

Computing the throughput of replicated workflows on heterogeneous platforms

Matthieu Gallet, Anne Benoit, Yves Robert, Bruno Gaujal

▶ **To cite this version:**

Matthieu Gallet, Anne Benoit, Yves Robert, Bruno Gaujal. Computing the throughput of replicated workflows on heterogeneous platforms. 2009. ensl-00365522

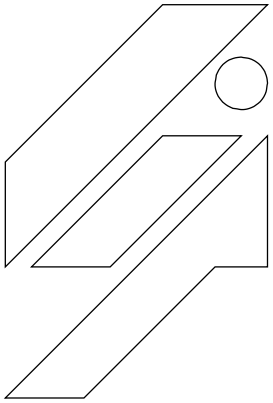
HAL Id: ensl-00365522

https://ens-lyon.hal.science/ensl-00365522

Preprint submitted on 3 Mar 2009

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.



Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Computing the throughput of replicated workflows on heterogeneous platforms

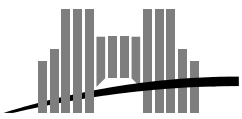
Anne Benoit [1,3,5] Matthieu Gallet [1,3,5]
Bruno Gaujal [2,4] Yves Robert [1,3,5]
[1] ENS Lyon [2] INRIA [3] Université de Lyon
[4] LIG laboratory, UMR 5217, CNRS – Grenoble INP – INRIA – UJF – UPMF, Grenoble, France
[5] LIP laboratory, UMR 5668, ENS Lyon – CNRS – INRIA – UCBL, Lyon, France

Février 2009

Research Report N° 2009-08

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr









Computing the throughput of replicated workflows on heterogeneous platforms

Anne Benoit [1,3,5] Matthieu Gallet [1,3,5] Bruno Gaujal [2,4] Yves Robert [1,3,5]

[1] ENS Lyon [2] INRIA [3] Université de Lyon

[4] LIG laboratory, UMR 5217, CNRS – Grenoble INP – INRIA – UJF – UPMF, Grenoble, France

[5] LIP laboratory, UMR 5668, ENS Lyon – CNRS – INRIA – UCBL, Lyon, France

Février 2009

Abstract

In this paper, we focus on computing the throughput of replicated workflows. Given a streaming application whose dependence graph is a linear chain, and a mapping of this application onto a fully heterogeneous platform, how can we compute the optimal throughput, or equivalently the minimal period? The problem is easy when workflow stages are not replicated, i.e., assigned to a single processor: in that case the period is dictated by the critical hardware resource. But when stages are replicated, i.e., assigned to several processors, the problem gets surprisingly complicated, and we provide examples where the optimal period is larger than the largest cycle-time of any resource. We then show how to model the problem as a timed Petri net to compute the optimal period in the general case, and we provide a polynomial algorithm for the one-port communication model with overlap. Finally, we report comprehensive simulation results on the gap between the optimal period and the largest resource cycle-time.

Keywords: Scheduling, workflows, heterogeneous platforms, period, critical resource, timed Petri nets.

Résumé

Dans ce papier, nous étudions le débit de graphes de tâches répliqués. Étant donnée une application de streaming dont le graphe de dépendance est une chaîne, et un placement de cette application sur une plate-forme hétérogène, comment pouvons-nous calculer le débit optimal, ou, de façon équivalente, la période minimale ? Ce problème est simple quand les différentes tâches ne sont traitées que par un seul processeur : dans ce cas, la période est donnée par le débit de la ou des ressources critiques. Cependant, quand les tâches sont répliquées, c'est-à-dire placées sur plusieurs processeurs, le problème devient étonnamment compliqué, et nous présentons des exemples d'instances sans aucune ressource critique, c'est-à-dire que chacune des ressources connaît des moments d'inactivité lors de l'exécution du système. Nous montrons comment calculer la période du système en utilisant les réseaux de Petri temporisés, et nous donnons un algorithme polynomial pour la calculer pour le modèle de communication avec overlap. Nous exposons également les résultats de nombreuses simulations montrant l'écart entre la période réelle entre le système et le maximum des temps d'occupation des ressources.

Mots-clés: Ordonnancement, graphes de tâches, plate-formes hétérogènes, période, ressource critiques, réseaux de Petri temporisés.

1 Introduction

In this paper we deal with streaming applications, or *workflows*, whose dependence graph is a linear chain composed of several stages. Such applications operate on a collection of data sets that are executed in a pipeline fashion [11, 10, 14]. They are a popular programming paradigm for streaming applications like video and audio encoding and decoding, DSP applications, etc [7, 13, 16]. Each data set is input to the linear chain and traverses it until its processing is complete. While the first data sets are still being processed by the last stages of the pipeline, the following ones have started their execution. In steady state, a new data set enters the system every $\mathcal{P}$ time-units, and several data sets are processed concurrently within the system. A key criterion to optimize is the *period*, or equivalently its inverse, the *throughput*. The period $\mathcal{P}$ is defined as the time interval between the completion of two consecutive data sets. With this definition, the system can process data sets at a rate $1/\mathcal{P}$ (the throughput).

The workflow is executed on a fully heterogeneous platform, whose processors have different speeds, and whose interconnection links have different bandwidths. When mapping application stages onto processors, we enforce the rule that any given processor will execute at most one stage. However, the converse is not true. If the computations of a given stage are independent from one data set to another, then two consecutive computations (different data sets) for the same stage can be mapped onto distinct processors. Such a stage is said to be *replicated*, using the terminology of Subhlok and Vondran [11, 12] and of the DataCutter team [4, 10, 15]. This corresponds to the *dealable* stages of Cole [6].

Given an application and a target heterogeneous platform, the problem to determine the optimal mapping (maximizing the throughput) has been shown NP-hard in [3]. The main objective of this paper is to assess the complexity of computing the throughput *when the mapping is given*. The problem is easy when workflow stages are not replicated, i.e., assigned to a single processor: in that case the period is dictated by the critical hardware resource. But when stages are replicated, i.e., assigned to several processors, the problem gets surprisingly complicated, and we provide examples where the optimal period is larger than the largest cycle-time of any resource. In other words, during the execution of the system, all the resources will be idle at some points. We then show how to use timed Petri nets to compute the optimal period in the general case, and we provide a polynomial algorithm for the one-port model with overlap. Finally, we report comprehensive simulation results on the gap between the optimal period and the largest resource cycle-time.

2 Framework

We deal with streaming applications, or *workflows*, whose dependence graph is a linear chain composed of n stages, called S_k ($0 \leq k \leq n-1$). Each stage S_k has a size w_k, expressed in FLOP, and needs an input file F_{k-1} of size δ_{k-1}, expressed in BYTES. Finally, S_k produces an output file F_k of size δ_k, which is the input file of stage S_{k+1}. All these sizes are independent of the data set. Note that S_0 produces the initial data and does not receive any input file, while S_{n-1} gathers the final data and does not send any file. Figure 1 shows a simple example of a 4-stage pipeline.

The workflow is executed on a fully heterogeneous platform with p processors. The speed of processor P_u is denoted as Π_u. We assume bidirectional links $\text{link}_{u,v} : P_u \rightarrow P_v$ between any processor pair P_u and P_v, with bandwidth $b_{u,v}$. These links are not necessarily physical, they can be logical. For instance, we can have a physical star-shaped platform, where all processors are linked to each other through a central switch. The time needed to transfer a file F_i from P_u to P_v is $\frac{\delta_i}{b_{u,v}}$, while the time needed to process S_k on P_u is $\frac{w_k}{\Pi_u}$. Two realistic common models are used for communications:

• **Overlap One-Port–** This first model permits overlap of communications by computations: any processor can simultaneously receive data set $i+1$, compute the result of data set i and send the resulting data set $i-1$ to the next processor. Requiring multi-threaded programs and full-duplex network interfaces, this model allows for a better use of computational resources.

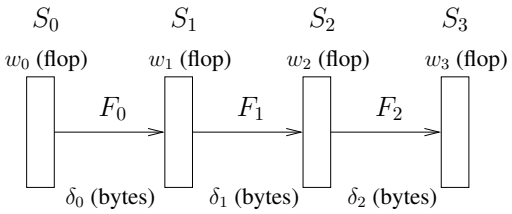


Figure 1. Example of a 4-stage pipeline.

• **STRICT ONE-PORT–** In this model, there is no overlap of communications by computations: a processor can either receive a given set of data, compute its result or send this result. This is the typical execution of a single-threaded program, with one-port serialized communications. Although leading to a less efficient use of physical resources, this model allows for simpler programs and hardware.

When mapping application stages onto processors, we enforce the rule that any given processor will execute at most one stage. However, the converse is not true. If the computations of a given stage are independent from one data set to another, then two consecutive computations (different data sets) for the same stage can be mapped onto distinct processors. Such a stage is said to be *replicated*, using the terminology of Subhlok and Vondran [11, 12] and of the DataCutter team [4, 10, 15]. This corresponds to the *dealable* stages of Cole [6]. Note that the computations of a replicated stage can be fully sequential for a given data set, what matters is that they do not depend from previous results for other data sets, hence the possibility to process different data sets in different locations. The following schema illustrates the replication of a stage S_k onto three processors:

$$\ldots\ S_{k-1} \begin{array}{ll} \nearrow & S_k \text{ on } P_1\text{: data sets } \mathbf{1, 4, 7, \ldots} \\ \rightarrow & S_k \text{ on } P_2\text{: data sets } \mathbf{2, 5, 8, \ldots} \\ \searrow & S_k \text{ on } P_3\text{: data sets } \mathbf{3, 5, 9, \ldots} \end{array} S_{k+1}\ \ldots$$

As outlined in the schema, the processors allocated to a replicated stage execute successive data sets in a round-robin fashion. This may lead to a load imbalance: more data sets could be allocated to faster processors. But this would imply out-of-order execution and would require a complicated data management if, say, a replicated stage is followed by a non-replicated one in the application pipeline. As a result, round-robin execution is enforced in all the papers referenced above, and we enforce this rule too.

The objective is to maximize the throughput ρ of the system, defined as the average number of data sets which can be processed within one time unit. Equivalently, we aim at minimizing the period $\mathcal{P}$, which is the inverse of the throughput and corresponds to the time-interval that separates two consecutive data sets entering the system. We can derive a lower bound for the period as follows. Let $C_{\text{exec}}(k)$ be the cycle-time of processor P_k. If we enforce the OVERLAP ONE-PORT model, then $C_{\text{exec}}(k)$ is equal to the maximum of its reception time $C_{\text{in}}(k)$, its computation time $C_{\text{comp}}(k)$, and its transmission time $C_{\text{out}}(k)$ [1]:

$$C_{\text{exec}}(k) = \max\left\{C_{\text{in}}(k), C_{\text{comp}}(k), C_{\text{out}}(k)\right\}.$$

If we enforce the STRICT ONE-PORT model, then $C_{\text{exec}}(k)$ is equal to the sum of the three operations:

$$C_{\text{exec}}(k) = C_{\text{in}}(k) + C_{\text{comp}}(k) + C_{\text{out}}(k).$$

In both models, the maximum cycle-time, $\mathcal{M}_{\text{ct}} = \max_{1\leq k\leq p} C_{\text{exec}}(k)$, is a lower bound for the period.

Given an application and a target heterogeneous platform, determining a mapping which maximizes the throughput has been shown to be a NP-hard problem in [3], even in the simple case where no stage can be replicated

[1] Note that $C_{\text{in}}(0)$ =$C_{\text{out}}(n-1)$ =0

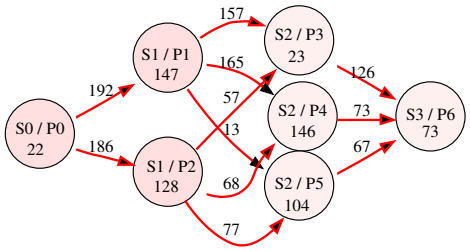


Figure 2. Example A: Mapping with replication: S_1 on 2 processors, S_2 on 3 processors.

(thereby enforcing a one-to-one mapping of stages to processors). The proof of [3] was given for the STRICT ONE-PORT model but can be easily extended to the OVERLAP ONE-PORT model. In this paper, we deal with the following problem, which in appearance looks simpler: given the mapping of stages to processors, how can we compute the period $\mathcal{P}$? If no stage is replicated, then the period is simply determined by the critical resource (maximum cycle-time): $\mathcal{P} = \mathcal{M}_{\text{ct}}$. Again, this problem is addressed in [3] for the STRICT ONE-PORT model but the same result can be easily shown for the OVERLAP ONE-PORT model. However, when stages are replicated, the previous result is no longer true, and we need to use more sophisticated techniques such as timed Petri nets.

3 Timed Petri net models

3.1 Mappings with replication

In this section, we aim at modeling mappings with timed Petri nets (TPNs) as defined in [1], in order to be able to compute the period of a given mapping. In the following only TPNs with the *event graph property* (each place has exactly one input and one output transition) will be considered (see [2]). We consider mappings where some stages may be replicated, as defined in Section 2: a stage can be processed by one or more processors. As already stated, two rules are enforced to simplify the model: a processor can process at most one stage, and if several processors are involved in the computation of one stage, they are served in a round-robin fashion. In all our Petri net models, the use of a physical resource during a time t (i.e., the computation of a stage or the transmission of a file from a processor to another one) is represented by a transition with a firing time t, and dependences are represented using places. Now, let us focus on the path followed in the pipeline by a single input data set, for a mapping with several stages replicated on different processors. Consider Example A described in Figure 2: the first data set enters the system and proceeds through processors P_0, P_1, P_3 and P_6. The second data set is first processed by processor P_0, then by processor P_2 (even if P_1 is available), by processor P_4 and finally by processor P_6. Paths followed by the first eight input data sets are summarized up in Table 1: as we can see, there are 6 different paths followed by the data sets, and then data set i takes the same path as data set $i - 6$. We have the following easy result:

Proposition 1. *Consider a pipeline of n stages $S_0, \ldots, S_{n-1}$, such that stage S_i is mapped onto m_i distinct processors. Then the number of paths followed by the input data in the whole system is equal to $m = \text{lcm}\,(m_0, \ldots, m_{n-1})$.*

Proof. Let m be the number of paths $\mathcal{P}_j$ followed by the input data. Assume that stage S_i is processed by processors $P_{i,0}, \ldots, P_{i,m_i-1}$. By definition, all paths are distinct. Moreover, the round-robin order is respected: path $\mathcal{P}_j$ is made of processors $\left(P_{0,j \mod m_0}, \ldots, P_{i,j \mod m_i}, \ldots, P_{n-1,j \mod m_{n-1}}\right)$. The first path $\mathcal{P}_0$ is made of $(P_{0,0}, P_{1,0}, \ldots, P_{n-1,0})$. By definition, m is the smallest positive integer, such that the $(m+1)$-th used path is identical to the first one:

$$\forall i \in \{0, \ldots, n-1\}, m \mod m_i = 0.$$

Input data	Path in the system
0	$P_0 \to P_1 \to P_3 \to P_6$
1	$P_0 \to P_2 \to P_4 \to P_6$
2	$P_0 \to P_1 \to P_5 \to P_6$
3	$P_0 \to P_2 \to P_3 \to P_6$
4	$P_0 \to P_1 \to P_4 \to P_6$
5	$P_0 \to P_2 \to P_5 \to P_6$
6	$P_0 \to P_1 \to P_3 \to P_6$
7	$P_0 \to P_2 \to P_4 \to P_6$

Table 1. Example A: Paths followed by the first input data.

Indeed, m is the smallest positive integer, which is divisible by each m_i, i.e., $m = \text{lcm}\,(m_0, \ldots, m_{n-1})$. □

The TPN model given here is the same flavor as what has been done to model jobshops with static schedules using TPNs [8]. Here, however, replication imposes that each path followed by the input data must be fully developed in the TPN: if P_0 appears in several distinct paths, as in Figure 2, there are several transitions corresponding to P_0. Furthermore, we have to add dependences between all the transitions corresponding to the same physical resource to avoid the simultaneous use of the same resource by different input data. These dependences differ according to the model used for communications and computations.

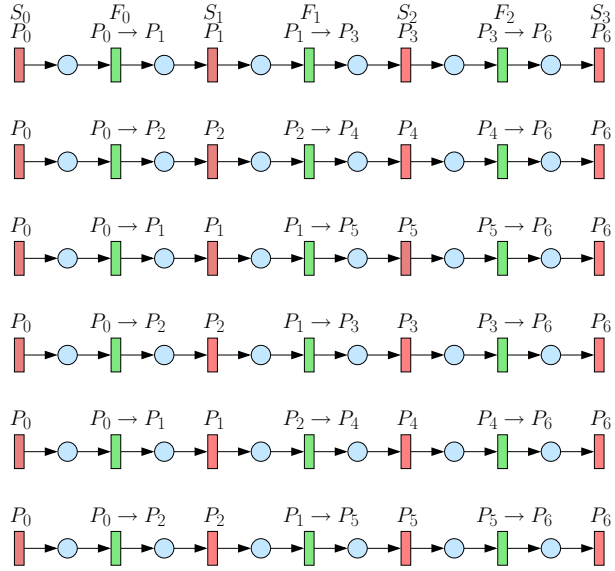
3.2 Overlap One-Port model

First, let us focus on the Overlap One-Port model: any processor can receive a file and send another one while computing. All paths followed by the input data in the whole system have to appear in the TPN. We use the notations of Proposition 1.

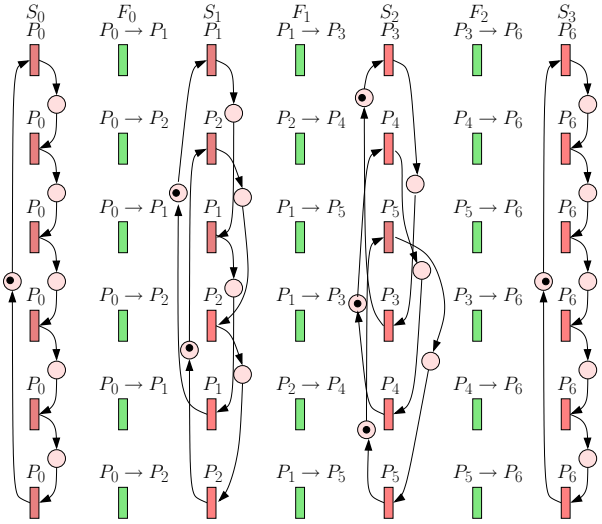
Let m denote the number of paths of our mapping. Then the i-th input data follows the $(i \mod m)$-th path, and we have a rectangular TPN, with m rows of $2n-1$ transitions, due to the n transitions representing the use of processors and the $n-1$ transitions representing the use of communication links. The i-th transition of the j-th row is named T_i^j. The time required to fire a transition T_{2i}^j (corresponding to the processing of stage S_i on processor P_u) is set to $\frac{w_i}{\Pi_u}$, and the one required by a transition T_{2i+1}^j (corresponding the transmission of a file F_i from P_u to P_v) is set to $\frac{\delta_i}{b_{u,v}}$.

Then we add places between these transitions to model the following set of constraints:

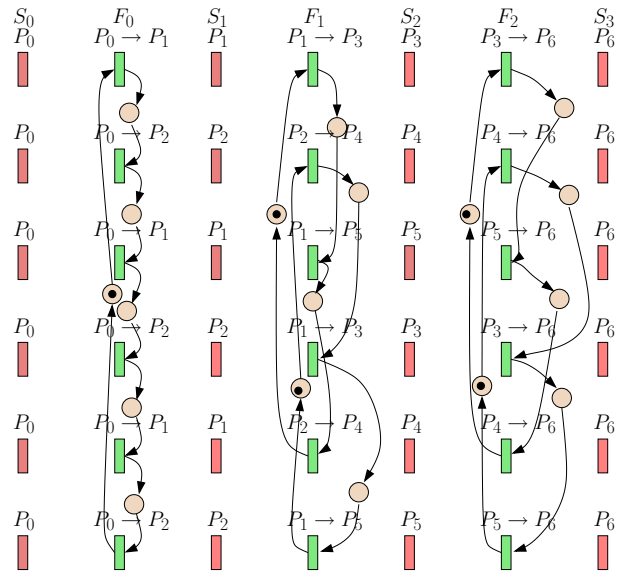
1. The file F_i cannot be sent before the computation of S_i: a place is added from T_{2i}^j to T_{2i+1}^j on each row. Similarly, the stage S_{i+1} cannot be processed before the end of the communication of F_i: a place is added from T_{2i+1}^j to $T_{2(i+1)}^j$ on each row j. All these places are shown in Figure 3(a).
2. When a processor appears in several rows, the round-robin distribution imposes dependences between these rows. Assume that processor P_i appears on rows $j_1, j_2, \ldots, j_k$. Then we add a place from $T_{2i}^{j_l}$ to $T_{2i}^{j_{l+1}}$ with $1 \leq l \leq k-1$, and a place from $T_{2i}^{j_k}$ to $T_{2i}^{j_1}$. All these places are shown in Figure 3(b).
3. The one-port model and the round-robin distribution of communications also impose dependences between rows. Assume that processor P_i appears on rows $j_1, j_2, \ldots, j_k$. Then we add a place from $T_{2i+1}^{j_l}$ to $T_{2i+1}^{j_{l+1}}$ with $1 \leq l \leq k-1$, and a place from $T_{2i+1}^{j_k}$ to $T_{2i+1}^{j_1}$ to ensure that P_i does not send two files simultaneously, if P_i does not compute the last stage. All these places are shown in Figure 3(c).
4. In the same way, we add a place from $T_{2i-1}^{j_l}$ to $T_{2i-1}^{j_{l+1}}$ with $1 \leq l \leq k-1$, and a place from $T_{2i-1}^{j_k}$ to $T_{2i-1}^{j_1}$ to ensure that P_i does not receive two files simultaneously, if P_i does not compute the first stage. All these places are shown in Figure 3(d).



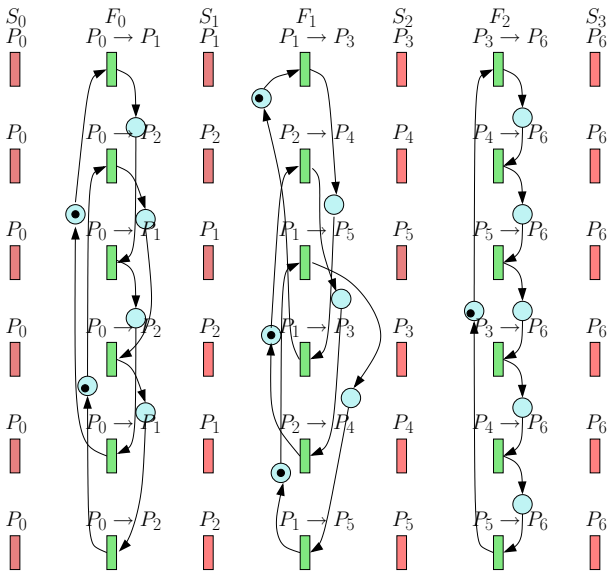
(a) Dependences between communications and computations.



(b) Dependences due to the round-robin distribution of computations.



(c) Dependences due to the round-robin distribution of outgoing communications.



(d) Dependences due to the round-robin distribution of incoming communications.

Figure 3. OVERLAP ONE-PORT **model: places imposed by the different constraints described in Subsection 3.2. Circuits model the round-robin distribution, and the single token in each circuit models the fact that any resource can process at most one job at a time.**

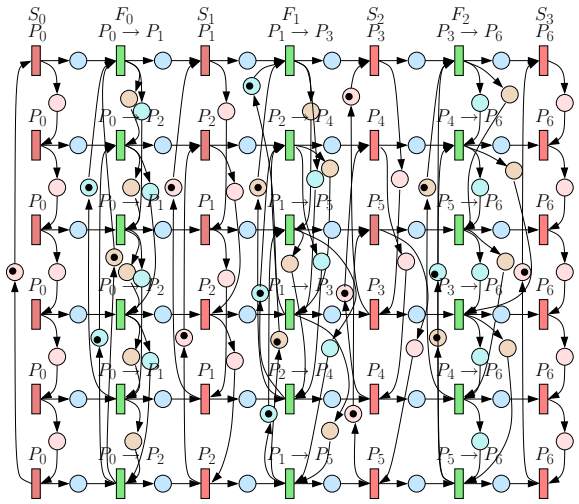


Figure 4. Complete TPN of Example A for the OVERLAP ONE-PORT model.

Finally, any resource before its first use is ready to compute or communicate, only waiting for the input file. Indeed, a token is put in every place going from a transition $T_i^{j_k}$ to a transition $T_i^{j_1}$, as defined in the previous lines. The complete TPN of Example A for the OVERLAP ONE-PORT model is given in Figure 4.

3.3 STRICT ONE-PORT model

In the STRICT ONE-PORT model, any processor can either send a file, receive another one, or perform a computation while these operations were happening concurrently in the OVERLAP ONE-PORT model. Hence, we require a processor to successively receive the data corresponding to an input file F_i, compute the stage S_{i+1} and send the file F_{i+1} before receiving the next data set of F_i. Paths followed by the input data are obviously the same as in Subsection 3.2, and the structure of the TPN remains the same (m rows of $2n-1$ transitions).

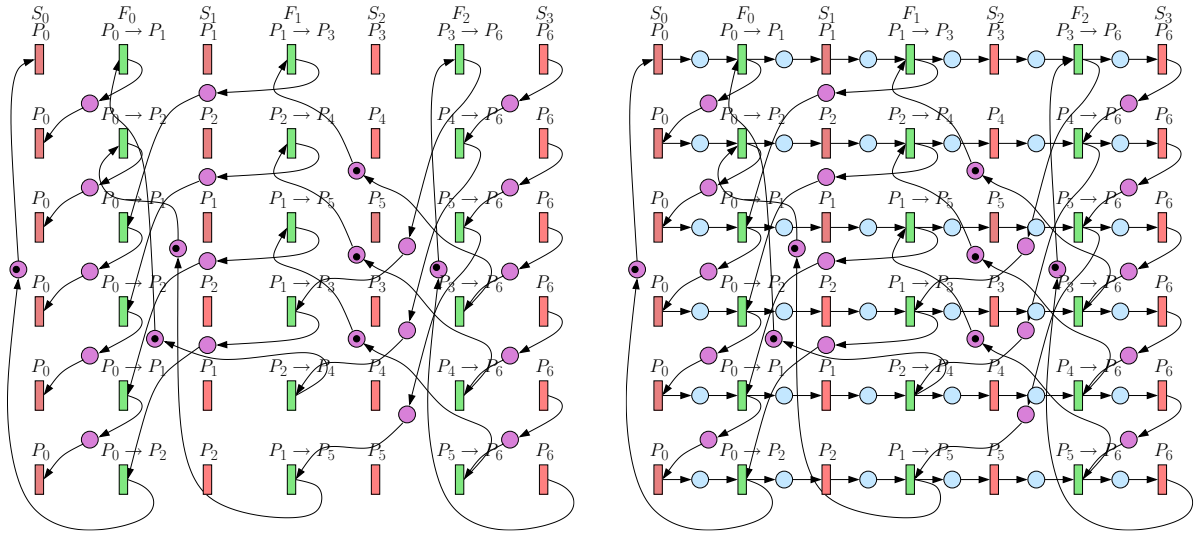
The first set of constraints is also identical to that of the OVERLAP ONE-PORT model, since we still have dependences between communications and computations, as in Figure 3(a). However, the other dependences are replaced by those imposed by the round-robin order of the STRICT ONE-PORT model. Indeed, when a processor appears in several rows, the round-robin order imposes dependences between these rows. Assume that processor P_i appears on rows $j_1, j_2, \ldots, j_k$. Then we add a place from $T_{2i+1}^{j_l}$ to $T_{2i-1}^{j_{l+1}}$ with $1 \leq l \leq k-1$, and a place from $T_{2i+1}^{j_k}$ to $T_{2i-1}^{j_1}$. These places ensure the respect of the model: the next reception cannot start before the completion of the current sequence reception-computation-sending. All these places are shown in Figure 5(a).

Any physical resource can immediately start its first communication, since it is initially only waiting for the input file. Thus a token is put in every place from a transition $T_i^{j_k}$ to a transition $T_i^{j_1}$, as defined in the previous lines. The complete TPN of Example A for the STRICT ONE-PORT model is given in Figure 5(b).

The automatic construction of the TPN in both cases has been implemented. The time needed to construct the Petri net is linear in its size: $\mathcal{O}(mn)$.

4 Computing mapping throughputs

TPNs with the event graph property make the computation of the throughput of a complex system possible through the computation of *critical cycles*, using $(\max, +)$ algebra [2]. For any cycle $\mathcal{C}$ in the TPN, let $\mathcal{L}(\mathcal{C})$ be its length (number of transitions) and $t(\mathcal{C})$ be the total number of tokens in places traversed by $\mathcal{C}$. Then a critical



(a) Dependences imposed by the round-robin distribution.

(b) Complete TPN.

Figure 5. STRICT ONE-PORT **model: places imposed by the different constraints described in Subsection 3.3.**

cycle achieves the largest ratio $\max_{\mathcal{C}\text{cycle}} \frac{\mathcal{L}(\mathcal{C})}{t(\mathcal{C})}$, and this ratio is the period $\mathcal{P}$ of the system: indeed, after a transitive period, every transition of the TPN is fired exactly once during a period of length $\mathcal{P}$ [2].

Critical cycles can be computed with softwares like ERS [9] or GreatSPN [5] with a complexity $\mathcal{O}(m^3n^3)$. By definition of the TPN, the firing of any transition of the last column corresponds to the completion of the last stage, i.e., to the completion of an instance of the workflow. Moreover, we know that all the m transitions (if m is still the number of rows of the TPN) of this last column are fired in a round-robin order. In our case, m data sets are completed during any period $\mathcal{P}$: the obtained throughput ρ is $\frac{m}{\mathcal{P}}$.

4.1 OVERLAP ONE-PORT model

The TPN associated to the OVERLAP ONE-PORT model has a regular structure, which facilitates the determination of critical cycles. In the complete TPN, places are linked to transitions either in the same row and oriented forward, or in the same column. Hence, any cycle only contains transitions belonging the same "column": we can split the complete TPN into $2n-1$ smaller TPNs, each sub-TPN representing either a communication or a computation. However, the size of each sub-TPN (the restriction of the TPN to a single column) is not necessarily polynomial in the size of the instance, due to the possibly large number of rows, equal to $m = \text{lcm}\,(m_0, \ldots, m_{n-1})$.

It turns out that a polynomial algorithm exists to find the weight $\mathcal{L}(\mathcal{C})/t(\mathcal{C})$ of a critical cycle: only a fraction of each sub-TPN is required to compute this weight, without computing the cycle itself. This is the main technical contribution of this paper, given in the following theorem.

Theorem 1. *Consider a pipeline of n stages $S_0, \ldots, S_{n-1}$, such that stage S_i is mapped onto m_i distinct processors. Then the average throughput of this system can be computed in time $\mathcal{O}\left(\sum_{i=0}^{n-2}\left((m_i m_{i+1})^3\right)\right)$.*

The complete proof of this theorem is given in the appendix A.

In Example A, a critical resource is the output port of P_0, whose cycle-time is equal to the period, 189. However it is possible to exhibit cases without critical resource: see for instance Example B presented in Figure 6. Its critical

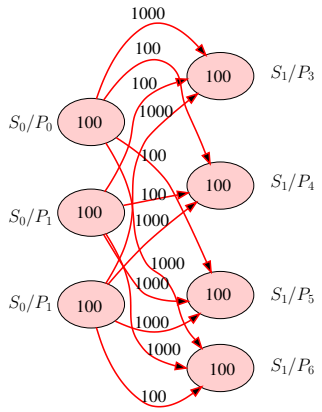


Figure 6. Example B: Stage 0 is replicated on 3 processors, and Stage 1 on 4 processors.

resource cycle-time is $\mathcal{M}_{\text{ct}} = 258.3$ and corresponds to the outgoing communications of P_2. It is strictly smaller than the actual period of the complete system, $\mathcal{P} = 291.7$.

4.2 STRICT ONE-PORT model

Cycles in the TPN associated to the STRICT ONE-PORT model are more complex and less regular, since corresponding TPNs have backward edges. An example of such a cycle is shown in Figure 8. The intuition behind these backward edges is that a processor P_u cannot compute an instance of S_i before having completely sent the result F_i of the previous instance of S_i to the next processor P_v. Thus, P_u can be slowed by P_v. As for the OVERLAP ONE-PORT model, there exist mappings for which all resources have idle times during a complete period. With the STRICT ONE-PORT model, this is the case for Example A, whose Gantt diagram is shown in Figure 7g the critical resource is P_2, which has a cycle-time $\mathcal{M}_{\text{ct}} = 215.8$, strictly smaller than the period $\mathcal{P} = 230.7$.

5 Experiments

In Section 4, we have shown examples of mappings without any critical resource, i.e., whose period is larger than any resource cycle-time, for both communication models. We have conducted extended experiments to assess whether such situations are very common or not. Several sets of applications and platforms were considered, with between 2 and 20 stages and between 7 and 30 processors. All relevant parameters (processor speeds, link bandwidths, number of processors computing the same stage) were randomly chosen uniformly within the ranges indicated in Table 2. Finally, each experiment was run for both models. We compared the inverse of the critical resource cycle-time and the actual throughput of the whole platform. A grand total of $5,152$ different experiments were run. Table 2 shows that the cases without critical resources are very rare. In fact no such case was actually found with the OVERLAP ONE-PORT model!

The computation times closely depends on the duplication factor of each stage: the computation of an example with 10 stages and 20 processors ranges from 2 to $150,000$ seconds on powerful machines such as a quadri-core server.

6 Conclusion

In this paper, we have studied the throughput of streaming applications mapped on heterogeneous platforms. The major originality of our work, and also its major difficulty, is that we consider stage replication. Although this

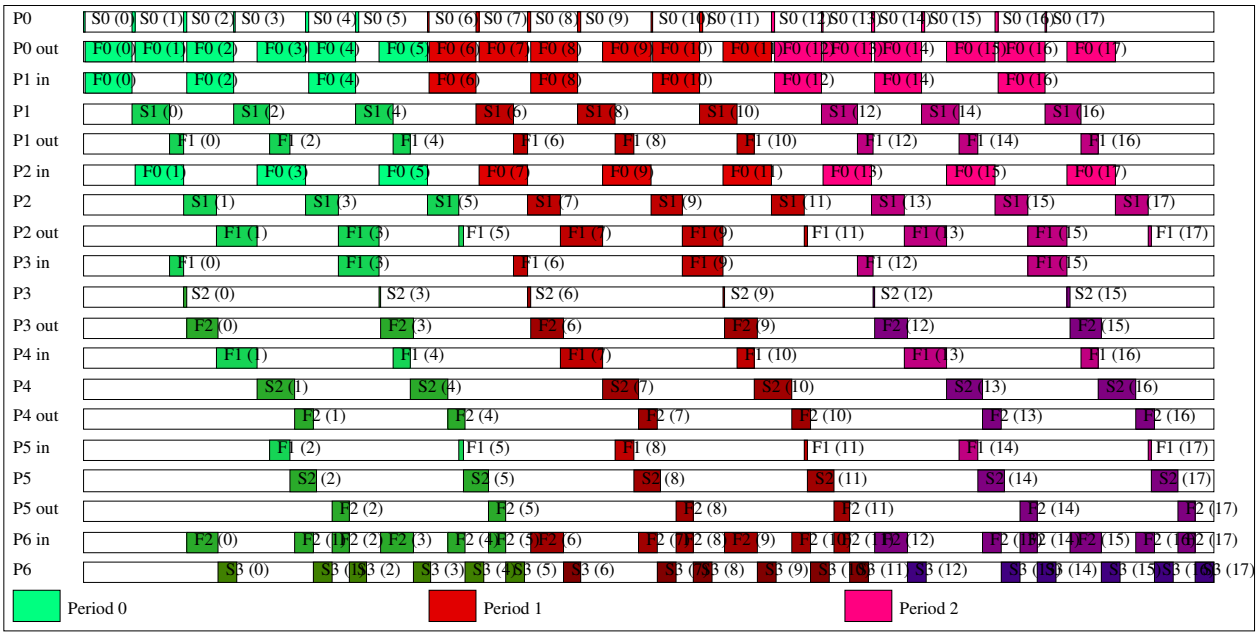


Figure 7. Gantt diagram of a schedule without critical resource.

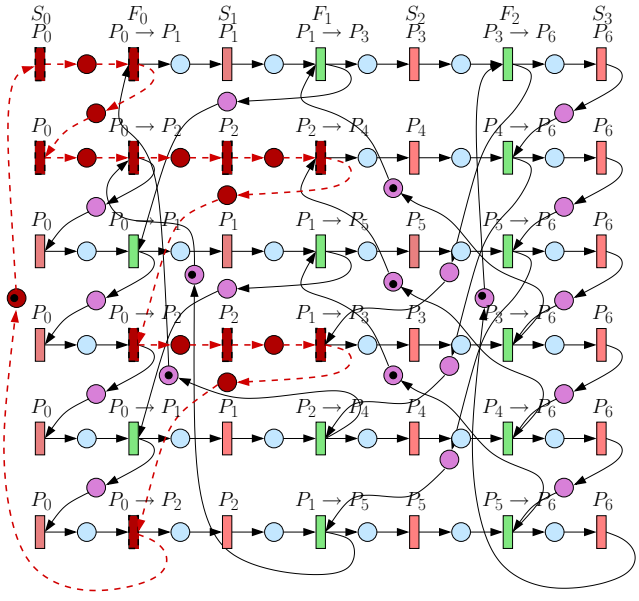


Figure 8. Complex critical cycles on Example A.

Size (stages, processors)	Computation times	Communication times	#exp without critical resource / total
With overlap:			
(10, 20) and (10, 30)	between 5 and 15	between 5 and 15	0 / 220
(10, 20) and (10, 30)	between 10 and 1000	between 10 and 1000	0 / 220
(20, 30)	between 5 and 15	between 5 and 15	0 / 68
(20, 30)	between 10 and 1000	between 10 and 1000	0 / 68
(2, 7) and (3, 7)	1	between 5 and 10	0 / 1000
(2, 7) and (3, 7)	1	between 10 and 50	0 / 1000
Without overlap:			
(10, 20) and (10, 30)	between 5 and 15	between 5 and 15	14 / 220 (diff less than 9%)
(10, 20) and (10, 30)	between 10 and 1000	between 10 and 1000	0 / 220
(20, 30)	between 5 and 15	between 5 and 15	5 / 68 (diff less than 7%)
(20, 30)	between 10 and 1000	between 10 and 1000	0 / 68
(2, 7) and (3, 7)	1	between 5 and 10	10 / 1000 (diff less than 3%)
(2, 7) and (3, 7)	1	between 10 and 50	0 / 1000

Table 2. Numbers of experiments without critical resource.

technique is classical in the literature, the computation of the throughput of such complex mappings has not been addressed yet (at the best of our knowledge). We have introduced TPNs (timed Petri nets) to determine the critical cycles of the mapping. The complexity of throughput evaluation depends on the communication model. Even the simple round-robin distribution implies complex interactions between involved resources, resulting in schedules without any critical resource: there exist schedules, such that all resources remain partially idle, and this is true for both models. However, experiments show that such cases are very rare under the OVERLAP ONE-PORT model. In addition, we have established the polynomial complexity of the problem for this OVERLAP ONE-PORT model, while it remains open for the STRICT ONE-PORT model.

This paper was focused on static platforms, opening the way to future work on finding good schedules on dynamic platforms, whose speeds and bandwidths are modeled by random variables.

References

[1] F. Baccelli, G. Cohen, and B. Gaujal. Recursive equations and basic properties of timed petri nets. *Journal of Discrete Event Dynamic Systems*, 1(4), 1992.

[2] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. Wiley, 1992.

[3] A. Benoit and Y. Robert. Mapping pipeline skeletons onto heterogeneous platforms. *J. Parallel Distributed Computing*, 68(6):790–808, 2008.

[4] M. D. Beynon, T. Kurc, A. Sussman, and J. Saltz. Optimizing execution of component-based applications using group instances. *Future Generation Computer Systems*, 18(4):435–448, 2002.

[5] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN: Graphical editor and analyzer for timed and stochastic petri nets. *Performance Evaluation*, 24(1-2):47–68, 1995.

[6] M. Cole. Bringing Skeletons out of the Closet: A Pragmatic Manifesto for Skeletal Parallel Programming. *Parallel Computing*, 30(3):389–406, 2004.

[7] DataCutter Project: Middleware for Filtering Large Archival Scientific Datasets in a Grid Environment. http://www.cs.umd.edu/projects/hpsl/ResearchAreas/DataCutter.htm.

[8] H. Hillion and J.-M. Proth. Performance evaluation of job shop systems using timed event graphs. *IEEE Transaction on Automatic Control*, 34(1):3–9, 1989.

[9] A. Jean-Marie. ERS: a tool set for performance evaluation of discrete event systems. http://www-sop.inria.fr/mistral/soft/ers.html.

[10] M. Spencer, R. Ferreira, M. Beynon, T. Kurc, U. Catalyurek, A. Sussman, and J. Saltz. Executing multiple pipelined data analysis operations in the grid. In *2002 ACM/IEEE Supercomputing Conference*. ACM Press, 2002.

[11] J. Subhlok and G. Vondran. Optimal mapping of sequences of data parallel tasks. In *Proc. 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP'95*, pages 134–143. ACM Press, 1995.

[12] J. Subhlok and G. Vondran. Optimal latency-throughput tradeoffs for data parallel pipelines. In *ACM Symposium on Parallel Algorithms and Architectures SPAA'96*, pages 62–71. ACM Press, 1996.

[13] K. Taura and A. A. Chien. A heuristic algorithm for mapping communicating tasks on heterogeneous resources. In *Heterogeneous Computing Workshop*, pages 102–115. IEEE Computer Society Press, 2000.

[14] N. Vydyanathan, U. Catalyurek, T. Kurc, P. Saddayappan, and J. Saltz. Toward optimizing latency under throughput constraints for application workflows on clusters. In *Euro-Par'07: Parallel Processing*, LNCS 4641, pages 173–183. Springer Verlag, 2007.

[15] N. Vydyanathan, U. Catalyurek, T. Kurc, P. Saddayappan, and J. Saltz. A duplication based algorithm for optimizing latency under throughput constraints for streaming workflows. In *ICPP'2008, the Int. Conf. on Parallel Processing*, pages 254–261. IEEE Computer Society Press, 2008.

[16] Q. Wu and Y. Gu. Supporting distributed application workflows in heterogeneous computing environments. In *14th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE Computer Society Press, 2008.

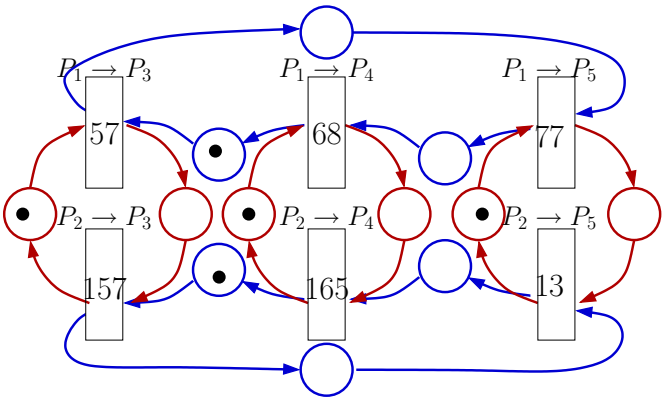


Figure 9. Sub-TPN corresponding to the transmission of F_1 in Example A (OVERLAP ONE-PORT model).

A Proof of Theorem 1

Theorem 1. *Let us consider a set $(S_0, \ldots, S_{i+1})$ of stages, and a one-to-many mapping of them, such that S_i is mapped on m_i processors. Then the average throughput of this system can be computed in time $\mathcal{O}\left(\sum_{i=0}^{n-2}\left((m_i m_{i+1})^3\right)\right)$.*

Proof. We saw that the throughput of the platform is given by the weight of a critical cycle. As said before, a critical cycle can only be found in a column of transitions, and we have two cases:

- transitions correspond to the computation of a stage S_i,
- transitions correspond to the transmission of a file F_i.

The first case is the simplest one: each transition appears in exactly one cycle, and each cycle passes through exactly one physical resource (all the transitions correspond to the same stage S_i on the same processor P_u). Thus, the time passed during a complete period of length $\mathcal{P}$ by a processor P_u is exactly equal to $\left(\frac{w_i}{\Pi_u}\right)\left(\frac{m}{m_i}\right)$ (we recall that m_i is the number of processors devoted to the computation of S_i). This is also the weight of this cycle. Thus, if a critical cycle of the TPN appears in such a column, then its average weight is easy to compute. The running time to compute critical cycles for those columns is $\mathcal{O}\left(\sum_{i=0}^{n-1} m_i\right)$.

The second case is more complex: each transition appears in exactly two cycles. The first cycle is created by the round-robin distribution on the output port of the emitter, and the second one comes from the round-robin distribution on the input port of the receiver. By construction, this sub-TPN is made of several elemental cycles, each elemental cycle corresponding to the successive receptions of F_i by a processor participating to the computation of S_{i+1}, or to the successive transmissions of F_i by a processor working on S_i. If any critical cycle passes through both types of elemental cycles, then all resources can have idle times in the final schedule, as shown in Figure 12, representing the Gantt chart of the first instances of Example B. This example, presented in Figure 6, is made of a single communication, whose sub-TPN is displayed in Figure 10; a critical cycle is drawn with dotted arrows.

This communication of F_i involves m_i senders and m_{i+1} receivers. The transmission of F_1 in example C, displayed in Figure 11, is used for a better understanding of our proof. Let m be the least common multiple of $(m_0, \ldots, m_{n-1})$, and p the greatest common divisor of m_i and m_{i+1}. Let u be equal to m_i/p and v be equal to m_{i+1}/p. Then the complete sub-TPN $\mathcal{G}$ is made of p connected components, each of them based on $c = \frac{m}{\text{lcm}(m_i, m_{i+1})}$ patterns P of size $u \times v$. One of these components is shown in Figure 13. Let $\mathcal{C}^a$ be a critical

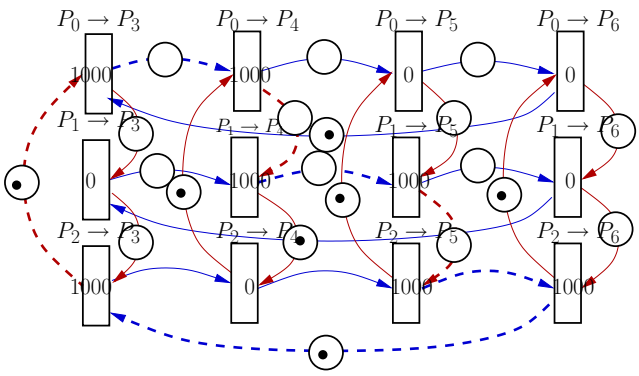


Figure 10. Sub-TPN corresponding to the transmission of F_0 in Example B (OVERLAP ONE-PORT model).

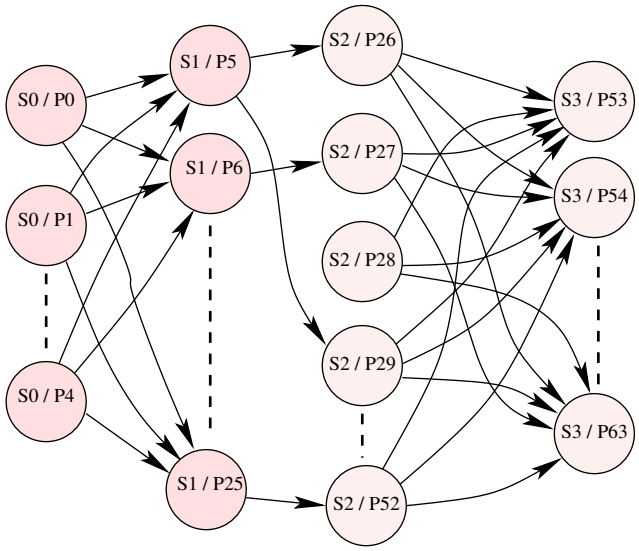


Figure 11. Example C: Stages are respectively replicated on 5, 21, 27 and 11 processors.

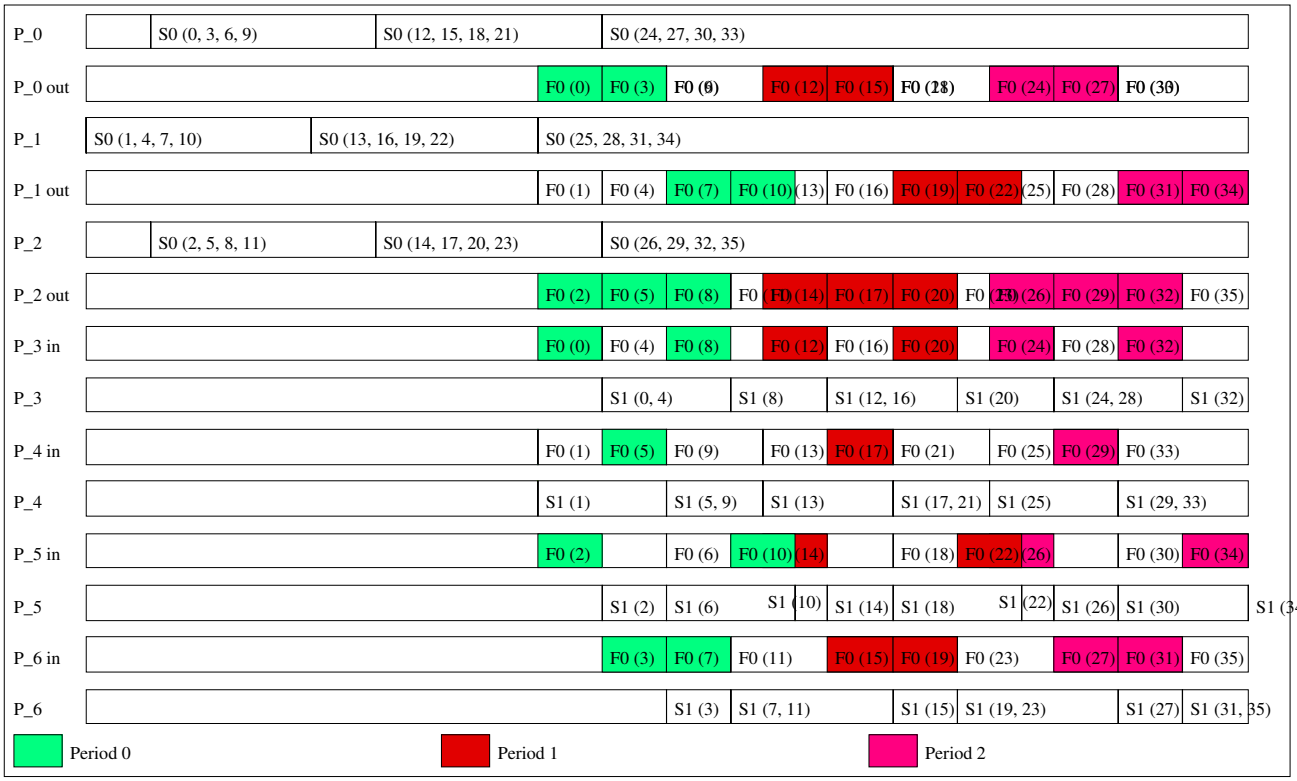


Figure 12. Gantt diagram of the first three periods of Example B.

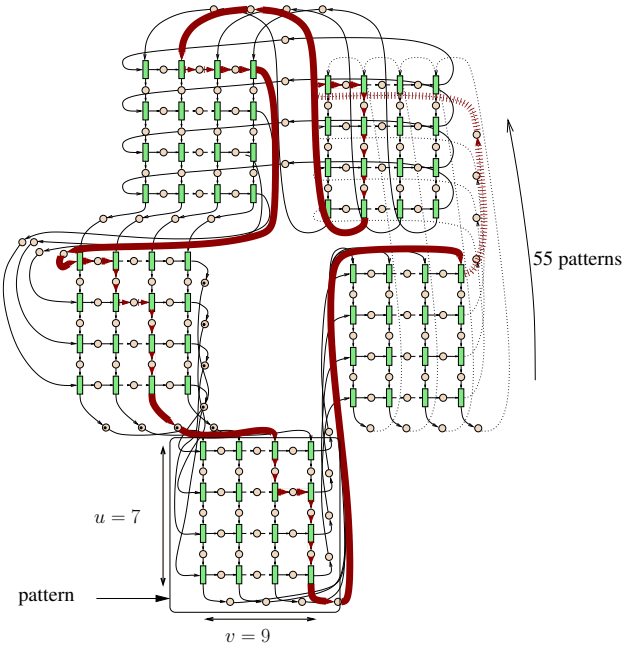


Figure 13. A complete connected component $\mathcal{G}$ (corresponding to Example C).

cycle. By definition of a cycle, $\mathcal{C}^a$ is contained in one of the p connected components. Thus, without any loss of generality, we now assume that the complete sub-TPN is reduced to a single connected component.

In the case of Example C, $m_0 = 5$, $m_1 = 21$, $m_2 = 27$ and $m_3 = 11$. Thus, we have $m = 10395$, $p = 3$, $c = 55$, $u = 7$ and $v = 9$. There are 3 connected components, reflecting the fact that any sender communicates with only 9 distinct receivers. As example, P_5 only communicates with P_{26}, P_{29}, P_{32}, ..., P_{50}, and P_6 only communicates with P_{27}, P_{30}, P_{33}, ..., P_{51}. Let us call x_{ij}^k the transition on column i ($0 \leq i < u$), row j ($0 \leq j < v$) and pattern k ($0 \leq k < c$).

The structure of any connected component is very regular:

- if $0 \leq i < u$, then there is a place from x_{ij}^k to $x_{(i+1)j}^k$, corresponding to the round-robin on the receiver,
- if $0 \leq j < v$, then there is a place from x_{ij}^k to $x_{i(j+1)}^k$, corresponding to the round-robin on the sender,
- if $0 \leq k < c$, then there is a place from $x_{(u-1)j}^k$ to x_{0j}^{k+1} and from $x_{i(v-1)}^k$ to x_{i0}^{k+1},
- there is a place from $x_{(u-1)j}^{c-1}$ to x_{0j}^0 and from $x_{i(v-1)}^{c-1}$ to x_{i0}^0.

Thus, any critical cycle passes through all patterns of $\mathcal{G}$. Now, let us call $\mathcal{G}'$ the smaller graph made of a single pattern of $\mathcal{G}$. $\mathcal{G}'$ has uv transitions, denoted by x_{ij} (with $0 \leq i < u$ and $0 \leq j < vb$) and $2uv$ places, such that:

- if $0 \leq i < u$, then there is a place from x_{ij} to $x_{(i+1)j}$,
- if $0 \leq j < v$, then there is a place from x_{ij} to $x_{i(j+1)}$,
- there is a place from $x_{(u-1)j}$ to x_{0j} and from $x_{i(v-1)}$ to x_{i0},

In figure 14, we can see this graph $\mathcal{G}'$, corresponding the full graph shown in Figure 13.

We need some other definitions:

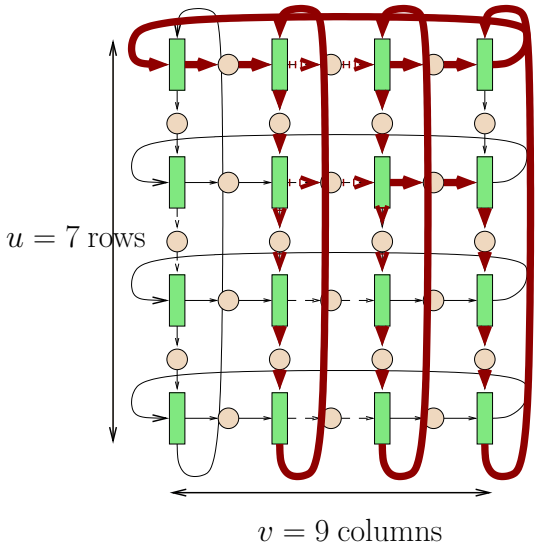


Figure 14. A single pattern $\mathcal{G}'$.

- If we consider a cycle $\mathcal{C}^a$ in $\mathcal{G}$, then by construction of P, the only way to pass through P is to enter by either the first column or the first line. Let k^a be the number of such entrances. Similarly, if we consider a cycle $\mathcal{C}^a$ in $\mathcal{G}'$, let k^a be the number of places $x_{(u-1)j} \rightarrow x_{0j}$ and $x_{i(v-1)} \rightarrow x_{i0}$.
- Let $\mathcal{L}^a$ be the sum of all transitions of a cycle $\mathcal{C}^a$.
- If $\mathcal{C}^a = \left(x_{i_0,j_0}^{k_0}, x_{i_1,j_1}^{k_1}, \ldots, x_{i_a,j_a}^{k_a}\right)$ is a cycle in $\mathcal{G}$, let $\mathcal{C}^b = (x_{i_0,j_0}, x_{i_1,j_1}, \ldots, x_{i_a,j_a})$ be its *projection* in $\mathcal{G}'$; by construction, the same place can appears many times in $\mathcal{C}^b$.
- A cycle $\mathcal{C}^a$ in $\mathcal{G}'$ can be dived into $\mathcal{G}$ to obtain a cycle $\mathcal{C}^b$ in $\mathcal{G}$. This transformation is shown in Figure 15.
- On the contrary, a cycle $\mathcal{C}^a$ in $\mathcal{G}$ can be projected on $\mathcal{G}'$ to obtain a cycle $\mathcal{C}^b$ in $\mathcal{G}'$. This transformation is shown in Figure 16.

Obviously, if $\mathcal{C}^a$ is a cycle in $\mathcal{G}$, then k^a is a multiple of p, the total number of patterns in $\mathcal{G}$. Now, by construction of the sub-TPN, there is a single token in each place between the last and the first pattern. Thus, the number of tokens in $\mathcal{C}^a$ is equal to k^a/p.

1. Let $\mathcal{C}^1$ be any critical cycle of $\mathcal{G}$. Its weight (or length) is $\mathcal{L}^1$, and the number of tokens is equal to k^1/p. Since $\mathcal{C}^1$ is critical, $\mathcal{L}^1 \times \frac{p}{k^1}$ is maximal.
2. Let $\mathcal{C}^2$ be the projection of $\mathcal{C}^1$ in $\mathcal{G}'$. By construction of $\mathcal{C}^2$, $k^2 = k^1$ and $\mathcal{L}^2 = \mathcal{L}^1$. However, there is no reason for $\mathcal{C}^2$ to be elemental. We split $\mathcal{C}^2$ into $\left(\mathcal{C}_1^2, \ldots, \mathcal{C}_{r_2}^2\right)$, where $\mathcal{C}_i^2$ is elemental. Morever, we have $\sum_{i=1}^{r_2} \mathcal{L}_i^2 = \mathcal{L}^2$ and $\sum_{i=1}^{r_2} k_i^2 = k^2$.
3. Let $\mathcal{C}^3$ be one of the $\mathcal{C}_i^2$ such that $\mathcal{L}^3/k^3 \geq \mathcal{L}^2/k^2$. Such an $\mathcal{C}_i^2$ exists, otherwise we have a contradiction: assume that we have

$$\forall i, \mathcal{L}_i^2/k_i^2 < \mathcal{L}^2/k^2$$
$$\Leftrightarrow \forall i, \mathcal{L}_i^2/k_i^2 < \frac{\sum_{j=1}^{r_2} \mathcal{L}_i^2}{\sum_{j=1}^{r_2} k_i^2}$$

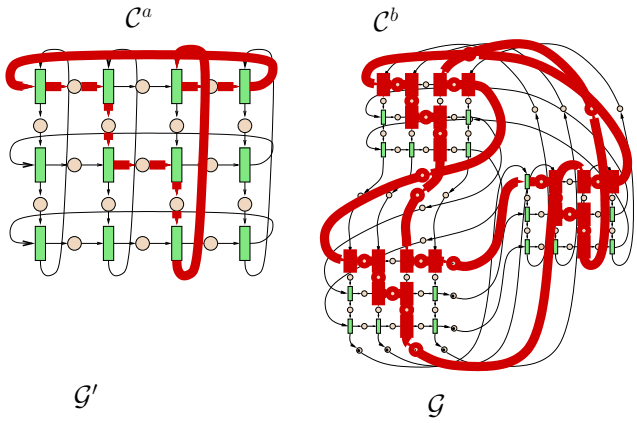


Figure 15. Diving $\mathcal{C}^a$ from $\mathcal{G}'$ to $\mathcal{G}$ to obtain $\mathcal{C}^b$.

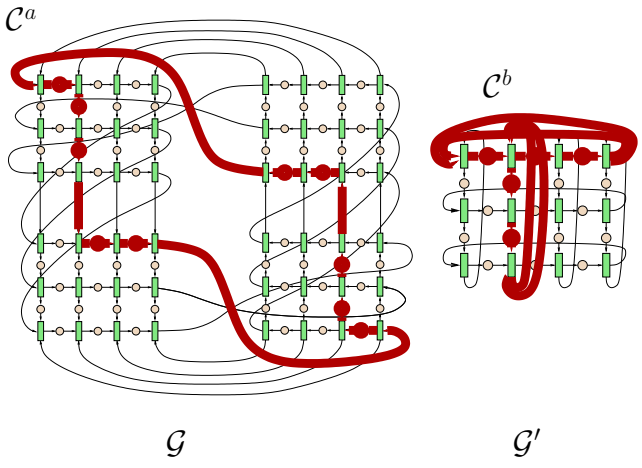


Figure 16. Projection of $\mathcal{C}^a$ from $\mathcal{G}$ to $\mathcal{G}'$ to obtain $\mathcal{C}^b$.

$$\Leftrightarrow \forall i, \mathcal{L}_i^2 \sum_{j=1}^{r_2} k_j^2 < k_i^2 \sum_{j=1}^{r_2} \mathcal{L}_j^2$$

We can sum these inequalities:

$$\Rightarrow \sum_{i=1}^{r_2} \left(\mathcal{L}_i^2 \sum_{j=1}^{r_2} k_j^2 \right) < \sum_{i=1}^{r_2} \left(\mathcal{L}_i^2 \sum_{j=1}^{r_2} k_j^2 \right)$$

$$\Leftrightarrow \left(\sum_{i=1}^{r_2} \mathcal{L}_i^2 \right) \left(\sum_{j=1}^{r_2} k_j^2 \right) < \left(\sum_{i=1}^{r_2} \mathcal{L}_i^2 \right) \left(\sum_{j=1}^{r_2} k_j^2 \right)$$

This last inequality is obviously wrong, showing that our $\mathcal{C}^3$ exists.

4. Let $\mathcal{C}^4$ be any elemental cycle of $\mathcal{G}'$, such that $\mathcal{L}^4/k^4$ is maximal. Since $\mathcal{C}^3$ is elemental, we have $\mathcal{L}^4/k^4 \geq \mathcal{L}^3/k^3$. Such a critical cycle can be found in time $O\left((uv)^3\right)$[2].

5. Let $\mathcal{C}^5$ the diving of $\mathcal{C}^4$ in $\mathcal{G}$. $\mathcal{C}^5$ is made of $c = \mathrm{lcm}(p, k^4)/k^4$ copies of $\mathcal{C}^4$. Thus, we have $\mathcal{L}^5 = n\mathcal{L}^4$ and $k^5 = nk^4$. Finally, $\mathcal{L}^5/k^5 = \mathcal{L}^4/k^4$. Again, there is no reason for $\mathcal{C}^5$ to be an elemental cycle. We split $\mathcal{C}^5$ into $\left(\mathcal{C}_1^5, \ldots, \mathcal{C}_{r_5}^5\right)$, where $\mathcal{C}_i^5$ is elemental.

6. Let $\mathcal{C}^6$ be one of the $\mathcal{C}_i^5$ such that $\mathcal{L}^6/k^6 \geq \mathcal{L}^5/k^5$. As before, we can ensure that $\mathcal{C}^6$ exists, and $\mathcal{C}^6$ is elemental. Moreover, the number of tokens in $\mathcal{C}^6$ is equal to k^6/p.

7. Finally, we have:
$$\mathcal{L}^6/k^6 \geq \mathcal{L}^5/k^5 = \mathcal{L}^4/k^4 \geq \mathcal{L}^3/k^3 \geq \mathcal{L}^2/k^2 = \mathcal{L}^1/k^1$$
Since p is positive, we have:
$$\mathcal{L}^6 p/k^6 \geq \mathcal{L}^5 p/k^5 = \mathcal{L}^4 p/k^4 \geq \mathcal{L}^3 p/k^3 \geq \mathcal{L}^2 p/k^2 = \mathcal{L}^1 p/k^1$$
We know that $\mathcal{L}^1 p/k^1$ is maximal; since $\mathcal{C}^6$ is an elemental cycle, we have: $\mathcal{L}^6/k^6 = \mathcal{L}^1/k^1$ and thus,
$$\mathcal{L}^4/k^4 = \mathcal{L}^1/k^1$$
.

We have shown that:

- $\mathcal{C}^4$ has the same critical weight as $\mathcal{C}^1$,
- $\mathcal{C}^4$ can be found without any knowledge on $\mathcal{G}$ nor $\mathcal{C}^1$,
- $\mathcal{C}^4$ is computed over $\mathcal{G}'$, which has a polynomial size.

Hence, even if the sub-TPN has an exponential size, the length of its critical cycles can be found in polynomial time for each of its connected components.

The Karp's algorithm runs in time $\mathcal{O}\left((uv)^2\right)$. Since we run this algorithm on all the p connected components, the total running time for the i-th communication is $\mathcal{O}\left((uv)^2 p\right) = \mathcal{O}\left((m_i m_{i+1})^2\right)$.

Thus, the total running time for all communications is $\mathcal{O}\left(\sum_{i=0}^{n-2} \left((m_i m_{i+1})^2\right)\right)$.

□