# Filter placement on a pipelined architecture

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

## Abstract

*In this paper, we explore the problem of mapping filtering query services on chains of heterogeneous processors. Two important optimization criteria should be considered in such a framework. The period, which is the inverse of the throughput, measures the rate at which data sets can enter the system. The latency measures the response time of the system in order to process one single data set entirely. We provide a comprehensive set of complexity results for period and latency optimization problems, with proportional or arbitrary computation costs, and without or with communication costs. We present polynomial algorithms for problems whose dependence graph is a linear chain (hence a fixed ordering of the filtering services). For independent services, the problems are all NP-complete except latency minimization with proportional computation costs, which was shown polynomial in [6].*

## 1 Introduction

We consider the problem of mapping a set of filtering query services onto a heterogeneous array of processors. This work is based upon a recent paper by Srivastava, Munagala and Widom [14]. We extend the results of [14] along several important directions, including the answer to an open question stated in their paper.

Filtering query services operate on a continuous stream of data-sets. They resemble classical pipelined workflow graphs [8, 18, 22]. A workflow graph contains several *nodes*, and these nodes are connected to each other using first-in-first-out *channels*. Data is input into the graph using input channel(s) and the outputs are produced on the output channel(s). The goal is to map each node onto some processor so as to optimize some scheduling objective. Since data continually flows through these applications, typical objectives of the scheduler are *throughput* maximization (or equivalently *period* minimization, where the period is defined as the inverse of the throughput) and/or *latency* (also called response time) minimization [19, 20, 4, 21].

Filtering services are workflow nodes with the additional property that they can *filter* their input data by a certain amount, according to their *selectivity*. Consider a service $C_i$ with selectivity $\sigma_i$: if the incoming data is of size $\delta$, then the outgoing data will be of size $\delta \times \sigma_i$. The initial data is of size $\delta_0$. We see that the data is shrunk (hence the term "filter") when $\sigma_i < 1$ but it can also be expanded if $\sigma_i > 1$. The main application of filtering services is query optimization over web services [14, 15, 6], an increasingly important application with the advent of Web Service Management Systems [9, 12]. Note that the approach also applies to general data streams [2] and to database predicate processing [7, 11].

Srivastava, Munagala and Widow [14] consider the following problem: given (i) a set of $n$ independent filtering services, or simply *services* $C_1$ to $C_n$, and (ii) a linear array of $m$ heterogeneous processors $S_1$ to $S_m$, how to map the services onto the processors so as to minimize the latency, i.e. the total time needed by each data set to traverse all the services. In the framework of [6], the ordering of the processors along the chain is fixed. On the contrary, a service can be mapped on any processor. Hence, we look for a permutation $\pi$ of the services and for an allocation function $\mathsf{alloc}$ which maps these services onto the processors while respecting the order induced by the permutation. The predecessors of a service are all services that are mapped *before* that service, be it on previous processors or on the same processor. Analytically, the predecessors of $C_i$ are $C_j$ where $\pi(j) < \pi(i)$.

The cost of executing a service depends (i) upon the processor it is assigned to and (ii) upon the combined selectivity of its predecessors. As for (i), each service has a different cost on each processor: the execution of service $C_i$ on processor $S_u$ takes time $C_{i,u}$. These costs may be *arbitrary*, or in some cases they take the form $C_{i,u} = \frac{\mathsf{w}_i}{\mathsf{s}_u}$: they are *proportional* to an amount of work $\mathsf{w}_i$ required by the service, and inversely proportional to the speed $\mathsf{s}_u$ of the processor. In the latter case, two different services have the same execution time ratios on two different processors; proportional costs are also called *uniform* costs in the scheduling literature [5]. As for (ii), the cost of executing a service is modified by all its predecessors: if $\mathsf{Pred}(C_i)$ denotes the set of all predecessors of $C_i$ in the mapping, then its execution cost on processor $S_u$ is $\left(\prod_{C_j \in \mathsf{Pred}(C_i)} \sigma_j\right) \times c_{i,u}$. Basically, we see there are two ways to decrease the final cost of a service: (i) map it on a server that executes it fast; and (ii) map it as a successor of services with small selectivities.

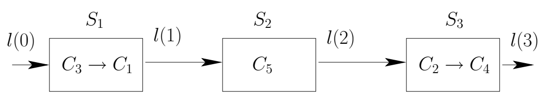


**Figure 1. Example.**

In the example of Figure 1, the permutation $\pi$ is equal to $[3, 1, 5, 2, 4]$ and the allocation function given by $\mathsf{alloc}(1) = \mathsf{alloc}(3) = 1$, $\mathsf{alloc}(5) = 2$ and $\mathsf{alloc}(2) = \mathsf{alloc}(4) = 3$. For instance, the execution cost of $C_5$ is $\sigma_3\sigma_1 C_{5,2}$.

We can finally state the problem addressed by Srivastava, Munagala and Widow [14]: they aim at minimizing the latency, defined as the sum of the latter execution costs for all services (to be precise, there are also communication costs, see Section 2 for a detailed analytical formulation). They show that with *proportional* costs the latency minimization problem can be solved via a (polynomial) dynamic programming algorithm, but with *arbitrary* costs they leave the complexity as an open question. One major contribution of this paper is to show the NP-completeness of the latency minimization problem with arbitrary costs, thereby assessing the additional difficulty induced by non-uniform machines.

We extend the results of [14] in another important direction: we investigate the situation where services are no longer independent but instead where they are ordered along a linear chain of precedence. In this case, both services and processors are arranged according to a fixed prescribed order. This problem is the extension of the well known chains-to-chains problem [13] to the case where nodes have a selectivity, and it has a great practical significance because linear dependence chains are ubiquitous in workflow applications (see [16, 17] and the references therein).

The last extension relates to the period minimization problem, which is only alluded to in [14]. In fact it is stated as a load-balancing problem which we reformulate as follows: the objective is to minimize the maximum cycle-time of a processor. Here, the cycle-time of a processor is the sum of the costs of all services assigned to it. In other words, the latency is the sum of the cycle-times, while the period is their maximum. The problem is easily shown NP-hard for proportional costs (use two identical processors, and a straightforward reduction from the 2-Partition problem, see Theorem 1). The problem becomes less straightforward when services are ordered along a linear chain rather than being independent, and we provide a comprehensive complexity analysis with arbitrary or proportional costs.

Altogether, the main objective in this paper is to assess the complexity of the different variants of these period and latency minimization problems, with independent or linearly ordered services, with arbitrary or proportional costs, and with or without communication costs between processors. In particular, we show the polynomial complexity of all problem instances with ordered services. For independent services, the major result is the NP-completeness of latency minimization of latency with arbitrary costs and no communication costs.

The rest of the paper is organized as follows. In Section 2 we detail the framework. Section 3 is devoted to a survey of related work. Section 4 presents the complexity results concerning period minimization, and Section 5 is the counterpart for latency minimization. We provide some final remarks and future research directions in Section 6.

## 2 Framework

This section is devoted to a precise statement of the optimization problems that we consider.

The application consists in a set of $n$ services $C_1, ..., C_n$, where service $C_i$ is characterized by its selectivity $\sigma_i$. Consecutive data sets must be processed by each service. For each data set, an initial set of tuples is input to the first service; the final result is the (shrunk or expanded) set of tuples output from the last service.

The basic network topology that we consider is a linear chain of $m$ processors $S_1, ..., S_m$. Processor $S_u$ can only send data to $S_{u+1}$, for $1 \leq u \leq m-1$. This corresponds to a hierarchical network, where $S_1$ is the processor acquiring the data. Processor $S_m$ is at the top of the hierarchy, and outputs the tuples of each data set that were processed through all services.

We define below the different variants of the problem.

### 2.1 Service ordering

The more flexible problem is the case with no precedence constraints as in [14]: services are independent, and they can be applied on the data in any order. This problem is called *Free* ordering.

We also consider the case in which services are totally ordered along a linear dependence chain: there is a precedence constraint from $C_i$ to $C_{i+1}$ for $1 \leq i \leq n-1$. We note this problem as the *Ordered* instance.

### 2.2 Service costs

The execution of service $C_i$ on processor $S_u$ takes a time $C_{i,u}$. In the most general instance, these costs are *Arbitrary*.

However, for uniform machines, costs $C_{i,u}$ take the form $C_{i,u} = \frac{\mathsf{w}_i}{\mathsf{s}_u}$, where $\mathsf{w}_i$ is the amount of work required by the service, and $\mathsf{s}_u$ is the speed of processor $S_u$. We refer to such costs as *Proportional* costs.

### 2.3 Communication costs

We consider two models of platforms, with or without communication costs. For the model with communication costs, we use the same framework as [14]. They consider a model without computation/communication overlap: a server cannot compute some data and communicate with another server at the same time, these actions are serialized.

Let $\text{ALLOC}_u$ denote the set of services that are mapped on processor $S_u$. Let $\text{PRED}_u$ be the set of services mapped on processors $S_v$ before $S_u$:

$$\text{PRED}_u = \{C_j \mid \exists v < u, \mathsf{alloc}(C_j) = S_v\}$$

Equivalently, $\text{PRED}_u = \bigcup_{v=1}^{u-1} \text{ALLOC}_v$. Finally, let $\text{UPTO}_u$ denote the set of services that are mapped before $S_u$, plus those mapped onto $S_u$:

$$\text{UPTO}_u = \text{PRED}_u \cup \text{ALLOC}_u$$

The communication cost between servers $S_u$ and $S_{u+1}$ is given by the value

$$C_{comm}(u) = l(u) \times \prod_{C_j \in \text{UPTO}_u} \sigma_j$$

where $l(u)$ is the inverse of the bandwidth of the link from $S_u$ to $S_{u+1}$. Indeed, the output of $S_u$ is filtered by all services mapped before $S_u$, and by those mapped on $S_u$, it is thus the set $\text{UPTO}_u$. We take into account the cost $C_{comm}(0)$ of input for processor $S_1$ and the cost $C_{comm}(m)$ of output for processor $S_m$. The corresponding bandwidths $l(0)$ and $l(m)$ corresponds to the communication links between the platform and the external world (the user).

The model with communication costs is denoted by *Cost* and the model without by *NoCost*.

### 2.4 Objective function

Different cost functions are considered. The period of the mapping is limited by the slowest (bottleneck) processor. The objective to minimize the period is denoted as PER. Another objective is minimize the latency, that is the sum of the costs incurred by all services in the mapping (objective denoted as LAT). This corresponds to the time required for one data set to be processed by all the services.

Formally, we define the period and the latency using $\text{ALLOC}_u$, $\text{PRED}_u$, and $\text{UPTO}_u$, which correspond to the sets of services mapped on, before, and up to $S_u$ respectively. Note that $\text{PRED}_u \subset \mathsf{Pred}(C_j) \subset \text{UPTO}_u$ for each service $C_j \in \text{ALLOC}_u$: $\mathsf{Pred}(C_j)$, the predecessors of $C_j$, are all services mapped onto preceding processors, plus those mapped on $S_u$ before $C_j$. To simplify notations, suppose that services in $\text{ALLOC}_u$ are placed in order $C_1 \to C_2 \to ... \to C_k$. We obtain the following computation cost $C_{comp}(u)$ for processor $S_u$:

$$C_{comp}(u) = \left( \prod_{C_j \in \text{PRED}_u} \sigma_j \right) \sum_{i=1}^{k} \left( \prod_{q=1}^{i-1} \sigma_q \right) \times C_{i,u}$$

For a model without communication cost, $C_{comp}(u)$ is the cycle-time of processor $S_u$. The period is

$$T_{\mathsf{period}} = \max_{1 \leq u \leq m} \{C_{comp}(u)\}$$

and the latency is

$$T_{\mathsf{latency}} = \sum_{u=1}^{m} C_{comp}(u)$$

For a model with communication cost, we need to take into account $C_{comm}(u)$. Since we consider a model with no overlap, computations and communications are serialized and we obtain a period

$$T_{\mathsf{period}} = \max_{1 \leq u \leq m} \{C_{comm}(u-1) + C_{comp}(u) + C_{comm}(u)\}$$

and a latency

$$T_{\mathsf{latency}} = C_{comm}(0) + \sum_{u=1}^{m} (C_{comp}(u) + C_{comm}(u))$$

### 2.5 Taxonomy of problems

We denote each problem by $XYZ - Obj$, where:

- $X = O|F$ denotes the service ordering (*Ordered* or *Free*);
- $Y = P|A$ denotes the service costs (*Proportional* or *Arbitrary*);
- $Z = C|N$ denotes the communication costs (*Cost* or *NoCost*);
- $Obj =$ PER|LAT denotes the objective function.

For instance, FAC-LAT is the problem of minimizing the latency with no precedence constraints between services, arbitrary service costs, and with communication costs.

In addition, * denotes any instance of the problem, thus F**-LAT denotes the problem of minimizing the latency with no precedence constraints between services, for any kind of service and communication costs.

## 3 Related work

As stated in the introduction, the main reference for this work is the paper by Srivastava, Munagala and Widow [14]. In fact, we utilize the very same application framework and execution model as those of [14]. Therefore, we refer the reader to [14], and to the many references therein, for further motivations of this study. In a word, applications include all domains where clients need to query multiple web services simultaneously, in a transparent and integrated fashion. As stated in Section 1, we extend their study in several directions.

Papers [15, 6, 3] deal with the same line of problems. They also consider filtering services, but in a very different framework. They investigate the mapping of filtering workflow applications with arbitrary dependence graphs onto fully connected platforms (the interconnection graph is a clique, and there is no prescribed ordering of the processors). They restrict to one-to-one mappings: a processor can only execute a single service. Several complexity results are established for period and latency optimization with these hypotheses.

In [1], the authors consider a set of jobs characterized by a certain success probability and a reward. The resulting problem is similar to a filtering workflow problem, but they maximize the reward while we minimize the cost. They present a polynomial algorithm in the case of a single server, and they prove that the problem becomes NP-complete when considering 2 servers.

Several papers aim at mapping applications whose dependence graph is a linear pipeline: see [16, 17] for homogeneous platforms, and [4] for heterogeneous platforms. These papers do not deal with filtering services (in other words, each service has a selectivity equal to 1). Finally, please refer to [3] for related work on mapping workflows whose graphs can be arbitrary DAGs (Directed Acyclic Graphs).

## 4 Period minimization

In this section we prove the NP-completeness of problems F**-PER (all problems with free ordering), and we present a polynomial algorithm for problems O**-PER (all problems with fixed ordering).

### 4.1 Free ordering

**Theorem 1.** *All problems F**-*PER *are NP-hard.*

**Proof**. We show that FPN-PER is NP-hard. All other problems are more difficult instances since *Proportional* is a particular case of *Arbitrary*, and *NoCost* a particular case of *Cost*.

The proof is straightforward. Consider the associated decision problem: given a period $K$, is there a mapping whose period does not exceed $K$? The problem is obviously in NP: given a period and a mapping, it is easy to check in polynomial time whether it is valid or not. The NP-completeness is obtained by reduction from 2-PARTITION [10]. Let $\mathcal{I}_1$ be an instance of 2-PARTITION: given a set $X = \{x_1, ..., x_n\}$, does there exist a subset $I$ such that $\sum_{x_i \in I} x_i = \frac{1}{2} \sum_{x_j \in X} x_j$? We construct the instance $\mathcal{I}_2$ with $n$ services and 2 servers such that:

- $\forall 1 \leq i \leq n, \sigma_i = 1$
- $\forall 1 \leq i \leq n, \mathsf{w}_i = x_i$
- $s_1 = s_2 = 1$
- $K = \frac{1}{2} \sum_{x_j \in X} x_j$

The size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$. Suppose that $\mathcal{I}_1$ has a solution $I$. We construct $\mathsf{alloc}$ such that: $\forall i, \mathsf{alloc}(i) = 1 \iff x_i \in I$. Then, the period of the mapping is $P = \max\{\sum_{x_i \in I} x_i, \sum_{x_i \notin I} x_i\}$, that means $P = K$. that means $\mathcal{I}_2$ has a solution. Suppose now that $\mathcal{I}_2$ has a solution. Let $I = \{x_i | \mathsf{alloc}(C_i) = S_1\}$. By hypothesis, we have $\sum_{x_i \in I} x_i \leq K$ and $\sum_{x_i \notin I} x_i = 2K - \sum_{x_i \in I} x_i \leq K$. We can conclude that $\sum_{x_i \in I} x_i = \frac{1}{2} \sum_{x_j \in X} x_j$. Then, $\mathcal{I}_1$ has a solution. This concludes the proof. □

### 4.2 Fixed ordering

**Data**: $n$ services of selectivities $\sigma_1, ..., \sigma_n$,
$m$ servers with a matrix of costs $C$, and
a vector of communication costs $l$
**Result**: a mapping $G$ optimizing the latency

$P(0,1) = l(m-1) + l(m)$;
**for** $j = 2$ **to** $m$ **do**
  $P(0,j) =$
  $\max\{l(m-j) + l(m-j+1), P(0,j-1)\}$;
**end**
**for** $i = 1$ **to** $n$ **do**
  $P(i,1) = l(m-1) + C_{n-i+1,m}+$
   $\sigma_{n-i+1}(P(i-1,1) - l(m-1))$;
  $\forall 1 \leq k \leq i,\ \mathsf{alloc}(i,1,n-k+1) = m$;
**end**
**for** $j = 2$ **to** $m$ **do**
  **for** $i = 1$ **to** $n$ **do**
    $\forall\ 0 \leq r \leq i,\ f(r) = \max\{l(m-j)+$
     $\sum_{q=1}^{r} \prod_{p=1}^{q-1} \sigma_{n-i+p} C_{n-i+q,m-j+1}+$
     $\prod_{p=1}^{r} \sigma_{n-i+p} l(m-j+1),$
     $\prod_{p=1}^{r} \sigma_{n-i+p} P(i-r,j-1)\}$;
    $k = argmin_{1 \leq r \leq i}\{f(r)\}$;
    $P(i,j) = f(k)$;
    $\forall 1 \leq q \leq k,$
     $\mathsf{alloc}(i,j,n-i+q) = m-j+1$;
    $\forall n-i < q < n-i+k,$
     $\mathsf{alloc}(i,j,q) = \mathsf{alloc}(i-k,j-1,q)$
  **end**
**end**

**Algorithm 1**: Optimal algorithm for OAC-PER.

**Theorem 2.** *Algorithm 1 computes the optimal mapping for problem OAC-*PER *in time* $O(m \times n^3)$.

**Proof**. Let $\mathcal{I}$ be an instance of OAC-PER. We prove by induction that for any pair $(i,j)$, the value $P(i,j)$ returned by Algorithm 1 is the optimal period on the instance $\mathcal{I}_{i,j}$ restricted to the last $i$ services and the last $j$ servers. Moreover, $\mathsf{alloc}(i,j,.)$ is the corresponding allocation function.

First, we compute the values $P(0,j)$ and $P(i,1)$ for $1 \leq j \leq m$ and $1 \leq i \leq n$. In these cases, there is only one possible mapping: for $P(0,j)$, there are no services to map; for $P(i,1)$, all services must be mapped onto the last server. Thus the computed period is optimal.

Now we consider the placement of the remaining services. Suppose that for all $j' < j$ and for all $i$, $P(i,j')$ is optimal. Then we show that $P(i,j)$ also is optimal. We define, for all $0 \leq r \leq i$, $f(r)$ as the period obtained by placing the $r$ first services on server $m-j+1$ and the other services optimally onto the next servers. We prove that the minimum of the values $f(r)$ is the optimal value for $P(i,j)$. Let $\mathsf{alloc}^*$ be an allocation of the last $i$ services on the last $j$ servers and $P^*$ be the period of this mapping. Let $S = \{i \mid \mathsf{alloc}^*(i) = m-j+1\}$, and $k = |S|$. Let $P'$ be the period on $\mathsf{alloc}^*$ for the last $i-k$ services on the last $j-1$ servers. By the hypothesis, $P' \geq P(i-k,j-1)$ and

$$\begin{aligned} P(i,j) \leq & \max\{l(j) + \textstyle\sum_{i' \in S} \prod_{q \in S, q < i'} \sigma_q \times C_{i',m-j+1} \\ & + \textstyle\prod_{q \in S} \sigma_q l(j+1), \prod_{q \in S} \sigma_q P(i-k,j-1)\} \\ \leq & \max\{l(j) + \textstyle\sum_{i' \in S} \prod_{q \in S, q < i'} \sigma_q \times C_{i',m-j+1} \\ & + \textstyle\prod_{q \in S} \sigma_q l(j+1), \prod_{q \in S} \sigma_q P'\} \\ \leq & P^* \end{aligned}$$

Since this is true for any mapping leading to a period $P^*$, $P(i,j)$ is the optimal period. We can conclude that $P(n,m)$ is the optimal period for instance $\mathcal{I}$. □

**Corollary 1.** *Problems O**-*PER *have polynomial complexity.*

**Proof**. The most difficult problem of O**-PER is OAC-PER, which is polynomial due to Theorem 2. □

## 5 Latency minimization

In this section, we present a polynomial algorithm for problems O**-LAT (fixed ordering) and we prove the NP-completeness of problems FA*-LAT. Recall that problems FP*-LAT are showed to be polynomial in [14]. With arbitrary costs instead of proportional costs, the problem becomes NP-hard, even in the absence of communications.

### 5.1 Fixed ordering

We derive an optimal algorithm for problems OAN-LAT and OAC-LAT. The algorithm for OAN-LAT (without communications) is presented only because it is simpler to understand than the algorithm for OAC-LAT (with communications). The complexity is the same for both cases.

**Theorem 3.** *Algorithm 2 computes the optimal mapping for problem OAN-*LAT *in time* $O(n^3 m)$.

**Proof**. Let $\mathcal{I}$ be an instance of OAN-LAT. We prove by induction that for any pair $(i,j)$, the value $L(i,j)$ returned by Algorithm 2 is the optimal latency on the instance $\mathcal{I}_{i,j}$ restricted to the last $i$ services and the last

**Data**: $n$ services of selectivities $\sigma_1, ..., \sigma_n \leq 1$ and $m$ servers with a matrix of costs $C$
**Result**: a mapping $G$ optimizing the latency

**for** $j = 1$ **to** $m$ **do**
  $L(0, j) = 0;$
**end**
**for** $i = 1$ **to** $n$ **do**
  $L(i, 1) = C_{n-i+1,m} + \sigma_{n-i+1} L(i-1, 1);$
  $\forall 1 \leq k \leq i,\ \mathsf{alloc}(i, 1, n-k+1) = m;$
**end**
**for** $j = 2$ **to** $m$ **do**
  **for** $i = 1$ **to** $n$ **do**
    $\forall\ 0 \leq l \leq i,\ f(l) = \sum_{i'=1}^{l} \left(\prod_{q=1}^{i'-1} \sigma_{n-i+q}\right) C_{n-i+i',m-j+1} + \left(\prod_{q=1}^{l} \sigma_{n-i+q}\right) L(i-l, j-1);$
    $k = argmin_{0 \leq l \leq i}\{f(l)\};$
    $L(i, j) = f(k);$
    $\forall 1 \leq q \leq k,$
      $\mathsf{alloc}(i, j, n-i+q) = m-j+1;$
    $\forall k < q \leq i, \quad \mathsf{alloc}(i, j, n-i+q) = \mathsf{alloc}(i-k, j-1, n-i+q);$
  **end**
**end**

**Algorithm 2**: Optimal algorithm for OAN-LAT.

$j$ servers. Moreover, $\mathsf{alloc}(i, j, .)$ is the corresponding allocation function.

First, we compute the values $P(0, j)$ and $P(i, 1)$ for $1 \leq j \leq m$ and $1 \leq i \leq n$. In these cases, there is only one possible mapping: either there are no services to map, or all services must be mapped onto the last server. Thus the computed latency is optimal.

Suppose that for all $j' < j$ and for all $1 \leq i \leq n$, $L(i, j')$ is optimal. Then we prove that for all $i$, $L(i, j)$ also is the optimal latency. Let $\mathsf{alloc}^*$ be an allocation of the last $i$ services on the last $j$ servers and $L^*$ be the latency of this mapping. Let $S = \{i \mid \mathsf{alloc}^*(i) = m-j+1\}$, and $k = |S|$. Let $L'$ be the latency on $\mathsf{alloc}^*$ for the last $i-k$ services on the last $j-1$ servers. By hypothesis, $L' \geq L(i-k, j-1)$, and

$$
\begin{aligned}
L(i,j) &\leq f(k) \\
&\leq \sum_{i' \in S} \prod_{q \in S, q < i'} \sigma_q \times C_{i',m-j+1} \\
&\quad + (\prod_{q \in S} \sigma_q) L(i-k, j-1) \\
&\leq \sum_{i' \in S} \prod_{q \in S, q < i'} \sigma_q \times C_{i',m-j+1} + (\prod_{q \in S} \sigma_q) L' \\
&\leq L^*
\end{aligned}
$$

Since this is true for any mapping leading to a latency $L^*$, $L(i, j)$ is the optimal latency. We can conclude that $L(n, m)$ is the optimal latency for instance $\mathcal{I}$. □

**Data**: $n$ services of selectivities $\sigma_1, ..., \sigma_n$, $m$ servers with a matrix of costs $C$ and a vector of communication cost $l$
**Result**: a mapping $G$ optimizing the latency

**for** $j = 1$ **to** $m$ **do**
  $L(0, j) = \sum_{j'=m-j+1}^{m} l(j');$
**end**
**for** $i = 1$ **to** $n$ **do**
  $L(i, 1) = l(m-1) + C_{n-i+1,m} + \sigma_{n-j+1}(L(i-1, 1) - l(m-1));$
  $\forall 1 \leq k \leq i,\ \mathsf{alloc}(i, 1, n-k+1) = m;$
**end**
**for** $j = 2$ **to** $m$ **do**
  **for** $i = 1$ **to** $n$ **do**
    $\forall\ 0 \leq l \leq i, \quad f(l) = l(m-j+1) + \sum_{i'=1}^{l} \left(\prod_{q=1}^{i'-1} \sigma_{n-i+q}\right) C_{n-i+i',m-j+1} + \left(\prod_{q=1}^{l} \sigma_{n-i+q}\right) L(i-l, j-1);$
    $k = argmin_{0 \leq l \leq i}\{f(l)\};$
    $L(i, j) = f(k);$
    $\forall 1 \leq q \leq k,$
      $\mathsf{alloc}(i, j, n-i+q) = m-j+1;$
    $\forall k < q \leq i, \quad \mathsf{alloc}(i, j, n-i+q) = \mathsf{alloc}(i-k, j-1, n-i+q);$
  **end**
**end**

**Algorithm 3**: Optimal algorithm for OAC-LAT.

**Theorem 4.** *Algorithm 3 compute the optimal mapping for problem OAC-*LAT *in time $O(n^3 m)$.*

**Proof**. The proof is similar to that for Theorem 3. We merely add communication costs in the equations. □

## 5.2 Free ordering

This section is devoted to assessing the most difficult complexity result of this paper: the NP-completeness of latency minimization with arbitrary costs, even without taking communications into account. This important result closes the open question raised in [14].

**Theorem 5.** *Problem FAN-*LAT *is NP-complete.*

**Proof**. We consider the associated decision problem: given a latency $K$, is there a mapping of latency less than $K$? The problem is obviously in NP: given a latency and a mapping, it is easy to check in polynomial time whether it is valid or not.

The NP-completeness is obtained by reduction from 2-PARTITION [10], as in Theorem 1, but the reduction is quite involved. Let $\mathcal{I}_1$ be an instance from 2-PARTITION: given a set $X = \{x_1, ..., x_n\}$, does it 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 $A > \frac{4}{3} n 3^n \times x_M^3$. We construct the instance $\mathcal{I}_2$ with $n+1$ services and 3 servers such that:

- $\forall i \leq n, C_{i,1} = \frac{x_i}{A}$
- $\forall i \leq n, C_{i,2} = 3\left(\frac{3A}{A-x_M}\right)^n$
- $\forall i \leq n, C_{i,3} = 0$
- $\forall i \leq n, \sigma_i = 1 - \frac{x_i}{A} + \beta \frac{x_i^2}{A^2}$
- $C_{n+1,1} = C_{n+1,3} = 3\left(\frac{3A}{A-x_M}\right)^n$
- $C_{n+1,2} = \frac{2A+S}{2A-2S}$
- $\sigma_{n+1} = 1$
- $K = C_{n+1,2} - \frac{3S^2}{8A(A-S)} + \frac{n3^n\beta^n x_M^3}{A^3}$

The size of $\mathcal{I}_2$ is polynomial in the size of $\mathcal{I}_1$: the greatest value in $\mathcal{I}_2$ is $A$ and $\log(A)$ is linear in $n$.

Suppose that $\mathcal{I}_1$ has a solution $I$. We place the services $C_i$ with $i \in I$ in any order as a linear chain on server $S_1$. Then, $C_{n+1}$ is placed on $S_2$, and finally the remaining services are placed on $S_3$. The cost of the services on $S_3$ is null; that means that the latency $L$ of the system is the latency of $C_{n+1}$. Let $k = |I|$, and for $1 \leq i \leq k$, let $c'_i$ be the cost of the $i$-th service of $I$ on the chain on server $S_1$, and let $\sigma'_i$ be its selectivity.

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

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

Suppose now that $\mathcal{I}_2$ has a solution. By construction of $C_{n+1,1}$ and $C_{n+1,3}$, we can see that the service $C_{n+1}$ has to be mapped onto $S_2$ in the solution of $\mathcal{I}_2$. Similary, there can be no service $C_i$ $(i \leq n)$ on $S_2$. Let $L$ be the latency of $C_{n+1}$ and $I$ be its set of predecessor. Suppose that there is a service $C_i$ with $i \in I$ on $S_2$, we have the latency $L_i$ of $C_i$ such that

$$
\begin{aligned}
L_i & \geq 3\left(\frac{A}{A-x_M}\right)^n \times \prod_{i\leq n} \sigma_i \\
& > 3 \\
& > K
\end{aligned}
$$

This proves that all the services of $I$ are mapped on $S_1$. We prove as in the previous computation that

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

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. □

**Corollary 2.** *Problem FAC-LAT is NP-complete.*

# 6 Conclusion

In this paper, we have studied the problem of mapping filtering services onto a linear array of heterogeneous processors. We have assessed the complexity of this problem for the optimization of two different criteria, the period and the latency. The following table summarizes the complexity of all problem instances:

| model | PER | LAT |
|---|---|---|
| O** | Polynomial | Polynomial |
| FP* | NP-complete | Polynomial |
| FA* | NP-complete | NP-complete |

We point out that the introduction of communication costs never changes the complexity of a given problem. We have presented new polynomial algorithms for all polynomial problem instances, except for problems FP*-LAT which were solved in [14].

As future work, it would be very interesting to derive approximation algorithms and lower bounds for all NP-hard instances. Also, allowing some services to be replicated would allow to decrease the period of the mappings, while data-parallelizing some other services would allow to decrease both period and latency. To the best of our knowledge, such extensions, which are well-known and widely used in the context of classical pipelined workflows have never been addressed for filtering services. This is an interesting but algorithmically challenging direction to explore.

## Acknowledgment

We thank the reviewers for their comments and suggestions. This work was supported in part by the ANR StochaGrid project.

## References


[1] 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`.

[2] 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*, pages 407–418. ACM Press, 2004.

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

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

[5] P. Brucker. *Scheduling Algorithms*. Springer-Verlag, 2004.

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

[7] S. Chaudhuri and K. Shim. Optimization of queries with user-defined predicates. *ACM Trans. Database Systems*, 24(2):177–228, 1999.

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

[9] 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`.

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

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

[12] M. Ouzzani and A. Bouguettaya. Query processing and optimization on the web. *Distributed and Parallel Databases*, 15(3):187–218, 2004.

[13] A. Pinar and C. Aykanat. Fast optimal load balancing algorithms for 1D partitioning. *J. Parallel Distributed Computing*, 64(8):974–996, 2004.

[14] U. Srivastava, K. Munagala, and J. Widom. Operator placement for in-network stream query processing. In *PODS'05: Proceedings of the 24th ACM Symposium on Principles of Database Systems*, pages 250–258. ACM Press, 2005.

[15] 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*, pages 355–366. VLDB Endowment, 2006.

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

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

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

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

[20] N. Vydyanathan, U. Catalyurek, T. Kurc, P. Saddayappan, and J. Saltz. Optimizing latency and throughput of application workflows on clusters. Research Report OSU-CISRC-4/08-TR17, Ohio State University, Columbus, OH, Apr. 2008. Available at `ftp://ftp.cse.ohio-state.edu/pub/tech-report/2007`.

[21] Q. Wu, J. Gao, M. Zhu, N. Rao, J. Huang, and S. Iyengar. On optimal resource utilization for distributed remote visualization. *IEEE Trans. Computers*, 57(1):55–68, 2008.

[22] 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.