Instantiation of a Generic Model for Load Balancing with Intelligent Algorithms

Vesna Sesum-Cavic and Eva Kühn

Vienna University of Technology, Institute for Computer Languages,
Space Based Computing Group, Argentinierstraße 4, 1040 Wien
{vesna,eva}@complang.tuwien.ac.at

Abstract. In peer-to-peer networks, an important issue is the distribution of load having an impact on the overall performance of the system. The answer could be the application of an intelligent approach that leads to autonomic self-organizing infrastructures. In this position paper, we briefly introduce a framework model for load balancing that allows various load-balancing algorithms to be plugged-in, and that uses virtual shared-memory-based communication known to be advantageous for the communication of autonomous agents in order to enable the collaboration of load-balancing agents. As the main contribution, we show how the biological concepts of bees can be mapped to the load-balancing problem, explain why we expect that bee intelligence can outperform other (un)intelligent approaches, and present an instantiation of the model with the bee intelligence algorithm. This load-balancing scheme focuses on two main policies: a transfer and a location policy for which we suggest some improvements.

Keywords: load balancing, bee intelligence, autonomous agents.

1 Introduction

The significance of Load Balancing (LB) in distributed systems is well known. The goal is to obtain an improved performance of the complete system. Dynamic LB algorithms can contribute towards efficiently distributing a load at run time, i.e., appropriately transferring work from heavily loaded nodes to idle or lightly loaded nodes.

Many different approaches cope with the LB problem. We will briefly enumerate the most important ones and classify them. The first group consists of different conventional approaches that do not make use of any kind of intelligence; for example, sender- and receiver-initiated negotiation [25], a gradient model [19], random algorithm [10], diffusion algorithm [7], hierarchical approaches [22], an economic-based model [2], and game theoretical approaches [11]. The second group includes theoretical improvements of LB algorithms using different mathematical tools and estimations [3]; however, with a lack of implementations and real benchmarks. The third group contains approaches that use intelligent algorithms: an evolutionary approach [5] and an ant colony optimization approach [12]. These intelligent approaches successfully cope with LB. Nevertheless, the improvement of several issues (scalability, the quality of the solution, general model, flexibility, etc.) is still open. In a general

K.A. Hummel and J.P.G. Sterbenz (Eds.): IWSOS 2008, LNCS 5343, pp. 311–317, 2008.

investigation of non-pheromone-based algorithms (bee intelligence), in order to compare them to pheromone-based algorithms, Lemmens et al. [18] concluded that the former were significantly more efficient in finding and collecting food and also that they were more scalable.

Our intention is to build a general LB self-organized architecture with exchangeable and combinable algorithms. The communication is based on the shared data model supported by space-based architecture [15]. We have developed such a general model for LB [17]; and now, as the second step, we propose to apply an intelligent algorithm to this model. In this paper, we propose an adaptation of bee intelligence for the realization of LB policies. In Section 2, we give a short summary of a general LB model. Section 3 contains a description of the bee intelligence and the motivation why to use it for LB. Section 4 describes algorithms and the mapping to the LB problem. Section 5 concludes this position paper with an outlook for our future work on this topic.

2 Overview of the SILBA Model

SILBA [17] is a model for LB and stands for "self-initiative load-balancing agents". Instead of having a central coordinator, autonomous agents exist in SILBA. They operate in a network with a dynamically changing amount of nodes, and decide on their own when to pick up or push back work, providing the necessary basis for self-organization. The main idea is to have a framework that allows the plugging of different LB algorithms for comparison. The benchmarks are instantiated with different algorithms, policies and parameters. Moreover, an execution environment for SILBA exists which uses shared data structures[1] for the collaboration of the LB worker agents. This indirect communication allows for more autonomy of the agents, [23] and [13]. The shared data structures are the Internet addressable resources [16]; they maintain collaboration information like pheromones, overlay topology, and other LB relevant parameters. The concurrently running agents either retrieve, or subscribe to this information being notified in near-time about changes, or modify it. Requestors continuously put tasks to any node in the P2P infrastructure, i.e., to a so-called shareable task-list data structure at this node.

In a first implementation, [24] and [17], we have populated the model by not yet intelligent algorithms and have compared it with a centralized approach [14], demonstrating that autonomously deciding agents can improve performance and scalability. The different benchmarks were promising and showed that the agents based approach achieved better results in all test examples (about 27% on average).

In this paper, we suggest how to extend the LB algorithms towards intelligent ones, based on biological self-organized systems. An LB algorithm consists of policies. Every policy has its own autonomous goal. The most important policies are the transfer policy (TP) and the location policy (LP). TP determines whether (and in which form) a resource should participate in load distribution and in that sense, how the classification of resources is done, using certain parameters (described below); whereas the LP determines a suitable partner of a particular resource for LB [25]. The LB algorithm regulates the coordinated interplay of these policies. The parameters of TP

[1] We have implemented the data structures with the XVSM shared data space (www.xvsm.org).