HAL
open science

Non-Cooperative Scheduling of Multiple Bag-of-Task Applications

Arnaud Legrand, Corinne Touati

To cite this version:

Arnaud Legrand, Corinne Touati. Non-Cooperative Scheduling of Multiple Bag-of-Task Applications. INFOCOM, May 2007, Anchorage, Alaska. inria-00153577

HAL Id: inria-00153577
https://inria.hal.science/inria-00153577v1

Submitted on 11 Jun 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Non-Cooperative Scheduling of Multiple Bag-of-Task Applications

Arnaud Legrand and Corinne Touati*
Laboratoire LIG, Grenoble, CNRS-INRIA, MESCAL project, France,
arnaud.legrand@imag.fr, corinne.touati@imag.fr

inria-00153577, version 1 - 11 Jun 2007

***Abstract*— Multiple applications that execute concurrently on heterogeneous platforms compete for CPU and network resources. In this paper we analyze the behavior of K non-cooperative schedulers using the optimal strategy that maximize their efficiency while fairness is ensured at a system level ignoring applications characteristics. We limit our study to simple single-level master-worker platforms and to the case where each scheduler is in charge of a single application consisting of a large number of independent tasks. The tasks of a given application all have the same computation and communication requirements, but these requirements can vary from one application to another. In this context, we assume that each scheduler aims at maximizing its throughput. We give closed-form formula of the equilibrium reached by such a system and study its performance. We characterize the situations where this Nash equilibrium is optimal (in the Pareto sense) and show that even though no catastrophic situation (Braess-like paradox) can occur, such an equilibrium can be arbitrarily bad for any classical performance measure.**

I. Introduction

The recent evolutions in computer networks technology, as well as their diversification, yield to a tremendous change in the use of these networks: applications and systems can now be designed at a much larger scale than before. Large-scale distributed platforms (Grid computing platforms, enterprise networks, peer-to-peer systems) result from the collaboration of many people. Thus, the scaling evolution we are facing is not only dealing with the amount of data and the number of computers but also with the number of users and the diversity of their needs and behaviors. Therefore computation and communication resources have to be *efficiently* and *fairly* shared between users, otherwise users will leave the group and join another one. However, the variety of user profiles requires resource sharing to be ensured at a system level. We claim that even in a perfect system where every application competing on a given resource receives the same share and where no degradation of resource usage (e.g., packet loss or context switching overhead) occurs when a large number of applications use a given resource, non-cooperative usage of the system leads to important application performance degradation and resource wasting. In this context, we make the following contributions:

- We present a simple yet realistic situation where a fair and Pareto-optimal *system-level* sharing fails to achieve an efficient *application-level* sharing. More precisely, we study the situation where multiple applications consisting of large numbers of independent identical tasks execute concurrently on heterogeneous platforms and compete for CPU and network resources. SETI@home [1], the Mersenne prime search [2], ClimatePrediction.NET [3], Einstein@Home [4], processing data of the Large Hadron Collider [5] are a few examples of such typical applications. As the tasks of a given application all have the same computation and communication requirements (but these requirements can vary for different applications), each scheduler aims at maximizing its throughput. This framework had previously been studied in a cooperative centralized framework [6]. In the previous context, at any instant, cooperation led to a dedication of resources to applications. The system-level resource sharing aspect was therefore not present and is extensively described in Section II of the present paper.
- We characterize in Section II-D the optimal selfish strategy for each scheduler (i.e. the scheduling strategy that will maximize its own throughput in all circumstances and adapt to external usage of resources) and propose equivalent representations of such non-cooperative schedulers competition (see Section III).
- The particularities of these representations enable us to characterize the structure of the resulting Nash equilibrium as well as closed-form values of the throughput of each application (see Section III-D).
- Using these closed-form formulas, we derive in Section IV a necessary and sufficient condition on the system parameters (in term of bandwidth, CPU speed, ...) for the non-cooperative equilibrium to be Pareto-optimal.
- We briefly study in Section IV-C the well-known "price of anarchy" [7]. Unfortunately, this metric does not enable one to distinguish Pareto optimal points from non-Pareto optimal ones. That is why we propose an alternate definition, the "*selfishness degradation factor*".
- When studying properties of Nash equilibria, it is important to know whether paradoxical situations like the ones exhibited by Braess in his seminal work [8] can occur. In such situations, the addition of resource (a new machine, more bandwidth or more computing power in our framework) can result in a simultaneous degradation of the performance of *all* the users. Such situations only occur when the equilibrium is not Pareto-optimal, which may be the case in this framework. We investigate in Section IV-D whether such situations can occur in our considered scenario and conclude with a negative answer.

*The authors would like to thank the University of Tsukuba and the Japan Society for the Promotion of Science for supporting this work.

- Last, we show in Section V, that even when the non-cooperative equilibrium is Pareto-optimal, the throughput of each application is far from being monotonous with a resource increase. This enables us to prove that this equilibrium can be arbitrarily bad for any of the classical performance measures (average, maximal, and minimum throughput).

Section VI concludes the paper with a discussion of extensions of this work and future directions of research. Due to space requirements, the proofs of the following theorems and propositions will be omitted in this paper. The interested reader is referred to [9] for detailed proofs.

II. Platform and Application Models

A. Platform Model

Our master-worker platform is made of $N+1$ processors $P_0, P_1, \ldots, P_N$. P_0 denotes the master processor, which does not perform any computation. Each processor P_n is characterized by its computing power W_n (in Mflop.s^{-1}) and the capacity of its connection with the master B_n (in Mb.s^{-1}). Last, we define the *communication-to-computation ratio* C_n of processor P_n as B_n/W_n. This model leads us to the following definition:

Definition 1. We denote by **physical-system** a triplet (N, B, W) where N is the number of machines, and B and W the vectors of size N containing the link capacities and the computational powers of the machines.

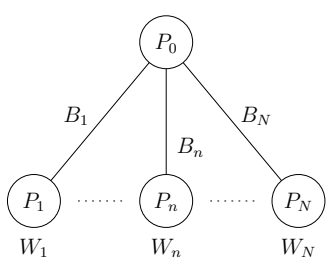


Fig. 1. Platform model: a single master and N workers

We assume that the platform performs an ideal fair sharing of resources among the various requests. More precisely, let us denote by $N_n^{(B)}(t)$ (resp. $N_n^{(W)}(t)$) the number of ongoing communication (resp. computation) from P_0 to P_n (resp. on P_n) at time t. The platform ensures that the amount of bandwidth received at time t by a communication from P_0 to P_n is exactly $B_n/N_n^{(B)}(t)$. Likewise, the amount of processor power received at time t by a computation on P_n is exactly $W_n/N_n^{(W)}(t)$. Therefore, the time T needed to transfer a file of size b from P_0 to P_n starting at time t_0 is such that

$$\int_{t=t_0}^{t_0+T} \frac{B_n}{N_n^{(B)}(t)} \cdot dt = b.$$

Likewise, the time T needed to perform a computation of size w on P_n starting at time t_0 is such that

$$\int_{t=t_0}^{t_0+T} \frac{W_n}{N_n^{(W)}(t)} \cdot dt = w.$$

Last, we assume that communications to different processors do not interfere. This amounts to say that the network card of P_0 has a sufficiently large bandwidth not to be a bottleneck. Therefore, a process running on P_0 can communicate with as many processors as it wants. This model is called *multi-port* [10], [11] and is reasonable when workers are spread over the Internet, are not too numerous and can make use of threads to manage communications.

B. Application Model

We consider K applications, A_k, $1 \leqslant k \leqslant K$. Each application is composed of a large set of independent, same-size tasks. We can think of each A_k as a bag of tasks, in which each task is a file that requires some processing. A task of application A_k is called a task of *type* k. We let w_k be the amount of computation (in Mflop) required to process a task of type k. Similarly, b_k is the size (in Mb) of (the file associated to) a task of type k. We assume that the only communication required is outwards from the master, i.e. the amount of data returned by the worker is negligible. This is a common hypothesis [12] as in steady-state, the output-file problem can be reduced to an equivalent problem with bigger input-files. Last, we define the *communication-to-computation ratio* c_k of tasks of type k as b_k/w_k. This model leads to the following definition:

Definition 2. We define a **user-system** a triplet (K, b, w) where K is the number of applications, and b and w the vectors of size K representing the size and the amount of computation associated to the different applications.

C. Global System

In the following our K applications run on our N processors and compete for network and CPU access:

Definition 3. A **system** S is a sextuplet (K, b, w, N, B, W), with K,b,w,N,B,W defined as for a user-system and a physical-system.

We assume that each application is scheduled by its own scheduler. As each application comprises a very large number of independent tasks, trying to optimize the makespan is known to be vainly tedious [13] especially when resource availability varies over time. Maximizing the throughput of a single application is however known to be a much more relevant metric in our context [6], [14]. More formally, for a given infinite schedule we can define $done_k(t)$ the number of tasks of type k processed in time interval $[0, t]$. The throughput for application k of such a schedule is defined as $\alpha_k = \liminf_{t\to\infty} \frac{done_k(t)}{t}$. Similarly we can define $\alpha_{n,k}$ the average number of tasks of type k performed per time-unit on the processor P_n. α_k and $\alpha_{n,k}$ are linked by the following linear equation $\alpha_k = \sum_n \alpha_{n,k}$. The scheduler of each application thus aims at maximizing its own throughput, i.e. α_k. However, as the applications are sharing the same set of resources, we have the following general constraints[1]:

Computation $\forall n \in [\![0, N]\!] : \sum_{k=1}^{K} \alpha_{n,k} \cdot w_k \leqslant W_n$ (1a)

[1]The notation $[\![a, b]\!]$ denotes the set of integers comprised between a and b, i.e. $[\![a, b]\!] = \mathbb{N} \cap [a, b]$.

Communication $\forall n \in [\![1, N]\!] : \sum_{k=1}^{K} \alpha_{n,k} \cdot b_k \leqslant B_n$ (1b)

These constraints enable to define upper-bounds on the throughput of any schedule. Moreover, a *periodic schedule* — one that begins and ends in exactly the same state — (we refer the interested reader to [15] for more details) can be built from any valid values for the α_k and $\alpha_{n,k}$ such that its throughput for all applications k is exactly $\alpha_k = \lim_{t\to\infty} \frac{done_k(t)}{t}$. When the number of tasks per application is large, this approach has the advantage of avoiding the NP-completeness of the makespan optimization problem. This further allows to only focus on the average steady-state values.

Remark 1. Consider a system with K applications running over N machines. The set of achievable utilities, that is to say the set of possible throughput α_k is given by

$$U(S) = \left\{ (\alpha_k)_{1\leqslant k\leqslant K} \;\middle|\; \begin{array}{l} \exists \alpha_{1,1}, \ldots, \alpha_{N,K}, \\ \quad \forall k \in [\![1, K]\!] : \sum_{n=1}^{N} \alpha_{n,k} = \alpha_k \\ \quad \forall n \in [\![1, N]\!] : \sum_{k=1}^{K} \alpha_{n,k} \cdot w_k \leqslant W_n \\ \quad \forall n \in [\![1, N]\!] : \sum_{k=1}^{K} \alpha_{n,k} \cdot b_k \leqslant B_n \\ \quad \forall n \in [\![1, N]\!], \forall k \in [\![1, K]\!] : \alpha_{n,k} \geqslant 0 \end{array} \right\}.$$

The utility set is hence convex and compact.

D. A Non-Cooperative Game

We first study the situation where only one application is scheduled on the platform. This will enable us to simply define the scheduling strategy that will be used by each player (scheduler) in the more general case where many applications are considered. When there is only one application, our problem reduces to the following linear program:

$$\text{MAXIMIZE } \sum_{n=1}^{N} \alpha_{n,1} \text{ UNDER THE CONSTRAINTS } \begin{cases} \text{(1a)} & \forall n \in [\![0, N]\!] : \alpha_{n,1} \cdot w_1 \leqslant W_n \\ \text{(1b)} & \forall n \in [\![1, N]\!] : \alpha_{n,1} \cdot b_1 \leqslant B_n \\ \text{(1c)} & \forall n, \quad \alpha_{n,1} \geqslant 0. \end{cases}$$

We can easily show that the optimal solution to this linear program is obtained by setting $\forall n,\ \alpha_{n,1} = \min\left(\frac{W_n}{w}, \frac{B_n}{b}\right)$. In a practical setting, this amounts to say that the master process will saturate each worker by sending it as many tasks as possible. On a stable platform W_n and B_n can easily be measured and the $\alpha_{n,1}$'s can thus easily be computed. On an unstable one this may be more tricky. However a simple acknowledgment mechanism enables the master process to ensure that it is not over-flooding the workers, while always converging to the optimal throughput.

In a multiple-applications context, each player (process) strives to optimize its own performance measure (considered here to be its throughput α_k) regardless of the strategies of the other players. Hence, in this scenario, each process constantly floods the workers while ensuring that all the tasks it sends are performed (e.g., using an acknowledgment mechanism). This adaptive strategy automatically cope with other schedulers usage of resource and *selfishly* maximize the throughput of each application[2]. As the players constantly adapt to each others' actions, they may (or not) reach some equilibrium, known in game theory as *Nash equilibrium* [16], [17]. In the remaining of this paper, we will denote by $\alpha_{n,k}^{(nc)}$ the rates achieved at such stable states.

[2]We suppose a purely non-cooperative game where no scheduler decides to "ally" to any other (i.e. no coalition is formed).

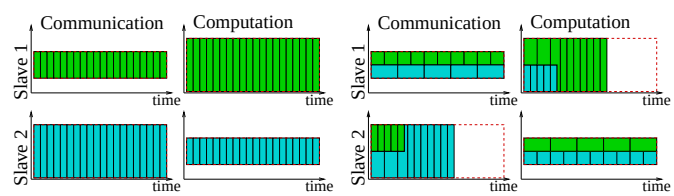
E. A simple example

Consider a system with two computers 1 and 2, with parameters $B_1 = 1$, $W_1 = 2$, $B_2 = 2$, $W_2 = 1$ and two applications of parameters $b_1 = 1$, $w_1 = 2$, $b_2 = 2$ and $w_2 = 1$. If the applications were collaborating such that application 1 was processed exclusively to computer 1 and application 2 in computer 2 (see Figure 2(a)), their respective throughput would be

$$\alpha_1^{(\text{coop})} = \alpha_2^{(\text{coop})} = 1.$$

Yet, with the non-cooperative approach (see Figure 2(b)), one can check that they only get a throughput of (the formal proof will be given in Theorem 1):

$$\alpha_1^{(nc)} = \alpha_2^{(nc)} = \frac{3}{4}$$



(a) With cooperation (b) Non-cooperative equilibrium.

Fig. 2. Non-cooperation can lead to inefficiencies.

In this example, one can easily check that, at the Nash equilibrium, for any worker, there is no resource waste: slave 1 (resp. slave 2) is communication (resp. computation) saturated i.e. equation (1b) (resp. equation (1a)) is an equality. However, communication-saturation implies computation idle times and vice-versa. Yet, when communication and computation idle times cohexist in a system, the non-cooperative behavior of the applications can lead to important inefficiencies, which we investigate further in the remaining of this paper.

III. Mathematical Formulation

In this section, we mathematically formulate the use of resources in the system. As in the multi-port model communication resources are all independent, we can study each worker separately. Hence, in this section, we study the use of resources on an arbitrary worker n of the system.

In a steady state, actions of the players will interfere on each resource in (a priori) a non predictable order and the resource usage may be arbitrarily complex (see Figure 3(a)). We hence propose in this section "equivalent" representations (in the sense that they conserve the throughput of each application on the considered worker) that will enable us to conclude this section with a closed-form expression of the $(\alpha_{n,k})_{\substack{1\leqslant n\leqslant N \\ 1\leqslant k\leqslant K}}$.

First note that for any given subset $\mathcal{K}$ of $[\![1, K]\!]$, we can define the fraction of time where all players of $\mathcal{K}$ (and only them) use a given resource. This enables us to reorganize the schedule into an equivalent representation (see Figure 3(b)) with at most $2^{|\mathcal{K}|}$ time intervals (the number of possible choices for the subset $\mathcal{K}$). In this representation, the fractions of time spent using a given resource (which can be a communication link or a processor) are perfectly equal to the ones in the original schedule. However such a representation is still too complex ($2^{|\mathcal{K}|}$ is a large value). Hence, we now explain how to build two more compact "equivalent" canonical representations (see Figure 3(c) and 3(d)).

A. Sequential Canonical Representation

The first compact form we define is called *sequential canonical representation* (see Figure 3(c)). If the schedulers were sending data one after the other on this link, the kth scheduler would have to communicate during exactly $\tau_{n,k}^{(B,seq)} = \frac{\alpha_{n,k}^{(nc)} b_k}{B_n}$ of the time to send the same amount of data as in the original scheduler. This value is called *sequential communication time ratio*. Similarly, we can define the *sequential computation time ratio* $\tau_{n,k}^{(W,seq)}$ as $\frac{\alpha_{n,k}^{(nc)} w_k}{W_n}$. We hence have the following relation between $\tau_{n,k}^{(B,seq)}$ and $\tau_{n,k}^{(W,seq)}$:

$$\tau_{n,k}^{(B,seq)} = \frac{c_k}{C_n} \tau_{n,k}^{(W,seq)}. \tag{2}$$

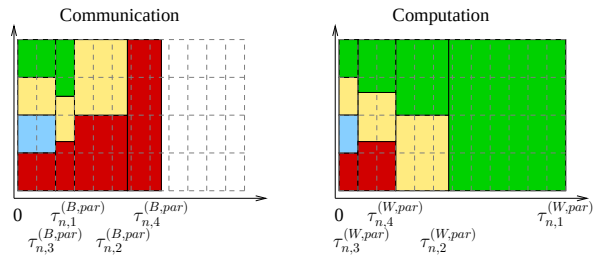
We can therefore obtain a canonical schedule (see Figure 3(c)) with at most $K+1$ intervals whose respective sets of players are $\{1\}$, $\{2\}$, $\ldots$, $\{K\}$, $\emptyset$. This communication scheme is thus called sequential canonical representation and has the same $\alpha_{n,k}^{(nc)}$ values as the original schedule. However, communication and computation times have all been decreased as each scheduler is now using the network link and the CPU exclusively. We will see later that this information loss does not matter for multi-port schedulers.

B. Parallel Canonical Representation

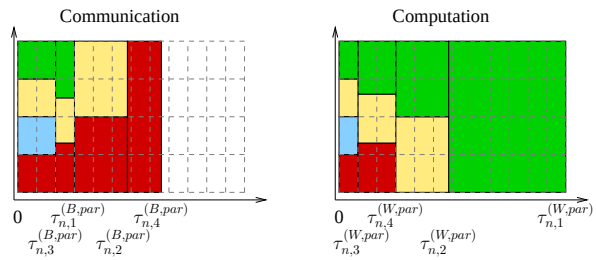
The second compact form we define is called *parallel canonical representation* (see Figure 3(d)). In this scheme, resource usage is as conflicting as possible. Let us denote by $\tau_{n,k}^{(B,par)}$ (resp. $\tau_{n,k}^{(W,par)}$) the fraction of time spent by player k to communicate with P_n (resp. to compute on P_n) in such a configuration. $\tau_{n,k}^{(B,par)}$ is the *parallel communication time ratio* and $\tau_{n,k}^{(W,par)}$ is the *parallel computation time ratio*. We can easily prove that such representation is unique (see the extended version [9]) and we can therefore obtain a canonical schedule (see Figure 3(d)) with at most $K+1$ intervals whose respective player sets are $\{1, \ldots, K\}$, $\{2, \ldots, K\}$, $\ldots$, $\{K\}$, and $\emptyset$. This communication scheme is called parallel canonical representation and has the same $\alpha_{n,k}^{(nc)}$ values as the original schedule. However, communication times have all been increased as each scheduler is now interfering with as many other schedulers as possible.

C. Particularities of Multi-port Selfish Schedulers

The same reasonings can be applied to computation resources and therefore, for a given worker, both communication and computation resources can be put in any of these two canonical forms (see Figure 4(a)).



(a) Parallel canonical form of an arbitrary schedule



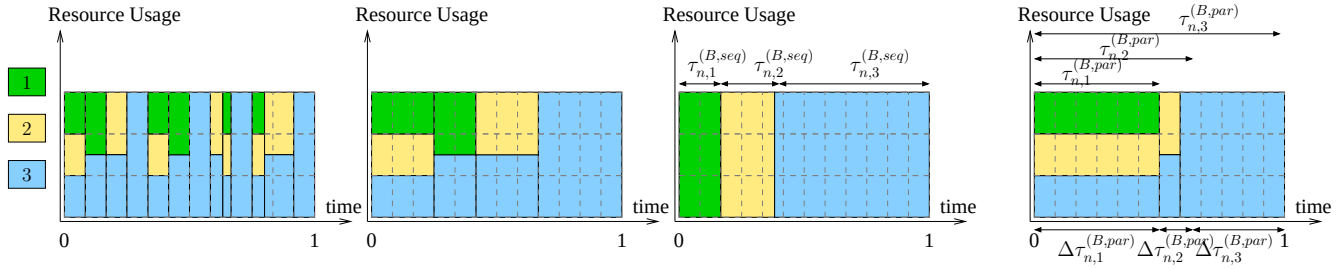
(b) Parallel canonical schedule for a given processor under the non-cooperative assumption. Application 3 (blue) and 4 (red) are *communication saturated*: they receive the same amount of bandwidth. Application 1 (green) and 2 (yellow) are *computation saturated*: they receive the same amount of CPU.

Fig. 4. Parallel canonical schedules

As we have seen in Section II-D, the scheduling algorithm used by the players consists in constantly flooding workers. Hence it is impossible that both $\tau_{n,k}^{(B,par)}$ and $\tau_{n,k}^{(W,par)}$ are smaller than 1. A player k is thus said to be either *communication-saturated* on worker n ($\tau_{n,k}^{(B,par)} = 1$) or *computation-saturated* on worker n ($\tau_{n,k}^{(W,par)} = 1$).

Proposition 1. If there is a communication-saturated application then $\sum_{k=1}^{K} \tau_{n,k}^{(B,seq)} = 1$. Similarly, if there is a computation-saturated application then $\sum_{k=1}^{K} \tau_{n,k}^{(W,seq)} = 1$.

As two computation-saturated players k_1 and k_2 receive the same amount of computation power and compute during the same amount of time, we have $\alpha_{n,k_1}^{(nc)} w_{k_1} = \alpha_{n,k_2}^{(nc)} w_{k_2}$. Therefore $c_{k_1} \leqslant c_{k_2}$ implies $\alpha_{n,k_1}^{(nc)} b_{k_1} \leqslant \alpha_{n,k_2}^{(nc)} b_{k_2}$, hence $\tau_{n,k_1}^{(B,par)} \leqslant \tau_{n,k_2}^{(B,par)}$ and $\tau_{n,k_1}^{(B,seq)} \leqslant \tau_{n,k_2}^{(B,seq)}$. The same reasoning holds for two communication-saturated players as well as for a mixture of both. As a consequence, in a multi-port setting, players should be first sorted according to their c_k to build the canonical schedule. The very particular structure of this schedule (see Figure 4(b)) will enable us in the following section to give closed-form formula for the $\alpha_{n,k}^{(nc)}$. All these remarks can be summarized in the following proposition:



(a) Complex arbitrary schedule

(b) Sorted schedule

(c) Sequential canonical representation: areas are preserved but using times are minimized

(d) Parallel canonical representation: areas are preserved but using times are maximized

Fig. 3. Various schedule representations. Each application is associated to a color: Application 1 is green, application 2 is yellow and application 3 is blue. The area associated to each application is preserved throughout all transformations.

Proposition 2. Let us consider an equilibrium and denote by $\mathcal{B}_n$ the set of communication-saturated applications on worker n and by $\mathcal{W}_n$ the set of computation-saturated applications on worker n. If $c_1 \leqslant c_2 \leqslant \cdots \leqslant c_K$, then there exists $m \in [\![0, K]\!]$ such that $\mathcal{W}_n = [\![1, m]\!]$ and $\mathcal{B}_n = [\![m+1, K]\!]$. We have:

- Sequential representation: Communications:

$$\tau_{n,1}^{(B,seq)} \leqslant \cdots \leqslant \tau_{n,m}^{(B,seq)} < \overbrace{\tau_{n,m+1}^{(B,seq)} = \cdots = \tau_{n,K}^{(B,seq)}}^{\mathcal{B}_n} \text{ ¡ } 1$$

  Computations:

$$1 > \underbrace{\tau_{n,1}^{(W,seq)} = \cdots = \tau_{n,m}^{(W,seq)}}_{\mathcal{W}_n} > \tau_{n,m+1}^{(W,seq)} \geqslant \cdots \geqslant \tau_{n,K}^{(W,seq)}$$

- Parallel representation: Communications:

$$\tau_{n,1}^{(B,par)} \leqslant \cdots \leqslant \tau_{n,m}^{(B,par)} < \overbrace{\tau_{n,m+1}^{(B,par)} = \cdots = \tau_{n,K}^{(B,par)}}^{\mathcal{B}_n} = 1$$

  Computations:

$$1 = \underbrace{\tau_{n,1}^{(W,par)} = \cdots = \tau_{n,m}^{(W,par)}}_{\mathcal{W}_n} > \tau_{n,m+1}^{(W,par)} \geqslant \cdots \geqslant \tau_{n,K}^{(W,par)}$$

D. Closed-form Solution of the Equations

The closed-form solutions of the equilibrium are defined by Theorem 1. Its complete proof, as well as the proofs of the other propositions and theorems presented in the remaining of this paper can be found in the extended version [9].

Theorem 1. We assume $c_1 \leqslant c_2 \leqslant \cdots \leqslant c_K$. Let us denote by $\mathcal{W}_n$ the set of players that are computation-saturated and by $\mathcal{B}_n$ the set of players that are communication-saturated on a given arbitrary worker n.

1) If $\sum_k \frac{C_n}{c_k} \leqslant K$ then $\mathcal{W}_n = \emptyset$ and

$$\forall k, \alpha_{n,k}^{(nc)} = \frac{B_n}{K.b_k}.$$

2) Else, if $\sum_k \frac{c_k}{C_n} \leqslant K$ then $\mathcal{B}_n = \emptyset$ and

$$\forall k, \alpha_{n,k}^{(nc)} = \frac{W_n}{K.w_k}.$$

3) Else, $\mathcal{B}_n$ and $\mathcal{W}_n$ are non-empty and there exists an integer $m \in [\![1; K-1]\!]$ such that

$$\frac{c_m}{C_n} < \frac{m - \sum_{k=1}^{m} \frac{c_k}{C_n}}{K - m - \sum_{k=m+1}^{K} \frac{C_n}{c_k}} < \frac{c_{m+1}}{C_n}.$$

Then, we have $\mathcal{W}_n = \{1, \ldots, m\}$ and $\mathcal{B}_n = \{m+1, \ldots, K\}$ and

$$\begin{cases} \alpha_{n,k}^{(nc)} = \frac{B_n}{b_k} \frac{|\mathcal{W}_n| - \sum_{p \in \mathcal{W}_n} \frac{c_p}{C_n}}{|\mathcal{W}_n||\mathcal{B}_n| - \sum_{p \in \mathcal{W}_n} c_p \sum_{p \in \mathcal{B}_n} \frac{1}{c_p}} & \text{if } k \in \mathcal{B}_n \\ \alpha_{n,k}^{(nc)} = \frac{W_n}{w_k} \frac{|\mathcal{B}_n| - \sum_{p \in \mathcal{B}_n} \frac{C_n}{c_p}}{|\mathcal{W}_n||\mathcal{B}_n| - \sum_{p \in \mathcal{W}_n} c_p \sum_{p \in \mathcal{B}_n} \frac{1}{c_p}} & \text{if } k \in \mathcal{W}_n \end{cases} \tag{3}$$

Sketch of the proof. If $\mathcal{B}_n = \emptyset$, then all applications use the CPU of P_n at any instant. Therefore they all receive the exact same amount of CPU, i.e. W_n/K. Hence we have $\alpha_{n,k}^{(nc)} = \frac{W_n}{K.w_k}$. Moreover, $\forall k, \tau_{n,k}^{(W,seq)} = 1/K$ and from (2) we have $1 \geqslant \sum_k \tau_{n,k}^{(B,seq)} = \sum_k \frac{c_k}{C_n} \tau_{n,k}^{(W,seq)} = \sum_k \frac{c_k}{KC_n}$. Hence $\sum_k \frac{C_n}{c_k} \leqslant K$. The case $\mathcal{W}_n = \emptyset$ is similar.

Let us now focus on the more interesting case where both $\mathcal{B}_n \neq \emptyset$ and $\mathcal{W}_n \neq \emptyset$. Using the definition of sequential communication and computation times, we have:

$$\begin{cases} \sum_{p \in \mathcal{B}_n} \tau_{n,p}^{(B,seq)} + \sum_{p \in \mathcal{W}_n} \tau_{n,p}^{(B,seq)} = 1 \\ \sum_{p \in \mathcal{B}_n} \tau_{n,p}^{(W,seq)} + \sum_{p \in \mathcal{W}_n} \tau_{n,p}^{(W,seq)} = 1 \end{cases} \tag{4}$$

Two applications from $\mathcal{B}_n$ communicate all the time. Therefore they send the exact same amount of data that we denote by $\tau_{\mathcal{B}}^{(B)}$: $\forall k \in \mathcal{B}_n, \alpha_{n,k}^{(nc)} \dfrac{B_n}{b_k} = \tau_{n,k}^{(B,seq)} = \tau_{\mathcal{B}}^{(B)}$. Similarly, we get $\forall k \in \mathcal{W}_n, \alpha_{n,k}^{(nc)} \dfrac{W_n}{w_k} = \tau_{n,k}^{(W,seq)} = \tau_{\mathcal{W}}^{(W)}$. From these relations and from (2), system (4) can be written:

$$\begin{cases} |\mathcal{B}_n| \tau_{\mathcal{B}}^{(B)} + \tau_{\mathcal{W}}^{(W)} \sum_{p \in \mathcal{W}_n} \frac{c_p}{C_n} = 1 \\ |\mathcal{W}_n| \tau_{\mathcal{W}}^{(W)} + \tau_{\mathcal{B}}^{(B)} \sum_{p \in \mathcal{B}_n} \frac{C_n}{c_p} = 1 \end{cases}$$

which can be easily solved to get (3).

Let m such that $m \in \mathcal{B}_n$ and $m+1 \in \mathcal{W}_n$. From (2) and (3) and Proposition 2, we can write:

$$\frac{c_{m+1}}{C_n} = \frac{\tau_{n,m+1}^{(B,seq)}}{\tau_{n,m+1}^{(W,seq)}} > \frac{\tau_{n,m+1}^{(B,seq)}}{\tau_{n,m}^{(W,seq)}} = \frac{m - \sum_{k=1}^{m} \frac{c_k}{C_n}}{K - m - \sum_{k=m+1}^{K} \frac{C_n}{c_k}}$$

and $\frac{c_m}{C_n} = \frac{\tau_{n,m}^{(B,seq)}}{\tau_{n,m}^{(W,seq)}} < \frac{\tau_{n,m+1}^{(B,seq)}}{\tau_{n,m}^{(W,seq)}} = \frac{m - \sum_{k=1}^{m} \frac{c_k}{C_n}}{K - m - \sum_{k=m+1}^{K} \frac{C_n}{c_k}}$

which leads to the condition on m. The reciprocity of the conditions on the sets relies on the application of the following technical result with $\gamma_k = c_k/C_n$.

Let $\gamma_1 < \cdots < \gamma_K$ be K positive numbers. We have:

1) If $\sum_k 1/\gamma_k \leqslant K$ then $\sum_k \gamma_k > K$;
2) If $\sum_k \gamma_k \leqslant K$ then $\sum_k 1/\gamma_k > K$;
3) If $\sum_k \gamma_k > K$ and $\sum_k 1/\gamma_k > K$, then there exists exactly one $m \in [\![1, K]\!]$ such that:

$$\gamma_m < \frac{\sum_{k=1}^{m} 1 - \gamma_k}{\sum_{k=m+1}^{K} 1 - \frac{1}{\gamma_k}} < \gamma_{m+1}. \qquad \blacksquare$$

Corollary 1. From these equations, we see that there always exists exactly one non-cooperative equilibrium.

IV. Inefficiencies and Paradoxes

In this section, we study the inefficiencies of the Nash equilibria, in the Pareto sense, and their consequences. Let us start by recalling the definition of the Pareto optimality.

Definition 4 (Pareto optimality). Let G be a game with K players. Each of them is defined by a set of possible strategies $\mathcal{S}_k$ and utility functions u_k defined on $\mathcal{S}_1 \times \cdots \times \mathcal{S}_K$.[3] A vector of strategy is said to be Pareto optimal if it is impossible to strictly increase the utility of a player without strictly decreasing the one of another. In other words, $(s_1, \ldots, s_K) \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_K$ is Pareto optimal if and only if:

$$\forall (s_1^*, \ldots, s_K^*) \in \mathcal{S}_1 \times \cdots \times \mathcal{S}_K,$$
$$\exists i, u_i(s_1^*, \ldots, s_K^*) > u_i(s_1, \ldots, s_K) \Rightarrow$$
$$\exists j, u_j(s_1^*, \ldots, s_K^*) < u_j(s_1, \ldots, s_K).$$

We recall that, in the considered system, the utility functions are the α_k, that is to say, the average number of tasks of application k processed per time-unit, while the strategies are the scheduling algorithms (i.e. which resources to use and when to use them).

In this section, we comment on the efficiency of the Nash equilibrium, in the case of a single worker (Section IV-A), and then of multiple workers (IV-B) and propose in Section IV-C a brief study of the well-known "price of anarchy". Unfortunately, this metric does not enable to distinguish Pareto optimal points from non-Pareto optimal ones. That is why we also propose an alternate definition, the "*selfishness degradation factor*". Last, it is known that when Nash equilibria are inefficient, some paradoxical phenomenon can occur (see, for instance [8]). We hence study in Section IV-D, the occurrence of Braess paradox in this system.

A. Single Processor

We can show that when the players (here the applications) compete in a non-cooperative way over a single processor, the resulting Nash equilibrium is Pareto optimal (see the extended version [9] for a detailed proof).

Proposition 3. On a single-processor system, the allocation at the Nash equilibrium is Pareto optimal.

[3]Note that the utility of a player depends on its own strategy and on the strategies of all the other players.

B. Multi-processors and Inefficiencies

Interestingly, although the Nash equilibria are Pareto optimal on any single-worker system, we show in this section that these equilibria are not always Pareto optimal for a system consisting of several processors.

We first exhibit this phenomenon on a simple example consisting of two machines and two applications (Section IV-B.1). We then provide a very simple characterization of the systems under which the non-cooperative competition leads to inefficiencies (Section IV-B.2).

1) Example of Pareto inefficiency: The Pareto optimality is a global notion. Hence, although for each single-processor system, the allocation is Pareto optimal, the result may not hold for an arbitrary number of machines. This phenomenon was illustrated in Section II-E.

2) Necessary and sufficient condition: We prove in [9] the following very simple characterization of the systems under which the non-cooperative competition leads to inefficiencies.

Theorem 2. Consider a system $S = (K, b, w, N, B, W)$ as defined in Definition 3. Suppose that the applications are not all identical, that is to say that there exists k_1 and k_2 such that $c_{k_1} < c_{k_2}$.

Then, the allocation at the Nash equilibrium is Pareto inefficient if and only if there exists two workers, namely n_1 and n_2 such that $\mathcal{W}_{n_1} = \emptyset$ and $\mathcal{B}_{n_2} = \emptyset$.

C. Measuring Pareto Inefficiency

We have seen that the Nash equilibrium of the system can be Pareto inefficient. A natural question is then "how much inefficient is it?". Unfortunately, measuring Pareto inefficiency is still an open question. It is hence the focus of this section.

1) Definitions: Papadimitriou [7] introduced the now popular measure "price of anarchy" that we will study in this section.

Let us consider an efficiency measure f on the α_k. For a given system S (i.e. platform parameters along with the description of our K applications), we denote by $\alpha_k^{(nc)}(S)$, the rates achieved on system S by the non-cooperative algorithm. For any given metric f, let $\left(\alpha_k^{(f)}(S)\right)_{1 \leqslant k \leqslant K}$ be a vector of optimal rates on system S for the metric f. We define the inefficiency $I_f(S)$ of the non-cooperative allocation for a given metric and a given system as

$$I_f(S) = \frac{f\left(\alpha_1^{(f)}(S), \ldots, \alpha_K^{(f)}(S)\right)}{f\left((\alpha_1^{(nc)}(S), \ldots, \alpha_K^{(nc)}(S)\right)} \geqslant 1.$$

Papadimitriou focuses on the profit metric Σ defined by $\Sigma(\alpha_1, \ldots, \alpha_K) = \frac{1}{K} \sum_{k=1}^{K} \alpha_k$. The price of anarchy ϕ_Σ is then be defined as the largest inefficiency:

$$\phi_\Sigma = \max_S I_\Sigma(S) = \max_S \frac{\sum_k \alpha_k^{(\Sigma)}(S)}{\sum_k \alpha_k^{(nc)}(S)} \geqslant 1.$$

2) Studying the Price of Anarchy on a simple example: Let us consider the following simple system $S_{M,K}$ defined by $N = 1$, $B_1 = 1$, $W_1 = 1$, $b = (\frac{1}{M}, 1, \ldots, 1)$, and $w = (\frac{1}{M}, 1, \ldots, 1)$. It is then easy to compute the following allocations (see Figure 5):

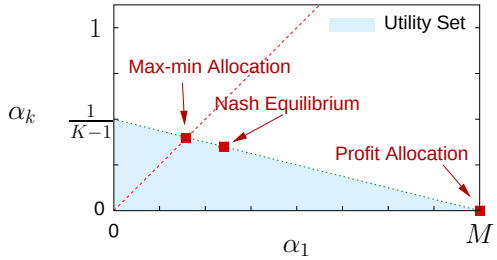


Fig. 5. Utility set and allocations for $S_{M,K}$ ($K = 3, M = 2$).

- $\alpha^{(nc)}(S_{M,K}) = \left(\frac{M}{K}, \frac{1}{K}, \ldots, \frac{1}{K}\right)$ corresponds to the non-cooperative allocation;
- $\alpha^{(\Sigma)}(S_{M,K}) = (M, 0, \ldots, 0)$ corresponds to the allocation optimizing the average throughput;
- $\alpha^{(\min)}(S_{M,K}) = \left(\frac{1}{K-1+1/M}, \ldots, \frac{1}{K-1+1/M}\right)$ corresponds to the max-min fair allocation [18];
- $\alpha^{(\Pi)}(S_{M,K}) = \left(\frac{M}{K}, \frac{1}{K}, \ldots, \frac{1}{K}\right)$ corresponds to the proportionally fair allocation which is a particular Nash Bargaining Solution [18]. Surprisingly, on this instance, this allocation also corresponds to the non-cooperative one.

Note that, $\alpha^{(\Sigma)}$, $\alpha^{(\min)}$, and $\alpha^{(\Pi)}$ are Pareto optimal by definition. One can easily compute

$$I_\Sigma(S_{M,K}) = \frac{M}{\frac{M}{K} + \frac{K-1}{K}} \xrightarrow[M\to\infty]{} K.$$

The price of anarchy is therefore unbounded. However, the fact that the non-cooperative equilibria of such instances are Pareto-optimal and have interesting properties of fairness (they correspond to a Nash Bargaining Solution [18]) questions the relevance of the *price of anarchy* notion as a Pareto efficiency measure.

Likewise, the inefficiency of the max-min fair allocation is equivalent to M for large values of M (as opposed to K for the non-cooperative equilibrium). It can hence be unbounded even for bounded number of applications and machines. This seems even more surprising as such points generally result from complex cooperations and are hence Pareto optimal. These remarks raise once more the question of the measure of Pareto inefficiency.

3) Selfishness Degradation Factor: The previous problems are not specific to the efficiency measure Σ. The same kind of behavior can be exhibited when using the $\min$ or the product of the throughputs. That is why we think that Pareto inefficiency should be measured as the *distance* to the Pareto border and not to a specific point.

Based on the definition of the Pareto optimality, one can define the concept of strict Pareto-superiority.

Definition 5 (Pareto-superiority). A utility point α is said *strictly Pareto-superior* to a point β if for all player k we have $\alpha_k > \beta_k$.

Obviously, a Pareto-optimal point is such that there is no achievable point strictly Pareto-superior to it. To quantify the degradation of Braess-like Paradoxes (the degree of Paradox), Kameda [19] introduced the Pareto-comparison of α and β as $\varrho(\alpha, \beta) = \min_k \frac{\alpha_k}{\beta_k}$. Therefore, α is strictly superior to β iff $\varrho(\alpha, \beta) > 1$. Intuitively ϱ represents the performance degradation between α and β. Using this definition, we propose the following definition of Pareto inefficiency: $I(S) = \max_{\alpha \in U(S)} \varrho(\alpha, \alpha^{(nc)}(S))$. Therefore $\alpha^{(nc)}(S)$ is Pareto inefficient as soon as $I(S) > 1$ and the larger $\max_\alpha I(\alpha, \alpha^{(nc)})$, the more inefficient the Nash equilibrium. The *selfishness degradation factor* can then be defined as

$$\phi = \max_S I(S) = \max_S \max_{\alpha \in U(S)} \min_k \frac{\alpha_k}{\alpha^{(nc)}_{n,k}(S)}.$$

A system (e.g., queuing network, transportation network, load-balancing, ...) that would be such that the Nash equilibria are always Pareto optimal would have a selfishness degradation factor equal to one. The selfishness degradation factor may however be unbounded on systems where non-cooperative equilibria are particularly inefficient. The relevance of this definition is corroborated by the fact that ε-approximations of Pareto-sets defined by Yannakakis and Papadimitriou [20] have a degradation factor of $1 + \varepsilon$. It can easily be shown that the systems studied in this article have a selfishness degradation factor larger than two but the exact value remains an open problem.

D. Braess-like Paradoxes

When studying properties of Nash equilibria in routing systems, Braess exhibited an example in which, by adding resource to the system (in his example, a route), the performance of all the users were degraded [8]. We investigate in this section whether such situations can occur in our scenario.

Let us consider a system (called "initial") and a second one (referred to as the "augmented" system), derived from the first one by adding some quantity of resource. Intuitively, the Nash equilibrium aug in the augmented system should be Pareto-superior to the one in the initial system ini. We say that a Braess paradox happens when ini is strictly Pareto-superior to point aug.

Obviously, every achievable state in the initial system is also achievable in the augmented system. Hence if a is an achievable point in the initial system and if b is a *Pareto* optimal point is the augmented one, then a cannot be strictly Pareto superior to b. Hence Braess paradoxes are consequences of the Pareto inefficiencies of the Nash equilibria.

We show that, even though the Nash equilibria may be Pareto inefficient, in the considered scenario, Braess paradoxes cannot occur.

Theorem 3. In the non-cooperative multi-port scheduling problem, Braess like paradoxes cannot occur.

Sketch of the proof. We first need to introduce the definition of equivalent subsystem. Consider a system $S = (K, b, w, N, B, W)$. We define the new subsystem $\widetilde{S} = (K, b, w, N, \widetilde{B}, \widetilde{W})$ by: for each worker n,

$$\widetilde{W}_n = \begin{cases} \sum_k \frac{B_n}{K c_k} & \text{if } \mathcal{W}_n = \emptyset, \\ W_n & \text{otherwise,} \end{cases}$$

$$\text{and} \quad \widetilde{B}_n = \begin{cases} \sum_k \frac{W_n c_k}{K} & \text{if } \mathcal{B}_n = \emptyset, \\ B_n & \text{otherwise.} \end{cases}$$

We now precise why $\widetilde{S}$ is said to be an equivalent subsystem of S. Consider a system $S = (K, b, w, N, B, W)$ and its Nash equilibrium $\alpha^{(nc)}$. One can check that:

i) The system $\widetilde{S}$ is a subsystem of S, i.e. for all worker n: $\widetilde{B}_n \leqslant B_n$ and $\widetilde{W}_n \leqslant W_n$.

ii) The Nash equilibrium $\widetilde{\alpha}^{(nc)}$ of the subsystem $\widetilde{S}$ verifies

$$\forall n, \forall k, \alpha_{n,k}^{(nc)} = \widetilde{\alpha}_{n,k}^{(nc)}.$$

iii) The Nash equilibrium $\widetilde{\alpha}^{(nc)}$ of the subsystem $\widetilde{S}$ is Pareto-optimal.

The conclusion of the proof relies on the following result: Consider two systems $S = (K, b, w, N, B, W)$ and $S' = (K, b, w, N, B', W')$ and their respective equivalent subsystems $\widetilde{S} = (K, b, w, N, \widetilde{B}, \widetilde{W})$ and $\widetilde{S}' = (K, b, w, N, \widetilde{B}', \widetilde{W}')$. Suppose that $\forall n, B'_n \geqslant B_n$ and $W'_n \geqslant W_n$ then $\forall n, \widetilde{B}'_n \geqslant \widetilde{B}_n$ and $\widetilde{W}'_n \geqslant \widetilde{W}_n$. ■

V. Performance Measures

In this section we show that unexpected behavior of some typical performance measures can occur even for Pareto optimal situations. To ensure optimality of the Nash equilibrium, we consider applications running on a single processor (Proposition 3).

We recall that the Pareto optimality is a global performance measure. Hence, it is possible that, while the resources of the system increase (either by the adding of capacity to a link or of computational capabilities to a processor), a given performance measure decreases while the equilibrium remains Pareto optimal. The aim of this section is to illustrate this phenomenon on some typical performance measures.

More precisely, we show the non-monotonicity of the maximal throughput, of the minimal throughput and of the average throughput. We finally end this section with a numerical example where these measures decrease simultaneously with the increase of the resource.

A. Lower Bound on the Maximal Degradation

In the following, we suppose that only one of the resource of the system increases. By symmetry, we suppose that the computational capacity (W_1) is constant, while the link capacity B_1 increases.[4]

Let us introduce $\underline{B} = WK/\sum_k \frac{1}{c_k}$ and $\overline{B} = \frac{W}{K}\sum_k c_k$. From Theorem 1, when considering the equations at the Nash equilibrium, we can distinguish 3 cases:

- 2 "saturated" situations that are:
  - $\boldsymbol{sat\mathcal{W}_n}$ If $B \leqslant \underline{B}$, then $\alpha_k^{(nc)} = \frac{B}{K.b_k}$, i.e. the throughput of each application is proportional to B.
  - $\boldsymbol{sat\mathcal{B}_n}$ If $B \geqslant \overline{B}$ then $\alpha_k^{(nc)} = \frac{W}{K.w_k}$, i.e. the throughput of each application is constant with respect with B.
- 1 "continuous" situation when $\underline{B} < B < \overline{B}$

Obviously, in the "saturated" situations, the throughput $\alpha_k^{(nc)}$ are increasing or constant and the order between the applications is preserved. (I.e. if for $B \leqslant \underline{B}$ (resp. $B \geqslant \overline{B}$), $\alpha_{k_1}^{(nc)} \leqslant \alpha_{k_2}^{(nc)}$ then for all $B' \leqslant \underline{B}$ (resp. $B' \geqslant \overline{B}$) we have $\alpha_{k_1}^{(nc)} \leqslant \alpha_{k_2}^{(nc)}$.)

To simplify the analysis, we consider the degradation obtained when $B = \underline{B}$ compared to the situation where $B = \overline{B}$. It is hence a lower bound on the actual maximum achievable degradation.

Consider an arbitrary application k. We write $\alpha_k^{(nc)}{}_{\text{before}}$ (resp. $\alpha_k^{(nc)}{}_{\text{after}}$) the value of its throughput when $B = \underline{B}$ (resp. $B = \overline{B}$). Hence, $\alpha_k^{(nc)}{}_{\text{before}} = \frac{B}{Kb_k} = \frac{W}{b_k \sum_p \frac{1}{c_p}}$ and $\alpha_k^{(nc)}{}_{\text{after}} = \frac{W}{Kw_k}$.

Remark 2. Note that $\frac{\alpha_{n,k}^{(nc)}{}_{\text{before}}}{\alpha_{n,k}^{(nc)}{}_{\text{after}}} = \frac{K}{c_k \sum_p 1/c_p}$. The lower bound on the degradation is hence proportional to $1/c_k$ and is therefore maximal for the application with the smallest coefficient c_k. For instance, if $\forall p \neq k, c_p = K$ and $c_k = 1$, then $\frac{\alpha_{\text{before}}}{\alpha_{\text{after}}} \sim K/2$. Hence, when the number of applications grows to infinity, the degradation of the application having the smaller c_k also grows to infinity.

We can now easily show that even in a single processor system, the maximal (and minimal) throughput can strictly decrease with the adding of resource. Note that:

- if $B_n \leqslant \underline{B}$ the application k having the highest (resp. smallest) throughput $\alpha_{n,k}^{(nc)}$ is the one having the smallest (resp. highest) value of b_k.
- if $B \geqslant \overline{B}$ the application k having the highest (resp. smallest) throughput $\alpha_{n,k}^{(nc)}$ is the one whose w_k is the smallest (resp. highest).

Hence, a lower bound on the maximal degradation of the maximal (resp. minimal) throughput is $\frac{K}{\sum_k 1/c_k}\frac{\min_k w_k}{\min_k b_k}$ (resp. $\frac{K}{\sum_k 1/c_k}\frac{\max_k w_k}{\max_k b_k}$). Therefore, for appropriate choices of $\min_k b_k$ and $\min_k w_k$ (for all c_k fixed), the degradation can be chosen arbitrarily large.

Finally, note that $\frac{\sum_k \alpha_{n,k}^{(nc)}{}_{\text{before}}}{\sum_k \alpha_{n,k}^{(nc)}{}_{\text{after}}} = \frac{K}{\sum_k 1/c_k} \cdot \frac{\sum_k 1/b_k}{\sum_k 1/w_k}$. Hence, for some combinations of w_k, b_k and c_k, the degradation of the average throughput can be arbitrarily large (e.g., for $w = (1, 1/\varepsilon, \ldots, 1/\varepsilon)$ and $b = (1, 1/\varepsilon^2, \ldots, 1/\varepsilon^2)$, we get a ratio of $K\frac{1+(K-1)\varepsilon^2}{(1+(K-1)\varepsilon)^2} \xrightarrow[\varepsilon \to 0]{} K$).

B. Numerical Example

We end this section with an example in which all the performance measures we considered are simultaneously degraded when the bandwidth B of the link connecting the master to the worker is increased.

Consider the example represented in Fig. 6. Observe that when the bandwidth B is $245/24 \simeq 10.208$ the three measures (namely the higher throughput, the lower throughput and the average throughput) have lower values than when the bandwidth B is only equal to $560/73 \simeq 7.671$.

As $\min_k w_k = w_4$ and $\min_k b_k = b_4$, then the application having the higher throughput in both $sat\mathcal{B}_n$ and $sat\mathcal{W}_n$ is application 4, and a lower bound of the degradation is $112/73$.

As $\max_k w_k = w_3$ and $\max b_k = b_1$, then the application having the lower throughput is application 3 in $sat\mathcal{B}_n$ and application 1 in $sat\mathcal{W}_n$, and a lower bound of the degradation is $84/73$.

[4] In the following, we will omit the subscript "1" as only one worker is considered in the system.

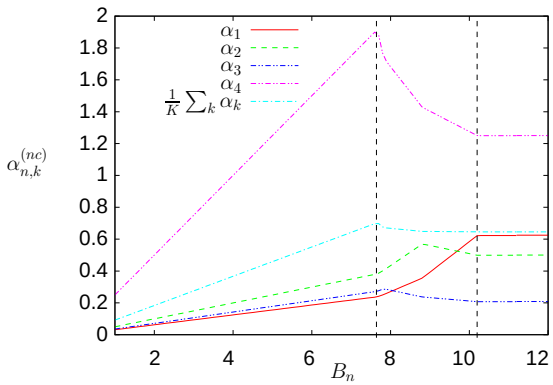


Fig. 6. The three performance measures can simultaneously decrease with the resource: b = { 8,5,7,1}, w = { 4,5,12,2}, K=4, W =10.

Finally, a lower bound of the degradation of the average performance is $2466/2263$.

VI. Conclusion

We have presented a simple yet realistic situation where a fair and Pareto-optimal *system-level* sharing fails to achieve an efficient *application-level* sharing. Even though the system achieves a perfect sharing of resources between applications, the non-cooperative usage of the system leads to important application performance degradation and resource wasting. We have proved the existence and uniqueness of the Nash equilibrium in our framework and extensively studied its property. Surprisingly, the equilibrium is Pareto-optimal on each worker independently. However, it may not be globally Pareto-optimal. We have proved that no Braess-like paradoxical situations could occur, which is, to the best of our knowledge, the first situation where Pareto-inefficient non-cooperative equilibrium cannot lead to Braess-like paradox. However, some seemingly paradoxical situations can occur. Indeed, even when the equilibria are Pareto optimal, their performance can be arbitrarily bad for any classical performance measure.

This study led us to the natural question of the inefficency measure. After briefly commenting on the notion of "price of anarchy", we proposed a new definition, called SDF (Selfishness Degradation Factor).

The key hypothesis for deriving a closed-form description of the equilibria is the multi-port hypothesis. Under this hypothesis, some time information could be lost when using equivalent representations, which resulted in simpler equations than if a 1-port model had been used (i.e. if the master can communicate with only one worker at a given instant). Preliminary simulations with this model show that Braess-like paradoxes may occur. The understanding of such phenomena are crucial to large-scale system planing and development as there is no way to predict their apparition so far. Analytical characterizations of such a framework could provide significant insights on the key ingredients necessary to the occurrence of Braess-like paradoxes.

Last, we can conclude from this study that cooperation between applications is essential to avoid inefficiencies (even for simple applications constituted of a huge number of independent identical tasks). As far as the framework of this article is concerned, some steps in this direction have been given in [6] where some distributed algorithms were proposed and compared to an optimal but centralized one. However, in their work, there was a single scheduler whose duty was to achieve the best throughput for all applications while ensuring a max-min fair share. In a fully-decentralized setting, as considered in the present article, some form of cooperation (e.g., similar to the one proposed by [21] for elastic traffic in broadband networks) between different schedulers should be designed.

References

[1] SETI, URL: http://setiathome.ssl.berkeley.edu.

[2] Prime, URL: http://www.mersenne.org.

[3] "Berkeley Open Infrastructure for Network Computing," http://boinc.berkeley.edu.

[4] "Einstein@Home," http://einstein.phys.usm.edu.

[5] "Large Hadron Collider," http://lhc.web.cern.ch/lhc/.

[6] O. Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, Loris Marchal, and Yves Robert, "Centralized versus distributed schedulers multiple bag-of-task applications," in *International Parallel and Distributed Processing Symposium IPDPS'2006*. IEEE Computer Society Press, 2006.

[7] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *STACS*, 1998.

[8] D. Braess, "ber ein paradoxien aus der verkehrsplanung," *Unternehmensforschung*, vol. 12, pp. 258–68, 1968.

[9] A. Legrand and C. Touati, "Non-cooperative scheduling of multiple bag-of-task applications," INRIA, Tech. Rep. RR-5819, 2005.

[10] M. Banikazemi, J. Sampathkumar, S. Prabhu, D. Panda, and P. Sadayappan, "Communication modeling of heterogeneous networks of workstations for performance characterization of collective operations," in *HCW'99, the 8th Heterogeneous Computing Workshop*. IEEE Computer Society Press, 1999, pp. 125–133.

[11] A. Bar-Noy, S. Guha, J. S. Naor, and B. Schieber, "Message multicasting in heterogeneous networks," *SIAM Journal on Computing*, vol. 30, no. 2, pp. 347–358, 2000.

[12] C. Banino, Olivier Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, and Yves Robert, "Scheduling strategies for master-slave tasking on heterogeneous processor platforms," *IEEE Trans. Parallel Distributed Systems*, vol. 15, no. 4, pp. 319–330, 2004.

[13] P.-F. Dutot, "Complexity of master-slave tasking on heterogeneous trees," *European Journal of Operational Research*, 2004, special issue on the Dagstuhl meeting on Scheduling for Computing and Manufacturing systems.

[14] B. Hong and V. Prasanna, "Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput," in *International Parallel and Distributed Processing Symposium IPDPS'2004*. IEEE Computer Society Press, 2004.

[15] C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand, and Y. Robert, "Scheduling strategies for master-slave tasking on heterogeneous processor platforms," *IEEE Trans. Parallel Distributed Systems*, vol. 15, no. 4, pp. 319–330, 2004.

[16] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences USA*, vol. 36, pp. 48–49, 1950.

[17] ——, "Noncooperative games," *Annal of Mathematics*, vol. 54, pp. 286–295, 1951.

[18] C. Touati, E. Altman, and J. Galtier, "Generalised Nash bargaining solution for banwidth allocation," *Computer Networks*, 2006 (to appear).

[19] H. Kameda, "Bounds on benefits and harms of adding connections to noncooperative networks," in *NETWORKING 2004*, ser. LNCS, N. Mitrou, K. Kontovasilis, G. N. Rouskas, I. Iliadis, and L. Merakos, Eds., vol. 3042. Springer Verlag, 2006, pp. 405–417.

[20] C. H. Papadimitriou and M. Yannakakis, "On the approximability of trade-offs and optimal access of web sources," in *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2000, p. 86.

[21] H. Yaïche, R. R. Mazumdar, and C. Rosenberg, "A game theoretic framework for bandwidth allocation and pricing in broadband networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667–678, 2000.