Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Best effort strategy and virtual load for asynchronous iterative load balancing

Raphaël Couturier, Arnaud Giersch*, Mourad Hakem

FEMTO-ST Institute, Univ Bourgogne Franche-Comté, Belfort, France

A R T I C L E I N F O

Article history:
Received 16 December 2017
Received in revised form 18 March 2018
Accepted 1 April 2018
Available online 7 April 2018

Keywords:
Load balancing algorithm
Virtual load
Simulation with SimGrid

A B S T R A C T

Most of the time, asynchronous load balancing algorithms are extensively studied from a theoretical point of view. The Bertsekas and Tsitsiklis' algorithm [1] is undeniably the best known algorithm for which the asymptotic convergence proof is given. From a practical point of view, when a node needs to balance a part of its load to some of its neighbors, the algorithm's description is unfortunately too succinct, and no details are given on what is really sent and how the load balancing decisions are made. In this paper, we propose a new strategy called *best effort* which aims at balancing the load of a node to all its less loaded neighbors while ensuring that all involved nodes by the load balancing phase have the same amount of load. Moreover, since asynchronous iterative algorithms are less sensitive to communication delays and their variations [2], both load transfer and load information messages are dissociated. To speedup the convergence time of the load balancing process, we propose *a clairvoyant virtual load* heuristic. This heuristic allows a node receiving a load information message to integrate the future virtual load (if any) in its load's list, even if the load has not been received yet. This leads to have predictive snapshots of nodes' loads at each iteration of the load balancing process. Consequently, the notified node sends a real part of its load to some of its neighbors, taking into account the virtual load it will receive in the subsequent time-steps. Based on the SimGrid simulator, some series of test-bed scenarios are considered and several QoS metrics are evaluated to show the usefulness of the proposed algorithm.

1. Introduction

Load balancing algorithms are widely used in parallel and distributed applications to achieve high performances in terms of response time, throughput and resources usage. They play an important role and arise in various fields ranging from parallel and distributed computing systems to wireless sensor networks (WSN). The objective of load balancing is to orchestrate the distribution of the global load so that the load difference between the computational resources of the network is minimized as much as possible. Unfortunately, this problem is known to be **NP-hard** in its general form and heuristics are required to achieve sub-optimal solutions but in polynomial time complexity.

In this paper, we focus on asynchronous load balancing of non negative real numbers of *divisible loads* in homogeneous distributed systems. Loads can be divided in arbitrary *fine-grain* parallel parts size that can be processed independently of each other [3–5]. This model of divisible loads arises in a wide range of real-world applications. Common examples, among many, include signal processing, feature extraction and edge detection in image processing, records search in huge databases, average consensus in WSN, pattern search in Big data and so on.

In the literature, the problem of load balancing has been formulated and studied in various ways. The first pioneering work is due to Bertsekas and Tsitsiklis [1]. Under some specific hypothesis and *ping-pong* awareness conditions (see Section 4 for more details), an asymptotic convergence proof is derived.

Although Bertsekas and Tsitsiklis describe the necessary conditions to ensure the algorithm's convergence, there is no indication nor any strategy to really implement the load distribution. Consequently, we propose a new strategy called *best effort* that tries to balance the load of a node to all its less loaded neighbors while ensuring that all the nodes involved in the load balancing phase have the same amount of load. Moreover, most of the time, it is simpler to dissociate load information messages from data migration messages. Former ones allow a node to inform its neighbors about its current load. These messages are in fact very small and can often be sent very quickly. For example, if a computing itera-

* Corresponding author.
E-mail addresses: raphael.couturier@univ-fcomte.fr (R. Couturier), arnaud.giersch@univ-fcomte.fr (A. Giersch), mourad.hakem@univ-fcomte.fr (M. Hakem).

https://doi.org/10.1016/j.jocs.2018.04.002
1877-7503/© 2018 Elsevier B.V. All rights reserved.

tion takes a significant time (ranging from seconds to minutes), it is possible to send a new load information message to each involved neighbor at each iteration. The load is then sent, but the reception may take time when the amount of load is huge and when communication links are slow. Depending on the application, it may or may not make sense for the nodes to try to balance a part of their load at each computing iteration. But the time needed to transfer a load message from one node to another is often much longer than the time needed to transfer a load information message. So, when a node is notified that later it will receive a data message, it can take this information into account in its load's queue list for preventive purposes. Consequently, it can send a part of its predictive load to some of its neighbors if required. We call this trick the *clairvoyant virtual load* transfer mechanism.

The main contributions and novelties of our work are summarized in the following section.

2. Our contributions

- We propose a *best effort strategy* which proceeds greedily to achieve efficient local neighborhoods equilibrium. Upon local load imbalance detection, a *significant amount* of load is moved from a highly loaded node (initiator) to less loaded neighbors.
- Unlike earlier works, we use a new concept of virtual loads transfer which allows nodes to predict the future loads they will receive in the subsequent iterations. This leads to a noticeable speedup of the global convergence time of the load balancing process.
- The SimGrid simulator, which is known to handle realistic models of computation and communication in different types of platforms was used. Taking into account both loads transfers' costs and network contention is essential and has a real impact on the quality of the load balancing performances.

The reminder of the paper is organized as follows. Section 3 offers a review of the relevant approaches in the literature. Section 4 describes the Bertsekas and Tsitsiklis' asynchronous load balancing algorithm. Section 5 presents the best effort strategy which provides efficient local loads equilibrium. In Section 6, the clairvoyant virtual load scheme is proposed to speedup the convergence time of the load balancing process. In Section 7, a comprehensive set of numerical results that exhibit the usefulness of our proposal when dealing with realistic models of computation and communication is provided. Finally, some concluding remarks are made in Section 8.

3. Related works

In this section, the relevant techniques proposed in the literature to tackle the problem of load balancing in a general context of distributed systems are reviewed.

As pointed above, the most interesting approach to this issue has been proposed by Bertsekas and Tsitsiklis [1]. This algorithm, which is outlined in Section 4 for the sake of comparison, has been borrowed and adapted in many works. For instance, in [6] a static load balancing (called DASUD) for non negative integer number of divisible loads in arbitrary networks topologies is investigated. The term *"static"* stems from the fact that no loads are added or consumed during the load balancing process. The theoretical correctness proofs of the convergence property are given. Some generalizations of the same authors' own work for partially asynchronous discrete load balancing model are presented in [7]. The authors prove that the algorithm's convergence is finite and bounded by the straightforward network's diameter of the global equilibrium threshold in the network. In [8], a fault tolerant communication version is addressed to deal with average consensus in wireless sensor networks. The objective is to have all nodes converging to the average of their initial measurements based only on nodes' local information. A slight adaptation is also considered in [9] for dynamic networks with bounded delays asynchronous diffusion. The dynamical aspect stands at the communication level as the links between the network's resources may be intermittent.

Cybenko [10] proposes a *diffusion* approach for hypercube multiprocessor networks. The author targets both static and dynamic random models of work distribution. The convergence proof is derived based on the *eigenstructure* of the iteration matrices that arise in load balancing of equal amount of computational works. A static load balancing for both synchronous and asynchronous ring networks is addressed in [11]. The authors assume that at any time step, one token at the most (units of load) can be transmitted along any edge of the ring and no tokens are created during the balancing phase. They show that for every initial token distribution, the proposed algorithm converges to the stable equilibrium with tighter linear bounds of time step-complexity.

In order to achieve the load balancing of cloud data centers, a LB technique based on Bayes theorem and Clustering is proposed in [12]. The main idea of this approach is that the Bayes theorem is combined with the clustering process to obtain the optimal clustering set of physical target hosts leading to the overall load balancing equilibrium. Bidding is a market-technique for task scheduling and load balancing in distributed systems that characterize a set of negotiation rules for users' jobs. For instance, Izakian et al. [13] formulate a double auction mechanism for tasks-resources matching in grid computing environments where resources are considered as provider agents and users as consumer ones. Each entity participates in the network independently and makes autonomous decisions. A provider agent determines its bid price based on its current workload and each consumer agent defines its bid value based on two main parameters: the average remaining time and the remaining resources for bidding. Based on JADE simulator, the proposed algorithm exhibits better performances in terms of successful execution rates, resource utilization rates and fair profit allocation.

Choi et al. [14] address the problem of robust task allocation in arbitrary networks. The proposed approaches combine a bidding approach for task selection and a consensus procedure scheme for decentralized conflict resolution. The developed algorithms are proven to converge to a conflict-free assignment in both single and multiple task assignment problem. An online stochastic dual gradient LB algorithm, which is called DGLB, is proposed in [15]. The authors deal with both workload and energy management for cloud networks consisting of multiple geo-distributed mapping nodes and data Centers. To enable online distributed implementation, tasks are decomposed both across time and space by leveraging a dual decomposition approach. Experimental results corroborate the merits of the proposed algorithm.

In [16] a LB algorithm based on game theory is proposed for distributed data centers. The authors formulate the LB problem as a non-cooperative game among front-end proxy servers and characterize the structure of Nash equilibrium. Based on the obtained Nash equilibrium structure, they derive a LB algorithm to compute the Nash equilibrium. They show through simulations that the proposed algorithm ensures fairness among the users and a good average latency across all client regions. A hybrid task scheduling and load balancing dependent and independent tasks for master-slaves platforms are addressed in [17]. To minimize the response time of the submitted jobs, the proposed algorithm which is called DeMS is split into three stages: (i) communication overhead reduction between masters and slaves, (ii) task migration to keep the workload balanced and (iii) precedence task graphs partitioning.