles is limited: A change of the eager value by suppressors or accelerators only takes place if a task has been allocated somewhere in the system (Assumption: Monitoring accelerators and suppressors are constant during the initial self-configuration). It follows that in each allocation cycle at least one task will be allocated. Thus, in the case of a variable eager value we get the following worst case timing behavior for the self-configuration:

$$\text{Worst Case Timing Behavior } = m^2 \text{ cycles}$$

4 Simulation Results

We started to implement a hormone simulator in order to evaluate and demonstrate the behavior of our artificial hormone system approach. The first simulations confirmed the worst-case time bound for self-configuration. We also registered that the accelerators have to be smaller than the suppressors to get a stable task allocation. The reason is that if a task is scheduled, accelerators will be submitted to the neighbor cells to allocate cooperating tasks nearby (see section 2). If these accelerators are stronger than the suppressors (which prevent the task from being allocated onto another processing element) the task allocation will not be stable. The reason is that the modified eager values will continuously increase.

In the first configuration we have chosen a grid of 64 processing elements with 64 tasks to be distributed. The tasks were grouped in 8×8 cooperating tasks. Assuming light-weight tasks, the suppressors indication processor were chosen weak. Therefore it is possible that several tasks can run on one processing element. Figure 7 shows the simulation result. Coopearting tasks were scheduled right next to each other which would lead to a small communication overhead in a real-world scenario. These clusters are not scattered at all and several tasks were scheduled onto a single processing element.

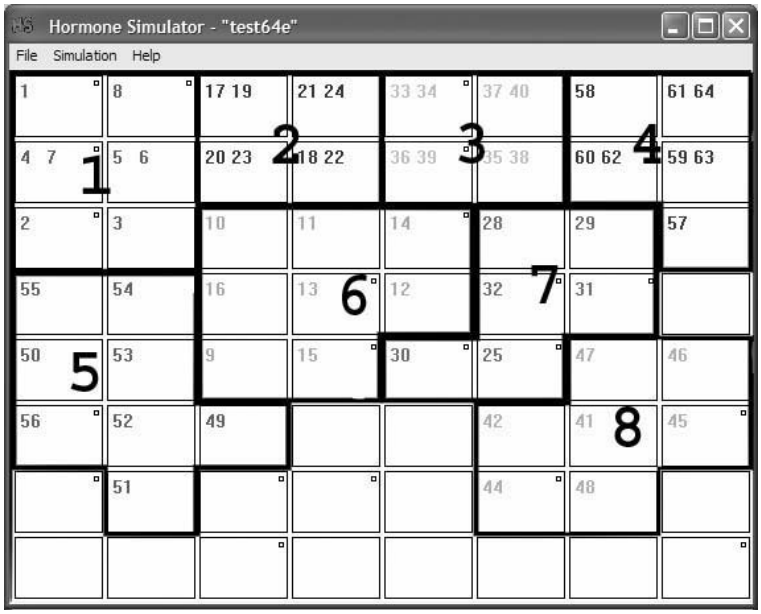

Fig. 7. 64 tasks in eight groups, weak suppressors

The second configuration is the same as the first one - the only difference is that the suppressors indicating processor load were chosen strong so that each processing element can execute exactly one task. As shown in figure 8 the related tasks form clusters again. This shows the efficiency of the artificial hormone system because, even under full load it is able to form entire clusters which are not scattered.

We also tested self-optimization by spontaneously increasing a task's eager value of a PE. As expected, the task moved to this PE if the eager value was high enough and the accelerators of the cooperating tasks were not too strong. In this way we were also able to move complete organs for optimization.

In addition, we simulated the failing of one or more PEs and the simulation results shows that the artificial hormone system was able to re-allocate the affected tasks if there were enough PEs able to take a task.

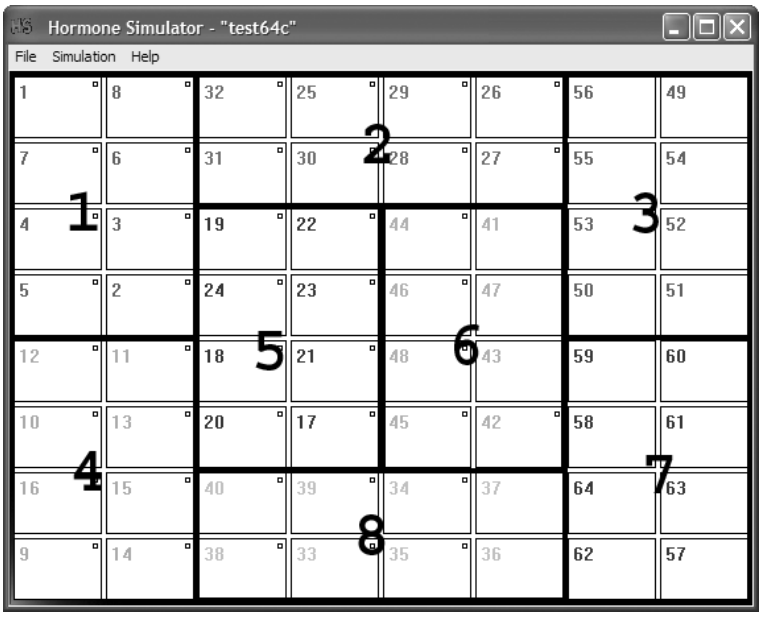

Fig. 8. 64 tasks in eight groups, strong suppressors

5 Related Work

There are several approaches for clustered task allocation in middleware. In [2], the authors present a scheduling algorithm distributing tasks onto a grid. It is implemented in the Xavantes Grid Middleware and arranges the tasks in groups. This approch is completely different from ours because it uses central elements for the grouping: The Group Manager (GM), a Process Manager (PM) and the Activity Managers (AM). Here, the GM is a single point of failure because, if it fails there is no possibility to get group information from this group anymore. In our approach there is no central task distribution instance and therefore the single point of failure can not occur.

Another approach is presented in [3]. The authors present two algorithms for task scheduling. The first algorithm, Fast Critical Path (FCP) makes sure time constrains to be kept. The second one, Fast Load Balancing (FLB) schedules the tasks so that every processor will be used. Using this strategy - especially the last one - it is not guaranteed that related tasks are scheduled nearby each other. In contrast to our approach, these algorithms do not include the failing of processing elements.

In [1], a decentralized dynamic load balancing approach is presented. Tasks are considered as particles which are influenced by forces like e.g. a load balancing force (results from the load potential) and a communication force (based on the communication intensities between the tasks). In this approach, the tasks are distributed according to the resultant of the different types of forces. A main difference to our approach is that we are able to provide time bounds for the self-configuration. Besides our approach cov-

ers self-healing which is absolutely not considered by this decentralized dynamic load balancing.

6 Conclusion and Further Work

We presented an artificial hormone system to allocate tasks to processing elements within a processor grid. The assignment is completely decentralized and holds self-X-features. Furthermore, we showed that we can guarantee tight upper bounds for the real-time behavior of the artificial hormone system for the self-configuration. We started testing the presented algorithms using a hormone simulator which confirmed the theoretical results so far.

As ongoing work, we will investigate additional quality properties of the artificial hormone system i.e. if it is possible to find time bounds for self-optimization and self-healing. We will also investigate if we can guarantee stability of the task assignment. Another question in this scope is how to find an optimal task assignment and is the artificial hormone system able to find it (if it exists)?

References

1. Hans-Ulrich Heiss and Michael Schmitz. Decentralized dynamic load balancing: The particles approach. In *Proc. 8th Int. Symp. on Computer and Information Sciences*, Istanbul, Turkey, November 1993.
2. F. R. L. Cicerre L. F. Bittencourt, E. R. M. Madeira and L. E. Buzato. A path clustering heuristic for scheduling task graphs onto a grid. In *3rd International Workshop on Middleware for Grid Computing (MGC05)*, Grenoble, France, 2005.
3. Andrei Radulescu and Arjan J. C. van Gemund. Fast and effective task scheduling in heterogeneous systems. In *IEEE Computer - 9th Heterogeneous Computing Workshop*, Cancun, Mexico, 2000.