Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
DAAAM Symposium

DYNAMIC THREAD ASSIGMENT IN A TANDEM OF THREADPOOLS INSPIRED BY THE ADAPTATION MECHANISM IN HONEYBEE FORAGING

RANDIC, M[irko]; JEDNAKOVIC, H[rvoje] & BLASKOVIC, B[runo]

Abstract: *Multithreaded server applications need to run in highly dynamic environments, where loads are continuously changing. Self-adaptation and self-optimization thus become desirable features of their behaviour. In this paper we demonstrated how to apply the honeybee recruitment mechanism to optimize the number of threads in a tandem of thread pools. Measurement of sojourn times show efficiency in processing requests that have particular characteristics related to service time distribution and CPU boundness factor.*
Key words: *multithreading, threadpool, CPU bound, self-management, honeybee foraging*

1. INTRODUCTION

Modern server applications are expected to process a huge number of requests simultaneously without noticeable degradation of performance, e.g. response times and throughput. Processing a request submitted by a client typically involves several steps that can be further divided into a number of subtasks that have to be processed in sequential order and are characterized by the resources they consume (CPUs, I/O, bandwidth, etc.). The subtasks are more or less CPU bound because different threads from a pool may occupy multiple resources simultaneously. This situation can be modelled by the CPU boundness factor r of the task that shows which proportion of overall service time belongs to pure CPU processing (Randić et al., 2010). In (Avi-Itzhak & Halfin, 1988) and (Weij et al., 2005) it is suggested that the response time and throughput of thread pooled applications greatly depends on the number of threads that are assigned to a particular pool and on the parameters of service-time distributions (mean and variance) that characterize each processing step. Various policies of assignment of a fixed number of threads to the pools are considered in (Weij et al., 2005).

In this paper we consider the possibility of implementing the recruitment mechanism that worker bees apply during foraging as a control mechanism for dynamic assignment of threads to two thread pools connected in tandem. We exploited the following similarities between honeybee foraging and the process of serving requests in a thread pooled server application:

- the entire task can be divided into two subtasks,
- the number of processing entities is constant and the entities are divided into two subgroups,
- the proportion of processing entities should be optimized to get the maximal flow of nectar i.e. the maximal request throughput.

It is important to stress that worker bees are autonomous and independent processing entities that act in a fixed rate. On the contrary, threads are not independent because of the common resource(s) they share. The degree of dependency is related to the CPU boundness factor of the tasks they are running. The honeybee recruitment mechanism gives maximal efficiency in the case of independent processing entities.

After this introduction we shall present the main characteristics of honeybee foraging. In the third section we shortly describe a model that is used to design and implement a prototype of Java server application composed of a tandem of thread pools. We also present results of measurement of sojourn times that requests spend in this application.

2. NECTAR FORAGING PROCESS

Foraging represents important collective behaviour through which insects collect food. Nectar foraging in a honeybee colony is a reliable and self-adaptive process because of its ability to perform dynamic task allocation. Moreover, foraging is achieved without any supervision or centralized or hierarchical control. To attain sufficient flexibility, a honeybee colony needs to divide the task of nectar foraging into two sequential subtasks done by two groups of workers: foragers and receivers. The first group, foragers, collect the nectar and return to the nest. However, instead of storing the material in the comb themselves, they search for a member of the second group, receiver bees, to which they directly transfer the material (Seeley, 1995). The receiver bees store the material in the comb. By appropriate task allocation, workers try to maximize the nectar flow.

2.1 Recruitment dances

In the honeybee colony recruitment tasks are represented through recruitment dances: the waggle dance (Frisch, 1967) and the tremble dance (Seeley, 1992, Seeley, 1995) in which the probability of performing a dance is related to the time it takes a forager to locate a receiver bee, i.e. its "queuing delay". Here forager bees respond to two threshold values: lower T_L and upper T_U. Above T_U or below T_L the recruitment dance is more probable to be performed than “no dance”. Seeley has presented empirical data for the threshold rule (Seeley, 1992). He demonstrated the empirical relationship between the duration of foragers' search delays and the probability of performing a dance or no dance at all.

$$\Phi_N = \Pr(\textit{does not dance} \mid D = d) = \begin{cases} \frac{dM}{T_m} & \textit{if } W \le d \le T_m \\ 2M - \frac{dM}{T_m} & \textit{if } T_m < d \le 2T_m \\ 0 & \textit{otherwise} \end{cases} \quad (1)$$

$$\Phi_W = \Pr(\textit{waggle dance} \mid D = d) = \begin{cases} 1 - \frac{dM}{T_m} & \textit{if } W \le d \le \frac{T_m}{2M} \\ \frac{1-M}{2} + \frac{M^2}{2M-1}\left(1 - \frac{d}{T_m}\right) & \textit{if } \frac{T_m}{2M} < d \le T_m \\ (1-M)\left(\frac{1}{2} + \frac{M}{2M-1}\left(1 - \frac{d}{T_m}\right)\right) & \textit{if } T_m < d \le 2T_m - \frac{T_m}{2M} \\ 0 & \textit{otherwise} \end{cases} \quad (2)$$

$$\Phi_T = \Pr(\textit{tremble dance} \mid D = d) = 1 - \Phi_N - \Phi_W \quad (3)$$

In his paper (Anderson, 1998) Anderson uses the approximation of this empirical data in the form of linearized probability functions with three parameters: M, T_m and W.

$M \in [0.5, 1]$ represents maximum probability of not dancing. $T_m \in [0, \infty]$ is the delay when the probability of not dancing is maximized. $W \in [0, \infty]$ is a constant search delay. The formalized threshold rule (Anderson 1998) is: "perform a waggle dance with prob. Φ_W as defined in (2)", "perform a tremble dance with probab. Φ_T as defined in (3)", or "perform no dance with probab. Φ_N as defined in (1)".

2.2 The main simulation result

In order to better understand the dynamics of the honeybee recruitment mechanism, we made simulations with the help of the SeSam tool http://ki.informatik.uni-wuerzburg.de/~sesam/. It provides a generic environment for modelling and experimenting with agent-based simulation. We have defined a model of a worker bee in the form of a SeSam activity diagram. For details see (Jednaković, 2010). The simulation, whose results are depicted in Fig. 1, was run with a changeable parameter of the nectar resource position. In fact, the value of this parameter was changed three times. The change of the resource position prolongs or shortens the duration of the forager tasks proportionally. Simulation results show that the recruitment mechanism establishes a new, optimal proportion of the forager and receiver very quickly.

3. EXPERIMENTAL SETUP

Our server application implements two thread pools with very large buffers to avoid rejections. The first pool consists of c_1 threads dedicated to processing the 1st subtask while the second pool consists of c_2 threads dedicated to processing the 2nd subtask. The total number of threads is fixed $C = c_1 + c_2 =$ const. The number of threads assigned to each pool can vary in time as a result of the implemented recruitment mechanism. This mechanism can be switched on or off. The recruitment mechanism is implemented as a dancing controller (Java class) connected to the buffer of the second pool. Request waiting times in the buffer are measured and averaged for groups of 20 requests. These average values d are used as input to calculate dance probabilities according to functions (1), (2) and (3). Recruitment decisions can be: "transfer a thread from 1st to 2nd pool", "transfer a thread form 2nd to 1st pool", or "no action".

A client that mimics the user request generator is connected to the server via 100 Mb/s LAN. It generates requests with Poisson arrival with intensity λ = 1/0.3 (req/sec) and exponential (c_v = 1) or gamma (c_v = 6) distributions of service times. Furthermore, requests are characterized by CPU boundness factor r. In the following two diagrams measured sojourn times are displayed as a function of parameter $p = T_{s2}/T_{s1}$, where p represents the proportion of service times of the two subtasks implied by requests. For each measurement C=10 and the probability function parameters are: W= 0, T_m= 0.1 (sec) and M=0.85. The diagram in Fig. 2 represents the situation where r =0 and ρ =4.4 i.e. when threads are independent and the effect of recruitment is maximal. It is evident that the recruitment mechanism retains constant sojourn time regardless of p. The diagram in Fig. 3 is related to the situation where r = 0.5, c_v = 6 and the CPU bound part of the load is ρ = 0.8. The effect of recruitment is considerably lower.

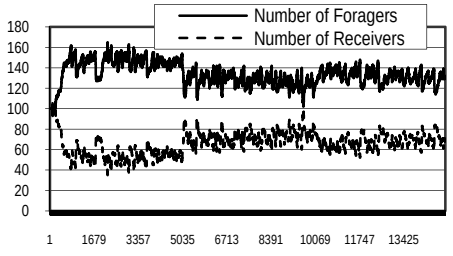

Fig. 1. Efficient optimization of the proportion between F and R

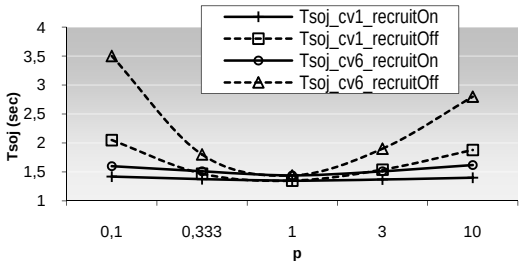

Fig. 2. Measured sojourn times for r = 0

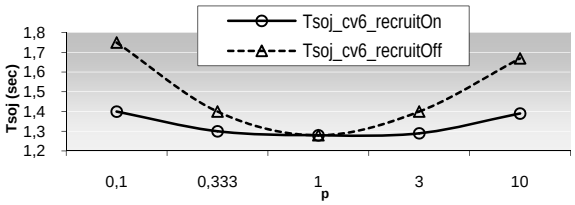

Fig. 3. Measured sojourn times for r = 0,5

4. CONCLUSION

From the biological point of view, the nectar foraging process is an important process in the life of a honeybee colony. From perspective of a software engineer, we see the foraging process as an inspiration that comes from bio world for solving the self-optimization problem in a tandem of thread pools.

In this paper we have demonstrated how the decision mechanism in the honeybee foraging process could be used in a thread pooled server to self-manage its performance. A bee colony attempts to maximize nectar throughput, just as server is directed to maximize the throughput of requests. Future research will focus on the implementation of the thread assignment mechanism described in this paper into a Tomcat web server. Sojourn time improvements obtained in the controlled experimental setting will be compared with those that could be achieved in a more realistic web server environment.

This work was carried out within research project 036-0362027-1640 "Knowledge-based network and service management", supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

5. REFERENCES

Anderson, C. (1998). Simulation of the Feedback and Regulation of Recruitment Dancing in Honey Bees, *Adv. Complex Systems*, Vol. 1, 1998, 267-282

Avi-Itzhak, B.;Halfin, S. (1988). Expected response times in a non-symetric time sharing queue with a limited number of service positions, *Proceedings of ITC 12*, 1988, 5.4B.2.1-7

Frisch, K von. (1967). *The Dance Language and Orientation of Bees*. Cabridge, MA: Harvard University Press

Jednaković, H. (2010). *Computer simulation of the honeybee nectar foraging process*, Bechelor thesis, (Cro.) Faculty of Electrical Engineering and Computing, Zagreb, Jun, 2010

Randić, M.; Blašković, B. & Dembitz Š., Analysis of the Mean Sojourn Time of Tasks Characterized by CPU Boundness in a QBD Model of a ThreadPool, paper submitted for publ.

Seeley, T.D. (1992). The tremble dance of the honey bee: message and meanings. *Behav. Ecol. Sociobiol.* Vol. 31, 375-383

Seeley, T.D. (1995). *The Wisdom of the Hive*. Cambridge, MA: Hravard University Press. 1995

Weij, W van der; Mei, R. D.; Gijsen B. M. & Phillipson, F. (2005) Optimal Server Assignment in a Two-layered Tandem of Multi-Server Queues, HET-NETs'05, P51/1-P51/12, 2005. *Accessed on: 2010-06-10*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.163.8481