Performance Evaluation 79 (2014) 134–145

Contents lists available at ScienceDirect

Performance Evaluation

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

Enhanced cluster computing performance through proportional fairness

Thomas Bonald [a,*,1], James Roberts [b,1]

[a] Télécom ParisTech, Paris, France
[b] IRT-SystemX, Paris-Saclay, France

ARTICLE INFO

Article history:
Available online 10 July 2014

Keywords:
Cluster computing
Multi-resource sharing
Proportional fairness
Dominant resource fairness

ABSTRACT

The performance of cluster computing depends on how concurrent jobs share multiple data center resource types such as CPU, RAM and disk storage. Recent research has discussed efficiency and fairness requirements and identified a number of desirable scheduling objectives including so-called dominant resource fairness (DRF). We argue here that proportional fairness (PF), long recognized as a desirable objective in sharing network bandwidth between ongoing data transfers, is preferable to DRF. The superiority of PF is manifest under the realistic modeling assumption that the population of jobs in progress is a stochastic process. In random traffic the strategy-proof property of DRF proves unimportant while PF is shown by analysis and simulation to offer a significantly better efficiency–fairness tradeoff.

1. Introduction

We consider a cluster of servers, exclusively reserved for executing a certain class of jobs, as a set of distinct resource pools. The resources in question include CPU, RAM, disk space and I/O bandwidth, for instance. A job must execute a certain number of tasks, each task requiring a given quantity of each type of resource. An example of this type of computing environment and its workload are described by Reiss et al. [1]. Each task has its own particular requirements profile specifying how much of each resource must be reserved: some tasks require proportionally more CPU than RAM, others require proportionally more RAM than I/O bandwidth, etc. The issue at hand is how a central scheduler should initiate and run tasks in parallel to fairly and efficiently share cluster resources between jobs. We argue in this paper that such resource sharing should realize the objective of Proportional Fairness (PF), that is, it should maximize the sum over jobs of logarithms of their number of tasks.

The multi-resource sharing problem was formulated in a recent paper by Ghodsi et al. [2] under the simplifying assumption that resource pools are homogeneous and infinitely divisible between jobs. All tasks of a given job are assumed to have identical profiles and, rather than allocating resources to discrete tasks, a central scheduler is supposed to assign shares of each resource in proportion to the task profile. Sharing objectives are expressed in terms of a number of desirable properties. In particular, it is required that allocations be strategy-proof in the sense that the owner of a job cannot gain a bigger share by lying about its actual task profile. The authors of [2] show that this property is not satisfied by many otherwise intuitively appealing allocations. They proceed to define an original strategy-proof allocation called Dominant Resource Fairness (DRF) and explain how this can be realized.

* Corresponding author.
E-mail addresses: thomas.bonald@telecom-paristech.fr (T. Bonald), james.roberts@irt-systemx.fr (J. Roberts).
[1] The authors are members of the LINCS, Paris, France. See www.lincs.fr.

http://dx.doi.org/10.1016/j.peva.2014.07.009
0166-5316/

Other authors have since come up with alternative strategies, based on different notions of fairness (e.g., [3,4]), or adapting DRF to account for more realistic cluster constraints (e.g., [5–7]). Gutman and Nisan [8] situate proposed fairness objectives like DRF [2] and "no justified complaints" [3] in a common economics framework and provide efficient polynomial time algorithms.

We believe the economics literature cited in [8] does not in fact constitute an appropriate background for modeling cluster resource sharing. This is because it completely ignores the dynamics of job arrivals and completions that characterize cluster workload [1]. These dynamics have an obvious impact on job completion times and interact closely with the applied allocation algorithm since a job finishes more or less quickly depending on the resources it is allocated. The notion of strategy-proofness must be revisited to account for the impact of false user requirements on the process of jobs in progress and their expected performance.

In advocating PF we draw on our understanding of bandwidth sharing performance in networks. PF was defined by Kelly et al. [9] as the bandwidth allocation that maximizes the sum of logs of flow rates, arguably thus realizing greater social welfare than alternative allocations like max–min fairness. We consider rather the impact of the sharing objective on flow completion times in dynamic traffic. In wired networks, it turns out that completion times are not highly dependent on the type of fairness imposed and DRF (equivalent here to max–min fairness) and PF have similar performance [10]. In wireless networks, on the other hand, the radio resource is measured in time slots per second rather than bit/s with the achievable bit rate per time slot depending significantly on user position. Sharing a channel equally in time, as realized in HDR/HSDPA systems for instance, actually corresponds to PF sharing in terms of bandwidth. In a mixed wired/wireless network, PF realizes a significantly more favorable efficiency–fairness tradeoff than max–min fairness [11].

To compare PF and DRF in dynamic traffic we adopt a simple Markovian traffic model assuming Poisson job arrivals and exponential job sizes. This simplifies analysis and clarifies the respective tradeoffs realized by the two allocation approaches. Simulation and analytical results confirm that PF performs significantly better than DRF. Given known insensitivity properties of fair resource sharing, we are confident that PF would be equally preferable under a more realistic traffic model [10].

In addition to discussing ideal sharing of infinitely divisible resources, Ghodsi et al. show how to approximately realize DRF in practice accounting for discrete, finite-size tasks. The scheduler algorithm preferentially launches tasks of the "most deprived" job. This is the job whose current share of its dominant resource is smallest. PF can be implemented similarly on re-defining "most deprived" in terms of the ideal shares determined for that allocation objective. The complexities of PF and DRF algorithms are similar and hardly constitute an implementation issue since the number of resource types to be shared is typically very small.

In the next section we define DRF and PF and demonstrate their respective sharing properties with respect to a static job population. We consider dynamic sharing in Section 3 assuming a fluid model where resources are infinitely divisible. The task-by-task implementations of DRF and PF are compared in Section 4. Finally, related work is discussed in Section 5 before we conclude.

2. Multi-resource sharing

We consider how multiple resources should be shared assuming a static population of jobs, each with a particular profile of per-task requirements. We adopt a fluid model where pools of resources are assumed infinitely divisible and compare two allocation strategies: dominant resource fairness (DRF) and proportional fairness (PF).

2.1. The fluid model

We consider J infinitely divisible pools of resources of respective capacities C_j, for $j = 1, \ldots, J$, to be shared by n jobs indexed by i. Each task of job i requires A_{ij} units of resource j. Denoting by φ_i the number of ongoing tasks of job i, we have the capacity constraints:

$$\sum_{i=1}^{n} \varphi_i A_{ij} \leq C_j,$$

for $j = 1, \ldots, J$. In the fluid model, we assume tasks are infinitesimally small and jobs can run a sufficiently large number of them to attain a given resource allocation. It then makes more sense to normalize resource capacities to 1 with $a_{ij} = A_{ij}/C_j$ representing fractional requirements. The (now) real numbers $\varphi_1, \ldots, \varphi_n$ are then considered as *task volumes* satisfying capacity constraints:

$$\sum_{i=1}^{n} \varphi_i a_{ij} \leq 1, \tag{1}$$

for $j = 1, \ldots, J$. The product $\varphi_i a_{ij}$ is the fraction of resource j allocated to job i. We also write capacity constraints (1) in matrix form $\varphi a \leq 1$, where inequality is understood component-wise.

We say that resource j is *saturated* if the corresponding capacity constraint is attained, i.e., if $\sum_i \varphi_i a_{ij} = 1$. Let a_i denote the vector whose jth component is a_{ij}. We say job i *needs* resource j if $a_{ij} > 0$ and we assume each job needs at least one resource. The *dominant resource* of job i is that for which the normalized requirement a_{ij} is largest.