Mapping filtering streaming applications with communication costs

Anne Benoit, Fanny Dufossé, Yves Robert
GRAAL team, LIP, ENS Lyon, France

Kunal Agrawal, MIT, USA

Scheduling for large-scale systems in Knoxville
May 14, 2009

Introduction and motivation

- Scheduling workshop: schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow): several data sets are processed by a set of services (or tasks)
  - Selectivity: services filter data
  - Dependencies: some freedom to order services
- Target platform
  - fully homogeneous, one-to-one mapping
  - different communication models (overlap, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)

Scheduling *filtering streaming applications* onto *homogeneous platforms with communication costs*

Introduction and motivation

- Scheduling workshop: schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow): several data sets are processed by a set of services (or tasks)
  - Selectivity: services filter data
  - Dependencies: some freedom to order services
- Target platform
  - fully homogeneous, one-to-one mapping
  - different communication models (overlap, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)

Scheduling *filtering streaming applications* onto *homogeneous platforms with communication costs*

Introduction and motivation

- Scheduling workshop: schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow): several data sets are processed by a set of services (or tasks)
  - Selectivity: services filter data
  - Dependencies: some freedom to order services
- Target platform
  - fully homogeneous, one-to-one mapping
  - different communication models (overlap, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)

Scheduling *filtering streaming applications* onto *homogeneous platforms with communication costs*

Introduction and motivation

- Scheduling workshop: schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow): several data sets are processed by a set of services (or tasks)
  - Selectivity: services filter data
  - Dependencies: some freedom to order services
- Target platform
  - fully homogeneous, one-to-one mapping
  - different communication models (overlap, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)

Scheduling *filtering streaming applications* onto *homogeneous platforms with communication costs*

Introduction and motivation

- Scheduling workshop: schedule an application onto a computational platform, with some criteria to optimize
- Target application
  - Streaming application (workflow): several data sets are processed by a set of services (or tasks)
  - Selectivity: services filter data
  - Dependencies: some freedom to order services
- Target platform
  - fully homogeneous, one-to-one mapping
  - different communication models (overlap, one- vs multi-port)
- Optimization criteria
  - period (inverse of throughput) and latency (execution time)

Scheduling *filtering streaming applications* onto *homogeneous platforms with communication costs*

Motivation

- Yesterday's talks...
- Fanny – filters without communication costs: difficulty to find the optimal mapping and dependencies, everything NP-hard for heterogeneous platforms
  Restrict to homogeneous platforms
- Loïc – standard workflows with no filtering: given a mapping, difficulty to order communications in order to minimize period and/or latency
  Similar problem for filters

Motivation

- Yesterday's talks...
- Fanny – filters without communication costs: difficulty to find the optimal mapping and dependencies, everything NP-hard for heterogeneous platforms
  Restrict to homogeneous platforms
- Loïc – standard workflows with no filtering: given a mapping, difficulty to order communications in order to minimize period and/or latency
  Similar problem for filters

Motivation

- Yesterday's talks...
- Fanny – filters without communication costs: difficulty to find the optimal mapping and dependencies, everything NP-hard for heterogeneous platforms
  Restrict to homogeneous platforms
- Loïc – standard workflows with no filtering: given a mapping, difficulty to order communications in order to minimize period and/or latency
  Similar problem for filters

Motivation

- Yesterday's talks...
- Fanny – filters without communication costs: difficulty to find the optimal mapping and dependencies, everything NP-hard for heterogeneous platforms
  Restrict to homogeneous platforms
- Loïc – standard workflows with no filtering: given a mapping, difficulty to order communications in order to minimize period and/or latency
  Similar problem for filters

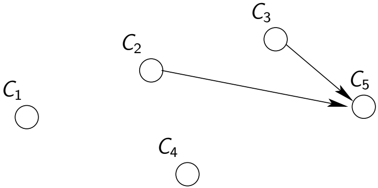
Motivation

- Yesterday's talks...
- Fanny – filters without communication costs: difficulty to find the optimal mapping and dependencies, everything NP-hard for heterogeneous platforms
  Restrict to homogeneous platforms
- Loïc – standard workflows with no filtering: given a mapping, difficulty to order communications in order to minimize period and/or latency
  Similar problem for filters

Outline

Framework: the application

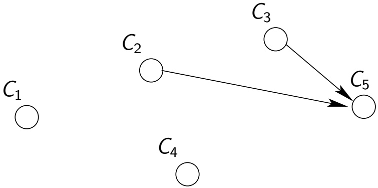
- Target application $\mathcal{A}$: set of services (or filters, or queries) linked by precedence constraints
- Streaming application: several data sets, each processed by every services
- Data communicated from one service to another



- $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ 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
- Service C_i: cost c_i and selectivity σ_i.

Framework: the application

- Target application $\mathcal{A}$: set of services (or filters, or queries) linked by precedence constraints
- Streaming application: several data sets, each processed by every services
- Data communicated from one service to another



- $\mathcal{A} = (\mathcal{F}, \mathcal{G})$ where $\mathcal{F} = \{C_1, C_2, \dots, C_n\}$ is the set of services and $\mathcal{G} \subset \mathcal{F} \times \mathcal{F}$ is the set of precedence constraints
- Service C_i: cost c_i and selectivity σ_i.

Framework: the platform

- Homogeneous platform with p servers (or processors)
- Identical server speed s: service cost does not depend upon the server it is mapped onto
- Servers interconnected by communication links of equal bandwidth b: cost $\frac{\delta}{b}$ for data of size δ
- δ_0: size of input data
- One-to-one mapping and identical servers: no mapping problems
- Impact of communication models on period and latency

Build an *execution plan*

Framework: the platform

- Homogeneous platform with p servers (or processors)
- Identical server speed s: service cost does not depend upon the server it is mapped onto
- Servers interconnected by communication links of equal bandwidth b: cost $\frac{\delta}{b}$ for data of size δ
- δ_0: size of input data
- One-to-one mapping and identical servers: no mapping problems
- Impact of communication models on period and latency

Build an *execution plan*

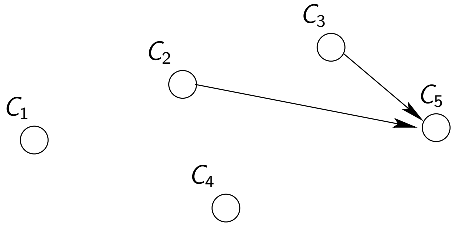
Framework: the platform

- Homogeneous platform with p servers (or processors)
- Identical server speed s: service cost does not depend upon the server it is mapped onto
- Servers interconnected by communication links of equal bandwidth b: cost $\frac{\delta}{b}$ for data of size δ
- δ_0: size of input data
- One-to-one mapping and identical servers: no mapping problems
- Impact of communication models on period and latency

Build an *execution plan*

Execution plan

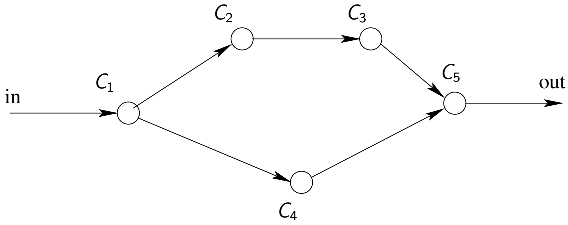
- *Plan* $PL = (EG, OL)$:
  - Execution graph $EG = (\mathcal{C}, \mathcal{E})$: all precedence relations in the mapping; nodes = services, arc $(C_i, C_j) \in \mathcal{E}$ if C_i precedes C_j in EG
    Ancest$_j(EG)$: ancestors of C_j in EG:
    $(C_i, C_j) \in \mathcal{G} \Longrightarrow C_i \in \text{Ancest}_j(EG)$



  - Operation list OL: captures the occurrence of each computation and each communication

Execution plan

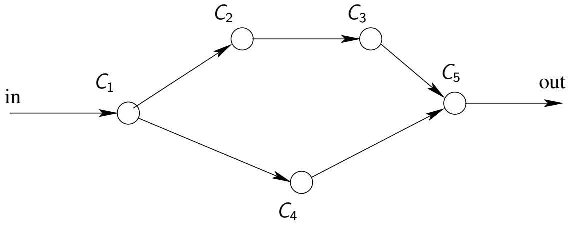
- *Plan* $PL = (EG, OL)$:
  - Execution graph $EG = (\mathcal{C}, \mathcal{E})$: all precedence relations in the mapping; nodes = services, arc $(C_i, C_j) \in \mathcal{E}$ if C_i precedes C_j in EG
    $\text{Ancest}_j(EG)$: ancestors of C_j in EG:
    $(C_i, C_j) \in \mathcal{G} \Longrightarrow C_i \in \text{Ancest}_j(EG)$



  - Operation list OL: captures the occurrence of each computation and each communication

Execution plan

- *Plan* $PL = (EG, OL)$:
  - Execution graph $EG = (\mathcal{C}, \mathcal{E})$: all precedence relations in the mapping; nodes = services, arc $(C_i, C_j) \in \mathcal{E}$ if C_i precedes C_j in EG
    Ancest$_j(EG)$: ancestors of C_j in EG:
    $(C_i, C_j) \in \mathcal{G} \Longrightarrow C_i \in \text{Ancest}_j(EG)$



  - Operation list OL: captures the occurrence of each computation and each communication

Service costs

- Lower bound of the time needed to receive input data from all the predecessors of C_k:

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

- Execution time of C_k on the server

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

- Outgoing communication lower bound

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

- Scaling: we can assume $\delta_0 = b = s = 1$ ($c_k \leftarrow \frac{b}{\delta_0} \cdot \frac{c_k}{s}$)

Service costs

- Lower bound of the time needed to receive input data from all the predecessors of C_k:

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

- Execution time of C_k on the server

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

- Outgoing communication lower bound

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

- Scaling: we can assume $\delta_0 = b = s = 1$ ($c_k \leftarrow \frac{b}{\delta_0} \cdot \frac{c_k}{s}$)

Communication models

- With overlap– full overlap of communications and computations: OVERLAP
  - A server can receive, compute and send (independent) data simultaneously
  - Multi-port communications: many incoming (resp. outgoing) communications can take place at the same time
  - Server: operate concurrently on different consecutive data sets
  - Execution time $C_{\text{exec}}(k) = \max\{C_{\text{in}}(k), C_{\text{comp}}(k), C_{\text{out}}(k)\}$
- Without overlap– communications and computations are sequential
  - $C_{\text{exec}}(k) = C_{\text{in}}(k) + C_{\text{comp}}(k) + C_{\text{out}}(k)$
  - Two variants: INORDER and OUTORDER
- Lower bound on period: $\mathcal{P} = \max_{1 \le k \le n} C_{\text{exec}}(k)$

Communication models

- With overlap– full overlap of communications and computations: OVERLAP
  - A server can receive, compute and send (independent) data simultaneously
  - Multi-port communications: many incoming (resp. outgoing) communications can take place at the same time
  - Server: operate concurrently on different consecutive data sets
  - Execution time $C_{exec}(k) = \max\{C_{in}(k), C_{comp}(k), C_{out}(k)\}$

- Without overlap– communications and computations are sequential
  - $C_{exec}(k) = C_{in}(k) + C_{comp}(k) + C_{out}(k)$
  - Two variants: INORDER and OUTORDER

- Lower bound on period: $\mathcal{P} = \max_{1 \leq k \leq n} C_{exec}(k)$

Communication models

- With overlap– full overlap of communications and computations: Overlap
  - A server can receive, compute and send (independent) data simultaneously
  - Multi-port communications: many incoming (resp. outgoing) communications can take place at the same time
  - Server: operate concurrently on different consecutive data sets
  - Execution time $C_{exec}(k) = \max\{C_{in}(k), C_{comp}(k), C_{out}(k)\}$
- Without overlap– communications and computations are sequential
  - $C_{exec}(k) = C_{in}(k) + C_{comp}(k) + C_{out}(k)$
  - Two variants: InOrder and OutOrder
- Lower bound on period: $\mathcal{P} = \max_{1 \leq k \leq n} C_{exec}(k)$

Communication model without overlap

- **InOrder** : each server completely processes a data set before starting the execution of the next one
- **OutOrder** : out-of-order execution allowed (for instance, start an incoming communication for data set $i+1$ while still processing data set i)
- Less idle time for the OUTORDER model: additional schedule flexibility
- Less flexible than multi-port models (used with overlap)
- Other combinations (with/without) overlap and (one-/multi-)port less realistic

Communication model without overlap

- **InOrder** : each server completely processes a data set before starting the execution of the next one
- **OutOrder** : out-of-order execution allowed (for instance, start an incoming communication for data set $i+1$ while still processing data set i)

- Less idle time for the OutOrder model: additional schedule flexibility
- Less flexible than multi-port models (used with overlap)

- Other combinations (with/without) overlap and (one-/multi-)port less realistic

Communication model without overlap

- **InOrder** : each server completely processes a data set before starting the execution of the next one
- **OutOrder** : out-of-order execution allowed (for instance, start an incoming communication for data set $i+1$ while still processing data set i)

- Less idle time for the OutOrder model: additional schedule flexibility
- Less flexible than multi-port models (used with overlap)

- Other combinations (with/without) overlap and (one-/multi-)port less realistic

Characterizing solutions

- Goal: find a plan $PL = (EG, OL)$ that minimizes the period (MinPeriod) or the latency (MinLatency);
- If EG is fixed, define the operation list OL (i.e., the schedule)
  - target schedule: cyclic and repeats for each data set
  - size proportional to the size of the plan (polynomial)
  - complete list of the time-steps at which every communication or computation begins and ends
  - $\text{BeginCalc}^n_{(i)}$ (resp. $\text{EndCalc}^n_{(i)}$): time-step where computation of C_i on data set n begins (resp. ends)
  - For each edge $C_i \rightarrow C_j$ in the plan, $\text{BeginComm}^n_{(i,j)}$ (resp. $\text{EndComm}^n_{(i,j)}$): time-step where communication $C_i \rightarrow C_j$ (edge in the plan) involving data set n begins (resp. ends)
  - Schedule: starts at time-step 0 with data set 0, cyclic behavior of period λ

Characterizing solutions: *OL*

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

- Different models $\longrightarrow$ different rules to guarantee a valid schedule: no resource contraint nor model hypothesis is violated
- All models are non-preemptive: once initiated, a communication or a communication cannot be interrupted
- Communications are synchronous
- Period and latency of a plan *PL* with a valid operation list:
  - $\mathcal{P} = \lambda$
  - $\mathcal{L} = \max\{\text{EndComm}^0_{(i,j)} | C_i \rightarrow C_j \in \mathcal{E}\}$.

Characterizing solutions: *OL*

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

- Different models $\longrightarrow$ different rules to guarantee a valid schedule: no resource contraint nor model hypothesis is violated
- All models are non-preemptive: once initiated, a communication or a communication cannot be interrupted
- Communications are synchronous
- Period and latency of a plan *PL* with a valid operation list:
  - $\mathcal{P} = \lambda$
  - $\mathcal{L} = \max\{\text{EndComm}^0_{(i,j)} | C_i \rightarrow C_j \in \mathcal{E}\}$.

Characterizing solutions: *OL*

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

- Different models $\longrightarrow$ different rules to guarantee a valid schedule: no resource contraint nor model hypothesis is violated
- All models are non-preemptive: once initiated, a communication or a communication cannot be interrupted
- Communications are synchronous
- Period and latency of a plan *PL* with a valid operation list:
  - $\mathcal{P} = \lambda$
  - $\mathcal{L} = \max\{\mathrm{EndComm}^{0}_{(i,j)} | C_i \rightarrow C_j \in \mathcal{E}\}$.

Summary of framework

- Application: set of filtering services, some dependencies between services
- Platform: fully homogeneous with 3 different communication models
- Goal:
  1. Build the execution plan
  2. Given the execution plan, build the operation list
- From an operation list, we can compute both period and latency of the schedule
- Objective: minimize period or latency
- $3 \times 2 \times 2 = 12$ problems to solve

Summary of framework

- Application: set of filtering services, some dependencies between services
- Platform: fully homogeneous with 3 different communication models
- Goal:
  1. Build the execution plan
  2. Given the execution plan, build the operation list
- From an operation list, we can compute both period and latency of the schedule
- Objective: minimize period or latency
- $3 \times 2 \times 2 = 12$ problems to solve

Summary of framework

- Application: set of filtering services, some dependencies between services
- Platform: fully homogeneous with 3 different communication models
- Goal:
    1. Build the execution plan
    2. Given the execution plan, build the operation list
- From an operation list, we can compute both period and latency of the schedule
- Objective: minimize period or latency
- $3 \times 2 \times 2 = 12$ problems to solve

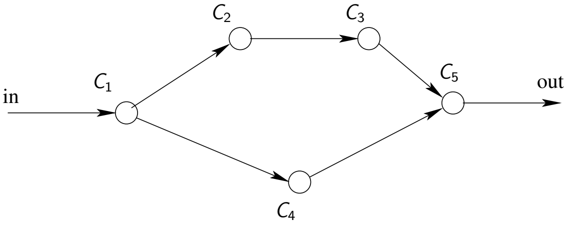
Summary of framework

- Application: set of filtering services, some dependencies between services
- Platform: fully homogeneous with 3 different communication models
- Goal:
  1. Build the execution plan
  2. Given the execution plan, build the operation list
- From an operation list, we can compute both period and latency of the schedule
- Objective: minimize period or latency
- $3 \times 2 \times 2 = 12$ problems to solve

Outline

(Ex1) Simple motivating example

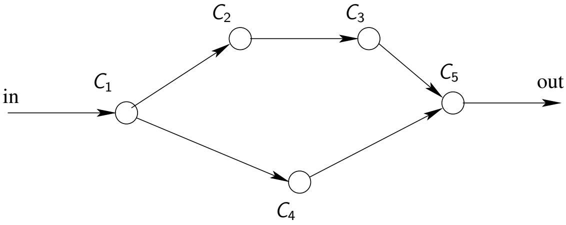
- 5 services of cost 4 and selectivity 1
- The execution graph EG is the following:



- Find the operation list which minimizes latency or period for each of the three communication models?

(Ex1) Simple motivating example

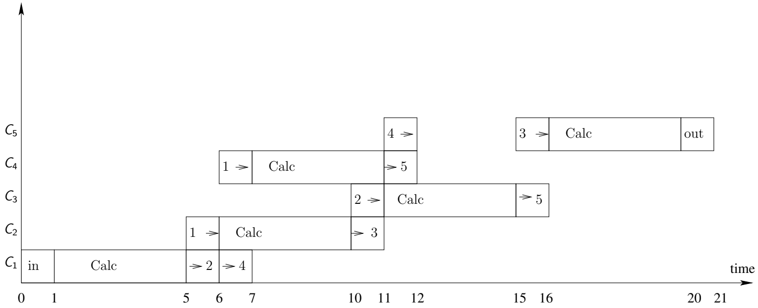
- 5 services of cost 4 and selectivity 1
- The execution graph EG is the following:



- Find the operation list which minimizes latency or period for each of the three communication models?

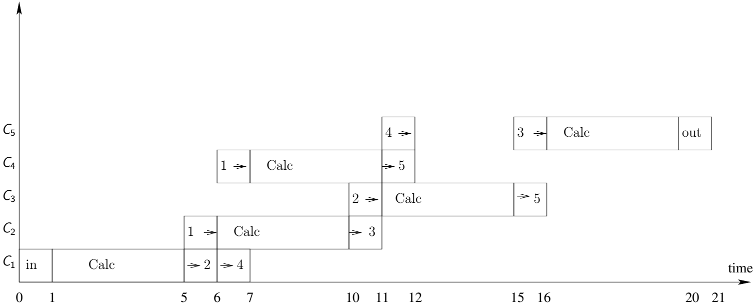
(Ex1) Minimizing latency

- One-port communications: InOrder and OutOrder identical for latency (consider one single data set, $\lambda = 21$)



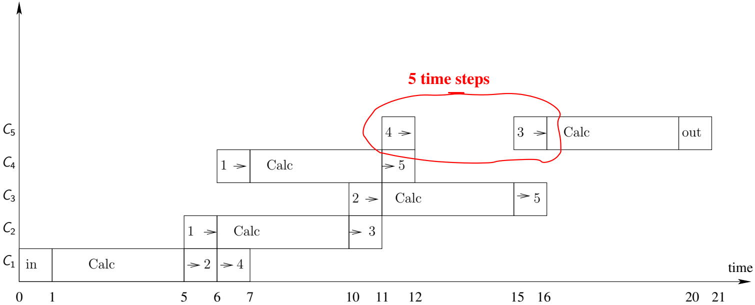
- Latency of 21, no idle time in the longest path, identical operation list for Overlap model

(Ex1) Minimizing period



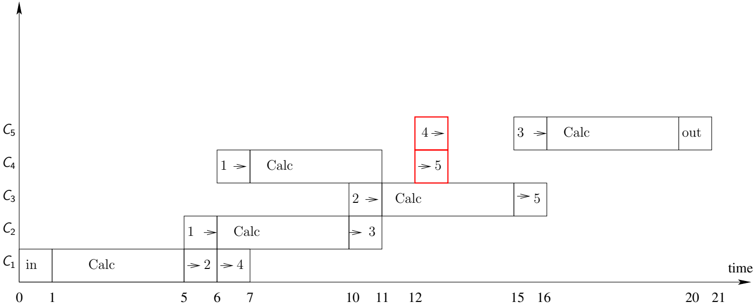
- Overlap: change $\lambda = 21$ into $\lambda = 5$ in previous OL, no resource conflict $\longrightarrow \mathcal{P} = 5$
- Can do better: $\mathcal{P} = 4$ if we move communication $C_4 \longrightarrow C_5$ at time-step 12 (optimal)

(Ex1) Minimizing period



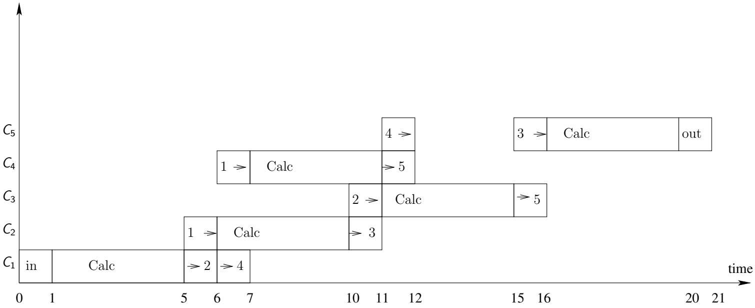
- OVERLAP: change $\lambda = 21$ into $\lambda = 5$ in previous OL, no resource conflict $\rightarrow \mathcal{P} = 5$
- Can do better: $\mathcal{P} = 4$ if we move communication $C_4 \rightarrow C_5$ at time-step 12 (optimal)

(Ex1) Minimizing period



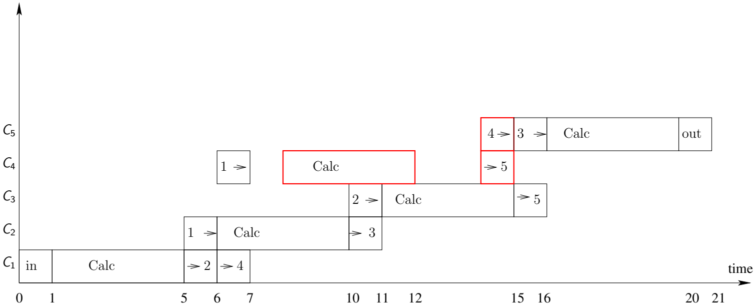
- Overlap: change $\lambda = 21$ into $\lambda = 5$ in previous OL, no resource conflict $\rightarrow \mathcal{P} = 5$
- Can do better: $\mathcal{P} = 4$ if we move communication $C_4 \rightarrow C_5$ at time-step 12 (optimal)

(Ex1) Minimizing period



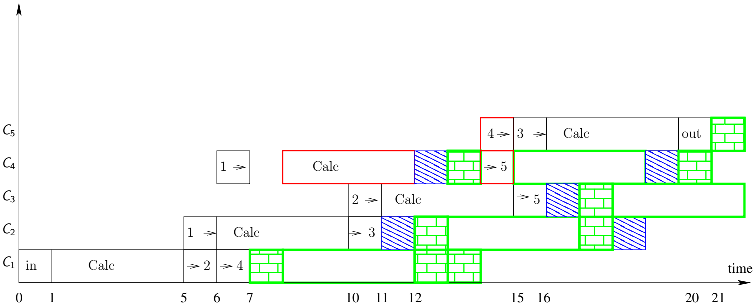
- Without overlap: difference between InOrder and OutOrder; minimal value: 7 (for C_1 and C_5)
- OutOrder
  - To obtain $\mathcal{P} = 7$, move $C_4 \rightarrow C_5$ at time-step 14, and computation of C_4 at time-step 8
  - Execution out of order: not valid for InOrder

(Ex1) Minimizing period



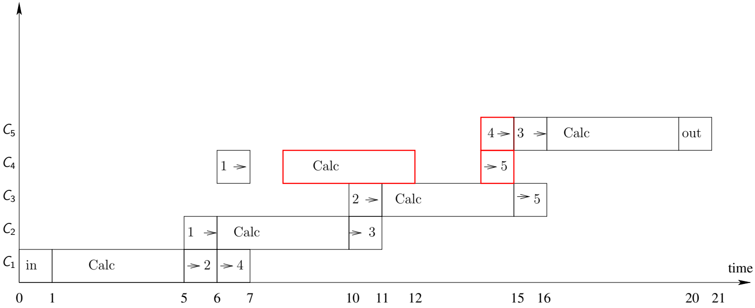
- Without overlap: difference between InOrder and OutOrder; minimal value: 7 (for C_1 and C_5)
- OutOrder
  - To obtain $\mathcal{P} = 7$, move $C_4 \rightarrow C_5$ at time-step 14, and computation of C_4 at time-step 8
  - Execution out of order: not valid for InOrder

(Ex1) Minimizing period



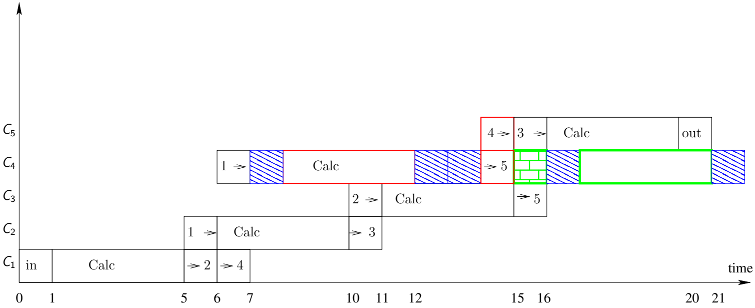
- Without overlap: difference between InOrder and OutOrder; minimal value: 7 (for C_1 and C_5)
- OutOrder
  - To obtain $\mathcal{P} = 7$, move $C_4 \to C_5$ at time-step 14, and computation of C_4 at time-step 8
  - Execution out of order: not valid for InOrder

(Ex1) Minimizing period



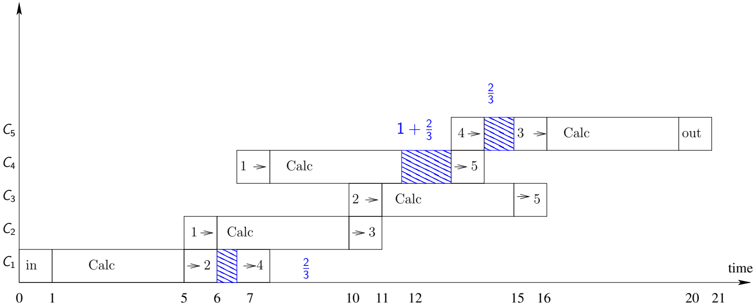
- InOrder
  - With the same operation list as for OutOrder...
  - ... $\mathcal{P} = 9$
  - Share the 3 slots of idle time: $\mathcal{P} = 7 + \frac{2}{3}$ ☺
  - (idle time comes from the difference of path lengths)

(Ex1) Minimizing period



- InOrder
  - With the same operation list as for OutOrder...
  - ... $\mathcal{P} = 9$
  - Share the 3 slots of idle time: $\mathcal{P} = 7 + \frac{2}{3}$ ☺
  - (idle time comes from the difference of path lengths)

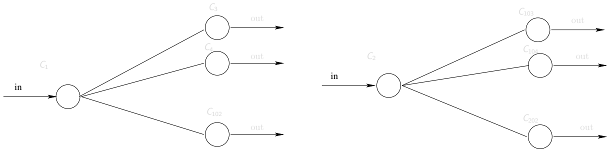
(Ex1) Minimizing period



- InOrder
  - With the same operation list as for OutOrder...
  - ... $\mathcal{P} = 9$
  - Share the 3 slots of idle time: $\mathcal{P} = 7 + \frac{2}{3}$ ☺
  - (idle time comes from the difference of path lengths)

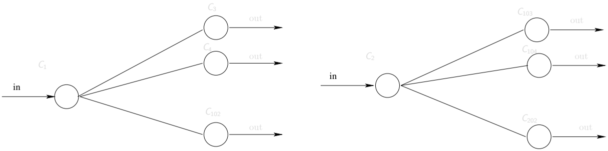
(Ex2) With and without communication costs

- 202 services:
  - C_1 and C_2 have selectivities $\sigma_i = 0.9999$ and costs $c_i = 100$
  - C_i for $3 \leq i \leq 202$ have $\sigma_i = 100$ and $c_i = \frac{100}{0.9999}$
- Does there exist a plan whose period does not exceed 100?
- Without communication cost, we obtain 100 by chaining C_1 and C_2, and by making C_2 the predecessor of all other services
- With communication costs, OVERLAP: period 200 (because of outgoing communications of C_2)
- Optimal solution (we lose the structure of chain):



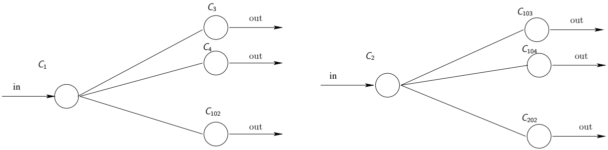
(Ex2) With and without communication costs

- 202 services:
  - C_1 and C_2 have selectivities $\sigma_i = 0.9999$ and costs $c_i = 100$
  - C_i for $3 \leq i \leq 202$ have $\sigma_i = 100$ and $c_i = \frac{100}{0.9999}$
- Does there exist a plan whose period does not exceed 100?
- Without communication cost, we obtain 100 by chaining C_1 and C_2, and by making C_2 the predecessor of all other services
- With communication costs, OVERLAP: period 200 (because of outgoing communications of C_2)
- Optimal solution (we lose the structure of chain):



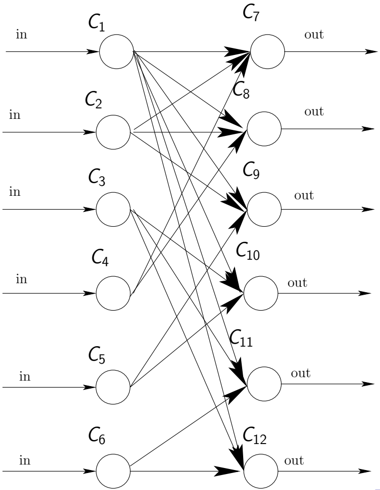
(Ex2) With and without communication costs

- 202 services:
  - C_1 and C_2 have selectivities $\sigma_i = 0.9999$ and costs $c_i = 100$
  - C_i for $3 \leq i \leq 202$ have $\sigma_i = 100$ and $c_i = \frac{100}{0.9999}$
- Does there exist a plan whose period does not exceed 100?
- Without communication cost, we obtain 100 by chaining C_1 and C_2, and by making C_2 the predecessor of all other services
- With communication costs, Overlap: period 200 (because of outgoing communications of C_2)
- Optimal solution (we lose the structure of chain):



(Ex3) One-port vs multi-port for latency

- 12 services, $c_i = \sigma_i = 1$ for all services except $\sigma_2 = \sigma_3 = 2$ and $\sigma_4 = \sigma_5 = \sigma_6 = 3$



(Ex3) One-port vs multi-port for latency

- Multi-port communications: C_1 to C_6 ends computing at time 2, all communications finish at time 8, and then it takes 6 time units of computation for C_7 to C_{12}, plus 6 time units to send the result to the outside world, $\mathcal{L} = 20$
- One-port communications: Idle time due to synchronization issues and $\mathcal{L} > 20$
  - Communications of weight 1 from C_1 to 6 other services, to be done at every time step
  - $\text{BeginComm}^0_{(1,j)} = 2$ and $\text{BeginComm}^0_{(1,k)} = 3$
  - No preemption: C_k idle between time 2 and 3 (other incoming communications greater than 1)

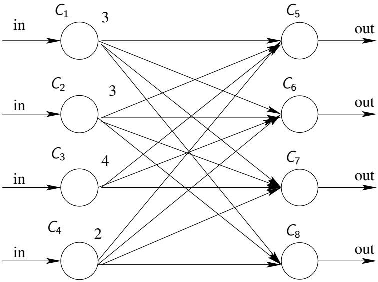
(Ex3) One-port vs multi-port for latency

- Multi-port communications: C_1 to C_6 ends computing at time 2, all communications finish at time 8, and then it takes 6 time units of computation for C_7 to C_{12}, plus 6 time units to send the result to the outside world, $\mathcal{L} = 20$

- One-port communications: Idle time due to synchronization issues and $\mathcal{L} > 20$
  - Communications of weight 1 from C_1 to 6 other services, to be done at every time step
  - $\text{BeginComm}^0_{(1,j)} = 2$ and $\text{BeginComm}^0_{(1,k)} = 3$
  - No preemption: C_k idle between time 2 and 3 (other incoming communications greater than 1)

(Ex4) One-port vs multi-port for period

- OVERLAP model, different data sets are processed concurrently, one- vs multi-port
- 4 services C_1 to C_4 with $c_i = 1$, $\sigma_1 = \sigma_2 = 3$, $\sigma_3 = 4$, and $\sigma_4 = 2$
- 4 services C_5 to C_8 with very small selectivities and communication costs



(Ex4) One-port vs multi-port for period

- Multi-port communications: $\mathcal{P} = 12$, maximum time needed for communications, bandwidth shared between links

- One-port communications: no idle time for $C_{\text{out}}(1) = C_{\text{out}}(2) = C_{\text{out}}(3) = 12$ and $C_{\text{in}}(5) = C_{\text{in}}(6) = C_{\text{in}}(7) = 12$

  More tricky than for latency but same idea: synchronization issues prevent the one-port model to achieve a period of 12

(Ex4) One-port vs multi-port for period

- Multi-port communications: $\mathcal{P} = 12$, maximum time needed for communications, bandwidth shared between links

- One-port communications: no idle time for $C_{\text{out}}(1) = C_{\text{out}}(2) = C_{\text{out}}(3) = 12$ and $C_{\text{in}}(5) = C_{\text{in}}(6) = C_{\text{in}}(7) = 12$

  More tricky than for latency but same idea: synchronization issues prevent the one-port model to achieve a period of 12

Outline

Period minimization, given an execution graph

Theorem

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.*

- OVERLAP model: all communications executed in time $T = \max_{1 \leq k \leq n}\{C_{\text{in}}(k), C_{\text{out}}(k)\}$: communication of size t is assigned a fraction t/T of available bandwidth
- Without overlap: involved reduction from RN3DM, a particular instance of 3-Dimensional Matching with two permutations (also called the permutation sums problem)
- The theorem holds for regular streaming applications

Period minimization, given an execution graph

Theorem

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.*

- OVERLAP model: all communications executed in time $T = \max_{1 \le k \le n}\{C_{\text{in}}(k), C_{\text{out}}(k)\}$: communication of size t is assigned a fraction t/T of available bandwidth
- Without overlap: involved reduction from RN3DM, a particular instance of 3-Dimensional Matching with two permutations (also called the permutation sums problem)
- The theorem holds for regular streaming applications

Period minimization, given an execution graph

Theorem

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.*

- OVERLAP model: all communications executed in time $T = \max_{1 \leq k \leq n}\{C_{\text{in}}(k), C_{\text{out}}(k)\}$: communication of size t is assigned a fraction t/T of available bandwidth
- Without overlap: involved reduction from RN3DM, a particular instance of 3-Dimensional Matching with two permutations (also called the permutation sums problem)
- The theorem holds for regular streaming applications

Period minimization, execution graph to find

Proposition

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.*

Property on the shape of the solution: reduces the search of optimal execution graph

Theorem

Problems MinPeriod-Overlap, MinPeriod-OutOrder *and* MinPeriod-InOrder *without dependence constraints are all NP-hard.*

Two involved reductions based on RN3DM, one for the case with overlap, one for the case without (which holds for both models)

+ polynomial instances for linear chain execution graphs

Latency minimization

Theorem

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.

Theorem

Problems MinLatency-Overlap, MinLatency-OutOrder *and* MinLatency-InOrder *without dependence constraints are all NP-hard.*

More involved reductions based on RN3DM

Outline

Related work

Srivastava et al– *Query optimization over web services*, with identical speed servers and no communications

Detti et al– Scheduling unreliable jobs on parallel machines: service selectivities correspond to job failure probabilities

Benoit et al– Extension to different-speed servers (IPDPS'09, Fanny's talk)

Traditional streaming applications– Data Cutter project in Colombus, Qishi Wu in Memphis, work in our GRAAL project, numerous talks at this workshop ☺

Related work

Srivastava et al– *Query optimization over web services*, with identical speed servers and no communications

Detti et al– Scheduling unreliable jobs on parallel machines: service selectivities correspond to job failure probabilities

Benoit et al– Extension to different-speed servers (IPDPS'09, Fanny's talk)

Traditional streaming applications– Data Cutter project in Colombus, Qishi Wu in Memphis, work in our GRAAL project, numerous talks at this workshop ☺

Conclusion

- Mapping filtering streaming applications on large-scale homogeneous platforms
- Communication models and their impact: 3 natural and realistic models
- Important problems addressed in this work:
  - 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?
- Complexity of all 12 optimization problems
- Solid theoretical foundations for the study of filtering streaming applications
- Several of these results apply to regular streaming applications

(To appear in SPAA'09)

Conclusion

- Mapping filtering streaming applications on large-scale homogeneous platforms
- Communication models and their impact: 3 natural and realistic models
- Important problems addressed in this work:
  - 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?
- Complexity of all 12 optimization problems
- Solid theoretical foundations for the study of filtering streaming applications
- Several of these results apply to regular streaming applications

(To appear in SPAA'09)

Future work

- Communication models with preemption: carefully assess the cost of interruptions
- Bi-criteria problems: given a threshold period, what is the optimal latency? and conversely, given a threshold latency, what is the optimal period?
  - All problem instances are NP-hard
  - Search for approximation algorithms
  - Design of efficient heuristics