Future Generation Computer Systems 49 (2015) 28–44

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Fair scheduling of bag-of-tasks applications on large-scale platforms

Javier Celaya *, Unai Arronategui

Dept. Informática e Ingeniería de Sistemas, Universidad de Zaragoza, María de Luna 1, 50018 Zaragoza, Spain

HIGHLIGHTS

- We present a scheduling model for fair resource sharing on large-scale platforms.
- It effectively aggregates information about application stretch.
- Task allocation is performed so that the maximum stretch is minimized.
- Our model is able to perform similar to a centralized implementation.
- The management overhead is bounded.

ARTICLE INFO

Article history:
Received 5 April 2014
Received in revised form
9 January 2015
Accepted 2 March 2015
Available online 11 March 2015

Keywords:
Fairness
Large-scale
Task scheduling

ABSTRACT

Users of distributed computing platforms want to obtain a fair share of the resources they use. With respect to the amount of computation, the most suitable measure of fairness is the *stretch*. It describes the slowdown that the applications suffer for being executed in a shared platform, in contrast to being executed alone. In this paper, we present a decentralized scheduling policy that minimizes the maximum stretch among user-submitted applications. With two reasonable assumptions, that can be deduced from existing system traces, we are able to minimize the stretch using only local information. In this way, we avoid a centralized design and provide scalability and fault tolerance. As a result, our policy performs just 11% worse than a centralized implementation, and largely outperforms other common policies. Additionally, it easily scales to hundreds of thousands of nodes. We presume that it can scale to millions with a minimal overhead. Finally, we also show that preemption is crucial to provide fairness in any case.

1. Introduction

It is common for a distributed computing platform to be shared among several users, for instance, a cluster giving service to several researchers of an academic institution, or a commercial cloud infrastructure attending millions of requests from around the world. All of them would like to obtain a fair share of the platform. However, the most common scheduling policies are incompatible with the fairness objective. They unbalance the share of the platform among users to maximize the global throughput, minimize the makespan or satisfy the negotiated SLA terms. So, it is the scheduling policy itself who must enforce the fair sharing of the platform among its users. While several such policies have been proposed [1–9], they have serious scalability limitations. They are usually implemented with a centralized design that relies on full knowledge of the platform and the workload. This prevents them from managing the scheduling of tasks on systems of thousands, or even millions of nodes, a scale that is becoming more common every day.

In this paper we present the *Fair Share Policy (FSP)*, a scheduling policy that allocates bag-of-tasks (BoT) applications with fairness in mind. It is a policy for STaRS [10], a scheduling model that can be implemented as part of a distributed computing platform. It provides scalability, fault-tolerance and the ability to support different scheduling policies. It is based on decentralized algorithms that eliminate the bottlenecks of a centralized design, and it is best suited for environments with millions of nodes. So, throughout the paper we assume that we deal with *a very large platform* and *no centralized scheduler*.

To measure the share of the platform, we consider the amount of computation that each user wants to get done. In this case, the most suited metric seems to be the maximum *stretch*, or slowdown [11,2]. The stretch of an application is defined as the ratio of its response time under the concurrent scheduling of applications to its response time when it is the only application executed on the platform. It is the user's perception of how slow its applications run due to its sharing the platform with other users.

* Corresponding author. Tel.: +34 876 55 55 31; fax: +34 976 76 19 14.
E-mail addresses: jcelaya@unizar.es (J. Celaya), unai@unizar.es (U. Arronategui).

http://dx.doi.org/10.1016/j.future.2015.03.002
0167-739X/

Let r_i be the release time of an application A_i, e_i its end time and Θ_i its response time in a platform dedicated to itself, its stretch S_i is calculated as

$$S_i = \frac{e_i - r_i}{\Theta_i}. \quad (1)$$

A perfectly fair share of the platform is obtained when all the applications obtain the same stretch. However, this is only possible with offline scheduling and divisible load. With these conditions, the scheduler can adjust the end time of each application to reach the optimum objective. Instead, we consider a classic configuration in distributed computing, with online scheduling and atomic tasks. So, the best tradeoff is obtained by minimizing the maximum stretch among all applications.

We already presented a first design of this policy in [12]. In this paper, we widely improve the estimation and representation of the stretch, to obtain much better results. In particular, the main problem we face is that computing Θ_i usually requires full knowledge of the platform. This is impractical with a decentralized design, so we assume two reasonable hypothesis:

- Each application has much less tasks than nodes in the platform.
- The distribution of computing power among nodes changes very little.

With these premises, we are able to minimize the maximum stretch without full knowledge of the platform.

The rest of the paper is organized as follows: Section 2 presents the related work on fair scheduling, both in centralized and decentralized environments. Then, Section 3 explains how we solve the problem of minimizing the maximum stretch without full knowledge of the platform. Section 4 gives a brief description of the architecture of STaRS, on which this paper is based. In Sections 5 and 6, we explain the details of the FSP policy. And finally, Section 7 presents the results of the experiments and in Section 8 we give our conclusions and a description of the future work.

2. Related work

Several works have approached the fair scheduling of different kinds of applications by minimizing the stretch. Benoit et al. [1] study the minimization of maximum stretch for concurrent BoT applications, like we do, but in a centralized setting. They show that interleaving tasks of several concurrent BoT applications perform better than scheduling each application after the other. Previously, Legrand et al. [2] focused on the scheduling of divisible load applications. In particular, they schedule applications that search large-scale genomic and proteomic sequence databanks. Casanova et al. [3] show the challenges of minimizing the stretch of parallel task graphs, due to their rigid constraints. They minimize both stretch and makespan, and show that relaxing completion times they can be still near to the perfectly fair schedule. OStrich [4] extends the concept of stretch to campaigns, or whole sets of jobs of the same user. In this way, they claim that the achieved stretch is proportional to the campaign size. Wu and Cao [13] use it to schedule dissemination-based applications. They propose a low-overhead solution to the fair scheduling of on-demand data broadcasts, where the contribution of each job to the system response time depends on the data size.

MapReduce [14], Hadoop [15] or Dryad [16] have schedulers with a trade-off between fairness and data locality. Quincy [17], which is built on Dryad, is a centralized solver that improves the throughput of jobs being allocated. For Hadoop, delay scheduling [5,18] achieves nearly optimal data locality while preserving fairness. While it includes the possibility of a distributed implementation, it follows a master–slave approach that can limit its scalability.

Cloud computing infrastructures offer users the possibility of sharing cost and resources among them. Different works enhance the possibilities of fair scheduling in this context. From min–max fair scheduling with constraints [6], to fair allocation of multiple resource types [19,7,8]. Also, there are proposals to provide hard delay guarantees with user fairness [9]. But none of them proposes a decentralized solution or addresses the challenge of fair scheduling in a scale of hundred of thousands, or even millions of nodes.

In our previous work [10], we included a survey of related work in decentralized scheduling of jobs for large-scale systems. But we have found very few proposal about fairness in large-scale architectures for cloud, grid or P2P computing systems. Östberg et al. [20] propose Aequus, a decentralized fairshare scheduler for grids. It tries to provide a similar share of the resources to every user based on how much they have been using them in the past. A hierarchical model, that reflects the grid organization (VOs, sites, projects, users, etc.), represents the share that can be granted to every user. Schedulers at each site use this information to assign a priority to each job. The information is distributed among the VOs and every scheduler is independent from each other. However, as a result, a single scheduler may need to access the information of all the users. On the contrary, we use a tree to both maintain the information and forward tasks towards execution nodes, so that each step is performed with local information only. Bertin et al. [21] present a fair decentralized scheduling algorithm for BoT applications with arbitrary communication-to-computation ratio. It is based on Lagrangian optimization and distributed gradient descent. As a result, when the algorithm converges, applications obtain a similar share of both computational and network resources. However, the largest tests are performed on 500-node platforms, much smaller than the hundred thousand nodes that we have tested. As the authors admit, there are still many aspects to improve, so we expect to see much better results. In the domain of HTC exascale systems, there are some decentralized architectures, based on work stealing, for load balancing of distributed jobs [22,23]. However, they do not deal with fairness.

The fair sharing of resources has been deeply studied before in other areas, like networking [24,25], which consider the amount of data to be transferred by each user. In particular, fair scheduling with a fully decentralized architecture has been proposed in mesh networks for sharing resources as bandwidth, time, frequency, relays, etc. [26–31]. But the challenge of large-scale networks was not considered in any of these works. Also, better results have been shown applying a decentralized model for fair scheduling in multiprocessors [32]. As in mesh networks, the scale being considered was quite low, with experiments of 8 CPUs and 10 threads.

3. Fairness in a decentralized BoT environment

There are several issues we have to deal with if we want to achieve fairness among BoT applications in a decentralized environment. We consider that a BoT application A_i consists of n_i tasks of equal length a_i, in millions of FLOPs. This is a common model [1,21], although BoT applications with variable-length tasks [33] will be tackled in the future. As stated by Eq. (1), to compute the stretch of an application A_i we need to calculate its response time if it was alone in the platform, Θ_i. The main problem we face is that calculating Θ_i requires full knowledge of the platform's characteristics. While easily performed in a centralized context, it is unthinkable in a decentralized one. So, we are going to limit the problem with two premises, that will let us obtain a good approximation of the stretch.

The first premise is that *each application has much less tasks than nodes in the platform* (although there is no limit in the number of