Scheduling pipeline workflows to optimize throughput, latency and reliability

Anne Benoit, Veronika Rehn-Sonigo, Yves Robert
GRAAL team, LIP, École Normale Supérieure de Lyon, France

Harald Kosch
University of Passau, Germany

University of Passau
June 26, 2008

Introduction and motivation

- Mapping applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline skeletons onto heterogeneous platforms

Introduction and motivation

- Mapping applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline skeletons onto heterogeneous platforms

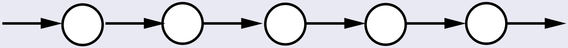
Introduction and motivation

- Mapping applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Mapping pipeline skeletons onto heterogeneous platforms

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

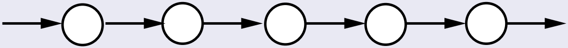
Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{FP}$, probability of failure of the application (i.e. some data-sets will not be processed)

Multi-criteria!

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

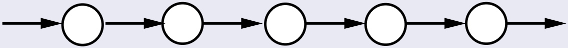
Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{FP}$, probability of failure of the application (i.e. some data-sets will not be processed)

Multi-criteria!

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

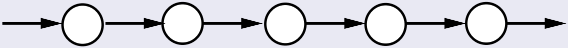
Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{FP}$, probability of failure of the application (i.e. some data-sets will not be processed)

Multi-criteria!

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

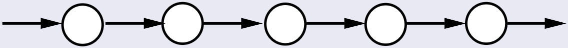
Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{FP}$, probability of failure of the application (i.e. some data-sets will not be processed)

Multi-criteria!

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Criteria to optimize?

Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

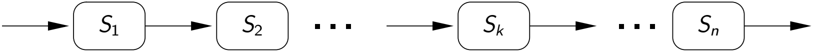
Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data set

Reliability: inverse of $\mathcal{FP}$, probability of failure of the application (i.e. some data-sets will not be processed)

Multi-criteria!

Rule of the game

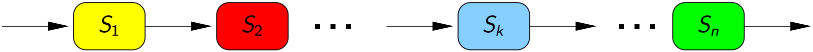
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



The pipeline application

Rule of the game

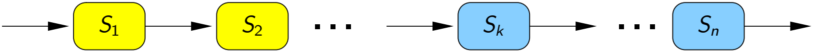
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



One-to-one Mapping

Rule of the game

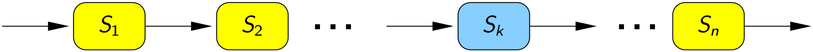
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



INTERVAL MAPPING

Rule of the game

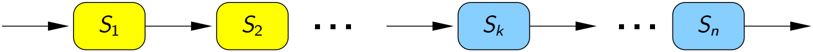
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



General Mapping

Rule of the game

- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



INTERVAL MAPPING

- Replication (one interval onto several processors) in order to increase reliability only: each data-set is processed by several processors

Major contributions

Theory

- Definition of multi-criteria mappings
- Problem complexity
- Linear programming formulation

Practice

- Heuristics for INTERVAL MAPPING on clusters
- Experiments: compare heuristics, evaluate their performance
- Simulation of a JPEG encoder application

Major contributions

Theory

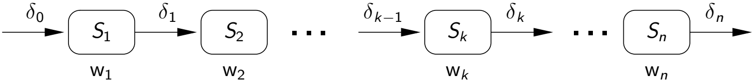
- Definition of multi-criteria mappings
- Problem complexity
- Linear programming formulation

Practice

- Heuristics for INTERVAL MAPPING on clusters
- Experiments: compare heuristics, evaluate their performance
- Simulation of a JPEG encoder application

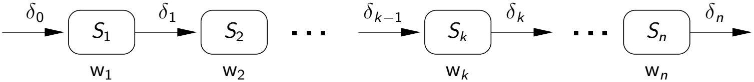
Outline

The application



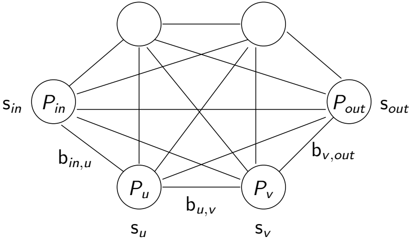
- n stages $\mathcal{S}_k$, $1 \leq k \leq$ n
- $\mathcal{S}_k$:
  - receives input of size δ_{k-1} from $\mathcal{S}_{k-1}$
  - performs w_k computations
  - outputs data of size δ_k to $\mathcal{S}_{k+1}$
- $\mathcal{S}_0$ and $\mathcal{S}_{n+1}$: virtual stages representing the outside world
- Classical application schema, for instance in image processing

The application



- n stages $\mathcal{S}_k$, $1 \leq k \leq$ n
- $\mathcal{S}_k$:
  - receives input of size δ_{k-1} from $\mathcal{S}_{k-1}$
  - performs w_k computations
  - outputs data of size δ_k to $\mathcal{S}_{k+1}$
- $\mathcal{S}_0$ and $\mathcal{S}_{n+1}$: virtual stages representing the outside world
- Classical application schema, for instance in image processing

The platform



- p processors P_u, $1 \leq u \leq \mathrm{p}$, fully interconnected
- s_u: speed of processor P_u
- bidirectional link $\mathrm{link}_{u,v} : P_u \rightarrow P_v$, bandwidth $\mathrm{b}_{u,v}$
- fp_u: failure probability of processor P_u (independent of the duration of the application, meant to run for a long time)
- one-port model with no overlap: each processor can either send, receive or compute at any time-step

Different platforms

Fully Homogeneous – Identical processors ($s_u = s$) and links ($b_{u,v} = b$): typical parallel machines

Communication Homogeneous – Different-speed processors ($s_u \neq s_v$), identical links ($b_{u,v} = b$): networks of workstations, clusters

Fully Heterogeneous – Fully heterogeneous architectures, $s_u \neq s_v$ and $b_{u,v} \neq b_{u',v'}$: hierarchical platforms, grids

Different platforms

Fully Homogeneous – Identical processors ($s_u = s$) and links ($b_{u,v} = b$): typical parallel machines

Failure Homogeneous– Identically reliable processors ($fp_u = fp_v$)

Communication Homogeneous – Different-speed processors ($s_u \neq s_v$), identical links ($b_{u,v} = b$): networks of workstations, clusters

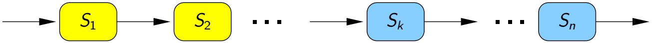
Fully Heterogeneous – Fully heterogeneous architectures, $s_u \neq s_v$ and $b_{u,v} \neq b_{u',v'}$: hierarchical platforms, grids

Failure Heterogeneous – Different failure probabilities ($fp_u \neq fp_v$)

Mapping problem: INTERVAL MAPPING

- Several consecutive stages onto the same processor(s)
- Increase computational load, reduce communications

- Partition of [1..n] into m intervals $I_j = [d_j, e_j]$
  (with $d_j \leq e_j$ for $1 \leq j \leq m$, $d_1 = 1$, $d_{j+1} = e_j + 1$ for $1 \leq j \leq m - 1$ and $e_m =$ n)



- Interval I_j mapped onto set of processors alloc(j) (replication)
- $k_j = |\text{alloc}(j)|$ processors executing I_j, $k_j \geq 1$.

Objective function?

Mono-criterion

- Minimize period $\mathcal{P}$
- Minimize latency $\mathcal{L}$
- Minimize failure probability $\mathcal{FP}$

Objective function?

Mono-criterion

- Minimize period $\mathcal{P}$
- Minimize latency $\mathcal{L}$
- Minimize failure probability $\mathcal{FP}$

Multi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{FP}$?
- Values which are not comparable

Objective function?

Mono-criterion

- Minimize period $\mathcal{P}$
- Minimize latency $\mathcal{L}$
- Minimize failure probability $\mathcal{FP}$

Multi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{FP}$?
- Values which are not comparable

- Minimize $\mathcal{P}$ for a fixed latency and failure
- Minimize $\mathcal{L}$ for a fixed period and failure
- Minimize $\mathcal{FP}$ for a fixed period and latency

Objective function?

Mono-criterion

- Minimize period $\mathcal{P}$
- Minimize latency $\mathcal{L}$
- Minimize failure probability $\mathcal{FP}$

Bi-criteria

- Period and Latency:
- Minimize $\mathcal{P}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed period

Objective function?

Mono-criterion

- Minimize period $\mathcal{P}$
- Minimize latency $\mathcal{L}$
- Minimize failure probability $\mathcal{FP}$

Bi-criteria

- Failure and Latency:
- Minimize $\mathcal{FP}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed failure

Interval Mapping problem - Period/Latency

- Period/Latency: no replication
- alloc(j) reduced to a single processor
- *Communication Homogeneous* platforms (easy to extend)

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{\mathsf{b}} + \frac{\sum_{i=d_j}^{e_j} \mathsf{w}_i}{\mathsf{s}_{\mathsf{alloc}(j)}} + \frac{\delta_{e_j}}{\mathsf{b}} \right\}$$

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{\mathsf{b}} + \frac{\sum_{i=d_j}^{e_j} \mathsf{w}_i}{\mathsf{s}_{\mathsf{alloc}(j)}} \right\} + \frac{\delta_n}{\mathsf{b}}$$

Interval Mapping problem - Period/Latency

- Period/Latency: no replication
- alloc(j) reduced to a single processor
- *Communication Homogeneous* platforms (easy to extend)

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b} \right\}$$

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} \right\} + \frac{\delta_n}{b}$$

Interval Mapping problem - Period/Latency

- Period/Latency: no replication
- alloc(j) reduced to a single processor
- *Communication Homogeneous* platforms (easy to extend)

$$\mathcal{P} = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{\mathsf{b}} + \frac{\sum_{i=d_j}^{e_j} \mathsf{w}_i}{\mathsf{s}_{\mathrm{alloc}(j)}} + \frac{\delta_{e_j}}{\mathsf{b}} \right\}$$

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{\mathsf{b}} + \frac{\sum_{i=d_j}^{e_j} \mathsf{w}_i}{\mathsf{s}_{\mathrm{alloc}(j)}} \right\} + \frac{\delta_n}{\mathsf{b}}$$

Interval Mapping problem - Latency/Reliability

- Latency/Reliability
- $\text{alloc}(j)$ is a set of k_j processors
- *Communication Homogeneous* platforms
- Output by only one processor (consensus between working processors)

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ k_j \times \frac{\delta_{d_j - 1}}{\mathsf{b}} + \frac{\sum_{i=d_j}^{e_j} \mathsf{w}_i}{\min_{u \in \text{alloc}(j)} (\mathsf{s}_u)} \right\} + \frac{\delta_n}{\mathsf{b}}$$

$$\mathcal{FP} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \text{alloc}(j)} \mathsf{fp}_u)$$

Interval Mapping problem - Latency/Reliability

- Latency/Reliability
- alloc(j) is a set of k_j processors
- *Communication Homogeneous* platforms
- Output by only one processor (consensus between working processors)

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ k_j \times \frac{\delta_{d_j - 1}}{\mathsf{b}} + \frac{\sum_{i=d_j}^{e_j} \mathsf{w}_i}{\min_{u \in \mathrm{alloc}(j)}(\mathsf{s}_u)} \right\} + \frac{\delta_n}{\mathsf{b}}$$

$$\mathcal{FP} = 1 - \prod_{1 \leq j \leq m} \left(1 - \prod_{u \in \mathrm{alloc}(j)} \mathrm{fp}_u\right)$$

Interval Mapping problem - Latency/Reliability

- Latency/Reliability
- alloc(j) is a set of k_j processors
- *Communication Homogeneous* platforms
- Output by only one processor (consensus between working processors)

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ k_j \times \frac{\delta_{d_j - 1}}{\mathrm{b}} + \frac{\sum_{i=d_j}^{e_j} \mathrm{w}_i}{\min_{u \in \mathrm{alloc}(j)} (\mathrm{s}_u)} \right\} + \frac{\delta_n}{\mathrm{b}}$$

$$\mathcal{FP} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \mathrm{alloc}(j)} \mathrm{fp}_u)$$

Working out an example: Period/Latency

$$\begin{array}{ccccccc} \mathcal{S}_1 & \longrightarrow & \mathcal{S}_2 & \longrightarrow & \mathcal{S}_3 & \longrightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$
No communications, no reliability issues

Optimal period?

Working out an example: Period/Latency

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$
No communications, no reliability issues

Optimal period?
$\mathcal{P} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($\mathcal{L} = 17$)

Optimal latency?

Working out an example: Period/Latency

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$
No communications, no reliability issues

Optimal period?
$\mathcal{P} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($\mathcal{L} = 17$)

Optimal latency?
$\mathcal{L} = 12$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$ ($\mathcal{P} = 12$)

Min. latency if $\mathcal{P} \leq 10$?

Working out an example: Period/Latency

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 14 & & 4 & & 2 & & 4 \end{array}$$

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$
No communications, no reliability issues

Optimal period?
$\mathcal{P} = 7$, $\mathcal{S}_1 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2$, $\mathcal{S}_4 \rightarrow P_3$ ($\mathcal{L} = 17$)

Optimal latency?
$\mathcal{L} = 12$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$ ($\mathcal{P} = 12$)

Min. latency if $\mathcal{P} \leq 10$?
$\mathcal{L} = 14$, $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

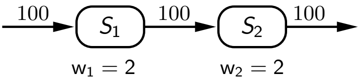
Outline

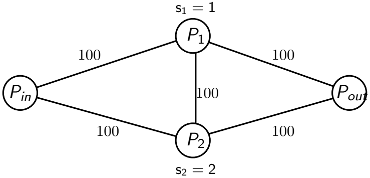
Complexity results: Latency - Com Hom

Lemma

On *Fully Homogeneous* and *Communication Homogeneous* platforms, the optimal interval mapping which minimizes latency can be determined in polynomial time.

- Assign whole pipeline to fastest processor!
- No intra communications to pay in this case.
- Only input and output com, identical for each mapping.



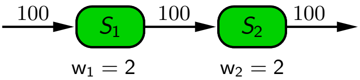


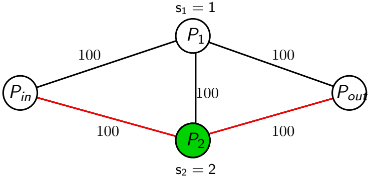
Complexity results: Latency - Com Hom

Lemma

On *Fully Homogeneous* and *Communication Homogeneous* platforms, the optimal interval mapping which minimizes latency can be determined in polynomial time.

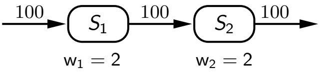
- Assign whole pipeline to fastest processor!
- No intra communications to pay in this case.
- Only input and output com, identical for each mapping.

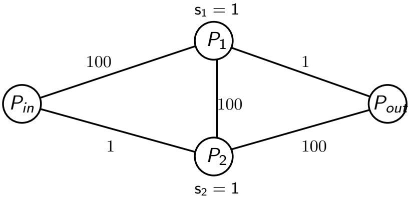




Complexity results: Latency - Het

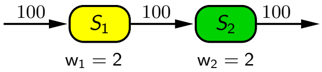
- *Fully Heterogeneous* platforms
- The interval of stages may need to be split

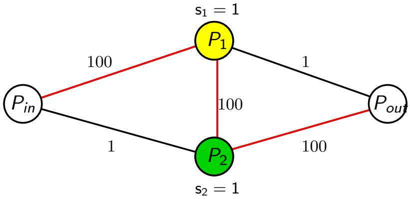




Complexity results: Latency - Het

- *Fully Heterogeneous* platforms
- The interval of stages may need to be split





Complexity results: Latency - Het

Lemma

On *Fully Heterogeneous* platforms, the optimal general mapping which minimizes latency can be determined in polynomial time.

Dynamic programming algorithm

Theorem

On *Fully Heterogeneous* platforms, finding an optimal one-to-one or interval mapping which minimizes latency is NP-hard.

One-to-one mapping: reduction from the Traveling Salesman Problem

Interval mapping: **involved reduction** from another graph problem

Complexity results: Latency - Het

Lemma

On *Fully Heterogeneous* platforms, the optimal general mapping which minimizes latency can be determined in polynomial time.

Dynamic programming algorithm

Theorem

On *Fully Heterogeneous* platforms, finding an optimal one-to-one or interval mapping which minimizes latency is NP-hard.

One-to-one mapping: reduction from the Traveling Salesman Problem

Interval mapping: **involved reduction** from another graph problem

Complexity results: Period

- Minimize the period on *Fully Homogeneous* platforms:
  - classical chains-on-chains problem
  - polynomial complexity
- *Communication Homogeneous* platforms: chains-on-chains, but with different speed processors!
  - the problem becomes NP-hard
  - **involved reduction**

Definition (HETERO-1D-PARTITION-DEC)

Given n elements $a_1, a_2, \ldots, a_n$, p values $s_1, s_2, \ldots, s_p$ and a bound K, can we find a partition of [1..n] into p intervals $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p$, and a permutation σ of $\{1, 2, \ldots, p\}$, such that $\max_{1 \leq k \leq p} \frac{\sum_{i \in \mathcal{I}_k} a_i}{s_{\sigma(k)}} \leq K$?

Complexity results: Reliability

Lemma

Minimizing the failure probability can be done in polynomial time.

- Formula computing global failure probability

$$\mathcal{FP} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \text{alloc}(j)} \text{fp}_u)$$

- Minimum reached by replicating whole pipeline as a single interval on all processors
- True for all platform types

Complexity results: Reliability

Lemma

Minimizing the failure probability can be done in polynomial time.

- Formula computing global failure probability

$$\mathcal{FP} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \text{alloc}(j)} \text{fp}_u)$$

- Minimum reached by replicating whole pipeline as a single interval on all processors
- True for all platform types

Outline

Complexity results - Latency/Period

- Interval mapping, *Fully Homogeneous* platforms
- Polynomial: dynamic programming algorithm

- Interval mapping, *Communication Homogeneous* platforms
- Period minimization: NP-hard
- Bi-criteria problems: NP-hard

Complexity results - Latency/Period

- Interval mapping, *Fully Homogeneous* platforms
- Polynomial: dynamic programming algorithm

- Interval mapping, *Communication Homogeneous* platforms
- Period minimization: NP-hard
- Bi-criteria problems: NP-hard

Summary of Latency/Failure complexity results

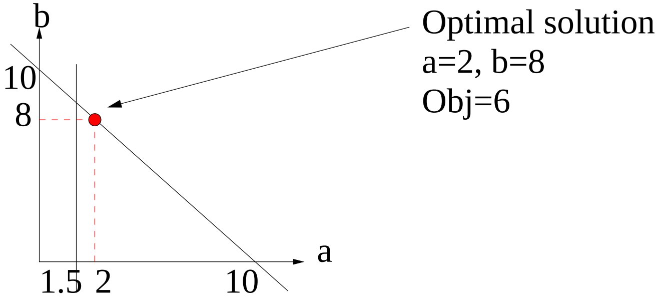
- Lemma-NoSplit: On *Fully Homogeneous* and *Communication Homogeneous-Failure Homogeneous* platforms, there is a mapping of the pipeline as a single interval which minimizes the failure probability (resp. latency) under a fixed latency (resp. failure probability) threshold.
- *Communication Homogeneous-Failure Homogeneous*: polynomial algorithms based on Lemma-NoSplit.
- *Communication Homogeneous-Failure Heterogeneous*: lemma not true, open complexity (probably NP-hard)
- *Fully Heterogeneous*: bi-criteria (decision problems associated to the) optimization problems are NP-hard.

details

Outline

Integer linear program

- Integer variables a, b
- Constraints $a \geq 0$, $b \geq 0$, $a + b \leq 10$, $2a \geq 3$
- Objective function: Maximize $(b - a)$



Integer linear program for our problems

- Latency/Period problem
- Integer LP to solve INTERVAL MAPPING on *Communication Homogeneous* platforms
- Many integer variables: no efficient algorithm to solve
- Approach limited to small problem instances
- Absolute performance of the heuristics for such instances

- Latency/Failure problem: no linear formulation because of strong non-linearity of failure probability formula

skip

Integer linear program for our problems

- Latency/Period problem
- Integer LP to solve INTERVAL MAPPING on *Communication Homogeneous* platforms
- Many integer variables: no efficient algorithm to solve
- Approach limited to small problem instances
- Absolute performance of the heuristics for such instances

- Latency/Failure problem: no linear formulation because of strong non-linearity of failure probability formula

skip

Linear program: variables

- T_{opt}: period or latency of the pipeline, depending on the objective function

Boolean variables:

- $x_{k,u}$: 1 if S_k on P_u
- $y_{k,u}$: 1 if $\mathcal{S}_k$ and S_{k+1} both on P_u
- $z_{k,u,v}$: 1 if $\mathcal{S}_k$ on P_u and S_{k+1} on P_v

Integer variables:

- first_u and last_u: integer denoting first and last stage assigned to P_u (to enforce interval constraints)

Linear program: variables

- T_{opt}: period or latency of the pipeline, depending on the objective function

Boolean variables:

- $x_{k,u}$: 1 if S_k on P_u
- $y_{k,u}$: 1 if $\mathcal{S}_k$ and S_{k+1} both on P_u
- $z_{k,u,v}$: 1 if $\mathcal{S}_k$ on P_u and S_{k+1} on P_v

Integer variables:

- first_u and last_u: integer denoting first and last stage assigned to P_u (to enforce interval constraints)

Linear program: constraints

Constraints on processors and links:

- $\forall k \in [0..\mathrm{n}+1], \qquad \sum_u x_{k,u} = 1$
- $\forall k \in [0..\mathrm{n}], \qquad \sum_{u \neq v} z_{k,u,v} + \sum_u y_{k,u} = 1$
- $\forall k \in [0..\mathrm{n}], \forall u, v \in [1..\mathrm{p}] \cup \{in, out\}, u \neq v, x_{k,u} + x_{k+1,v} \leq 1 + z_{k,u,v}$
- $\forall k \in [0..\mathrm{n}], \forall u \in [1..\mathrm{p}] \cup \{in, out\}, \qquad x_{k,u} + x_{k+1,u} \leq 1 + y_{k,u}$

Constraints on intervals:

- $\forall k \in [1..\mathrm{n}], \forall u \in [1..\mathrm{p}], \qquad \mathrm{first}_u \leq k.x_{k,u} + \mathrm{n}.(1 - x_{k,u})$
- $\forall k \in [1..\mathrm{n}], \forall u \in [1..\mathrm{p}], \qquad \mathrm{last}_u \geq k.x_{k,u}$
- $\forall k \in [1..\mathrm{n}-1], \forall u, v \in [1..\mathrm{p}], u \neq v,$
  $\mathrm{last}_u \leq k.z_{k,u,v} + \mathrm{n}.(1 - z_{k,u,v})$
- $\forall k \in [1..\mathrm{n}-1], \forall u, v \in [1..\mathrm{p}], u \neq v, \quad \mathrm{first}_v \geq (k+1).z_{k,u,v}$

Linear program: constraints

Constraints on processors and links:

- $\forall k \in [0..\text{n}+1], \quad \sum_u x_{k,u} = 1$
- $\forall k \in [0..\text{n}], \quad \sum_{u \neq v} z_{k,u,v} + \sum_u y_{k,u} = 1$
- $\forall k \in [0..\text{n}], \forall u, v \in [1..\text{p}] \cup \{in, out\}, u \neq v, x_{k,u} + x_{k+1,v} \leq 1 + z_{k,u,v}$
- $\forall k \in [0..\text{n}], \forall u \in [1..\text{p}] \cup \{in, out\}, \quad x_{k,u} + x_{k+1,u} \leq 1 + y_{k,u}$

Constraints on intervals:

- $\forall k \in [1..\text{n}], \forall u \in [1..\text{p}], \quad \text{first}_u \leq k.x_{k,u} + \text{n}.(1 - x_{k,u})$
- $\forall k \in [1..\text{n}], \forall u \in [1..\text{p}], \quad \text{last}_u \geq k.x_{k,u}$
- $\forall k \in [1..\text{n}-1], \forall u, v \in [1..\text{p}], u \neq v,$
  $\text{last}_u \leq k.z_{k,u,v} + \text{n}.(1 - z_{k,u,v})$
- $\forall k \in [1..\text{n}-1], \forall u, v \in [1..\text{p}], u \neq v, \quad \text{first}_v \geq (k+1).z_{k,u,v}$

Linear program: constraints

$$\forall u \in [1..p], \sum_{k=1}^{n} \left\{ \left(\sum_{t \neq u} \frac{\delta_{k-1}}{b} z_{k-1,t,u} \right) + \frac{w_k}{s_u} x_{k,u} + \left(\sum_{v \neq u} \frac{\delta_k}{b} z_{k,u,v} \right) \right\} \leq \mathcal{P}$$

$$\sum_{u=1}^{p} \sum_{k=1}^{n} \left[\left(\sum_{t \neq u, t \in [1..p] \cup \{in, out\}} \frac{\delta_{k-1}}{b} z_{k-1,t,u} \right) + \frac{w_k}{s_u} x_{k,u} \right] + \left(\sum_{u \in [1..p] \cup \{in\}} \frac{\delta_n}{b} z_{n,u,out} \right) \leq \mathcal{L}$$

Min period with fixed latency

$T_{\text{opt}} = \mathcal{P}$

$\mathcal{L}$ is fixed

Min latency with fixed period

$T_{\text{opt}} = \mathcal{L}$

$\mathcal{P}$ is fixed

Linear program: constraints

$$\forall u \in [1..p], \sum_{k=1}^{n} \left\{ \left(\sum_{t \neq u} \frac{\delta_{k-1}}{b} z_{k-1,t,u} \right) + \frac{w_k}{s_u} x_{k,u} + \left(\sum_{v \neq u} \frac{\delta_k}{b} z_{k,u,v} \right) \right\} \leq \mathcal{P}$$

$$\sum_{u=1}^{p} \sum_{k=1}^{n} \left[\left(\sum_{t \neq u, t \in [1..p] \cup \{in, out\}} \frac{\delta_{k-1}}{b} z_{k-1,t,u} \right) + \frac{w_k}{s_u} x_{k,u} \right] + \left(\sum_{u \in [1..p] \cup \{in\}} \frac{\delta_n}{b} z_{n,u,out} \right) \leq \mathcal{L}$$

Min period with fixed latency

$$T_{\text{opt}} = \mathcal{P}$$

$\mathcal{L}$ is fixed

Min latency with fixed period

$$T_{\text{opt}} = \mathcal{L}$$

$\mathcal{P}$ is fixed

Outline

Heuristics

- Back to the problem Period/Latency
- Target clusters: *Communication Homogeneous* platforms and Interval Mapping

Two sets of heuristics

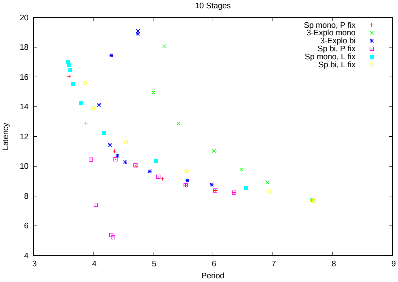
- Minimizing latency for a fixed period
- Minimizing period for a fixed latency

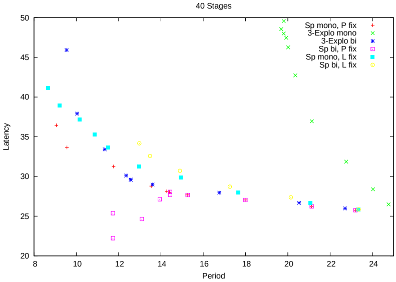
- Key idea: map the pipeline as a single interval then split the interval until stop criterion is reached
- Split: decreases period but increases latency

detailed heuristics

Heuristics comparison

- communication time $\delta_i = 10$, computation time $1 \leq w_i \leq 20$
- 10 processors



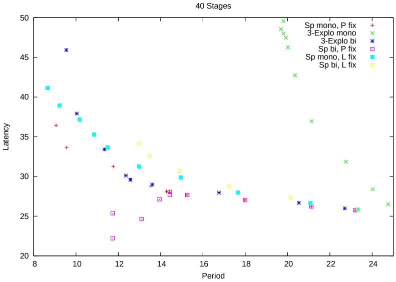


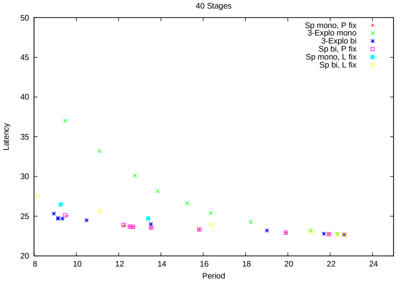
10 stages
☺ 2-Sp bi P
☹ 3-Sp mono P

40 stages
☺ 2-Sp mono P
☹ 3-Sp mono P

Heuristics comparison

- communication time $\delta_i = 10$, computation time $1 \leq w_i \leq 20$
- 10 vs. 100 processors





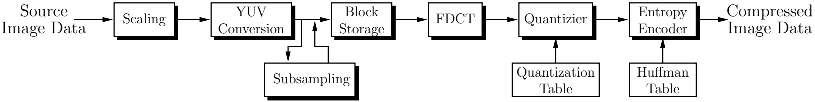
40 stages, 10 processors
☺ 2-Sp mono P
☹ 3-Sp mono P

40 stages, 100 processors
☺ 3-Sp bi P
☹ 3-Sp mono P

Real World Application

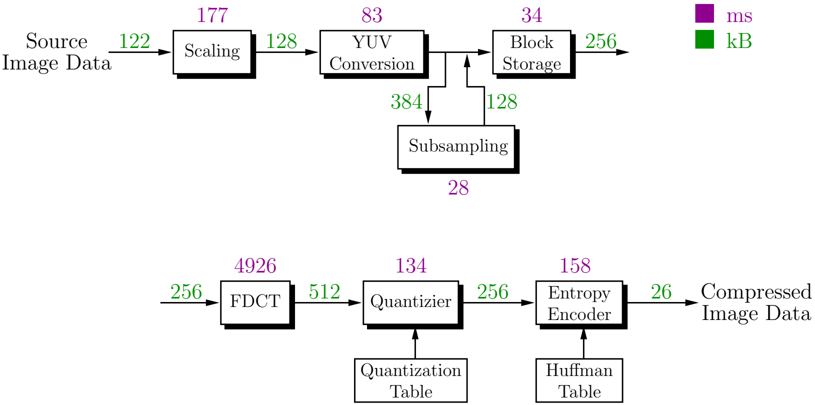
The JPEG encoder

- Image processing application
- JPEG: standardized interchange format
- Data compression
- 7 stages



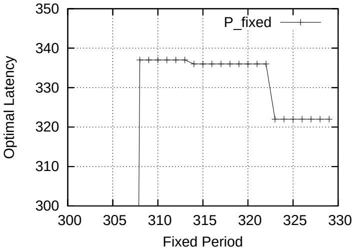
- Joint work with Harald Kosch, University of Passau, Germany

JPEG Encoder

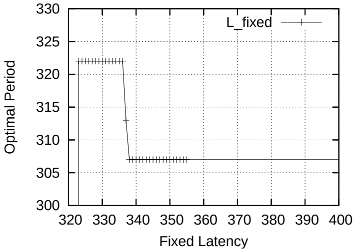


Simulation environment & bucket behavior

- MPI application, Message passing + sleep()
- (Homogeneous processors) - simulation of heterogeneity
- Mapping 7 stages on 10 processors



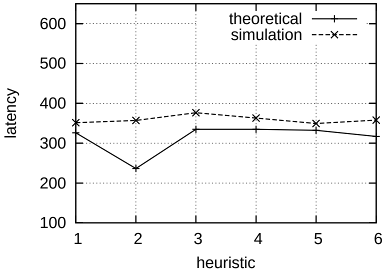
(a) Fixed P.



(b) Fixed L.

Results

- Heuristics vs LP: a simple heuristic always finds the optimal solution
- Comparison theory/experience: good except for one heuristic which violates threshold



Outline

Related work

Subhlok and Vondran– Extension of their work (pipeline on hom platforms)

Mapping pipelined computations onto clusters and grids– DAG [Taura et al.], DataCutter [Saltz et al.]

Energy-aware mapping of pipelined computations [Melhem et al.], three-criteria optimization

Mapping pipelined computations onto special-purpose architectures– FPGA arrays [Fabiani et al.]. Fault-tolerance for embedded systems [Zhu et al.]

Mapping skeletons onto clusters and grids– Use of stochastic process algebra [Benoit et al.]

Conclusion

Theoretical side

- Pipeline structured applications
- Multi-criteria mapping problem
- Complexity study: latency/period & latency/failure
- period/failure: mix difficulties of period (NP-hard) and failure (non-linear)

Practical side

- Design of several polynomial heuristics
- Simulation of a real world application
- Good results of the heuristics, even if not tuned for the JPEG application: close to LP solution

Future work

Theory

- Extension to stage replication (for period) and data-parallelism
- Extension to fork, fork-join and tree workflows
- Bounded multi-port communication model with overlap
- Add selectivity to stages (Web services)

Practice

- Design of new multi-criteria heuristics for fully heterogeneous platforms.
- Real experiments with bigger pipeline applications, using MPI
- Working on Scalable Video Coding applications in collaboration with Passau

RobSched'08

First International Workshop on **Robust Scheduling**

part of **ICPADS'08**, the 14th Int. Conf. on Parallel and Distributed Systems

December 8-10, 2008, Melbourne, Australia

- Scheduling algorithms for heterogeneous platforms
- Performance models
- Models of platform/application failures
- Fault tolerance issues
- Resource discovery and management
- Task and communication scheduling
- Task coordination and workflow
- Job scheduling
- Stochastic scheduling
- Scheduling applications for clusters and grids

Areas of scheduling, performance evaluation and fault tolerance.
Original, unpublished papers, as well as work-in-progress contributions.

July 4 - Full paper due
(6 IEEE-2-col. pages)
Aug. 22 - Notification
Sep. 9 - Final paper due
Dec. 8-10 - Workshop

Marco Aldinucci, Anne Benoit, Rajkumar Buyya, Henri Casanova, Anthony Chronopoulos, Murray Cole, Bruno Gaujal, Mourad Hakem, Aaron Harwood, Emmanuel Jeannot, Leila Kloul, Domenico Laforenza, Kiminori Matsuzaki, Rami Melhem, Gregory Mounie, Jean-Marc Nicod, Rajiv Ranjan, Yves Robert, Arnold Rosenberg, Uwe Schwiegelshohn, Oliver Sinnen, Magda Slawinska.

http://graal.ens-lyon.fr/~abenoit/conf/robsched08.html

Complexity results - Latency/Failure

Lemma NoSplit

On *Fully Homogeneous* and *Communication Homogeneous-Failure Homogeneous* platforms, there is a mapping of the pipeline as a single interval which minimizes the failure probability (resp. latency) under a fixed latency (resp. failure probability) threshold.

From an existing optimal solution consisting of more than one interval: easy to build a new optimal solution with a single interval

Complexity results - Latency/Failure

- *Communication Homogeneous-Failure Homogeneous*: Minimizing $\mathcal{FP}$ for a fixed $\mathcal{L}$

- Order processors in non-increasing order of s_j
- Find k maximum, such that

$$k \times \frac{\delta_0}{\mathsf{b}} + \frac{\sum_{1 \leq j \leq n} \mathsf{w}_j}{\mathsf{s}_k} + \frac{\delta_n}{\mathsf{b}} \leq \mathcal{L}$$

- Replicate the whole pipeline as a single interval onto the fastest k processors
- *Note that at any time s_k is the speed of the slowest processor used in the replication scheme*

Complexity results - Latency/Failure

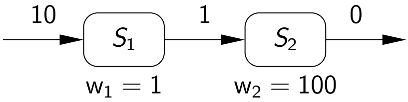
- *Communication Homogeneous* platforms-*Failure Homogeneous*: Minimizing $\mathcal{L}$ for a fixed $\mathcal{FP}$
- Find k minimum, such that

$$1 - (1 - \mathrm{fp}^k) \leq \mathcal{FP}$$

- Replicate the whole pipeline as a single interval onto the fastest k processors

Complexity results - Latency/Failure

- *Communication Homogeneous-Failure Heterogeneous*
- Lemma NoSplit not true: example
- One slow and reliable processor, s = 1, fp = 0.1
- Ten fast and unreliable processors, s = 100, fp = 0.8
- $\mathcal{L} \leq 22$, minimize $\mathcal{FP}$

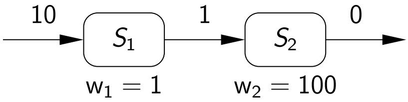


- One interval: $\mathcal{FP} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $\mathcal{FP} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$

- Open complexity (probably NP-hard)

Complexity results - Latency/Failure

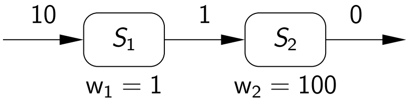
- *Communication Homogeneous-Failure Heterogeneous*
- Lemma NoSplit not true: example
- One slow and reliable processor, s = 1, fp = 0.1
- Ten fast and unreliable processors, s = 100, fp = 0.8
- $\mathcal{L} \leq 22$, minimize $\mathcal{FP}$



- One interval: $\mathcal{FP} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $\mathcal{FP} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$
- Open complexity (probably NP-hard)

Complexity results - Latency/Failure

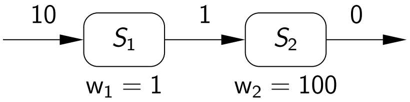
- *Communication Homogeneous-Failure Heterogeneous*
- Lemma NoSplit not true: example
- One slow and reliable processor, s = 1, fp = 0.1
- Ten fast and unreliable processors, s = 100, fp = 0.8
- $\mathcal{L} \leq 22$, minimize $\mathcal{FP}$



- One interval: $\mathcal{FP} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $\mathcal{FP} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$
- Open complexity (probably NP-hard)

Complexity results - Latency/Failure

- *Communication Homogeneous-Failure Heterogeneous*
- Lemma NoSplit not true: example
- One slow and reliable processor, $s = 1$, $fp = 0.1$
- Ten fast and unreliable processors, $s = 100$, $fp = 0.8$
- $\mathcal{L} \leq 22$, minimize $\mathcal{FP}$



- One interval: $\mathcal{FP} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $\mathcal{FP} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$

- Open complexity (probably NP-hard)

Complexity results - Latency/Failure

- *Fully Heterogeneous* platforms

Theorem

On *Fully Heterogeneous* platforms, the bi-criteria (decision problems associated to the) optimization problems are NP-hard.

- Reduction from 2-PARTITION: one single stage, processors of identical speed and $\text{fp}_j = e^{-a_j}$, $\text{b}_{in,j} = 1/a_j$ and $\text{b}_{j,out} = 1$

Back

Complexity results - Latency/Failure

- *Fully Heterogeneous* platforms

Theorem

On *Fully Heterogeneous* platforms, the bi-criteria (decision problems associated to the) optimization problems are NP-hard.

- Reduction from 2-PARTITION: one single stage, processors of identical speed and $\mathrm{fp}_j = e^{-a_j}$, $\mathrm{b}_{in,j} = 1/a_j$ and $\mathrm{b}_{j,out} = 1$

Back

Minimizing Latency for a Fixed Period (1/2)

2-Sp mono P: Splitting mono-criterion

- Map the whole pipeline on the fastest processor.
- At each step, select used processor j with largest period.
- Try to split its stage interval into 2 intervals, giving some stages to the next fastest processor j' in the list (not yet used).
- Split interval at any place, and either assign the first part of the interval on j and the remainder on j', or the other way round. Solution which minimizes $max(period(j), period(j'))$ is chosen if better than original solution.
- Break-conditions:
  Fixed period is reached or period cannot be improved anymore.

Minimizing Latency for a Fixed Period (2/2)

3-Sp mono P: 3-Splitting mono-criterion – Select used processor j with largest period and split its interval into three parts.

3-Sp bi P: 3-Splitting bi-criteria – More elaborated choice where to split: split the interval with largest period so that $max_{i\in\{j,j',j''\}}(\frac{\Delta latency}{\Delta period(i)})$ is minimized.

2-Sp bi P: Splitting bi criteria – Binary search over latency: at each step choose split that minimizes $max_{i\in\{j,j'\}}(\frac{\Delta latency}{\Delta period(j)})$ within the authorized latency increase.

$\Delta latency$: $\mathcal{L}$ after split - $\mathcal{L}$ before split

$\Delta period$: $\mathcal{P}(j)$ before split - $\mathcal{P}(j)$ after split

Minimizing Period for a Fixed Latency

2-Sp mono L: Splitting mono-criterion – Similar to **2-Sp mono P** with different break condition: splitting is performed as long as fixed latency is not exceeded.

2-Sp bi L: Splitting bi criteria – Similar to **2-Sp mono L**, but at each step choose solution that minimizes $max_{i\in\{j,j'\}}(\frac{\Delta latency}{\Delta period(i)})$ while fixed latency is not exceeded.

Back