Mapping Filtering Streaming Applications With Communication Costs

Kunal Agrawal
CSAIL, Massachusetts Institute of Technology, USA
kunal_ag@mit.edu

Anne Benoit, Fanny Dufossé and Yves Robert
LIP, École Normale Supérieure de Lyon, France
{Anne.Benoit|Fanny.Dufosse|Yves.Robert}@ens-lyon.fr

February 2009

LIP Research Report RR-2009-06

Abstract

In this paper, we explore the problem of mapping *filtering streaming applications* on large-scale homogeneous platforms, with a particular emphasis on communication models and their impact. Filtering application are streaming applications where each node also has a *selectivity* which either increases or decreases the size of its input data set. This selectivity makes the problem of scheduling these applications more challenging than the more studied problem of scheduling "non-filtering" streaming workflows.

We identify three significant realistic communication models. For each of them, we address the complexity of the following important problems:

1. Given an execution graph, how can one compute the period and latency? A solution to this problem is an operation list which provides the time-steps at which each computation and each communication occurs in the system.
2. Given a filtering workflow problem, how can one compute the schedule that minimizes the period or latency? A solution to this problem requires generating both the execution graph and the associated operation list.

Altogether, with three models, two problems and two objectives, we present 12 complexity results, thereby providing solid theoretical foundations for the study of filtering streaming applications.

Key words: query optimization, web service, streaming application, workflow, communication model, period, latency, complexity results.

1 Introduction

This paper addresses the problem of mapping *filtering streaming applications*, or *filtering workflows*, on parallel platforms. This mapping problem was first studied in the context of query optimization over web services: in [1, 2], the authors consider the case where the web services were to be mapped one-to-one (one service on one processor) onto identical servers. A recent extension considers the same problem with different-speed servers [3]. However, none of these works include communication costs in the analysis. In this paper, we introduce three different and realistic communication models, and we revisit the problem in this more challenging framework.

Filtering workflows resemble traditional pipelined workflows, a popular programming paradigm for streaming applications like video and audio encoding and decoding, DSP applications etc [4, 5, 6]. Filtering workflows consist of a collection of various services that must be applied on a stream of consecutive data sets. These services are represented using a workflow graph, which contains several *nodes* (the services), and these nodes are connected to each other using first-in-first-out *channels* (or dependence constraints between services). Data is input into the graph using input channel(s) and the outputs are produced on the output channel(s). Until here, the filtering workflows resemble regular workflows. However, in filtering workflows, the services *filter* incoming data by a certain amount. More precisely, each service C_i has a *selectivity* σ_i and an *elementary cost* c_i. If an input of size δ_i is provided to C_i, then the output is of size $\sigma_i\delta_i$ while the computation requirement of the service is $c_i\delta_i$. Therefore, when $\sigma_i < 1$, the service shrinks data (hence the name *filter*). On the other hand, when $\sigma > 1$, the service expands the data. Just as in regular workflows, the goal is to map these computations on some parallel platform and then organize computations and communications so as to optimize some scheduling objective. Since data continually flows through these applications, typical objectives of the scheduler are to minimize the *period* (which is defined as the inverse of the throughput) or the *latency* (also called response time) [7, 8, 9, 10].

Given a collection of services, we produce a *plan*, which is the combination of an *execution graph* and an *operation list*:

- The execution graph is a directed acyclic graph (DAG) that characterizes the predecessors and successors of each service onto the target platform. The set of edges in the execution graph must include all the original precedence edges between services, but may well include additional edges to further filter the incoming data of some services, thereby reducing their actual execution time.
- The operation list details the time-steps at which every computation and every communication takes place. We assume that the schedule is cyclic, so that the execution list can be specified concisely.

If there is an edge from service C_i to service C_j in the plan, then the output from service C_i is fed into the input of service C_j. Let δ_0 be the size of the input data sets to the services which have no predecessors. The size of the input data set to a service C_j is δ_0 times the product of the selectivities of all of its ancestors. Therefore, there is reason to add an edge to the execution graph that did not exist in the original precedence constraints. For instance, if we add an edge from C_i to C_j, where C_i has a small selectivity, then C_j will require a smaller computation time and have a smaller-sized input and output data set (thereby reducing its future communication costs). However, adding an extra edge means that we are adding to communication costs, since C_i now has to send the data to C_j, which it would not have to if that edge were not added.

All of the assumptions related to service costs and selectivities are those of Srivastava et al. [1, 2]. Although their papers mainly deal with query optimization over web services (already an increasingly important application with the advent of Web Service Management Systems [11, 12]), the approach applies to general data streams [13] and to database predicate processing [14, 15]. In addition, our framework is quite similar to the problem of scheduling unreliable jobs on parallel machines [16] where service selectivities correspond to job failure probabilities. Due to lack of space, please refer to [17] for more related work.

In this paper, we only consider mapping filtering workflows on *homogeneous machines*: each server has the same speed, and all servers are connected to each other by communication links of equal bandwidth.

As in previous work [1, 2], we consider *one-to-one* mappings, where each server has at most one service mapped to it. Clearly, for one-to-one mappings, the number of servers must be equal to or more than the number of services. We show in this paper that most period or latency minimization are NP-complete even in this setting[1]. Note that we do not need to specify which service is mapped onto which server, since all servers are equivalent. Instead, we have to generate the execution graph together with the operation list, in order to minimize the period or latency.

The emphasis of our work is on the impact of communication models. We consider two commonly used communication models. The *no overlap* communication model requires that at any point, a server can either compute, or receive an incoming communication, or send an outgoing communication. This models single threaded machines where every operation is serialized. We define two variants for this model, one where we enforce in-order execution, and another where we allow out-of-order execution (which means interleaving communications and computations of different data sets) so as to reduce the idle-time incurred by the serial ordering of the communications. In contrast, the *overlap* communication model considers the situation where a server can compute and send/receive communications at the same time. This calls for multi-threaded machines and parallel communications. In all models, both computations and communications are non-preemptive, which means that they cannot be interrupted once initiated. Also, communications are synchronous (by rendez-vous between the sender and the receiver). This synchronization between servers can cause idle times.

Our main findings is that computing the period or the latency in all these models turns out to be difficult. As already stated, the minimization problems (finding the optimal plan to minimize the period or the latency) are all NP-hard. This result is surprising, since polynomial algorithms exist for homogeneous machines when we do not model communication [1, 2]. Therefore, modeling communication costs explicitly has a huge impact on the difficulty of mapping filtering services. In addition, and quite unexpectedly, the "orchestration" problems (given an execution graph, find the optimal operation list) also are of combinatorial nature. Finally, the choice of the model has a tremendous impact on the values that can be achieved. Many of our results and counter-examples apply to regular workflows (without selectivities), and should be of great interest to the whole community interested in scheduling streaming applications.

This paper is organized as follows. Section 2 describes the framework of the problem in more details. Section 3 illustrates the difference between communication models with the help of several examples. The next two sections constitute the core of the paper. Section 4 is devoted the period minimization problem, while Section 5 is the counterpart for latency minimization. Finally we give some conclusions and perspectives in Section 6.

2 Framework

This section is devoted to a precise statement of the different models and optimization problems. We then give a formal definition of the period and of the latency. Surprisingly, these formal definitions require quite a complicated formulation, so we work out an exemple in full details, in order to illustrate the differences between all the models.

2.1 Plans

As stated above, the target application $\mathcal{A}$ is a set of services (or filters, or queries) linked by precedence constraints. We write $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ where $\mathcal{F} = \{C_1, C_2, \ldots, C_n\}$ is the set of services and $\mathcal{G} \subset \mathcal{F} \times \mathcal{F}$ is the set of precedence constraints. A service C_i is characterized by its cost c_i and its selectivity σ_i.

[1] In general mappings, we can map several services onto the same server. Problems with general mappings are straightforwardly shown NP-hard by reduction from 2-Partition or bin packing [18].

For the computing resources, we have a homogeneous platform with p servers (or processors) of same speed s: the cost of a service does not depend upon the server it is mapped onto. In fact, we we can identify a service C_i with its server, because we restrict to one-to-one mappings. All servers are connected to each other by communication links of equal bandwidth b. The cost for transmitting a data of size δ is $\frac{\delta}{b}$. Let δ_0 be the size of input data.

We have to build a *plan* $PL = (EG, OL)$, that is an execution graph $EG = (\mathcal{C}, \mathcal{E})$ that summarizes all precedence relations in the mapping, and an operation list OL that captures the occurrence of each computation and each communication. We deal with the operation lists in Section A, after having described the communication models. As for the execution graph $EG = (\mathcal{C}, \mathcal{E})$, the nodes in $\mathcal{C}$ are the services in $\mathcal{F}$ and input/output nodes. There is an arc $(C_i, C_j) \in \mathcal{E}$ if C_i precedes C_j in the execution. There are two types of such arcs: those induced by the set of precedence constraints $\mathcal{G}$, which must be enforced in any case, and those added to reduce the period or the latency. Let $\mathsf{Ancest}_j(EG)$ denote the set of all ancestors[2] of C_j in the execution graph EG. Only arcs from direct predecessors are kept in $\mathcal{E}$. In other words, if $(C_i, C_j) \in \mathcal{G}$, then we must have $C_i \in \mathsf{Ancest}_j(EG)$ [3].

For each service C_k in $\mathcal{F}$, let $\mathcal{S}_{\text{in}}(k)$ be the set of its direct predecessors in EG, and let $\mathcal{S}_{\text{out}}(k)$ be the set of its direct successors. Entry nodes are nodes C_k such that $\mathcal{S}_{\text{in}}(k) = \emptyset$; for each of them we add an input node to $\mathcal{C}$ to model input from the outside world. Similarly, for each exit node C_k in $\mathcal{C}$ (with $\mathcal{S}_{\text{out}}(k) = \emptyset$), we add an output node to $\mathcal{C}$. We define:

$$C_{\text{in}}(k) = \frac{\delta_0}{b} \sum_{C_i \in \mathcal{S}_{\text{in}}(k)} \left(\prod_{C_j \in \mathsf{Ancest}_i(EG)} \sigma_j \right)$$

$$C_{\text{comp}}(k) = \left(\prod_{C_j \in \mathsf{Ancest}_k(EG)} \sigma_j \right) \times \frac{\delta_0 . c_k}{s}$$

$$C_{\text{out}}(k) = \frac{\delta_0}{b} \times |\mathcal{S}_{\text{out}}(k)| \times \left(\prod_{C_j \in \mathsf{Ancest}_k(EG)} \sigma_j \right) \times \sigma_k$$

Here, $C_{\text{in}}(k)$ is a lower bound of the time needed to receive input data from all the predecessors of C_k. The input data from each predecessor C_i is of size $\delta_0 \prod_{C_j \in \mathsf{Ancest}_i(EG)} \sigma_j$, hence it requires $\frac{\delta_0}{b} \prod_{C_j \in \mathsf{Ancest}_i(EG)} \sigma_j$ time units for communication from C_i. We add the communication from all the ancestors to get the total incoming communication time $C_{\text{in}}(k)$. This lower bound may not be met because of idle times due to server synchronizations for the communications. However, we have not yet specified in which order the different communications take place. This specification requires discussion of communication models. We discuss variations of both one-port [19] and multi-port models [20] in Section 2.2.

The outgoing communication lower bound $C_{\text{out}}(k)$ is defined similarly, except that the outgoing communication to each successor is of same size. Finally, $C_{\text{comp}}(k)$ is the execution time of C_k on the server, with the appropriate size factor involving the selectivities of all its ancestors. We assume that each service without successor in the execution graph performs a single output communication (this models returning the results to the outside world). Before discussing further the communication models, we make two important remarks that apply to all variants:

- The selectivity of a service influences the execution time of *all* its successors (if any) in the mapping. In other words, a service is "filtered" or "expanded" by the combined selectivity of all its predecessors. This implies that selectivities are independent, and that the cost of join operations is negligible in

[2] The ancestors of a service are the services preceding it, and the predecessors of their predecessors, and so on.

[3] Equivalently, $\mathcal{G}$ must be included, in the transitive closure of $\mathcal{E}$.

front of the service costs. All these hypotheses are enforced in the literature, but further work could be devoted to generalizations of this simple model.

- Because everything is homogeneous, we can scale all service costs as $c_k \leftarrow \frac{b}{\delta_0} \cdot \frac{c_k}{s}$. This allows to let $\delta_0 = b = s = 1$ without any loss of generality in the previous expressions for $C_{\text{in}}(k)$, $C_{\text{comp}}(k)$ and $C_{\text{out}}(k)$. At the end of the computation, it just remains to scale by $\frac{\delta_0}{b}$ the computed period and latency to obtain the actual values.

2.2 Communication models

This section presents the various communication models. We first present the general overview and then we formally state the constraints and the rules of the three models. We present only an informal description of the models. Detailed formulas are provided in Appendix A.

With overlap– In the first model, we assume full overlap of communications and computations, so that a server can receive, compute and send (independent) data simultaneously. This model, denoted as OVERLAP, calls for multi-port communications: many incoming (resp. outgoing) communications can take place at the same time, sharing the incoming (resp. outgoing) bandwidth, provided that the total communication capacity of the server is never exceeded. Independent computations take place in parallel to these communications. In this model, the server may operate concurrently on different consecutive data sets: while receiving input for a given data set, it can execute computations for some older data set and sends output for some even older data set. We define execution time $C_{\text{exec}}(k)$ of a service/server pair C_k as the maximum execution time of the send, receive and compute operations of its service:

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

The period $\mathcal{P}$ is defined as the interval between the completion of consecutive data sets. With this definition, the system can process data sets at a rate $1/\mathcal{P}$ (the throughput). 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. In the overlap model, the lower bound on the period is the maximum of the quantities $C_{\text{exec}}(k)$ over all services C_k:

$$\mathcal{P} = \max_{1 \leq k \leq n} C_{\text{exec}}(k)$$

Given an execution graph, it turns out that we can generate an order of communication/computations that achieves this lower bound in the multi-port model: see Theorem 1 in Section 4. Note that determining the optimal execution graph is still NP-complete, and therefore, the period minimization problem is still difficult. See Appendix A for a complete list of the resource contraints that need to be satisfied for the OVERLAP model.

Without overlap– In the models without overlap, a server performs communications and computations sequentially (instead of in parallel). This is typical of an execution with single-threaded programs and (one-port) serialized communications. Despite its apparent simplicity, the model calls for two variants.

- **INORDER** : In the first variant, called INORDER, each server completely processes a data set before starting the execution of the next one; it receives incoming communications for data set number, say, i, one after the other; then it executes the computations for this data set, and then it sends the output data to all its successors, one communication after the other. Only after completing this whole set of operations can the processing of data set $i + 1$ be started (with the incoming communications).
- **OUTORDER** : In this second variant, we allow for out-of-order execution, namely starting some operation (say, an incoming communication) for data set $i + 1$ (or even $i + j$, $j \geq 2$) while still processing data set i.

From an architectural point of view, we emphasize that the INORDER and OUTORDER variants may be overly pessimistic, as modern processors are capable of some internal parallelism. However, both operation modes correspond to blocking send/receive MPI primitives [21], and servers may encounter idle time due to the synchronizations in both models. Nevertheless, we expect less idle time for the OUTORDER model than for the INORDER model, due to the additional schedule flexibility of the former model. Both variants lead to a computation cost for server/service C_k is bounded below by

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

As before, a lower bound on the period is the maximum of the execution times. But unlike the OVERLAP model with multi-port communications, this lower bound cannot always be reached: see the example in Section 2.3. Note that the multi-port model is more flexible: since it allows to send data to many other servers simultaneously, orchestrating the communications in the multi-port model is an easier task than for the one-port model We refer to Appendix A for a list of resource contraints to be enforced for each model. We emphasize that there is no closed-form formula for the period with the INORDER and OUTORDER models, which we believe is a new and surprising observation.

Latency– We have just seen that models have a strong impact on the computation of the period. This is also true for the latency (or response time), but to a lesser extent. The latency (or response time) is the time needed to execute a single data set entirely. The overlap/no-overlap distinction is no longer meaningful for optimizing this criterion. Indeed, we can always fully serialize the processing of each data set and minimize the execution time, or makespan, when processing a unique data set. In other words, we delay the processing of the next data set until the current one is completely executed, this suppresses all resource conflicts. With such a strategy, the period is equal to the latency, which in turn is equal to longest path from an input node to an output node in the plan. However, the choice between one-port or multi-port communications does have an impact on the latency. This is illustrated by the example presented in Section 3.2.

Other variants– Altogether, we have three models, one multi-port model with overlap and two one-port variants without overlap; the precise constraints that need be enforced are detailed in Appendix A. We note that other models can be introduced, for instance one-port communications with computation/communication overlap. However, we believe that we address the most realistic combinations: on single-threaded machines it is hard to avoid doing everything sequentially, and on multi-threaded machines, we can execute computations and (several) communications concurrently. Another possibility is to consider preemptive models where communication and/or computation can be interrupted, and the bandwidth of communication can vary during the communication. Such preemptive models are beyond the scope of this paper.

We point out that the effective difference between one-port and multi-port communications is not evident to assess: in most cases, the optimal solution for the multi-port model obeys the one-port constraints. We present in Section 3 examples of execution graphs were the optimal latency and period for the multi-port model are strictly smaller than the optimal ones for the one-port model.

Characterizing solutions– In this paper, we study two optimization problems: (i) MINPERIOD: find a plan $PL = (EG, OL)$ that minimizes the period; and (ii) MINLATENCY: find a plan $PL = (EG, OL)$ that minimizes the latency. For each problem instance, independently of the model and of the objective function, the solution includes the execution graph EG that describes the set Ancest_i for each service C_i. But this graph alone does not give enough information to compute the schedule, i.e., the moment at which each operation takes place. Indeed, we need the complete list of the time-steps at which every communication or computation begins and ends. Because the target schedule is cyclic and repeats for each data-set, such a

list has a size proportional to the size of the plan, which is at most quadratic in the number of services: this information remains polynomial in the problem size.

Formally, we define the operation list OL as follows:

- For each service C_i, $\mathsf{BeginCalc}^n_{(i)}$ is the time-step where the computation of C_i on data set number n begins, and $\mathsf{EndCalc}^n_{(i)}$ is the time-step where this computation ends
- For each edge $C_i \rightarrow C_j$ in the plan, $\mathsf{BeginComm}^n_{(i,j)}$ is the time-step where this communication involving data set number n begins, and $\mathsf{EndComm}^n_{(i,j)}$ is the time-step where this communication ends.
- The schedule starts at time-step 0 with the data set number 0, and we impose a cyclic behavior of period λ:

$$\begin{cases} \mathsf{BeginCalc}^n_{(i)} = \mathsf{BeginCalc}^0_{(i)} + \lambda \times n & \text{for each service } C_i \\ \mathsf{EndCalc}^n_{(i)} = \mathsf{EndCalc}^0_{(i)} + \lambda \times n & \text{for each service } C_i \\ \mathsf{BeginComm}^n_{(i,j)} = \mathsf{BeginComm}^0_{(i,j)} + \lambda \times n & \text{for each communication } C_i \rightarrow C_j \\ \mathsf{EndComm}^n_{(i,j)} = \mathsf{EndComm}^0_{(i,j)} + \lambda \times n & \text{for each communication } C_i \rightarrow C_j \end{cases}$$

To each model are associated different rules that must be satisfied by the operation list so that the schedule is *valid*: no resource contraint nor model hypothesis is violated (see Appendix A for these rules). Note that all models are non-preemptive: once initiated, a communication or a communication cannot be interrupted. Also, communications are synchronous, and the bandwidth assigned to a given communication remains the same during its whole execution (this is not really a restriction for the one-port model but it is an important one for the multi-port model). With the operation list we can define the period and the latency of a plan PL:

- The period is $\mathcal{P} = \lambda$
- The latency is $\mathcal{L} = \max\{\mathsf{EndComm}^0_{(i,j)} | C_i \rightarrow C_j \in \mathcal{E}\}$. Remember that output nodes execute a communication to the outside world, so that the longest path for data set number 0 ends by one such communication.

2.3 Example

In this section, we work out a little example in order to better understand the three models. Consider an instance with 5 services of cost 4 and of selectivity 1 without dependence constraints. Let the execution graph EG be the graph presented in Figure 1.

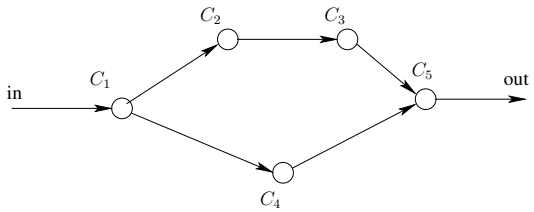

Figure 1: Example.

Latency– We start with the latency because it is simpler. Assume first one-port communications, hence the INORDER or OUTORDER models: remember that there is no difference between these models for computing the latency, in both cases we have to minimize the length of the longest path in the graph. If the first data is sent at time $t = 0$, then the computation of service C_1 is completed at time 5. Then the computation

of C_2 begins at time 6 if C_1 sends to C_2 at time 5 before sending to C_4 at time 6. The computation of C_3 begins at time 11. The computation of C_4 begins at time 7 and completes at time 11. Then, the communication between C_4 and C_5 can be done at time 12. In the meantime, C_3 completes its computation at time 15. Then, the computation of C_5 can begin at time 16 and is completed at time 20. With the last communication of C_5, this leads us to a latency of 21, which is easily seen to be the optimal value. This execution scheme corresponds to the following operation list: for computations, we have $\lambda = 21$, $\mathsf{BeginCalc}^0_{(1)} = 1$, $\mathsf{EndCalc}^0_{(1)} = 5$, $\mathsf{BeginCalc}^0_{(2)} = 6$, $\mathsf{EndCalc}^0_{(2)} = 10$, $\mathsf{BeginCalc}^0_{(3)} = 11$, $\mathsf{EndCalc}^0_{(3)} = 15$, $\mathsf{BeginCalc}^0_{(4)} = 7$, $\mathsf{EndCalc}^0_{(4)} = 11$, $\mathsf{BeginCalc}^0_{(5)} = 16$, and $\mathsf{EndCalc}^0_{(5)} = 20$. For communication times, we have $\mathsf{BeginComm}^0_{(in,1)} = 0$, $\mathsf{EndComm}^0_{(in,1)} = 1$, $\mathsf{BeginComm}^0_{(1,2)} = 5$, $\mathsf{EndComm}^0_{(1,2)} = 6$, $\mathsf{BeginComm}^0_{(1,4)} = 6$, $\mathsf{EndComm}^0_{(1,4)} = 7$, $\mathsf{BeginComm}^0_{(2,3)} = 10$, $\mathsf{EndComm}^0_{(2,3)} = 11$, $\mathsf{BeginComm}^0_{(3,5)} = 15$, $\mathsf{EndComm}^0_{(3,5)} = 16$, $\mathsf{BeginComm}^0_{(4,5)} = 11$, $\mathsf{EndComm}^0_{(4,5)} = 12$, $\mathsf{BeginComm}^0_{(5,out)} = 20$, and $\mathsf{EndComm}^0_{(5,out)} = 21$. With multi-port communications we cannot achieve a better latency for this example, so we derive the same solution. See Section 3.2 for an example where the multi-port latency is smaller than the one-port latency.

Period– Looking at the above operation list, we can obtain a period $\mathcal{P} = 5$ for the model OVERLAP: if we keep the same list and only change $\lambda = 21$ into $\lambda = 5$, we have no resource conflict. In fact we can achieve a period of 4 for the OVERLAP model, and this is clearly optimal as each computation has cost 4. To do so, we modify the following in the operation list: $\lambda = 4$, $\mathsf{BeginComm}^0_{(4,5)} = 12$, and $\mathsf{EndComm}^0_{(4,5)} = 13$. For example, between time 5 and 9, server C_1 receives data set number 3, computes the data set number 2, and sends data set number 1 to C_2 and C_4.

For the model OUTORDER, the minimal possible value for the period is 7: indeed for server C_5, there are two incoming communications of length 1, one computation of length 4 and one outgoing communication of length 1 (we get the same bound with C_1). This value cannot be obtained for service C_5 with the current operation list: the reception from C_4 for data set 1 (at time $12 + 7 = 19$ coincides with its computation for data set 0. To achieve a period 7, we can let $\mathsf{BeginComm}^0_{(4,5)} = 14$, and $\mathsf{BeginCalc}^0_{(4)} = 8$. We keep $\mathsf{BeginComm}^0_{(1,4)} = 6$, so there is an idle time between the end of this communication and the beginning of the computation. C_4 has another idle time at the end of this computation at time 12, and the cycle resumes for data set 1 at time $13 = 6 + 7 = \mathsf{BeginComm}^1_{(1,4)}$.

For the model INORDER, we have the same bound for the period as for the model OUTORDER, namely 7. With the previous operation list, we obtain a period 10 because of the cost of C_5: the beginning of the reception for data set 1 has to wait for the end of the emission of data set 0. This difference of 3 between 7 and 10 corresponds to the idle time between the end of the reception from C_4 and the beginning of the reception from C_3, which is the difference of the lengths of the path $C_1 \to C_4 \to C_5$ and of the path $C_1 \to C_2 \to C_3 \to C_5$. This idle time can be reduced (or shared) between C_1, C_4 and C_5 as follows. The time spent in computations and communications is 7 for C_1, 6 for C_4 and 7 for C_5 respectively. The optimal solution is to give an idle time $\frac{2}{3}$ for C_1, $1 + \frac{2}{3}$ for C_4 and $\frac{2}{3}$ for C_5. We obtain the following values: $\mathsf{BeginComm}^0_{(1,4)} = 6 + \frac{2}{3}$,$\mathsf{EndComm}^0_{(1,4)} = 7 + \frac{2}{3}$, $\mathsf{BeginCalc}^0_{(4)} = 7 + \frac{2}{3}$, $\mathsf{EndCalc}^0_{(4)} = 11 + \frac{2}{3}$, $\mathsf{BeginComm}^0_{(4,5)} = 13 + \frac{1}{3}$, and $\mathsf{EndComm}^0_{(4,5)} = 14 + \frac{1}{3}$. The other values do not change. We obtain a period $\frac{23}{3}$, which the reader may find surprising!

In this example, with the original operation list, we obtain three different values of the period for the three different models. In addition, and more interestingly, the optimal period is different for each model, and is obtained with a different operation list.

3 Counter-examples

In this section we give three examples to show the difficulty introduced by communication costs with the different models.

3.1 With and without communication cost

We start by showing the impact of communication costs on the optimal solution for the period: without communications, the optimal plan is always a linear chain for the services whose selectivities do not exceed 1 [1]. This property is no longer true in the OVERLAP model. An example is provided in Appendix B.1.

3.2 One-port/multi-port for latency

The example given in Section B.2 of the Appendix shows the difference between one-port and multi-port communications when computing the latency in the OVERLAP model. We point out that this result still holds for traditional workflows (without selectivities). To the best of our knowledge, this is a new and important observation for scheduling classical streaming applications.

3.3 One-port/multi-port for period

The example given in Section B.3 of the Appendix shows the difference between one-port and multi-port communications when computing the period in the OVERLAP model. The example with the period is more complicated than the one with the latency because we can have different data sets being processed concurrently. Just as for the latency, we point out that this result still holds for traditional workflows (without selectivities).

4 Period minimization

In this section, we study two problems related to period computation and minimization. First we address the following problem: given an execution graph, what is the complexity of determining the operation list that leads to the best period? We provide a polynomial algorithm for the OVERLAP model, and show that the problem is NP-hard for the INORDER and OUTORDER models. Then we address the general optimization problems MINPERIOD-OVERLAP, MINPERIOD-INORDER and MINPERIOD-OUTORDER: what is the complexity of determining the plan whose period is optimal? We show that these three problems are NP-hard.

4.1 Optimal period for a given execution graph

Theorem 1. *Given an execution graph, the problem of computing the operation list that leads to the optimal period has polynomial complexity with the* OVERLAP *model but is NP-hard with the* OUTORDER *and* INORDER *models.*

The proof of Theorem 1 is provided in Appendix C. Dealing with the OVERLAP model is not too difficult. Intuitively, in this model, all the communications can be executed in time

$$T = \max_{1 \leq k \leq n} \{C_{\text{in}}(k), C_{\text{out}}(k)\}$$

We just have to assign to any communication of size t a fraction t/T of the available bandwidth. Remember that we have normalized the bandwidth to $b = 1$. By doing so, the communications will be executed

in time T, and the sum of incoming or outgoing communications on any server is less than or equal to $b = 1$. On the other hand, the NP-completeness reduction for non-overlapping models is rather involved. We reduce the problem from RN3DM [22], a particular instance of 3-Dimensional Matching with two permutations (also called the permutation sums problem). We should point out that Theorem 1 holds for regular streaming applications (without selectivities). This is an important and new result in that context.

4.2 Computing the optimal period

We now address the complexity of the period minimization problem for the three models. Notice that the plan consists of both the execution graph and the operations list. As it turns out, computing the execution graph is NP-complete for all three period minimization problems. Therefore, even though we can compute the operations list for the OVERLAP model in polynomial time, the overall problem for computing a plan which minimizes the period is NP-complete.

On a positive note, we derived the following result on the structure of the optimal execution graph: for any instance of MINPERIOD without dependence constraints, and using any of the three models, there exists an optimal plan whose execution graph is a forest (see Proposition 4 in Appendix C for the proof). This "structural" result reduces the search of optimal execution graphs. Still, all minimization problems are NP-hard.

Theorem 2. *Problems* MINPERIOD-OVERLAP*,* MINPERIOD-OUTORDER *and* MINPERIOD-INORDER *without dependence constraints are all NP-hard.*

The proof of Theorem 2 is provided in Section C of the Appendix. Again, the NP-completeness reductions are quite involved.

We conclude this section by providing a particular polynomial instance of the problem: If we restrict the search for execution graphs and impose the restriction that the execution graph must be a linear chain, then the execution graph can be found in polynomial time using a greedy algorithm (see Proposition 8 in Appendix C for the proof). In addition, for this case, the operations list can be found quickly as well. Therefore, the problem of finding a plan which minimizes the period can be solved in polynomial time for all three communication models in this instance.

5 Latency minimization

This section is the counterpart of Section 4 for the latency. First we address the following problem: given an execution graph, what is the complexity of determining the operation list that leads to the best latency? This problem turns out to be NP-hard for all models (while determining the best period was polynomial for the OVERLAP model). The general optimization problems MINLATENCY-OVERLAP, MINLATENCY-INORDER and MINLATENCY-OUTORDER are all NP-hard. All these results imply technically involved reduction proofs.

5.1 Optimal latency for a given execution graph

As for the optimization of the period, the latency of a plan depends upon the operation list. We prove in this section that the computation of the optimal latency for a given execution graph is NP-hard for the three models.

Theorem 3. *Given an execution graph, the problem of computing the optimal operation list that leads to the optimal latency is NP-hard for the three models.*

The proof of Theorem 3 is provided in Appendix D. Also, we derive a polynomial case, namely computing the latency for a tree-shaped execution graph (see Proposition 12 in Appendix D). As for the period (Theorem 1), we point out that Theorem 3 holds for regular streaming applications (without selectivities). Again, this is an important and new result in that context.

5.2 Computing the optimal latency

In this section, we address the complexity of the latency minimization problem for the three models. Again, note that the fact that finding *the operations list given an execution graph* is NP-complete does not automatically imply that the problem of finding *the plan* is NP-complete. For example, the optimal plan may always consists of a simple execution graph for which the operations list can be computed in polynomial time. Therefore, in order to prove that the latency minimization problem is NP-complete, we have to argue that either (i) computing the execution graph that minimizes latency is NP-complete (as we did for the period minimization proofs) or (ii) that the plans that minimize latency contain the "difficult" execution graphs that do not allow to compute the best operation list easily. In this instance, we prove the following result using the second option.

Theorem 4. *Problems* MINLATENCY-OVERLAP*,* MINLATENCY-OUTORDER *and* MINLATENCY-INORDER *without dependence constraints are all NP-hard.*

The proof of Theorem 4 is provided in Appendix D. We conclude this section by providing the complexity of the problems where we impose the restriction that the execution graph must be a chain or a forest:

- The problem MINLATENCY when restricting to plans whose execution graphs are linear chains is polynomial for all models: see Proposition 16 in Appendix D for a proof.
- The problem MINLATENCY when restricting to plans whose execution graphs are forests is NP-hard for all models: see Proposition 17 in Appendix D for a proof.

6 Conclusion

In this paper, we have explored the problem of mapping filtering streaming applications on large-scale homogeneous platforms, with a particular emphasis on communication models and their impact. We have identified three natural and realistic communication models, with and without commmunication/computation overlap, and with one-port or bounded multi-port communications. We have addressed the following important problems:

- Given an execution graph, what is the complexity of computing the period or the latency?
- What is the complexity of the general period or latency minimization problem?

We have been able to provide the complexity of all the 12 optimization problems, thereby providing solid theoretical foundations for the study of filtering streaming applications. Several of our results apply to regular workflow applications, which broadens the scope and significance of our results to quite a large applicative framework.

In the future, we plan to explore models that allow preemption. This would require to carefully assess the cost of interruptions. Another important extension of this work would be to tackle bi-criteria problems: given a threshold period, what is the optimal latency? and conversely, given a threshold latency, what is the optimal period? All bi-criteria problems are trivially NP-hard (since mono-criterion problems already are) but we can search for approximation algorithms, or at least efficient heuristics.

References

[1] U. Srivastava, K. Munagala, J. Widom, and R. Motwani, "Query optimization over web services," in *VLDB '06: Proceedings of the 32nd Int. Conference on Very Large Data Bases*. VLDB Endowment, 2006, pp. 355–366.

[2] J. Burge, K. Munagala, and U. Srivastava, "Ordering pipelined query operators with precedence constraints," Stanford University, Research Report 2005-40, November 2005.

[3] A. Benoit, F. Dufossé, and Y. Robert, "On the complexity of mapping filtering services on heterogeneous platforms," LIP, ENS Lyon, France, Research Report 2008-30, Oct. 2008, short version to appear in IPDPS'09.

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

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

[6] 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.

[7] N. Vydyanathan, U. Catalyurek, T. Kurc, P. Saddayappan, and J. Saltz, "An approach for optimizing latency under throughput constraints for application workflows on clusters," Ohio State University, Columbus, OH, Research Report OSU-CISRC-1/07-TR03, Jan. 2007, available at ftp://ftp.cse.ohio-state.edu/pub/tech-report/2007.ShortversionappearsinEuroPar'2008.

[8] ——, "Optimizing latency and throughput of application workflows on clusters," Ohio State University, Columbus, OH, Research Report OSU-CISRC-4/08-TR17, Apr. 2008, available at ftp://ftp.cse.ohio-state.edu/pub/tech-report/2007.

[9] A. Benoit and Y. Robert, "Mapping pipeline skeletons onto heterogeneous platforms," *J. Parallel Distributed Computing*, vol. 68, no. 6, pp. 790–808, 2008.

[10] Q. Wu, J. Gao, M. Zhu, N. Rao, J. Huang, and S. Iyengar, "On optimal resource utilization for distributed remote visualization," *IEEE Trans. Computers*, vol. 57, no. 1, pp. 55–68, 2008.

[11] D. Florescu, A. Grunhagen, and D. Kossmann, "Xl: A platform for web services," in *CIDR 2003, First Biennial Conference on Innovative Data Systems Research*, 2003, on-line proceedings at http://www-db.cs.wisc.edu/cidr/program/p8.pdf.

[12] M. Ouzzani and A. Bouguettaya, "Query processing and optimization on the web," *Distributed and Parallel Databases*, vol. 15, no. 3, pp. 187–218, 2004.

[13] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom, "Adaptive ordering of pipelined stream filters," in *SIGMOD'04: Proceedings of the 2004 ACM SIGMOD Int. Conf. on Management of Data*. ACM Press, 2004, pp. 407–418.

[14] S. Chaudhuri and K. Shim, "Optimization of queries with user-defined predicates," *ACM Trans. Database Systems*, vol. 24, no. 2, pp. 177–228, 1999.

[15] J. M. Hellerstein, "Predicate migration: Optimizing queries with expensive predicates," in *In Proc. of the ACM SIGMOD Conf. on Management of Data*, 1993, pp. 267–276.

[16] A. Agnetis, P. Detti, M. Pranzo, and M. S. Sodhi, "Sequencing unreliable jobs on parallel machines," *Journal on Scheduling*, 2008, available on-line at http://www.springerlink.com/content/c571u1221560j432.

[17] A. Benoit, F. Dufossé, and Y. Robert, "Mapping filter services on heterogeneous platforms," LIP, ENS Lyon, France, Research Report 2008-19, Jun. 2008, available at graal.ens-lyon.fr/~abenoit/.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[19] P. Bhat, C. Raghavendra, and V. Prasanna, "Efficient collective communication in distributed heterogeneous systems," *Journal of Parallel and Distributed Computing*, vol. 63, pp. 251–263, 2003.

[20] B. Hong and V. Prasanna, "Bandwidth-aware resource allocation for heterogeneous computing systems to maximize throughput," in *Proceedings of the 32th International Conference on Parallel Processing (ICPP'2003)*. IEEE Computer Society Press, 2003.

[21] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra, *MPI the complete reference*. The MIT Press, 1996.

[22] W. Yu, H. Hoogeveen, and J. K. Lenstra, "Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard," *J. Scheduling*, vol. 7, no. 5, pp. 333–348, 2004.

Appendix

A Operation lists

Given a plan $PL = (EG, OL)$, the operation list OL defines the beginning and completion times of the computation of each service C_i (values $\mathsf{BeginCalc}^0_{(i)}$ and $\mathsf{EndCalc}^0_{(i)}$ for data set 0), and of each communication $C_i \to C_j$ for each edge in the plan (values $\mathsf{BeginComm}^0_{(i,j)}$ and $\mathsf{EndComm}^0_{(i,j)}$ for data set 0). The whole operation is cyclic and repeats every λ time-steps for a new data set.

Let B_i (resp. E_i) be the remainder of the Euclidian division of $\mathsf{BeginCalc}^0_{(i)}$ (resp. $\mathsf{EndCalc}^0_{(i)}$) by λ. Similarly, let $B_{(i,j)}$ (resp. $E_{(i,j)}$) be the remainder of the Euclidian division of $\mathsf{BeginComm}^0_{(i,j)}$ (resp. $\mathsf{EndComm}^0_{(i,j)}$) by λ.

Let $PL = (EG, OL)$ be a plan for an instance $\mathcal{A} = (\mathcal{F}, \mathcal{G})$. Recall that $C_{\text{comp}}(i) = \left(\prod_{C_k \in \mathsf{Ancest}_i(EG)} \sigma_k\right) c_i$ is the computation cost for C_i in the execution graph. Let $\delta(i,j) = \left(\prod_{C_k \in \mathsf{Ancest}_i(EG)} \sigma_k\right)$ be the cost of the communication from C_i to C_j whenever it exists. For consistency, note that $C_{\text{in}}(j) = \sum_{C_i \in \mathcal{S}_{\text{in}}(j)} \delta(i,j)$.

All models are non-preemptive: one initiated, a communication or a communication cannot be interrupted. Also, communications are synchronous, and the bandwidth assigned to a given communication remains the same during its whole execution (this is not really a restriction for the one-port model but it is an important one for the multi-port model)

One-port without overlap– A valid operation list for the models INORDER and OUTORDER should respect the following constraints:

- For each node C_i, $\mathsf{EndCalc}^0_{(i)} = \mathsf{BeginCalc}^0_{(i)} + C_{\text{comp}}(i)$ (computation time)
- For each edge $C_i \to C_j$, $\mathsf{EndComm}^0_{(i,j)} = \mathsf{BeginComm}^0_{(i,j)} + \delta(i,j)$ (communication time)
- For each node C_i, for each edge pair $C_j \to C_i$ and $C_k \to C_i$,
  - $\mathsf{EndComm}^0_{(j,i)} \leq \mathsf{BeginComm}^0_{(k,i)}$ or
  - $\mathsf{EndComm}^0_{(k,i)} \leq \mathsf{BeginComm}^0_{(j,i)}$

  This is the one-port constraint: for any service, two incoming communications for a same data set do not occur at the same time
- For each node C_i, for each edge pair $C_i \to C_j$ and $C_i \to C_k$,
  - $\mathsf{EndComm}^0_{(i,j)} \leq \mathsf{BeginComm}^0_{(i,k)}$ or
  - $\mathsf{EndComm}^0_{(i,k)} \leq \mathsf{BeginComm}^0_{(i,j)}$

  This is the counterpart for outgoing communications
- For each node C_i, for each edge $C_j \to C_i$,

$$\mathsf{EndComm}^0_{(j,i)} \leq \mathsf{BeginCalc}^0_{(i)}$$

  For any service, all incoming communications for a given data set are completed before the beginning of the computation

- For each node C_i, for each edge $C_i \rightarrow C_j$,

$$\mathsf{EndCalc}^0_{(i)} \leq \mathsf{BeginComm}^0_{(i,j)}$$

For any service, the computation is completed before the beginning of outgoing communications

For the model INORDER, we add the following constraint: for each node i, for each edge pair $C_j \rightarrow C_i$ and $C_i \rightarrow C_k$,

$$\mathsf{EndComm}^0_{(i,k)} \leq \mathsf{BeginComm}^1_{(j,i)} = \mathsf{BeginComm}^0_{(j,i)} + \lambda \quad (1)$$

Constraint (1) states that outgoing communications for a data set are completed before the beginning of incoming communications for the next data set.

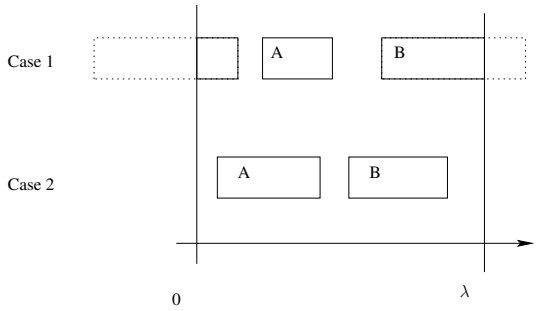

Figure 2: Cases for the INORDER model.

For the model OUTORDER, things get more complicated. We replace constraint (1) by the following set of constraints (see Figure 2):

- We forbid that an incoming communication and a computation take place at same time: for each edge $C_i \rightarrow C_j$,
  - $B_{(i,j)} \leq E_{(i,j)} \leq B_j \leq E_j$ (case 2, $A = C_i \rightarrow C_j$ and $B = C_j$) or
  - $B_j \leq E_j \leq B_{(i,j)} \leq E_{(i,j)}$ (case 2: $A = C_j$ and $B = C_i \rightarrow C_j$) or
  - $E_{(i,j)} \leq B_j \leq E_j \leq B_{(i,j)}$ (case 1: $A = C_j$ and $B = C_i \rightarrow C_j$) or
  - $E_j \leq B_{(i,j)} \leq E_{(i,j)} \leq B_j$ (case 1: $A = C_i \rightarrow C_j$ and $B = C_j$)
- We forbid that an outgoing communication and a computation take place at same time: for each edge $C_i \rightarrow C_j$,
  - $B_{(i,j)} \leq E_{(i,j)} \leq B_i \leq E_i$ (case 2: $A = C_i \rightarrow C_j$ and $B = C_i$) or
  - $B_i \leq E_i \leq B_{(i,j)} \leq E_{(i,j)}$ (case 2: $A = C_i$ and $B = C_i \rightarrow C_j$) or
  - $E_{(i,j)} \leq B_i \leq E_i \leq B_{(i,j)}$ (case 1: $A = C_i$ and $B = C_i \rightarrow C_j$) or
  - $E_i \leq B_{(i,j)} \leq E_{(i,j)} \leq B_i$ (case 1: $A = C_i \rightarrow C_j$ and $B = C_i$)
- We forbid that two different outgoing communications happen at the same time: for each edge pair $C_i \rightarrow C_j$ and $C_i \rightarrow C_k$,
  - $B_{(i,j)} \leq E_{(i,j)} \leq B_{(i,k)} \leq E_{(i,k)}$ (case 2: $A = C_i \rightarrow C_j$ and $B = C_i \rightarrow C_k$) or
  - $B_{(i,k)} \leq E_{(i,k)} \leq B_{(i,j)} \leq E_{(i,j)}$ (case 2: $A = C_i \rightarrow C_k$ and $B = C_i \rightarrow C_j$) or

- $E_{(i,j)} \leq B_{(i,k)} \leq E_{(i,k)} \leq B_{(i,j)}$ (case 1: $A = C_i \to C_k$ and $B = C_i \to C_j$) or
- $E_{(i,k)} \leq B_{(i,j)} \leq E_{(i,j)} \leq B_{(i,k)}$ (case 1: $A = C_i \to C_j$ and $B = C_i \to C_k$)

- We forbid that an incoming and an outgoing communications happen at the same time: for each edge pair $C_i \to C_j$ and $C_k \to C_i$,
  - $B_{(i,j)} \leq E_{(i,j)} \leq B_{(k,i)} \leq E_{(k,i)}$ (case 2: $A = C_i \to C_j$ and $B = C_k \to C_i$) or
  - $B_{(k,i)} \leq E_{(k,i)} \leq B_{(i,j)} \leq E_{(i,j)}$ (case 2: $A = C_k \to C_i$ and $B = C_i \to C_j$) or
  - $E_{(i,j)} \leq B_{(k,i)} \leq E_{(k,i)} \leq B_{(i,j)}$ (case 1: $A = C_k \to C_i$ and $B = C_i \to C_j$) or
  - $E_{(k,i)} \leq B_{(i,j)} \leq E_{(i,j)} \leq B_{(k,i)}$ (case 1: $A = C_i \to C_j$ and $B = C_k \to C_i$)

- We forbid that two different incoming communications happen at the same time: for each edge pair $C_j \to C_i$ and $C_k \to C_i$,
  - $B_{(j,i)} \leq E_{(j,i)} \leq B_{(k,i)} \leq E_{(k,i)}$ (case 2: $A = C_j \to C_i$ and $B = C_k \to C_i$) or
  - $B_{(k,i)} \leq E_{(k,i)} \leq B_{(j,i)} \leq E_{(j,i)}$ (case 2: $A = C_k \to C_i$ and $B = C_j \to C_i$) or
  - $E_{(j,i)} \leq B_{(k,i)} \leq E_{(k,i)} \leq B_{(j,i)}$ (case 1: $A = C_k \to C_i$ and $B = C_j \to C_i$) or
  - $E_{(k,i)} \leq B_{(j,i)} \leq E_{(j,i)} \leq B_{(k,i)}$ (case 1: $A = C_j \to C_i$ and $B = C_k \to C_i$)

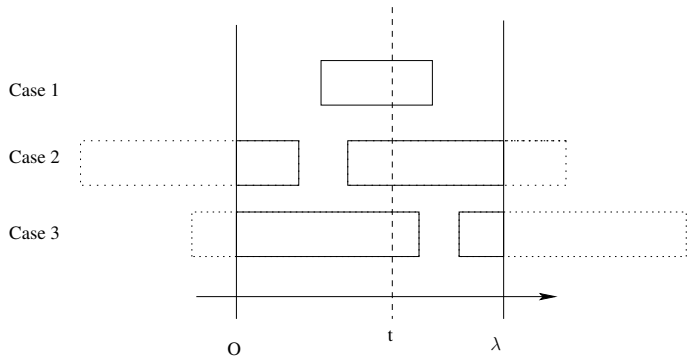

Figure 3: Cases for the OVERLAP model.

Multi-port with overlap– For the model OVERLAP, the servers can execute many incoming (outgoing) communications simultaneously. Bandwidth is shared between concurrent communications. Any communication on a server is assigned some ratio of the available bandwidth. This ratio does not change during the communication, hence the execution time of the communication is the cost of the communication multiplied by its bandwidth ratio. At any time, the sum of the ratios used should not exceed $b = 1$.

We define the set of incoming communications to C_i, that are beginning or in progress at time $t + k \times \lambda$ for k large enough (see Figure 3):

$$\mathcal{A}^i_{\text{in}}(t) = \{j \in \mathcal{S}_{\text{in}}(i) | B_{(j,i)} \leq t < E_{(j,i)} \text{ (case 1) or } E_{(j,i)} \leq B_{(j,i)} \leq t(2) \text{ (case 2) or } t < E_{(j,i)} \leq B_{(j,i)} \text{ (case 3)}\}$$

Similarly for outgoing communications from C_i at time $t + k \times \lambda$:

$$\mathcal{A}^i_{\text{out}}(t) = \{j \in \mathcal{S}_{\text{out}}(i) | B_{(i,j)} \leq t < E_{(i,j)} \text{ (case 1) or } E_{(i,j)} \leq B_{(i,j)} \leq t(2) \text{ (case 2) or } t < E_{(i,j)} \leq B_{(i,j)} \text{ (case 3)}\}$$

The operation list is valid if:

- For all node i, $\mathsf{EndCalc}^0_{(i)} = \mathsf{BeginCalc}^0_{(i)} + C_{\text{comp}}(i)$ (computation cost)
- For each node C_i and for each edge $C_j \to C_i$,

$$\sum_{k \in \mathcal{A}^i_{\text{in}}(B_{(j,i)})} \frac{\delta(k,i)}{\mathsf{EndComm}^0_{(k,i)} - \mathsf{BeginComm}^0_{(k,i)}} \leq 1$$

(incoming communications do not exceed the bandwidth)

$$\sum_{k \in \mathcal{A}^i_{\text{in}}(B_{(j,i)})} \frac{\delta(j,k)}{\mathsf{EndComm}^0_{(j,k)} - \mathsf{BeginComm}^0_{(j,k)}}$$

(outgoing communications do not exceed the bandwidth)

- For each edge $C_i \to C_j$,

$$\mathsf{EndComm}^0_{(i,j)} \leq \mathsf{BeginCalc}^0_{(j)} \text{ and } \mathsf{EndCalc}^0_{(i)} \leq \mathsf{BeginComm}_{(i,j)}$$

(for a given data set, incoming communications are completed before the computation, which itself is completed before outgoing communications)

B Counter-examples

B.1 With and without communication cost

This example shows the impact of communication costs on the optimal solution for the period: without communications, the optimal plan always is a linear chain for the services whose selectivities do not exceed 1 [1]. This property is no longer true in the OVERLAP model.

We present an instance of the problem MINPERIOD, and we study it without communication costs and with the model OVERLAP. We consider the following instance with 202 services:

- services C_1 and C_2 have selectivities $\sigma_i = 0,9999$ and costs $c_i = 100$
- services C_i for $3 \leq i \leq 202$ have selectivities $\sigma_i = 100$ and costs $c_i = \frac{100}{0,9999}$

Does there exist a plan whose period does not exceed 100?

For the model without communication cost, we obtain 100 by chaining C_1 and C_2, and by making C_2 the predecessor of all other services. But because of the outgoing communications of C_2, this gives us a period 200 with the model OVERLAP. We claim that the only solution with the model OVERLAP is the plan presented in Figure 4.

By contradiction, consider a plan with period less than 100. Let i be an integer with $3 \leq i \leq 202$. The cost of C_i is strictly greatest than 100, hence it must have C_1 or C_2 as a predecessor. But if C_1 precedes C_2, because of communication costs, C_1 can only have 99 other successors, and C_2 can only have 100 successors. There is a slot missing, unless some C_i has a successor C_j where $3 \leq i, j \leq 202$ and $i \neq j$. But then the computation time of C_j in G would be

$$\prod_{C_k \in \mathsf{Ancest}_j(G)} \sigma_k \times \sigma_i \times c_j \geq 0,9999^2 * 100 * 100 > 100$$

This establishes the claim.

As stated above, it is proved in [1] that in any optimal plan for MINPERIOD without communication costs, all services of selectivity less than one are chained. This example shows that such a structure is no longer always optimal with communication costs.

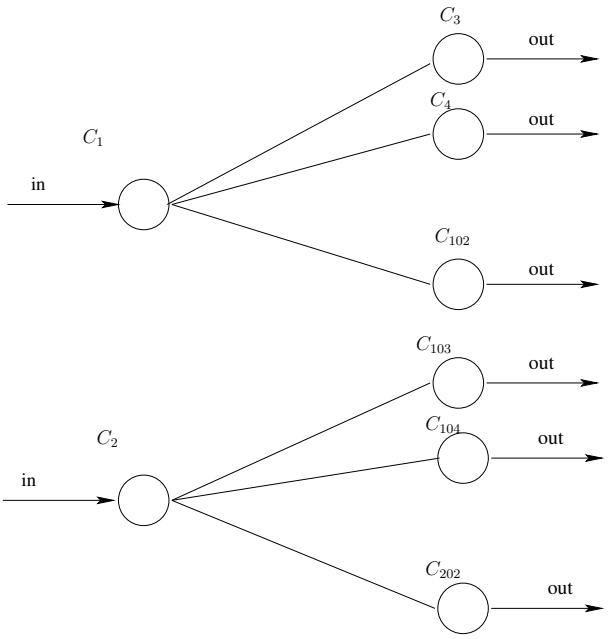

Figure 4: Optimal solution with communication costs.

B.2 One-port/multi-port for latency

In this example, we study the difference between one-port and multi-port communications when computing the latency in the OVERLAP model. Consider a problem instance with 12 services C_1 to C_{12}, all of unit cost. We assume that $\sigma_2 = \sigma_3 = 2$, $\sigma_4 = \sigma_5 = \sigma_6 = 3$, and the other selectivities are equal to 1. The execution graph EG is represented in Figure 5.

The computations of services C_1,...,C_6 can be completed at time 2. With multi-port communications, the communications between these services and services C_7,...,C_{12} can be executed within 6 time-steps; they all complete at time 8. The computations of services C_7,...,C_{12} complete at time 14 (the size of each input is 6), and the output communications to the outside world all complete at time 20, which leads to a latency $\mathcal{L} = 20$.

This latency cannot be achieved with one-port communications. To see this, first note that the computation of any service C_i with $1 \leq i \leq 6$ cannot be completed before time 2. Then the difference between the beginning of the computation of any service C_j with $7 \leq j \leq 12$ and the latency is at least 12, because of the cost of their computation and of their outgoing communication. Hence, to obtain a latency of 20, all communications from services C_1,...,C_6 to services C_7,...,C_{12} must be completed within 6 time-steps. But this value is equal to $C_{\text{out}}(i)$ for $1 \leq i \leq 6$ and to $C_{\text{in}}(j)$ for $7 \leq j \leq 12$: there cannot be any idle time on any service between the first incoming data and the last outgoing data. Suppose that there exists a valid operation list for one-port communications without idle-time and capable of executing all the communications within 6 time steps. Let C_j be the service such that $\mathsf{BeginComm}^0_{(1,j)} = 2$ and let C_k be the service such that $\mathsf{BeginComm}^0_{(1,k)} = 3$. We know that these two services exist because there cannot be any idle time on C_1. Then C_k is necessarily idle between time 2 and 3 because the only incoming communication on C_k of cost 1 is $C_1 \rightarrow C_k$. This contradiction proves that the optimal latency with one-port communications on this instance is strictly greater than 20.

We point out that this result still holds for traditional workflows (without selectivities). Indeed, the execution graph of the example can be viewed as the original DAG of a workflow where the weight of

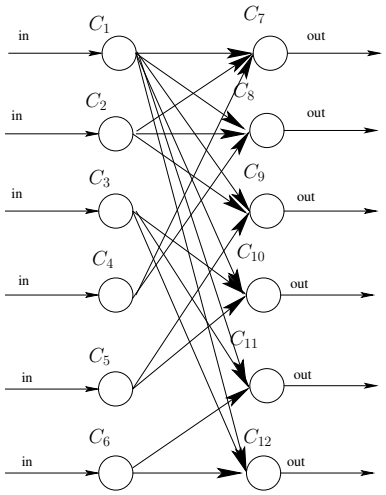

Figure 5: Execution graph for the example with the latency.

a node C_k is $C_{\text{comp}}(k)$ and where the volume of a communication from C_i to C_k is $\prod_{C_j \in \mathsf{Ancest}_i(EG)} \sigma_j$. To the best of our knowledge, this is a new and important observation for scheduling classical streaming applications.

B.3 One-port/multi-port for period

In this example, we study the difference between one-port and multi-port communications when computing the period in the OVERLAP model. The example with the period is more complicated than the one with the latency because we can have different data sets being processed concurrently. Consider the following problem instance with 8 services:

- $\forall i, c_i = 1$,
- $\sigma_1 = \sigma_2 = 3$,
- $\sigma_3 = 4$,
- $\sigma_4 = 2$,
- $\forall 4 < i \leq 8, \sigma_i = 1$

The execution graph EG is represented in Figure 6.

With the multi-port model, the optimal value of the period is given by the maximum time needed for communications, i.e. $\mathcal{P} = 12$. Can we obtain this value with one-port communications? Notice that $C_{\text{out}}(1) = C_{\text{out}}(2) = C_{\text{out}}(3) = 12$ and $C_{\text{in}}(5) = C_{\text{in}}(6) = C_{\text{in}}(7) = 12$. That means that if there exists a solution, then, on these servers, there will be no idle time in their communications.

Suppose that there exists a valid operation list of period $\lambda = 12$. Consider the steady-state operation, and let t be a time-step at which a communication from C_3 to C_5 begins. Then, as there is no idle time on C_3 for outgoing communications, there is a communication from C_3 at time $t + 4$ (suppose it goes to C_6) and at $t + 8$ (suppose it goes to C_7).

There is no idle time on C_5 for incoming communications either, hence there is an incoming communication on C_5 that begins at time $t + 4$. This communication is of size 2 or 3.

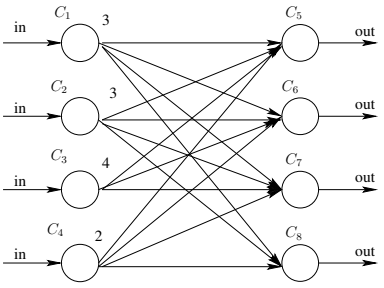

Figure 6: Execution graph for the example with the period.

Suppose first that this communication is of size 3. This case is represented in Figure 7. We can suppose that it comes from C_1. We study the beginning time of the communication from C_1 to C_7. There is no idle on C_1, and this server sends data to C_5 between $t+4$ and $t+7$: hence this communication begins at time $t+7$, $t+10$ or $t+13$ (or $t+1$ for the previous data set). Server C_7 receive data from C_3 between $t+8$ and $t+12$. That means that the communication from C_1 cannot begin at $t+7$, and nor at $t+10$. There only remains time $t+13$. There remains an idle slot between $t+12$ and $t+13$ for a communication, but there is no communication of size 1. As there is no idle time on C_7, we obtain a contradiction.

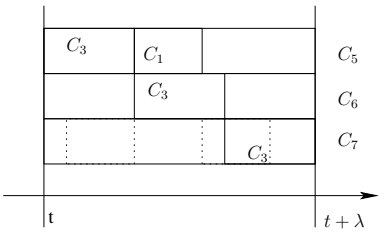

Figure 7: Case 1.

Suppose now that there is an incoming communication to C_5 of size 2 that begins at time $t+4$, followed by a communication of size 3 (suppose it comes from C_1). We study the beginning time of the communication from C_1 to C_6. Server C_1 sends data to C_5 from time $t+6$ to $t+9$ and it has no idle time. Then the communication from C_1 to C_6 can begin at time $t+9$ or $t+12$ (or t for the previous data set) or $t+15$ (or again $t+3$ for the previous data set). This communication cannot begin at time $t+3$ because it is of size 3 and the communication from C_3 to C_6 begins at time $t+4$. If it begins at time $t+9$, we obtain an idle time of size 1 between $t+8$ and $t+9$, and if it begins at time t, we obtain an idle time of size 1 between $t+3$ and $t+4$. However, we have seen in the previous case that this is not possible. We obtain a contradiction.

Altogether, for computing the period in the OVERLAP model, we have proven that multi-port communications are strictly "stronger" than one-port communications. Just as in Section 3.2 for the latency, we point out that this result still holds for traditional workflows (without selectivities).

C Period minimization

C.1 Optimal period for a given execution graph

Theorem 1. *Given an execution graph, the problem of computing the operation list that leads to the optimal period has polynomial complexity with the* OVERLAP *model but is NP-hard with the* OUTORDER *and*

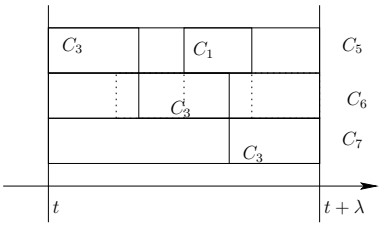

Figure 8: Case 2.

INORDER *models.*

The proof of Theorem 1 is given by Propositions 1, 2 and 3.

Proposition 1. *Given an execution graph, the problem of computing the operation list that leads to the optimal period has polynomial complexity with the* OVERLAP *model.*

Proof. Consider an execution graph EG for an application $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ Let $T = \max_{1 \leq k \leq n}\{C_{\text{exec}}(k)\}$. This value is a lower bound for period, and we prove that it can be met.

For each communication of size t, we assign to this communication a fraction t/T of the available bandwidth at the sender/receiver pair. That means that all communications will execute during T time-steps. For any server, the sum of the bandwidth ratios of incoming communications does not exceed 1, by definition of T. The same holds for outgoing communications. By definition of T, the computation time of each server also fits within the period.

We have not yet specified which data sets are operated upon by the different servers. But the previous discussion shows that every server can repeat its operations every T time-units without conflict. If suffices to let the first data set traverse the execution graph greedily: each communication is performed as soon as possible, and each computation is performed as soon as all the necessary data (all incoming communication) is available. We then repeat this scheme for every data set every T time units, and we obtain an operation list of period T. □

Proposition 2. *Given an execution graph, the problem of computing the operation list that leads to the optimal period is NP-hard with the* OUTORDER *model.*

Proof. We consider the associated decision problem and show that is NP-complete: given an application $\mathcal{A} = (\mathcal{F}, \mathcal{G})$, an execution graph EG for this application, and a bound K, does there exist an operation list for EG such that the period does not exceed K? This problem is obviously in NP: given $\mathcal{A}$, EG and an operation list, we have the period λ and check whether it does not exceed K. To establish completeness, we use a reduction from RN3DM [22]. We consider an instance $\mathcal{I}_1$ of this problem: given an integer vector $A = (A[1], \ldots, A[n])$ of size $n \geq 2$, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that:

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{2}$$

We can suppose that $2 \leq A[i] \leq 2n$ for all i and that $\sum_{i=1}^{n} A[i] = n(n+1)$, otherwise we know that the instance $\mathcal{I}_1$ has no solution. We associate to $\mathcal{I}_1$ an instance $\mathcal{I}_2$ with $2n + 5$ services without dependence constraints, all of selectivity 1, and whose costs are as follows:

- $c_1 = c_{2n+5} = n$ and $c_{2n+3} = c_{2n+4} = 2n + 1$
- $c_{2i} = 2n + 1$ for $1 \leq i \leq n + 1$ and $c_{2i+1} = 2n + 1 - A[i]$ for $1 \leq i \leq n$

- $\sigma_i = 1$ for $1 \leq i \leq 2n + 5$

The execution graph is represented in Figure 9. Finally, we let $K = 2n + 3$. The size of $\mathcal{I}_2$ is obviously linear in the size of $\mathcal{I}_1$.

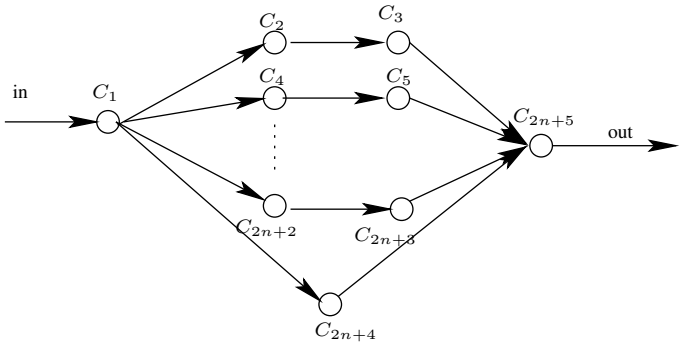

Figure 9: graph G.

Intuitively, we note that service C_1 has many successors and C_{2n+5} many predecessors. We need the ordering of the associated communications to compute the optimal period for this execution graph. We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution.

Suppose first that $\mathcal{I}_1$ has a solution λ_1, λ_2. We compute the following operation list for $\mathcal{I}_2$: C_1 first communicates with C_{2n+4}. Then services $C_2, C_4, \ldots, C_{2n}$ are fed in the ordering given by λ_1. Finally C_{2n+4} is the last service to receive data from C_1. Receptions by C_{2n+5} are done in the order C_1, $C_{2(n-\lambda_2(1))+3}$, $\ldots$, $C_{2(n-\lambda_2(n))+3}$, C_{2n+3}. With this orchestration, owing to Equation 2, the period is $2n + 3$.

Suppose now that $\mathcal{I}_2$ has a solution. For a data set k, suppose that the computation of C_{2n+2} begins at time i and that the computation of C_{2n+4} begins at time j. For services C_1, C_{2n+2} and C_{2n+4}, the sum of the costs of communications and of computations is equal to $2n + 3$. That means that there is no idle time for the associated servers. Hence at time $i - 1$ (resp. $j - 1$), there is a communication between servers C_1 and C_{2n+2} (resp. C_{2n+4}) for data set k. Hence service C_1 sends the result of its computation for data set k between time-steps $i - 1$ and j or between time-steps $j - 1$ and i. Therefore, $|j - i| + 1 \leq n + 2$. For services C_{2n+3}, C_{2n+4} and C_{2n+5}, the sum of the costs of communications and of computations is equal to $2n + 3$. That means that there is no idle time for the associated servers. Hence the computation of data set k on C_{2n+3} and C_{2n+4} are completed at time $i + 4n + 3$ and $j + 2n + 1$ respectively. C_{2n+5} receives the corresponding data from C_{2n+3} and C_{2n+4} at time $i + 4n + 3$ and $j + 2n + 1$ respectively. Then service C_{2n+5} receives the data for the computation of data set k between time $i + 4n + 1$ and $j + 2n$. Hence $|(i + 4n + 3) - (j + 2n + 1)| + 1 = |(i - j) + 2n + 2| + 1 \leq n + 2$. We obtain $j - i = n + 1$. As a consequence, for $1 \leq i \leq n$, the communication from C_1 to C_{2i} is done between time j and $j + n$ and the communication between C_{2i+1} and C_{2n+5} is done between time $j + 2n + 2$ and $j + 3n + 2$. Let λ_1 be the ordering of communications from C_1 to services $C_2, ..., C_{2n}$ and λ_2 be the permutation such that $n + 1 - \lambda_2$ is the ordering of communication from $C_3, ..., C_{2n+1}$ to C_{2n+5}. We obtain

$$\begin{array}{rc} & \forall i, \lambda_1(i) + (2n+1) + 1 + (2n+1-A[i]) + \lambda_2(i) = 4n+3 \\ \iff & \forall i, \lambda_1(i) + \lambda_2(i) = A[i]. \end{array}$$

This completes the proof. □

Proposition 3. *Given an execution graph, the problem of computing the operation list that leads to the optimal period is NP-hard with the* INORDER *model.*

Proof. We use the same reduction as for Proposition 2, because the optimal operation list fulfilled the constraints of the INORDER model. □

C.2 Computing the optimal period

Proposition 4. *For any instance of* MINPERIOD *without dependence constraints, and using any of the three models, there exists an optimal plan whose execution graph is a forest.*

Proof. For this proof, for any execution graph $EG = (\mathcal{C}, \mathcal{E})$, we define the number of added predecessors of a vertex $v \in \mathcal{C}$ as $na(v) = 0$ if v has zero or one direct predecessor, and as $na(v) = p - 1$ if v has $p \geq 2$ direct predecessors. We also define the number of added predecessors of EG as $na(EG) = \sum_{v \in \mathcal{C}} na(v)$.

Let $\mathcal{I}$ be an instance of MINPERIOD. Let EG be the execution graph of an optimal plan for this instance which has the minimal number of added predecessors. Suppose that EG is not a forest. Let C be a service of minimal depth with at least direct predecessors. Let C_1, C_2 be two of these predecessors.

Suppose first that C_1 and C_2 have no common predecessor. Let P_1 and P_2 the paths from an entry node to C_1 and from an entry node to C_2. There are unique by construction of C. Let Σ_1 be the product of selectivities on P_1 and Σ_2 the product of selectivities on P_2. If $\Sigma_1 \geq 1$ (resp. $\Sigma_2 \geq 1$), we can remove the edge $C_1 \to C$ (resp. $C_2 \to C$): this will decrease the product of selectivities for C as well as the output communication cost of C_1 (resp.C_2). If $\Sigma_1 < 1$ and $\Sigma_2 < 1$, let C_2' be the root of the path P_2. C_2' is an entry node of G by construction of P_2. If we remove the edge $C_1 \to C$ and add an edge $C_1 \to C_2'$, the product of selectivities for C does not change, and the product of selectivities for the services of P_2 decreases. These two operations strictly decrease the number of added predecessors of the graph EG and does not increase the period. This contradicts the hypothesis that EG is a optimal graph for the period with a minimal number of added predecessors.

Suppose now that C_1 and C_2 have common predecessors. Let C_1' (resp. C_2') be the predecessor of C_1 (resp. C_2) of minimal depth such that C_1' (resp. C_2') is not a predecessor of C_2 (resp. C_1). Let C' be the direct predecessor of C_1' and C_2'. Let P_1 (resp. P_2) be the path from C_1' (resp. C_2') to C_1 (resp. C_2). Let Σ_1 be the product of selectivities on P_1 and Σ_2 the product of selectivities on P_2. If $\Sigma_1 \geq 1$ (resp. $\Sigma_2 \geq 1$), we can remove the edge $C_1 \to C$ (resp. $C_2 \to C$): this will decrease the product of selectivities for C as well as the output communication cost of C_1 (resp.C_2). If $\Sigma_1 < 1$, we can remove the edge $C_1 \to C$ and add an edge $C_1 \to C_2'$. The product of selectivities for C does not change, and the product of selectivities for the services of P_2 decreases. These two operations strictly decrease the number of added predecessors of the graph EG and does not increase the period, a contradiction.

We can conclude that an optimal execution graph for the period whose number of added predecessors is minimal, necessarily is a forest. □

Theorem 2. *Problems* MINPERIOD-OVERLAP, MINPERIOD-OUTORDER *and* MINPERIOD-INORDER *without dependence constraints are all NP-hard.*

The proof of Theorem 2 is given by Propositions 5, 6 and 7.

Proposition 5. *The problem* MINPERIOD-OVERLAP *without dependence constraints is NP-hard.*

Proof. We consider the associated decision problem and show that is NP-complete: given n services without dependence constraints and a bound K, is there a plan whose period does not exceed K? This problem is obviously in NP: given a plan, that is an execution graph together with an operation list, we are given the period, and we can check that the operation list is valid in polynomial time. To establish the completeness, we use a reduction from RN3DM [22]. Consider an instance $\mathcal{I}_1$ of RN3DM: given an integer vector $A = (A[1], \ldots, A[n])$ of size n, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{3}$$

We can suppose that $2 \leq A[i] \leq 2n$ for all i and that $\sum_{i=1}^{n} A[i] = n(n+1)$, otherwise we know that the instance has no solution. We associate to $\mathcal{I}_1$ an instance $\mathcal{I}_2$ of RN3DM with $3n$ services without dependence constraints. For convenience we denote these services as $C_{1,i}$, $C_{2,i}$, and $C_{3,i}$ for $1 \leq i \leq n$. We let $K = \frac{3}{2}$. Service costs and selectivities are the following:

- $\forall i$, $c_{1,i} = K$, $c_{2,i} = K \times \frac{2}{b+1}$ and $c_{3,i} = \frac{K}{a^2} \times \gamma^{-A[i]}$
- $\forall i$, $\sigma_{1,i} = \sigma_{2,i} = a \times \gamma^i$ and $\sigma_{3,i} = \frac{K}{b^2}$
-

Here, a, b and γ are rational numbers such that

- $\sqrt[2n]{\frac{3}{4}} < a < b < \sqrt[2n]{\frac{3,2}{4}}$ with $2^n \times a \in \mathbb{N}$ and $2^n \times b \in \mathbb{N}$
- $1 < \gamma < \sqrt[n]{\frac{b}{a}}$ with $2^n \times \gamma \in \mathbb{N}$

We have $a < b < 1$ and $\gamma \leq 2$, so their numerators are bounded by 2^{n+1}. Altogether, a, b and γ can be represented with $O(n)$ bits, which is polynomial in the size $O(n)$ of $\mathcal{I}_1$ (we have n services). But we need to prove that we can find such numbers. For a fixed n, we can find two rational numbers a and b with denominator 2^n if the function $f(n) = \sqrt[2n]{\frac{3,2}{4}} - \sqrt[2n]{\frac{3}{4}} - 2 * 2^{-n}$ is positive. We have

$$\begin{aligned} f'(n) &= -\ln(\tfrac{3,2}{4})\tfrac{1}{2n^2} * \sqrt[2n]{\tfrac{3,2}{4}} + \ln(\tfrac{3}{4})\tfrac{1}{2n^2} \sqrt[2n]{\tfrac{3}{4}} + 2\ln(2)^{-}n \\ &\sim (\ln(\tfrac{3}{4}) - \ln(\tfrac{3,2}{4}))\tfrac{1}{2n^2} < 0 \end{aligned}$$

We obtain that f tends to 0 as n tends to $+\infty$, and that f' is negative for n big enough. This proves that there exists n_0 such that $\forall n > n_0, f(n) > 0$. This gives the existence of a and b for n large enough.

Now we have a and b rational numbers with denominator 2^n and both smaller than 1. Then, in worst case, $1 < \gamma < \sqrt[n]{\frac{2^n}{2^n-1}}$. Let $g(n) = \sqrt[n]{\frac{2^n}{2^n-1}} - 1 - 2^{-2n}$. Then

$$\begin{aligned} g'(x) &= \tfrac{2}{\sqrt[n]{2^n-1}}(\tfrac{\ln(2^n-1)}{n^2} - \tfrac{\ln(2)2^n}{n(2^n-1)}) + 2\ln(2)2^{-2n} \\ &= \tfrac{2}{\sqrt[n]{2^n-1}}(\tfrac{\ln(2)}{n} + \tfrac{\ln(1-2^{-n})}{n^2} - \tfrac{\ln(2)}{n(1-2^{-n})}) + 2\ln(2)2^{-2n} \\ &\sim \tfrac{2}{\sqrt[n]{2^n-1}}(\tfrac{\ln(2)}{n} - \tfrac{2^{-n}}{n^2} - \tfrac{\ln(2)}{n}(1+2^{-n})) + 2\ln(2)2^{-2n} \\ &\sim \tfrac{2}{\sqrt[n]{2^n-1}}(-\tfrac{2^{-n}}{n^2} - \tfrac{\ln(2)}{n}2^{-n}) + 2\ln(2)2^{-2n} \\ &\sim -\tfrac{2}{\sqrt[n]{2^n-1}} \times \tfrac{\ln(2)}{n}2^{-n} < 0 \end{aligned}$$

This proves the existence of γ for n big enough.

Finally, all costs and selectivities are rational numbers whose numerators and denominators are of the order at most $O(2^{n^2}$, hence the size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$.

We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Suppose first that $\mathcal{I}_1$ has a solution λ_1, λ_2. We prove that the plan whose execution graph is represented in Figure 10 is a solution of $\mathcal{I}_2$. For all i, j,$c_{1,i} = K$ and $\sigma_i \times C_{2,j} \leq \frac{2b}{b+1} \times K \leq K$ since $b < 1$. In addition, all communication costs are less than one and $K > 1$, then all services $C_{1,i}$ and $C_{2,i}$ respect the bound on the period. For any i, the incoming communication volume to $C_{3,i}$ is less than K. The outgoing communication volume is at most $\max_j\{\sigma_{1,j}\} \times \max_j\{\sigma_{2,j}\} \times c_{3,i} = b \times b \times \frac{K}{b^2} = K$. Concerning the computation of $C_{3,i}$, we obtain a cost

$$\begin{aligned} C_{\text{comp}}(3,i) &= \sigma_{1,\lambda_1(i)}\sigma_{2,\lambda_2(i)}c_{(3,i)} \\ &= a^2\gamma^{\lambda_1(i)+\lambda_2(i)} \times \tfrac{K}{a^2} \times \gamma^{-A[i]} \\ &= K \end{aligned}$$

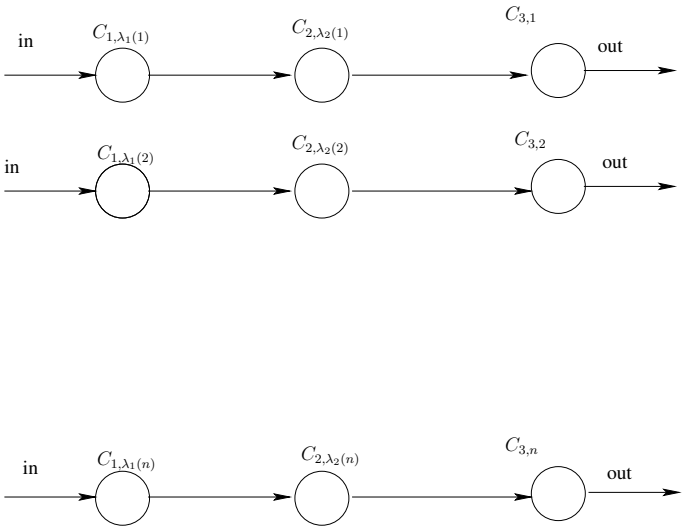

Figure 10: instance $\mathcal{I}_2$

We conclude that this plan is a valid solution of $\mathcal{I}_2$.

Suppose now that $\mathcal{I}_2$ has a solution. We prove that there exists λ_1 and λ_2 such that this solution has the plan of Figure 10. For all i, $c_{2,i} = K \times \frac{2}{b+1} > P$ since $b < 1$ and $c_{3,i} = \frac{K}{a^2} \times \gamma^{-A[i]} \geq \frac{K}{a^2} > K$. That means that these services cannot be entry nodes in a solution. For all i, j, $\sigma_{1,i} \times c_{3,j} \geq a \times \frac{K}{a^2}\frac{a^2}{b^2} \geq K\frac{a}{b^2} > K$ since $\frac{a}{b^2} > 1$. That means that in a solution, $C_{3,i}$ has at least 2 predecessors. Suppose that there exist i, j such that $C_{3,i}$ is predecessor of $C_{3,j}$. The outgoing communication of $C_{3,i}$ is at least: $a^{2n} \times \frac{K^2}{b^4} \geq \frac{9}{4b^2} \times K > K$. We obtain a contradiction. Suppose that there exists a service $C_{1,i}$ or $C_{2,i}$ with at least two direct successors. Then the outgoing communication of this service has a cost at least $2a^2 > \frac{3}{2} = K$. This prove that the services $C_{3,i}$ are arranged on n independent chains of length at least 3. In addition, the services $C_{2,i}$ cannot be entry nodes of the plan. This proves that the plan of the solution has the structure of Figure 10. Let λ_1, λ_2 be two permutations such that the predecessors of the service $C_{3,i}$ are $C_{1,\lambda_1(i)}$ and $C_{2,\lambda_2(i)}$.

The computation cost of each service $C_{3,i}$ is smaller than K:

$$\begin{array}{rll} & \forall i & C_{\text{comp}}(3,i) \leq K \\ \iff & \forall i & P \times \gamma^{\lambda_1(i)+\lambda_2(i)-A[i]} \leq K \\ \iff & \forall i & \gamma^{\lambda_1(i)+\lambda_2(i)-A[i]} \leq 1 \\ \iff & \forall i & \lambda_1(i) + \lambda_2(i) - A[i] \leq 0 \\ \iff & \forall i & \lambda_1(i) + \lambda_2(i) - A[i] = 0 \end{array}$$

Altogether, we have proven that $\mathcal{I}_1$ has a solution. This concludes the proof. □

Proposition 6. *The problem* MINPERIOD-OUTORDER *without dependence constraints is NP-hard.*

Proof. We consider the associated decision problem and show that is NP-complete: given n services without dependence constraints and a bound K, is there a plan whose period does not exceed K? As before, this problem is obviously in NP. To establish the completeness, we use a reduction from RN3DM [22]. Consider an instance $\mathcal{I}_1$ of RN3DM: given an integer vector $A = (A[1], \ldots, A[n])$ of size n, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{4}$$

We can suppose that $2 \leq A[i] \leq 2n$ for all i and that $\sum_{i=1}^{n} A[i] = n(n+1)$, otherwise we know that the instance has no solution.

Let $x_i = y_i = n - i$ and $z_i = A[i]$ for $1 \leq i \leq n$, and let $\alpha = (1 + 2^{-n})$. We construct the following instance $\mathcal{I}_2$ of our problem. We have a set $\mathcal{F} = \{C_0, C_1^x, \ldots, C_n^x, C_1^y, \ldots, C_n^y, C_1^z, \ldots, C_n^z\}$ of $3n+1$ services. Each service C_i^s, where $s \in \{x, y, z\}$ has a selectivity of σ_i^s and the computation cost of c_i^s. Here is how we pick the selectivities and the computation costs for the services.

1. Let $m = 2n$. For n enough big, we have $\alpha^m < (1 + \epsilon)$, where $\epsilon = 1/(2n)$.
2. Compute $K = (1 + \epsilon)/(\epsilon\alpha^m)$. Note that $K = (1 + \epsilon)/(\epsilon\alpha^m) > (1 + \epsilon)/(1.5\epsilon) > 2n/1.5 > n + 2$ for $n \geq 7$, since $\alpha^m < (1 + \epsilon) < 1.5$.
3. For the first service, we set selectivity $\sigma_0 = 1/(\alpha^m(1 + \epsilon))$ and $c_0 = K - 1 - n\sigma_0$. This is feasible, since $c_0 = K - 1 - n\sigma_0 > n + 2 - n + 1 = 1$. Therefore, the computation cost is always positive.
4. For services C_i^x, where $1 \leq i \leq n$, we set the selectivity such that $\sigma_i^x = \alpha^{x_i}$. Therefore, $1 < \sigma_i^x < 1 + \epsilon$. We pick $c_i^x = K/\sigma_0 - \sigma_i^x - 1$. Again, the computation cost is positive.
5. For the next n services C_i^y where $1 \leq i \leq n$ we set selectivity $\sigma_i^y = (1 + \epsilon)\alpha^{y_i}$ so that $1 + \epsilon < \sigma_i^y < (1 + \epsilon)^2$. Similarly, choose $c_i^y = K/(\sigma_0(1 + \epsilon)) - 1 - \sigma_i^y$.
6. For the next n services, C_i^z for $1 \leq i \leq n$, we pick c_i^z and σ_i^z such that $1 + \sigma_i^z + c_i^z = \alpha^{z_i} K$ and set $\sigma_i^z = (1 + 2\epsilon)$.

The size of $\mathcal{I}_2$ is clearly polynomial in the size of $\mathcal{I}_1$.

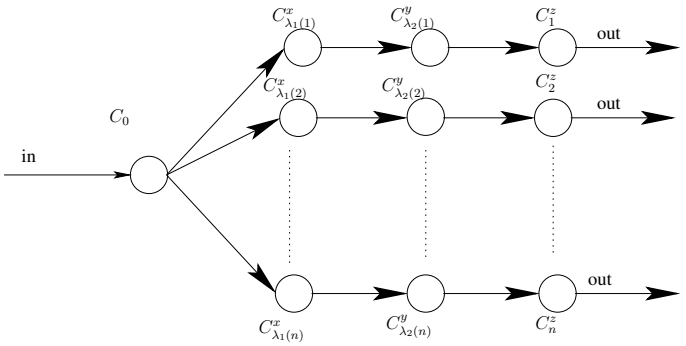

Figure 11: Structure of the optimal execution graph.

We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Suppose first that $\mathcal{I}_1$ has a solution with permutations a and b. Then, we prove that $\mathcal{I}_2$ has a solution of the form shown in Figure 11. C_0 appears first, and has n outgoing links that input into services C_1^x through C_n^x. All other services just have one output. For $1 \leq i \leq n$, the output of service C_i^x goes to service $C^y_{\lambda_1(i)}$ and the output of service $C^y_{\lambda_1(i)}$ goes into the input of service $C^z_{\lambda_2(i)}$. We now prove that the period of this mapping is exactly K.

1. The period of the first service C_0 is $1 + c_o + n\sigma_0 = 1 + K - 1 - n\sigma_0 + n\sigma_0 = K$, by construction.
2. The period of services C_i^x is $\sigma_0(1 + c_i^x + \sigma_i^x) = \sigma_0(1 + K/\sigma_0 - \sigma_i^x - 1 + \sigma_i^x) = K$.
3. The next n services are generated using the values from set Y. The period of service $C^y_{\lambda_1(i)}$ for all $1 \leq i \leq n$ is $\sigma_0\sigma_i^x(1 + c^y_{\lambda_1(i)} + \sigma^y_{\lambda_1(i)}) = \sigma_0\sigma_i^x(1 + K/(\sigma_0(1 + \epsilon)) - 1 - \sigma^y_{\lambda_1(i)} + \sigma^y_{\lambda_1(i)}) = \sigma_0\sigma_i^x K/(\sigma_0(1 + \epsilon)) < K$, since $\sigma_i^x < (1 + \epsilon)$

4. The period of service $f_{\lambda_2(i)}$ for all $1 \leq i \leq n$ is

$$\begin{aligned} P &= \sigma_0 \sigma_i^x \sigma^y_{\lambda_1(i)}(1 + c^z_{\lambda_2(i)} + \sigma^z_{\lambda_2(i)}) \\ &= \sigma_0 . \alpha^{x_i}(1+\epsilon)\alpha^{y_{\lambda_1(i)}} . K\alpha^{z_{\lambda_2(i)}} \\ &= \sigma_0 \alpha^{2n}(1+\epsilon)K \\ &= \frac{1}{(1+\epsilon)\alpha^{2n}}\alpha^{2n}(1+\epsilon)K \\ &= K \end{aligned}$$

We now prove that if we have a solution to $\mathcal{I}_2$, then we have a solution to $\mathcal{I}_1$. Say we have a mapping of services with period is at most K.

Observation 1. *C_0 must appear first in the mapping, and all other services must come after C_0.*

Proof. If a service C_i^s, $s \in \{x, y, z\}$ is not after C_0, its period is at least $1 + c_i^s + \sigma_i^s$, since the selectivity of all other services is greater than 1.

1. For services C_i^x, the period is at least $1 + c_i^x + \sigma_i^x = K/\sigma_0 > K$, since $\sigma_0 < 1$.
2. For services C_i^y, we have period $1 + c_i^y + \sigma_i^y = K/(\sigma_0(1+\epsilon)) > K$ since $\sigma_0 = 1/((1+\epsilon)\alpha^{2n}) < 1/(1+\epsilon)$
3. For services C_i^z, we have $1 + \sigma_i^z + c_i^z > K\alpha^{2n} > K$ by definition.

□

Observation 2. *By construction, we know that for the period to be less than K, C_0 can have at most n outgoing communications.*

Observation 3. *All of C_0's outgoing links go directly into C_1^x through C_n^x.*

Proof. We know that $c_i^x = K/\sigma_0 - \sigma_i - 1$. Assume for the sake of contradiction that C_i^x is after some service C_j^s, $s \in \{x, y, z\}$. Then the period of C_i^x is $\sigma_0\sigma_j^s(1 + c_i^x + \sigma_i^x) = \sigma_0\sigma_j^s(K/\sigma_0) > K$ since all $\sigma_j^s > 1$. □

Observation 4. *Each of these services C_1^x through C_n^x can have at most one outgoing branch.*

Proof. If C_i^x had two branches, the period would be $\sigma_0(1 + c_i^x + 2\sigma_i^x) = \sigma_0(1 + K/\sigma_0 - 1 - \sigma_i^x + 2\sigma_i^x) = K + \sigma_0\sigma_i^x > K$. □

Observation 5. *The outgoing branches from C_1^x through C_n^x go into distinct services C_1^y through C_n^y, not necessarily in order.*

Proof. Assume for contradiction that we can put service C_j^s, $s \in \{y, z\}$ between C_k^x and C_i^y. The period of C_i^y is $\sigma_0\sigma_k^x\sigma_j^s(1 + c_i^y + \sigma_i^y) > \sigma_0(1+\epsilon)(1 + K/(\sigma_0(1+\epsilon)) - 1 - \sigma_i^y + \sigma_i^y) = K$, since $\sigma_k^x > 1$ and $\sigma_j^s > (1+\epsilon)$. □

Observation 6. *Again, these services C_1^y through C_n^y can have only one outgoing edge.*

Proof. If they had 2 edges and if service C_k^x precedes this service C_i^y, their period is

$$\begin{aligned} P &= \sigma_0\sigma_k^x(1+c_i^y+2\sigma_i^y) \\ &> \sigma_0(K/(\sigma_0(1+\epsilon))+\sigma_i^y) \\ &> K/(1+\epsilon)+\sigma_0(1+\epsilon) \\ &= K/(1+\epsilon)+K\epsilon(1+\epsilon)/(1+\epsilon)^2 \\ &= K. \end{aligned}$$

□

Observation 7. *Finally, all of the services C_i^z are on these outgoing n edges.*

Therefore, the structure of the graph to get a period of K is exactly as is shown in figure.
Let us consider the service C_k^z, which is chained after services C_i^x and C_j^y. The period of service C_k^z is

$$\begin{aligned} \sigma_0\sigma_i^x\sigma_j^y(1+c_k^z+\sigma_k^z) &\leq K \\ \sigma_0.\alpha^{x_i}(1+\epsilon)\alpha^{y_j}K\alpha^{z_k} &\leq K \\ \sigma_0(1+\epsilon)K\alpha^{(x_i+y_j+z_k)} &\leq K \\ \sigma_0(1+\epsilon)\alpha^{(x_i+y_j+z_k)} &\leq 1 \\ \frac{1}{(1+\epsilon)\alpha^{2n}}(1+\epsilon)\alpha^{(x_i+y_j+z_k)} &\leq 1 \\ \alpha^{(x_i+y_j+z_k)} &\leq \alpha^{2n} \\ x_i+y_j+z_k &\leq 2n \end{aligned}$$

Since all the sums are less than or equal to $2n$ and $\sum_{i=1}^{n} A[i] = n(n+1)$, all sums have to be equal to $2n$. Because $x_i + y_j + z_k = 2n \Leftrightarrow i + j = A[k]$, we have a solution to $\mathcal{I}_1$. This concludes the proof. □

Proposition 7. *The problem* MINPERIOD-INORDER *without dependence constraints is NP-hard.*

Proof. We use the same reduction as for Proposition 7, because the optimal operation list fulfilled the constraints of the INORDER model. □

C.3 Problems on chains

Proposition 8. *The problem* MINPERIOD *when restricting to linear chain execution graphs is polynomial for all models.*

Proof. On a chain of servers, the models INORDER and OUTORDER are equivalent: they lead to the same value of the period. Consider an optimal execution graph EG for either model. Let $C_i \to C_j$ be two successive services. We suppose that

$$\max\{1+c_i+\sigma_i, \sigma_i(1+c_j+\sigma_j)\} \leq \max\{1+c_j+\sigma_j, \sigma_j(1+c_i+\sigma_i)\}$$

otherwise we can exchange their positions. Let $c'_k = 1+c_k+\sigma_k$ for all k. If $\sigma_i, \sigma_j \leq 1$, we have $c'_i \leq c'_j$, and if $\sigma_i, \sigma_j \geq 1$, then we have $\frac{\sigma_i}{c'_i} \leq \frac{\sigma_j}{c'_j}$ and the only remaining case is $\sigma_i < 1$ and $\sigma_j > 1$. Therefore the problem can be solved by the following greedy algorithm: place services of selectivity less than 1 by increasing value of c'_k, and then have them followed by services of selectivity at least 1 arranged by increasing value of $\frac{\sigma_k}{c'_k}$.

Similarly, for the model OVERLAP, we can suppose that

$$\max\{1, c_i, \sigma_i, \sigma_i c_j, \sigma_i\sigma_j\} \leq \max\{1, c_j, \sigma_j, \sigma_j c_i, \sigma_j\sigma_i\}$$

Let $c'_k = \max\{1, c_k\}$ for all k. Then we see that $\max\{c'_i, \sigma_i c'_j\} \leq \max\{c'_j, \sigma_j c'_i\}$. We obtain the same greedy algorithm as above with the new value of c'_k. □

D Latency minimization

D.1 Optimal latency for a given execution graph

Theorem 3. *Given an execution graph, the problem of computing the optimal operation list that leads to the optimal latency is NP-hard for the three models.*

The proof of Theorem 3 is given by Propositions 9, 10 and 11.

Proposition 9. *Given an execution graph, the problem of computing the optimal operation list that leads to the optimal latency is NP-hard for the model* OUTORDER.

Proof. We consider the associated decision problem: given an application $\mathcal{A} = (\mathcal{F}, \mathcal{G})$, an execution graph EG for this application, and a bound K, does it exist an operation list for EG such that the latency does not exceed K? This problem is obviously in NP: given $\mathcal{A}$, EG and an operation list, we can compute $\max\{\mathsf{EndComm}^0_{(i,j)} |\ C_i \to C_j \in \mathcal{E}\}$ and check whether this value does not exceed K.

The NP-completeness is obtain by reduction from RN3DM. Let $\mathcal{I}_1$ be an instance of RN3DM: given an integer vector $A = (A[1], \ldots, A[n])$ of size n, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{5}$$

We can suppose that $\sum_{i=1}^{n} A[i] = n(n+1)$, otherwise we know that the instance has no solution. We associate to $\mathcal{I}_1$ an instance $\mathcal{I}_2$ with $n+2$ services C_0 to C_{n+1}, without dependence constraints, all of selectivity 1, and whose costs are as follows:

- $c_0 = c_{n+1} = 1$
- $c_i = B[i] = n - A[i] + n^2$ for $1 \leq i \leq n$

We let $K = n + 4 + n^2$. The execution graph is a fork-join plan EG represented in Figure 12. The size of the instance $\mathcal{I}_2$ is linear in the size of the instance $\mathcal{I}_1$.

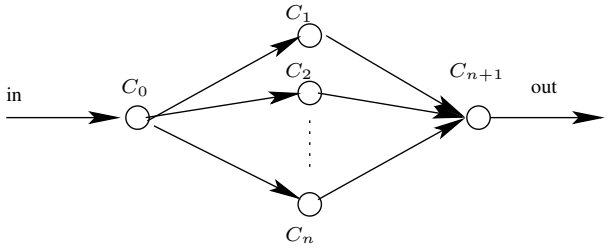

Figure 12: The fork-join execution graph.

We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Suppose first that $\mathcal{I}_1$ has a solution λ_1, λ_2. Then for $1 \leq i \leq n$, the services C_i are fed in the order λ_1 and the receptions are done in the order $n+1-\lambda_2$. For service C_i, the computation begins at time $2+\lambda_1(i)$ and is completed at time $2+\lambda_1(i)+B[i]$. There remain $\lambda_2(i)$ communications to do when the service C_i sends its data to service C_{n+1}. So the final latency is at least $l(i) = \lambda_1(i) + B[i] + \lambda_2(i) + 4$ for any i. In fact, we see that the latency $\mathcal{L}$ is equal to equal to $\max_i l(i)$. Hence $\mathcal{L} \leq \max_i \lambda_1(i) + B[i] + \lambda_2(i) + 4 = n + 4 + n^2$. That proves that $\mathcal{I}_2$ has a solution.

Suppose now that $\mathcal{I}_2$ has a solution. Let λ_1 be the sending order from C_0 and λ_2 be a permutation such that $n + 1 - \lambda_2$ is the order of receptions by C_{n+1}. For service C_i, the computation begins at time $2 + \lambda_1(i)$

and is completed at time $2 + \lambda_1(i) + B[i]$. There remain $\lambda_2(i)$ communications to do when the service C_i send its data to service C_{n+1}. So the final latency $\mathcal{L}$ is at least $l(i) = \lambda_1(i) + B[i] + \lambda_2(i) + 4$ for all i, and we have $\mathcal{L} \leq K =\leq n + 4 + n^2$. Hence $\lambda_1(i) + B[i] + \lambda_2(i) \leq n + n^2$, or $\lambda_1(i) + \lambda_2(i) \leq A[i]$ for all i. Summing up, we see that all these inequalities are in fact equalities, hence a solution to $\mathcal{I}_1$[4]. □

Proposition 10. *Given an execution graph, the problem of computing the optimal operation list that leads to the optimal latency is NP-hard for the model* INORDER.

Proof. We use the same reduction as for Proposition 9, because the optimal operation list fulfilled the constraints of the INORDER model. □

Proposition 11. *Given an execution graph, the problem of computing the optimal operation list that leads to the optimal latency is NP-hard for the model* OVERLAP.

Proof. This proof uses the reduction in the proof of Proposition 9. Let $\mathcal{I}_1$ be an instance of RN3DM. Let $\mathcal{I}_2$ be the instance associated to $\mathcal{I}_1$ by this proof. Let EG be the execution graph presented in Figure 12. A valid solution for the model OUTORDER is valid for the model OVERLAP. We show the converse: for any valid solution involving multi-port communications, there is a solution involving only one-port communications and whose latency is at least as good. Intuitively, when there is a single predecessor common to several nodes, the best is to feed these nodes sequentially. Sharing the bandwidth would only delay the first communications without accelerating the other ones.

Suppose that there exists a valid operation list OL_1 for OVERLAP on instance $\mathcal{I}_2$. Let λ_1 be the completion order of the communications from C_0. We construct an operation list OL_2 such that all communications are still done in the order λ_1 but their assigned bandwidth ratios are now all equal to 1 (which means that the communications are done sequentially in OL_2). For $1 \leq i \leq n$, the communication $C_0 \rightarrow C_i$ is completed not later in OL_2 than in OL_1: if it is the $\lambda_1(i)$-th communication, it is completed at least at time $\lambda_1(i)$ after the end of the computation of C_0 in OL_1, and this value is obtained in OL_2.

Let λ_2 be the reverse order of the beginning of the communications to C_{n+1} in OL_1. In OL_2, we execute these communications in the order $n + 1 - \lambda_2$ with bandwidth ratio 1. The time needed for the the last i sends in OL_2 is not larger in OL_1, because it is equal to i in OL_2 and at least this value in OL_1.

In this plan, a valid operation list for the model OVERLAP is therefore valid for the model OUTORDER with the hypothesis $\lambda \geq \max\{\mathsf{EndComm}^0_{(i,j)} |\ C_i \rightarrow C_j \in \mathcal{E}\}$. This concludes the proof. □

Data: tree T
Result: latency L
if *T is restricted to a leaf C_i* **then**
  $L = c_i$
else
  Let $T_1, ..., T_k$ be the subtrees of the children of the root C_0 of T;
  for $i = 1$ ***to*** k **do** Compute the optimal latency L_i of the subgraph T_i;
  Let σ be a permutation such that $L_{\sigma(1)} \leq ... \leq L_{\sigma(k)}$;
  Let C_j be the root of T;
  Let $L = 1 + c_0 + \sigma_0 \times \max_{1 \leq i \leq k}\{(k - i + 1) + L_{\sigma(i)}\}$;
end

Algorithm 1: Computation of the latency on a tree.

Proposition 12. *Algorithm 1 computes in time $O(n \log(n))$ the optimal latency of a plan whose execution graph is a tree.*

[4]Note that we did not define λ in $\mathcal{I}_2$. As pointed out before, we can always enforce a period λ equal to $\mathcal{L} = K$ to avoid any resource conflict.

Proof. First we note that for tree-shaped execution graph all models are equivalent with respect to the latency: as explained in the proof of Proposition 11, one-port communications are always dominant.

For any node, the algorithm feeds the subtrees by non-increasing latency, which is clearly optimal.

□

D.2 Computing the optimal latency

Theorem 4. *Problems* MINLATENCY-OVERLAP*,* MINLATENCY-OUTORDER *and* MINLATENCY-INORDER *without dependence constraints are all NP-hard.*

The proof of Theorem 4 is given by Propositions 13, 14 and 15.

Proposition 13. *The problem* MINLATENCY-OUTORDER *without dependence constraints is NP-hard.*

Proof. We consider the associated decision problem: given a period L, is there a mapping of latency less than L? The problem is obviously in NP: given a $\mathcal{A}$, EG and OL, we can compute $\max\{\mathsf{EndComm}^0_{(i,j)} | C_i \to C_j \in \mathcal{E}\}$ and check wether this value does not exceed K.

The NP-completeness is obtained by reduction from RN3DM, a special instance of 3-dimensional matching. Let $\mathcal{I}_1$ be an instance of RN3DM: given an integer vector $A = (A[1], \ldots, A[n])$ of size $n \geq 2$, does there exist two permutations λ_1 and λ_2 of $\{1, 2, \ldots, n\}$ such that

$$\forall 1 \leq i \leq n, \quad \lambda_1(i) + \lambda_2(i) = A[i] \tag{6}$$

We can suppose that $2 \leq A[i] \leq 2n$ for all i and that $\sum_{i=1}^{n} A[i] = n(n+1)$, otherwise we know that the instance has no solution. We construct an instance $\mathcal{I}_2$ with $n+2$ services as follows:

- one service F (where F stands for *fork*) with cost c_f and selectivity σ_f both equal to $\frac{1}{20n}$
- n services C_i, $1 \leq i \leq n$, with cost $c_i = 10n - A[i]$ and selectivity $\sigma_i = \sigma = 1 - \frac{1}{2n}$
- one service J (where J stands for *join*) with cost $c_j = 1$ and selectivity $\sigma_j = 200n^2 - 1$

Given this construction, we ask whether this set of services be arranged to obtain a latency at most $K = \frac{1}{2} + 10n\sigma^n + \frac{1}{20n}$. The size of $\mathcal{I}_2$ is clearly polynomial in the size of $\mathcal{I}_1$.

We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Suppose first that $\mathcal{I}_1$ has a solution with permutations λ_1 and λ_2. Then, we prove that $\mathcal{I}_2$ has a solution whose plan is a fork-join graph with F as the source, having the n services C_i as its children, and J being the final node, successor of all the n services C_i. The services C_i are fed in order λ_1 and the reception are done in order $n + 1 - \lambda_2$. For service C_i, the computation begins at $c_f + \sigma_f \lambda_1(i) = \sigma_f \lambda_1(i) + \frac{1}{20n}$ and is completed at $c_f + \sigma_f(\lambda_1(i) + c_i)$. There remain $\lambda_2(i)$ communications of size $\sigma_f \sigma \leq \sigma_f$ to execute when service C_i sends its data to final service J. The start-up time of J is bounded by $\max_i l(i)$, where

$$l(i) = c_f + \sigma_f(\lambda_1(i) + c_i + \sigma\lambda_2(i)) \leq c_f + \sigma_f(\lambda_1(i) + 10n - A[i] + \lambda_2(i)) = c_f + \sigma_f(10n) = \frac{1}{2} + \frac{1}{20n}$$

Hence this plan achieves a latency at most

$$\frac{1}{2} + \frac{1}{20n} + \sigma_f \sigma^n (c_j + \sigma_j) = \frac{1}{2} + 10n\sigma^n + \frac{1}{20n} = K$$

This proves that $\mathcal{I}_2$ has a solution.

We now prove that if we have a solution to $\mathcal{I}_2$, then we have a solution to $\mathcal{I}_1$. Say we have a mapping of services with latency at most K. Note that $\sigma^n < \sigma < 1$. Note also that $0.7 > \sigma^n > \frac{1}{2}$ for all n, because the sequence $u_n = \left(1 - \frac{1}{2n}\right)^n$ is non-decreasing (and converges to $\frac{1}{\sqrt{(e)}}$, where e is the base of the natural logarithm). As a consequence, we have $K < 1/2 + 7n$.

We show that the plan necessarily is a fork-join. We make some preliminary observations:

- if one service C_i has no predecessor, then the latency is at least $c_i + \sigma \geq c_i \geq 8n > K$
- if service J has no predecessor, then the latency is at least $c_j + \sigma_j = 200n^2 > K$
- if service J is a direct successor of service F, then the latency is at least $c_f + \sigma_f(c_j + \sigma_j) \geq 10n > L$

So we know that F is a predecessor of all nodes and that the predecessors of J include F and at least one of the C_i. Assume (by contradiction) that we have exactly $k < n$ services C_i in the list of the predecessors of J. The latency obtained this way is at least L_k:

$$L_k = c_f + \sigma_f(\min(c_i) + \sigma^k(c_j + \sigma_j) \geq \frac{1}{20n} + \frac{1}{20n}\left(8n + \sigma^k 200n^2\right)$$

We derive $L_k - K \geq 10n(\sigma^k - \sigma^n) - \frac{1}{10}$. But

$$\sigma^k - \sigma^n \geq \sigma^{n-1} - \sigma^n = \sigma^{n-1}(1-\sigma) \geq \sigma^n(1-\sigma) \geq \frac{1}{2}\frac{1}{2n} = \frac{1}{4n}$$

Hence $L_k - K \geq \frac{10n}{4n} - \frac{1}{10} > 0$, the desired contradiction. Hence J is a successor of all other services.

There remains to show that each service C_i is a direct successor of F to obtain the fork-join plan. But if two services C_i and C_j were serialized, the latency would be at least L', where

$$L' = c_f + \sigma_f(\min(c_i) + \sigma \min(c_i) + \sigma^n(c_j + \sigma_j)$$

We get $L' - K \geq \frac{1}{20n} 8n(1+\sigma) - \frac{1}{2}$. But $\sigma > \frac{3}{4}$ since $n \geq 2$ hence $L' - K > 0$, again a contradiction.

Now that we have the fork-join plan, we assume that the services C_i are fed in order λ_1 and the reception are done in order $n + 1 - \lambda_2$. For service C_i, the latency is at least $l(i) = c_f + \sigma_f(\lambda_1(i) + c_i + \sigma\lambda_2(i))$ and we must have $l(i) \leq \frac{1}{2} + \frac{1}{20n}$ for all i. We have the following case analysis:

- If for some i we had $\lambda_1(i) + \lambda_2(i) > A[i]$, then $\lambda_1(i) + \lambda_2(i) \geq A[i] + 1$ (because we deal with integers) and we would derive

  $$l(i) = c_f + \sigma_f(10n + \lambda_1(i) + \lambda_2(i) - A[i] + (\sigma - 1)\lambda_2(i))$$

  and $l(i) - \frac{1}{2} - \frac{1}{20n} \geq \sigma_f(1 - \frac{\lambda_2(i)}{2n}) > 0$, a contradiction.
- If for some i we had $\lambda_1(i) + \lambda_2(i) < A[i]$, then by symmetry we would have some i' such that $\lambda_1(i') + \lambda_2(i') > A[i']$. This is because $\sum_i \lambda_1(i) + \lambda_2(i) = n(n+1) = \sum_i A[i]$. Using i' we obtain a contradiction as above.
- We conclude that $\lambda_1(i) + \lambda_2(i) = A[i]$ for all i, hence a solution to $\mathcal{I}_1$.

This concludes the proof.

□

Proposition 14. *The problem* MINLATENCY-INORDER *without dependence constraints is NP-hard.*

Proof. We use the same reduction as for Proposition 13, because the optimal operation list fulfilled the constraints of the INORDER model.

□

Proposition 15. *The problem* MINLATENCY-OVERLAP *without dependence constraints is NP-hard.*

Proof. The reasoning for the proof of Proposition 11 can be applied to Proposition 13, because the optimal plans have the same structure. □

D.3 Problems on chains and forests

Proposition 16. *The problem* MINLATENCY *when restricting to plans whose execution graphs are linear chains is polynomial for all models.*

Proof. Let EG be an optimal chain for the latency. Suppose that C_i is the direct predecessor of C_j. We have

$$\begin{array}{rl} & 1 + c_i + \sigma_i + \sigma_i c_j \leq 1 + c_j + \sigma_j + \sigma_j c_i \\ \iff & \frac{1-\sigma_i}{1+c_i} \geq \frac{1-\sigma_j}{1+c_j} \end{array}$$

We obtain the following greedy algorithm : order the services by decreasing values $\frac{1-\sigma_i}{1+c_i}$. □

Proposition 17. *The problem* MINLATENCY *when restricting to plans whose execution graphs are forests is NP-hard for all models.*

Proof. We consider the associated decision problem: given a latency K, is there a plan with an execution graph that is a forest of latency less than K? We have proved in Subsection 5.1 that for an execution graph that is a forest we can compute the optimal operation list for the latency in polynomial time. That means that the problem is in NP.

The NP-completeness is obtained by reduction from 2-Partition [18]. Let $\mathcal{I}_1$ be an instance from 2-Partition: given an integer set $X = \{x_1, ..., x_n\}$, does there exist a subset I such that $\sum_{x_i \in I} x_i = \frac{1}{2}\sum_{x_j \in X} x_j$? Let $x_M = \max_{x_i \in X}\{x_i\}$, $S = \sum_{x_j \in X} x_j$, $\beta = \frac{A-S}{2A+S}$ and $A > \frac{4}{3}n3^n\beta^n \times x_M^3$. We construct an instance $\mathcal{I}_2$ with $n+1$ services such that:

- $\forall i \leq n, c_i = \frac{x_i}{A}$
- $\forall i \leq n, \sigma_i = 1 - \frac{x_i}{A} + \beta\frac{x_i^2}{A^2}$
- $c_{n+1} = \frac{2A+S}{2A-2S}$
- $\sigma_{n+1} = 1$
- $K = c_{n+1} - \frac{3S^2}{8A(A-S)} + \frac{n3^n\beta^n x_M^3}{A^3}$

The size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$.

We now show that $\mathcal{I}_1$ has a solution if and only if $\mathcal{I}_2$ has a solution. Suppose first that $\mathcal{I}_1$ has a solution I. We place the services of I in a chain, as predecessors of C_{n+1} in any order. The remaining services are placed in parallel without any predecessor. Their latency is smaller than 1 and $K > 1$. That means that $\mathcal{I}_2$ has a solution if and only if the latency L of c_{n+1} is smaller than K. Suppose that the services in I are placed in the order $C'_1, ..., C'_{k-1}$ along the chain, and let $C_{n+1} = C'_k$. We have:

$$\begin{array}{rcl} L & = & \sum_{i<k}\prod_{j<i}\sigma'_j c'_i + \prod_{j<k}\sigma'_j c_{n+1} \\ & \leq & \sum_{i<k}\frac{x'_i}{A}(1 - \sum_{j<i}\frac{x'_j}{A} + 3^n\beta^n(\frac{x_M}{A})^2) \\ & & + c_{n+1}(1 - \sum_{i<k}\frac{x'_i}{A} + \beta\sum_{i<k}(\frac{x'_i}{A})^2 + \sum_{i<k}(\frac{x'_i}{A})^2 + 2\sum_{i<j<k}\frac{x'_i x'_j}{A^2} + 3^n\beta^n\frac{x_M^3}{A^3}) \\ & \leq & c_{n+1} + \sum_{i<k}\frac{x'_i}{A}(1 - c_{n+1}) + \sum_{i<k}(\frac{x'_i}{A})^2 c_{n+1}(\beta+1) + \sum_{i<j<k}\frac{x'_i x'_j}{A^2}(2c_{n+1} - 1) + n3^n\beta^n\frac{x_M^3}{A^3} \\ & \leq & c_{n+1} + \sum_{i<k} x'_i(\frac{-3S}{2A(A-S)}) + \sum_{i<k} x'^2_i(\frac{3}{2A(A-S)}) + \sum_{i<j<k} x'_i x'_j(\frac{1}{A(A-S)}) + n3^n\beta^n\frac{x_M^3}{A^3} \\ & \leq & c_{n+1} + (\frac{3}{2A(A-S)})(-S\sum_{i<k} x'_i + \sum_{i<k} x'^2_i + 2\sum_{i<j<k} x'_i x'_j) + n3^n\beta^n\frac{x_M^3}{A^3} \\ & \leq & c_{n+1} + (\frac{3}{2A(A-S)})(\frac{S}{2} - \sum_{i<k} x'_i)^2 - (\frac{3}{2A(A-S)})\frac{S^2}{4} + n3^n\beta^n\frac{x_M^3}{A^3} \\ & \leq & K \end{array}$$

Then, the instance $\mathcal{I}_2$ has a solution.

Suppose now that $\mathcal{I}_2$ has a solution. Let L be the latency of C_{n+1} and I be its set of predecessors. The plan is a forest, which means that the services of I are chained. We prove as in the previous computation that

$$L \geq (K + (\frac{3}{2A(A-S)})(\frac{S}{2} - \sum_{i \in I} x'_i)^2) - 2n3^n\beta^n\frac{x_M^3}{A^3}$$

By hypothesis, we have $L \leq K$. Hence:

$$\begin{array}{ll} & (\frac{3}{2A(A-S)})(\frac{S}{2} - \sum_{i \in I} x'_i)^2) \leq 2n3^n\beta^n\frac{x_M^3}{A^3} \\ \iff & (\frac{S}{2} - \sum_{i \in I} x'_i)^2 \leq 4n3^n\beta^n\frac{x_M^3(A-S)}{3A^2} \end{array}$$

By construction of A, we have $4n3^n\beta^n\frac{x_M^3(A-S)}{3A^2} \leq 4n3^n\beta^n\frac{x_M^3}{3A} < 1$. This proves that $(\frac{S}{2} - \sum_{i \in I} x'_i)^2 = 0$. Then I is a valid solution for the instance $\mathcal{I}_1$. This concludes the proof. □