

Online Allocation of Splittable Clients to Multiple Servers on Large Scale Heterogeneous Platforms

Olivier Beaumont¹ and Lionel Eyraud-Dubois¹ and Hejer Rejeb¹
and Christopher Thraves¹ †

¹INRIA Bordeaux – Sud-Ouest, University of Bordeaux, LaBRI

Dans cet article, nous considérons l'allocation dynamique (online) d'un très grand nombre de tâches identiques et indépendantes sur une plate-forme maîtres-esclaves. Initialement, plusieurs nœuds maîtres possèdent ou génèrent les tâches qui sont ensuite transférées et traitées par des nœuds esclaves. L'objectif est de maximiser le débit (*i.e.*, le nombre fractionnaire de tâches qui peuvent être traité en une unité de temps, en régime permanent, par la plate-forme). Nous considérons que les communications se déroulent suivant le modèle multi-port à degré borné, dans lequel plusieurs communications peuvent avoir lieu simultanément sous réserve qu'aucune bande passante ne soit dépassée et qu'aucun serveur n'ouvre simultanément un nombre de connections supérieur à son degré maximal. Sous ce modèle, la maximisation du débit correspond au problème Maximum-Throughput-Bounded-Degree (MTBD) qui a été analysé dans [BEDRT08]. Il a été montré que le problème est NP-Complet au sens fort mais qu'une augmentation (additive) de ressources minimale (de 1) sur le degré maximal des serveurs permet de le résoudre en temps polynomial. Dans cet article, nous considérons une extension de MTBD à la situation plus réaliste, dans le contexte des plates-formes de calcul à grande échelle, dans laquelle les nœuds esclaves rejoignent et quittent dynamiquement la plate-forme à des instants arbitraires (problème *online* MTBD). Nous montrons tout d'abord qu'aucun algorithme complètement à la volée (c.-à.-d. qui n'autorise pas les déconnexions) ne peut conduire à un facteur d'approximation constant, quelle que soit l'augmentation de ressources utilisée. Ensuite, nous montrons qu'il est en fait possible de maintenir à tout instant la solution optimale (avec une augmentation de ressource additive de 1) en ne réalisant à chaque modification de la plate-forme qu'une déconnection et qu'une nouvelle connection par maître.

Keywords: task allocation, large scale platforms, desktop grid computing, online algorithms, resource augmentation.

1 Introduction

Nowadays, scientific research has brought challenging calculations to solve intractable problems for a single machine. Desktop Grids have appeared as an interesting and cheap solution to perform such computations. Desktop Grids take benefit of machine idle times in a network to, altogether, perform a hard computation. On the Internet, where only 5 to 10% of the computational power of personal computers is used, platforms like BOINC [And04] uses volunteer machines to perform hard computations in mathematics, biology, medicine or to search for intelligent life outside earth in the case of SETI@home. Each machine performs a small chunk of a huge computation, small enough not to disturb the work of the volunteer machines. All the applications running on these platforms consist in a huge number of independent tasks and all communications take place under master-worker paradigm (*i.e.*, there are no direct communication between workers). In this context, we consider the problem of allocating a large number of independent, equal-sized tasks to a heterogeneous large scale computing platform.

†This work was partially supported by the French ANR project Alpage.

We model the platform using a set of servers that initially hold (or generate) the tasks to be processed by a set of workers (volunteers). All resources have different speeds of communication and computation, and we model contentions using the bounded multi-port model. Under this model, a processor can be involved simultaneously in several communications, provided that neither its incoming nor its outgoing bandwidths is exceeded. But, for the sake of realism, another parameter needs to be introduced in order to bound the number of simultaneous connections that can be opened and handled at a server node.

Formally, let S_j be a server, and let b_j and d_j denote the outgoing bandwidth of server S_j and the maximal number of connections that it can handle simultaneously, respectively. Also, let us denote by w_i the capacity of client C_i . Finally, let us denote by w_i^j the fraction of the outgoing bandwidth allocated by server S_j to client C_i . Then, a valid allocation must satisfy the following conditions

$$\forall j, \quad \sum_i w_i^j \leq b_j \quad \text{outgoing bandwidth constraint at } S_j \quad (1)$$

$$\forall j, \quad \text{Card}\{i, w_i^j > 0\} \leq d_j \quad \text{degree constraint at } S_j \quad (2)$$

$$\forall i, \quad \sum_j w_i^j \leq w_i \quad \text{capacity constraint at } C_i \quad (3)$$

Therefore, MTBD is defined as follows

$$\text{Maximize } \sum_j \sum_i w_i^j \text{ under constraints (1), (2) and (3).}$$

Under this model, it was proved in [BEDRT08] that the additional parameter (2) makes MTBD problem NP-Complete. On the other hand, a polynomial time algorithm, called SEQ, has been proposed. It is based on the use of a small resource augmentation on the maximal number of connections a server can handle simultaneously. More specifically, the throughput achieved using algorithm SEQ and $d_j + 1$ connections for server S_j is at least the same as the optimal one using d_j connections.

Due to the dynamic nature of large scale heterogeneous platforms and the results mentioned above, it becomes interesting to study MTBD in the more realistic dynamic scenario, when clients join and leave the system at any moment.

2 Model

In order to formalize the dynamism of the platform, online MTBD is defined using rounds. A new round begins each time a client joins or leaves the system. Let \mathcal{LC}^t be the list of clients in round t (with their respective capacity constraints). Client C joins the system at round t if $C \in \mathcal{LC}^t \setminus \mathcal{LC}^{t-1}$. Equivalently, Client C leaves the system at round t if $C \in \mathcal{LC}^{t-1} \setminus \mathcal{LC}^t$. Thus, arrivals and departures of clients only take place at the beginning of a round and exactly one arrival or one departure of a client takes place at each round, i.e., $|\mathcal{LC}^t \setminus \mathcal{LC}^{t-1}| + |\mathcal{LC}^{t-1} \setminus \mathcal{LC}^t| = 1$ for all t . Let us denote by \mathcal{LS} the list of servers (with their respective bandwidths and the number of connections they can handle simultaneously). Therefore Online MTBD consists in

Solving MTBD for each round using the corresponding input sets \mathcal{LS} and \mathcal{LC}^t .

Measure In order to compare the performance of online algorithms that achieve the same throughput, we introduce the following definitions. Let us denote by $w_i^j(t)$ the fraction of outgoing bandwidth allocated by server S_j to client C_i during round t . Then, we say that client C_i is *connected* to server S_j in round t if $w_i^j(t) > 0$. Furthermore, let $S_j(\mathcal{LC}^t) = \{C_i, w_i^j(t) > 0\}$ be the set of clients connected to server S_j during round t . Then, we define the set of *new connections* to server S_j in round t as $S_j(\mathcal{LC}^t) \setminus S_j(\mathcal{LC}^{t-1})$, the set of clients connected to server S_j in round t that were not connected to server S_j in round $t - 1$. Similarly, we define the set of *disconnections* of server S_j during round t as $S_j(\mathcal{LC}^{t-1}) \setminus S_j(\mathcal{LC}^t)$, the set of clients connected to server S_j in round $t - 1$ that were not connected during round t . At last, we say that the connection between server S_j and client C_i *changed* during round t if $w_i^j(t-1) \neq w_i^j(t)$, and both are positive (C_i is connected to S_j in rounds $t - 1$ and t , but with different capacities). Let us denote by $Ch_j(\mathcal{LC}^t)$ the set of all the clients connected to server S_j during round t and whose connections changed between round $t - 1$ and round t .

In the former context, clients connected to a server may change from one round to the next one. Hence, in each round a server will have to establish new connections, clients will be disconnected or their assigned bandwidths will change. Disconnections are for free in terms of time cost. On the other hand, new connections and changes in already existing connections require to manipulate TCP connections and therefore are time expensive. Hence, in order to limit the cost induced at each round, it is necessary to reduce as much as possible new connections and changes in already existing connections. And then, if two online algorithms achieve the same throughput, the algorithm that produces less new connections and changes in already existing connections per round is to be preferred. On the other hand, and even assuming that disconnections are for free, we prefer algorithms that reduce the number of disconnections per round. Therefore, we introduce the following definitions to measure and compare the algorithms that solves online MTBD problem and that achieve the same throughput.

Definition 2.1 Let \mathcal{A} be an algorithm solving the online version of MTBD. It is said that \mathcal{A} produces at most l disconnections per round per server if $\max_t \max_{S_j \in \mathcal{LS}} |S_j(\mathcal{LC}^{t-1}) \setminus S_j(\mathcal{LC}^t)| = l$.

Definition 2.2 Let \mathcal{A} be an algorithm solving the online version of MTBD. It is said that \mathcal{A} produces at most l new connections per round per server if $\max_t \max_{S_j \in \mathcal{LS}} |S_j(\mathcal{LC}^t) \setminus S_j(\mathcal{LC}^{t-1})| = l$.

Definition 2.3 Let \mathcal{A} be an algorithm solving the online version of MTBD. It is said that \mathcal{A} produces at most l changes on the already existing connections per round per server if $\max_t \max_{S_j \in \mathcal{LS}} |Ch_j(\mathcal{LC}^t)| = l$.

3 Results

Now, we can formally state the results presented in this work. Using definitions 2.1, 2.2, and 2.3 we define an algorithm \mathcal{A} as *fully online* if it produces zero disconnections nor changes in bandwidth allocations, and at most one new connection each time a new client arrives. During a round when a client leaves the system, \mathcal{A} produces one disconnection (the natural one), but no new connections, and no changes in already existing connections. On the other hand, we say that an algorithm uses *additive resource augmentation ratio* α if it connects at most $d_j + \alpha$ clients to server S_j , while the original degree constraint is d_j , as stated in Equation (2). Moreover, let us denote by $I = \{I_t\}_{t \in T}$ an instance, where I_t describes changes that take place at the beginning of round t . Also, let us denote by $OPT(I_t)$ the optimal throughput on instance I at round t , and by $\mathcal{A}(I_t)$ the throughput provided by an algorithm \mathcal{A} on instance I at round t . Hence, we can state the first result about unapproximability of the problem.

Theorem 3.1 Given a resource augmentation factor α and a constant k ; there exists an instance I of online MTBD and a value t , such that for any fully online algorithm \mathcal{A} used to solve online MTBD,

$$\mathcal{A}(I_{t'}) < \frac{1}{k} OPT(I_{t'}) \quad \text{for all } t' \geq t.$$

The algorithm The previous result underlines the interest of knowing how many new connections, disconnections and changes in already existing connections are required in order to achieve some desired throughput. In our case, since it is known that with additive resource augmentation equal to 1 it is possible to obtain the optimal throughput, how many new connections per round are required to achieve the optimal throughput for online MTBD assuming additive resource augmentation equal to 1.

Let us start with the seed of the algorithm, the criterion to connect clients from a list to a single server. Let $\mathcal{LC} = C_1, C_2, \dots, C_n$ be the list of clients, and assume that \mathcal{LC} is ordered by increasing values of the capacities of the clients, $w_1 \leq w_2 \leq \dots \leq w_n$. Let $\mathcal{LC}(l, k) = \sum_{i=l}^k w_i$ denote the sum of the capacities of the clients in \mathcal{LC} between C_l and C_k , both included. Additionally, let b and d denote the outgoing bandwidth and degree constraint of a server S . Let $S(\mathcal{LC})$ be the set of clients in C connected with server S .

Definition 3.2 If there exists an index l in \mathcal{LC} such that $\mathcal{LC}(l, l+d-1) < b$ and $\mathcal{LC}(l+1, l+d) \geq b$ then, $S(\mathcal{LC}) := C_l, C_{l+1}, \dots, C_{l+d-1}, C'_{l+d}$, where C'_{l+d} is the fraction of C_{l+d} such that $\mathcal{LC}(l, l+d-1) + w'_{l+d} = b$. Otherwise either $\mathcal{LC}(1, d) \geq b$ or $\mathcal{LC}(n-d+1, n) < b$. When $\mathcal{LC}(1, d) \geq b$, then $S(\mathcal{LC}) := C_1, C_2, \dots, C_l, C'_{l+1}$, where l is the index such that $\mathcal{LC}(1, l) < b$ and $\mathcal{LC}(1, l+1) \geq b$, and C'_{l+1} is the fraction of C_{l+1} such that $\mathcal{LC}(1, l) + w'_{l+1} = b$. It is possible that $w_1 \geq b$, in that case l is considered to be 0 and

$S(\mathcal{L}C)$ only contains C'_1 , the fraction of C_1 with size equal to b . Finally, when $\mathcal{L}C(n-d+1, n) < b$, then $S(\mathcal{L}C) := C_{n-d+1}, C_{n-d+2}, \dots, C_{n-1}, C_n$.

Once Definition 3.2 is applied to connect clients in $\mathcal{L}C$ with server S , the list of clients $\mathcal{L}C$ is actualized. To do that, $S(\mathcal{L}C)$ is removed from the list and the remaining part of the split client (if it exists) is reinserted in the list following its size so as to maintain the increasing order. Let us denote by $\mathcal{L}C_S$ the *actualized* list of clients. Besides the above explained way to connect clients of a list with one server, it is possible to recursively connect clients from a list with several servers. Therefore, let $\mathcal{L}S = S_1, S_2, \dots, S_m$ be the list of servers. Hence, applying Definition 3.2 and then actualizing the list of clients consecutively, first to S_1 , then to S_2 and so on until S_m , each server is assigned a list of clients. We therefore present *Online Algorithm SEQ* (OSEQ), an algorithm that solves online MTBD. In each round, algorithm OSEQ recomputes the connections for all the servers using Definition 3.2. Algorithm 1 provides a formal description of Algorithm OSEQ, and the complete presentation can be found in [BEDRT09].

Algorithm 1 Algorithm OSEQ

```

At round  $t$ ;
Set  $\mathcal{L}S$  the list of servers;
Set  $\mathcal{L}C = \text{sort}(\mathcal{L}C^t)$  the ordered list of clients at round  $t$ ;
for  $j = 1$  to  $|\mathcal{L}S|$ : do
    Compute  $S_j(\mathcal{L}C)$  the set of clients in  $\mathcal{L}C$  to connect with server  $S_j$  applying Def. 3.2
    Set  $\mathcal{L}C = \mathcal{L}C_{S_j}$ , the actualized list of clients.
end for
RETURN  $S_i(\mathcal{L}C^t) = S_i(\mathcal{L}C)$  for  $1 \leq i \leq |\mathcal{L}S|$ ; THE ALLOCATION AT ROUND  $t$ .
Wait for round  $t + 1$ ;

```

The following two theorems assess the performance of Algorithm OSEQ.

Theorem 3.3 *The throughput achieved by algorithm OSEQ at any round is at least as much as the optimal throughput without resource augmentation.*

Theorem 3.4 *At each server and at each round, Algorithm OSEQ produces at most 1 new connection, 1 disconnection, and 2 changes in bandwidth allocation for already existing connections.*

Due to lack of place, we refer the interesting reader to the research report [BEDRT09], where all proofs can be found.

4 Conclusions

In this paper, we proved that it is not possible to provide a constant approximation ratio for online MTBD (online throughput maximization) when the algorithm is fully online, even allowing any additive resource augmentation (Theorem 3.1). We also present an algorithm that provides the optimal throughput using the smallest possible additive resource augmentation ($\alpha = 1$), and allowing the smallest possible number of disconnections per round per server (at most 1 as stated Theorem 3.4). Finally, the cost in terms of new connections or changes in already existing connections of OSEQ algorithm is at most 3 (one new connection and two changes in already existing connections), while the lower bound is 1 (Corollary of Theorem 3.1).

References

- [And04] D.P. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 365–372, 2004.
- [BEDRT08] O. Beaumont, L. Eyraud-Dubois, H. Rejeb, and C. Thraves. Allocation of Clients to Multiple Servers on Large Scale Heterogeneous Platforms. *INRIA Research Report*, inria-00346394 v1, 2008.
- [BEDRT09] O. Beaumont, L. Eyraud-Dubois, H. Rejeb, and C. Thraves. Online Allocation of Splittable Clients to Multiple Servers on Large Scale Heterogeneous Platforms. *INRIA Research Report*, inria-00384475 v1, 2009.