Multi-criteria Scheduling of Pipeline Workflows

Anne Benoit Harald Kosch* Veronika Rehn-Sonigo
Yves Robert

GRAAL team, LIP
École Normale Supérieure de Lyon
France

*University of Passau
Germany

Working Group
December 6th, 2007

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
communication homogeneous 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
communication homogeneous platforms

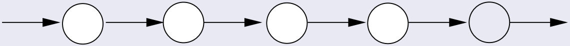
Introduction and motivation

- 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
communication homogeneous platforms

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Multi-criteria?

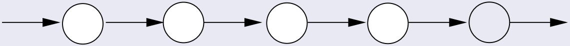
Period: time interval between the beginning of execution of two consecutive data sets

Latency: maximal time elapsed between beginning and end of execution of a data set

Bi-criteria!

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Multi-criteria?

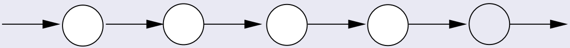
Period: time interval between the beginning of execution of two consecutive data sets

Latency: maximal time elapsed between beginning and end of execution of a data set

Bi-criteria!

Multi-criteria scheduling of workflows

Workflow



Several consecutive data-sets enter the application graph.

Multi-criteria?

Period: time interval between the beginning of execution of two consecutive data sets

Latency: maximal time elapsed between beginning and end of execution of a data set

Bi-criteria!

Why restrict to pipelines?

Pipeline: linear application graph

Chains-on-chains partitioning problem

- no communications
- identical processors

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

5 7 3 4 | 8 1 3 8 | 2 9 7 | 3 5 2 3 6

$T_{\text{period}} = 20$

If processors have different speeds?

Why restrict to pipelines?

Pipeline: linear application graph

Chains-on-chains partitioning problem
- no communications
- identical processors

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

5 7 3 4 | 8 1 3 8 | 2 9 7 | 3 5 2 3 6

$T_{\text{period}} = 20$

If processors have different speeds?

Why restrict to pipelines?

Pipeline: linear application graph

Chains-on-chains partitioning problem

- no communications
- identical processors

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

5 7 3 4 | 8 1 3 8 | 2 9 7 | 3 5 2 3 6

$T_{\text{period}} = 20$

If processors have different speeds?

Why restrict to pipelines?

Pipeline: linear application graph

Chains-on-chains partitioning problem
- no communications
- identical processors

Load-balance **contiguous** tasks

5 7 3 4 8 1 3 8 2 9 7 3 5 2 3 6

With $p = 4$ identical processors?

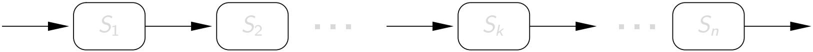
5 7 3 4 | 8 1 3 8 | 2 9 7 | 3 5 2 3 6

$T_{\text{period}} = 20$

If processors have different speeds?

Rule of the game

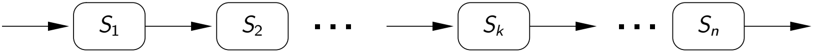
- Map each pipeline stage on a single processor
- Goal: minimize period AND minimize latency
- Several mapping strategies



The pipeline application

Rule of the game

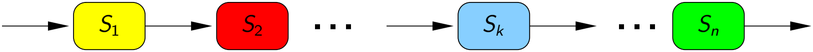
- Map each pipeline stage on a single processor
- Goal: minimize period AND minimize latency
- Several mapping strategies



The pipeline application

Rule of the game

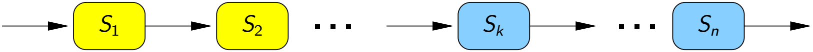
- Map each pipeline stage on a single processor
- Goal: minimize period AND minimize latency
- Several mapping strategies



ONE-TO-ONE MAPPING

Rule of the game

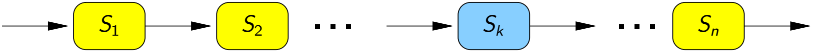
- Map each pipeline stage on a single processor
- Goal: minimize period AND minimize latency
- Several mapping strategies



INTERVAL MAPPING

Rule of the game

- Map each pipeline stage on a single processor
- Goal: minimize period AND minimize latency
- Several mapping strategies



General Mapping

Major contributions

Theory
- Definition of bi-criteria mapping
- Problem complexity
- Linear programming formulation

Practice
- Heuristics for INTERVAL MAPPING on clusters
- Experiments to compare heuristics and evaluate their performance
- Simulation of a real world application

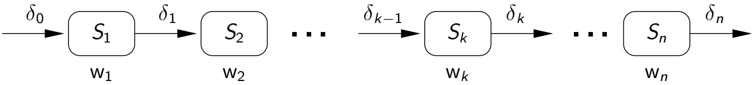
Major contributions

Theory	Definition of bi-criteria mapping Problem complexity Linear programming formulation
Practice	Heuristics for INTERVAL MAPPING on clusters Experiments to compare heuristics and evaluate their performance Simulation of a real world application

Outline

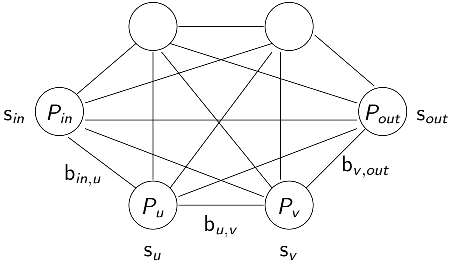
1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

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

The platform



- p processors P_u, $1 \leq u \leq$ p, fully interconnected
- s_u: speed of processor P_u
- bidirectional link $\text{link}_{u,v} : P_u \rightarrow P_v$, bandwidth $b_{u,v}$
- one-port model: 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

In this talk we restrict to *Communication Homogeneous* platforms!

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

In this talk we restrict to *Communication Homogeneous* platforms!

Mapping problem: Interval Mapping

- 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 $e_m = \text{n}$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

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

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

Mapping problem: Interval Mapping

- 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 $e_m = \text{n}$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

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

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

Mapping problem: INTERVAL MAPPING

- 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 $e_m = \text{n}$)
- Interval I_j mapped onto processor $P_{\text{alloc}(j)}$

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

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

Objective function?

Mono-criterion

- Minimize T_{period}
- Minimize T_{latency}

Bi-criteria

- How to define it?
  Minimize $\alpha.T_{\text{period}} + \beta.T_{\text{latency}}$?
- Values which are not comparable

- Minimize T_{period} for a fixed latency
- Minimize T_{latency} for a fixed period

Objective function?

Mono-criterion

- Minimize T_{period}
- Minimize T_{latency}

Bi-criteria

- How to define it?
  Minimize $\alpha.T_{\text{period}} + \beta.T_{\text{latency}}$?
- Values which are not comparable

- Minimize T_{period} for a fixed latency
- Minimize T_{latency} for a fixed period

Objective function?

Mono-criterion

- Minimize T_{period}
- Minimize T_{latency}

Bi-criteria

- How to define it?
  Minimize $\alpha.T_{\text{period}} + \beta.T_{\text{latency}}$?
- Values which are not comparable

- Minimize T_{period} for a fixed latency
- Minimize T_{latency} for a fixed period

Objective function?

Mono-criterion

- Minimize T_{period}
- Minimize T_{latency}

Bi-criteria

- How to define it?
  Minimize $\alpha.T_{\text{period}} + \beta.T_{\text{latency}}$?
- Values which are not comparable

- Minimize T_{period} for a fixed latency
- Minimize T_{latency} for a fixed period

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

Complexity results

Lemma

The optimal mapping which minimizes latency can be determined in polynomial time.

Assign whole pipeline to fastest processor!
No communications to pay in this case.

Complexity results

Lemma

The optimal mapping which minimizes latency can be determined in polynomial time.

Assign whole pipeline to fastest processor!
No communications to pay in this case.

Complexity results

Minimize the period?

Chains-on-chains problem with different speed processors!

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 $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p$, with $\mathcal{I}_k = [d_k, e_k]$ and $d_k \leq e_k$ for $1 \leq k \leq p$, $d_1 = 1$, $d_{k+1} = e_k + 1$ for $1 \leq k \leq p-1$ and $e_p = n$, and a permutation σ of $\{1, 2, \ldots, p\}$, such that

$$\max_{1 \leq k \leq p} \frac{\sum_{i \in \mathcal{I}_k} a_i}{s_{\sigma(k)}} \leq K \quad ?$$

Complexity results

Minimize the period?

Chains-on-chains problem with different speed processors!

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 $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p$, with $\mathcal{I}_k = [d_k, e_k]$ and $d_k \leq e_k$ for $1 \leq k \leq p$, $d_1 = 1$, $d_{k+1} = e_k + 1$ for $1 \leq k \leq p-1$ and $e_p = n$, and a permutation σ of $\{1, 2, \ldots, p\}$, such that

$$\max_{1 \leq k \leq p} \frac{\sum_{i \in \mathcal{I}_k} a_i}{s_{\sigma(k)}} \leq K \quad ?$$

Complexity results

Minimize the period?

Chains-on-chains problem with different speed processors!

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 $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p$, with $\mathcal{I}_k = [d_k, e_k]$ and $d_k \leq e_k$ for $1 \leq k \leq p$, $d_1 = 1$, $d_{k+1} = e_k + 1$ for $1 \leq k \leq p-1$ and $e_p = n$, and a permutation σ of $\{1, 2, \ldots, p\}$, such that

$$\max_{1 \leq k \leq p} \frac{\sum_{i \in \mathcal{I}_k} a_i}{s_{\sigma(k)}} \leq K \quad ?$$

Complexity results

Theorem 1

The HETERO-1D-PARTITION-DEC problem is NP-complete.

Involved reduction

Theorem 2

The period minimization problem for pipeline graphs is NP-complete.

Direct consequence from Theorem 1

All bi-criteria optimization problems are NP-complete on *Communication Homogeneous* platforms.

Complexity results

Theorem 1

The HETERO-1D-PARTITION-DEC problem is NP-complete.

Involved reduction

Theorem 2

The period minimization problem for pipeline graphs is NP-complete.

Direct consequence from Theorem 1

All bi-criteria optimization problems are NP-complete on *Communication Homogeneous* platforms.

Complexity results

Theorem 1

The HETERO-1D-PARTITION-DEC problem is NP-complete.

Involved reduction

Theorem 2

The period minimization problem for pipeline graphs is NP-complete.

Direct consequence from Theorem 1

All bi-criteria optimization problems are NP-complete on *Communication Homogeneous* platforms.

Complexity results

Theorem 1

The HETERO-1D-PARTITION-DEC problem is NP-complete.

Involved reduction

Theorem 2

The period minimization problem for pipeline graphs is NP-complete.

Direct consequence from Theorem 1

All bi-criteria optimization problems are NP-complete on *Communication Homogeneous* platforms.

Complexity results

Theorem 1

The HETERO-1D-PARTITION-DEC problem is NP-complete.

Involved reduction

Theorem 2

The period minimization problem for pipeline graphs is NP-complete.

Direct consequence from Theorem 1

All bi-criteria optimization problems are NP-complete on *Communication Homogeneous* platforms.

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

Integer linear programming

- 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

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 procs and links:

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

Constraints on intervals:

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

Linear program: constraints

Constraints on procs 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

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

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

Min period with fixed latency

$T_{\text{opt}} = T_{\text{period}}$

T_{latency} is fixed

Min latency with fixed period

$T_{\text{opt}} = T_{\text{latency}}$

T_{period} is fixed

Linear program: constraints

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

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

Min period with fixed latency

$$T_{\text{opt}} = T_{\text{period}}$$

T_{latency} is fixed

Min latency with fixed period

$$T_{\text{opt}} = T_{\text{latency}}$$

T_{period} is fixed

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

Heuristics

- Target clusters: *Communication Homogeneous* platforms and INTERVAL MAPPING
- n stages
- p processors

Two sets of heuristics

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

Minimizing Latency for a Fixed Period (1/2)

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, 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-Explo mono: 3-Exploration mono-criterion – Select used processor j with largest period and split its interval into three parts.

3-Explo bi: 3-Exploration 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.

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.

$\Delta latency$: T_{latency} after split - T_{latency} before split

$\Delta period$: $T_{\text{period}}(j)$ before split - $T_{\text{period}}(j)$ after split

Minimizing Period for a Fixed Latency

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

Sp bi L: Splitting bi criteria – Similar to **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.

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

Plan of experiments

- Assess performance of polynomial heuristics
- Random applications, $n \in \{5, 10, 20, 40\}$ stages
- Random *Communication Homogeneous* platforms, $p = 10$ and $p = 100$ processors
- $b = 10$, proc. speed between 1 and 20
- Relevant parameters: ratios $\frac{\delta}{b}$ and $\frac{w}{s}$
- Average over 50 similar random appli/platform pairs

Plan of experiments

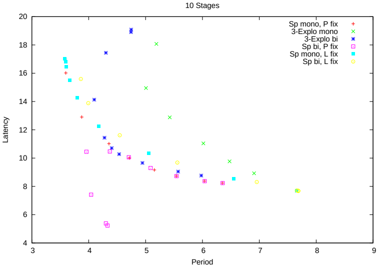
- Assess performance of polynomial heuristics

- Random applications, $n \in \{5, 10, 20, 40\}$ stages
- Random *Communication Homogeneous* platforms, $p = 10$ and $p = 100$ processors

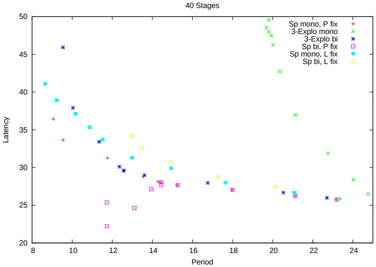
- $b = 10$, proc. speed between 1 and 20
- Relevant parameters: ratios $\frac{\delta}{b}$ and $\frac{w}{s}$
- Average over 50 similar random appli/platform pairs

Experiment 1 - balanced comm/comp, hom comm

- communication time $\delta_i = 10$
- computation time between 1 and 20
- 10 processors



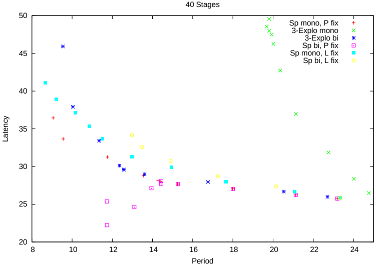
10 stages.
☺ Sp bi P
☹ 3-Explo mono



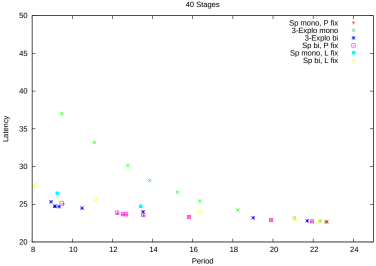
40 stages.
☺ Sp mono P
☹ 3-Explo mono

Experiment 1 - balanced comm/comp, hom comm

- communication time $\delta_i = 10$
- computation time between 1 and 20
- 10 vs. 100 processors



40 stages, 10 procs.
☺ Sp mono P
☹ 3-Explo mono



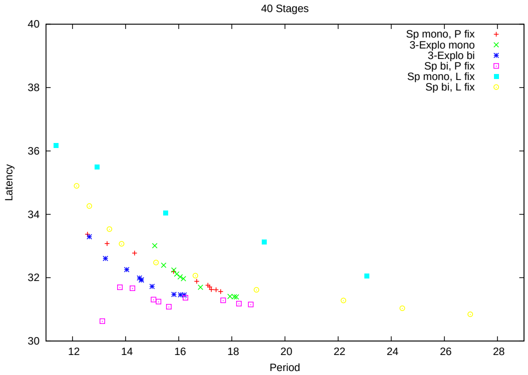
40 stages, 100 procs.
☺ 3 Explo bi
☹ 3-Explo mono

Experiment 2 - balanced comm/comp, het comm

- communication time between 1 and 100
- computation time between 1 and 20

100 processors.
40 stages.

☺ Sp bi P
☹ 3-Explo mono



Experiment 3 - large computations

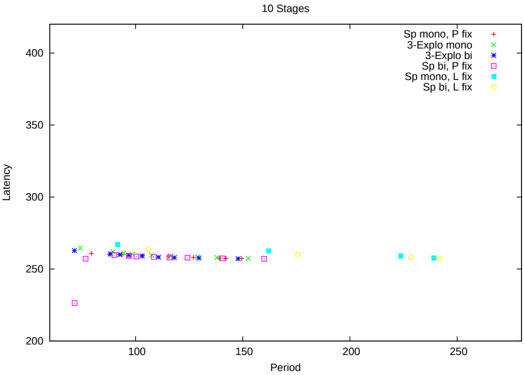
- communication time between 1 and 20
- computation time between 10 and 1000

100 processors.

5 stages.

☺ Sp bi P

☹ Sp mono L



Experiment 4 - small computations

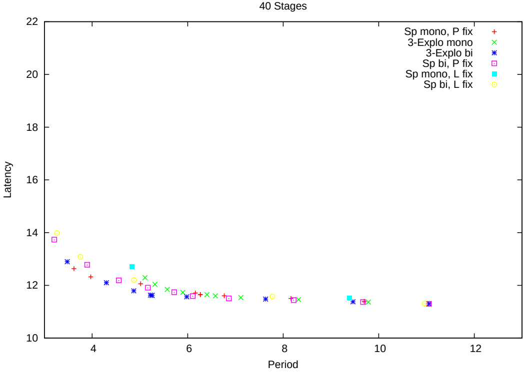
- communication time between 1 and 20
- computation time between 0.01 and 10

100 processors.

5 stages.

☺ 3-Explo bi

☹ Sp mono L



Failure Thresholds for 10 procs

Failure threshold: largest fixed value (latency or period) for which a heuristic does not find a solution.

Exp.	Heuristic	Number of stages			
		5	10	20	40
E1	Sp mono P	3.0	3.3	5.0	5.0
	3-Explo mono	3.0	4.7	9.0	18.0
	3-Explo bi	3.0	4.0	5.0	5.0
	Sp bi P	3.3	3.3	6.0	10.0
	Sp mono L	4.5	6.0	13.0	25.0
	Sp bi L	4.5	6.0	13.0	25.0
E3	Sp mono P	50.0	70.0	100.0	250.0
	3-Explo mono	50.0	140.0	450.0	950.0
	3-Explo bi	50.0	90.0	250.0	400.0
	Sp bi P	100.0	140.0	300.0	650.0
	Sp mono L	140.0	270.0	500.0	1000.0
	Sp bi L	140.0	270.0	500.0	1000.0

Small values are good !

☺ Sp mono P
☹ 3-Explo mono

Failure Thresholds for 10 procs

Failure threshold: largest fixed value (latency or period) for which a heuristic does not find a solution.

Exp.	Heuristic	Number of stages			
		5	10	20	40
E1	Sp mono P	3.0	3.3	5.0	5.0
	3-Explo mono	3.0	4.7	9.0	18.0
	3-Explo bi	3.0	4.0	5.0	5.0
	Sp bi P	3.3	3.3	6.0	10.0
	Sp mono L	4.5	6.0	13.0	25.0
	Sp bi L	4.5	6.0	13.0	25.0
E3	Sp mono P	50.0	70.0	100.0	250.0
	3-Explo mono	50.0	140.0	450.0	950.0
	3-Explo bi	50.0	90.0	250.0	400.0
	Sp bi P	100.0	140.0	300.0	650.0
	Sp mono L	140.0	270.0	500.0	1000.0
	Sp bi L	140.0	270.0	500.0	1000.0

Small values are good !

☺ Sp mono P

☹ 3-Explo mono

Failure Thresholds for 10 procs

Failure threshold: largest fixed value (latency or period) for which a heuristic does not find a solution.

Exp.	Heuristic	Number of stages			
		5	10	20	40
E1	Sp mono P	3.0	3.3	5.0	5.0
	3-Explo mono	3.0	4.7	9.0	18.0
	3-Explo bi	3.0	4.0	5.0	5.0
	Sp bi P	3.3	3.3	6.0	10.0
	Sp mono L	4.5	6.0	13.0	25.0
	Sp bi L	4.5	6.0	13.0	25.0
E3	Sp mono P	50.0	70.0	100.0	250.0
	3-Explo mono	50.0	140.0	450.0	950.0
	3-Explo bi	50.0	90.0	250.0	400.0
	Sp bi P	100.0	140.0	300.0	650.0
	Sp mono L	140.0	270.0	500.0	1000.0
	Sp bi L	140.0	270.0	500.0	1000.0

Small values are good !

☺ Sp mono P

☹ 3-Explo mono

Failure Thresholds for 10 procs

Failure threshold: largest fixed value (latency or period) for which a heuristic does not find a solution.

Exp.	Heuristic	Number of stages			
		5	10	20	40
E1	Sp mono P	3.0	3.3	5.0	5.0
	3-Explo mono	3.0	4.7	9.0	18.0
	3-Explo bi	3.0	4.0	5.0	5.0
	Sp bi P	3.3	3.3	6.0	10.0
	Sp mono L	4.5	6.0	13.0	25.0
	Sp bi L	4.5	6.0	13.0	25.0
E3	Sp mono P	50.0	70.0	100.0	250.0
	3-Explo mono	50.0	140.0	450.0	950.0
	3-Explo bi	50.0	90.0	250.0	400.0
	Sp bi P	100.0	140.0	300.0	650.0
	Sp mono L	140.0	270.0	500.0	1000.0
	Sp bi L	140.0	270.0	500.0	1000.0

Small values are good !

☺ Sp mono P
☹ 3-Explo mono

Summary of experiments

- Performance of bi-criterion heuristics highly depends on the number of available processors.
- Small number of processors:
  - Sp mono P and Sp mono L
  - Small latencies: Sp bi P
- Increasing number of processors:
  - Sp bi P and Sp bi L

Summary of experiments

- Performance of bi-criterion heuristics highly depends on the number of available processors.
- Small number of processors:
  - Sp mono P and Sp mono L
  - Small latencies: Sp bi P
- Increasing number of processors:
  - Sp bi P and Sp bi L

Summary of experiments

- Performance of bi-criterion heuristics highly depends on the number of available processors.
- Small number of processors:
  - Sp mono P and Sp mono L
  - Small latencies: Sp bi P
- Increasing number of processors:
  - Sp bi P and Sp bi L

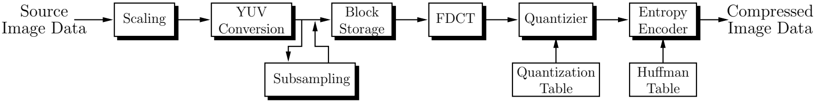
Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

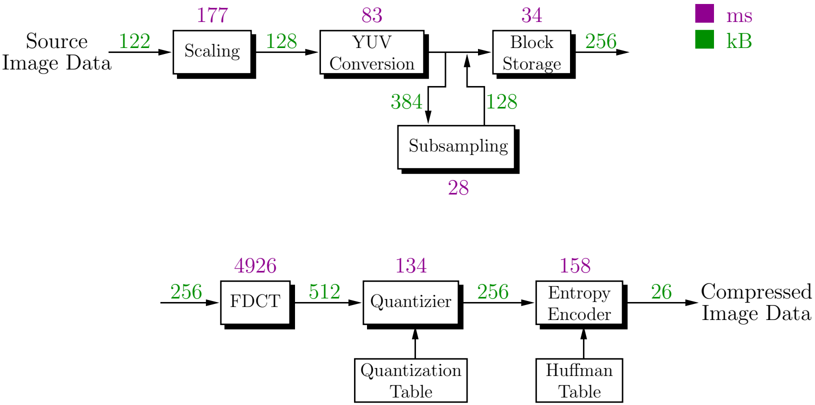
Real World Application

The JPEG encoder

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



JPEG Encoder



Simulation environment

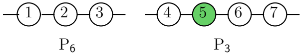
- MPI application
- Message passing + sleep()
- Homogeneous processors (Salle Europe)
- Simulation of heterogenity
- Mapping 7 stages on 10 processors

Influence of the fixed parameter on the solution

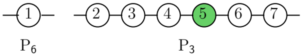
LP solutions:

minimize latency

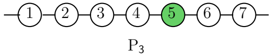
$P_{fix} = 310$ $L_{opt} = 337,575$



$P_{fix} = 320$ $L_{opt} = 336,729$

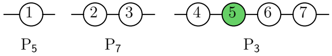


$P_{fix} = 330$ $L_{opt} = 322,700$

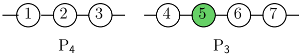


minimize period

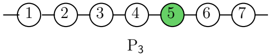
$L_{fix} = 370$ $P_{opt} = 307,319$



$L_{fix} = 340$ $P_{opt} = 307,319$



$L_{fix} = 330$ $P_{opt} = 322,700$



Overview of the different solutions

Minimize latency with $T_{period} = 310$

Algorithm	Intervals	Processors	Latency	Simu
LP	[1-3][4-7]	6,3	337,575	
Sp mono P	[1-3][4-7]	4,3	337,575	308,2
3-Explo mono	[1][2-3][4-7]	4,6,3	350,57	310,02
3-Explo mono	[1][2-3][4-7]	6,4,3	350,57	310,06
Sp bi P	does not succeed		(322,7)	307,02

Overview of the different solutions

Minimize period with $T_{\text{latency}} = 370$

Algorithm	Intervals	Processors	Period	Simu
LP	[1][2-3][4-7]	5,7,3	307,319	
Sp mono L	[1-3][4-7]	4,3	307,319	308,15
Sp bi L	[1-7]	3	322,7	307,00

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Application Simulation
7. Conclusion

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

- Bi-criteria mapping problem on *Communication Homogeneous* platforms
- Pipeline structured applications
- Complexity study
- Linear programming formulation

Practical side

- Design of several polynomial heuristics
- Extensive simulations to compare their performance
- Simulation of a real world application
- Evaluation

Future work

Theory

- Extension to stage replication
- Extension to fork, fork-join and tree workflows
- Multi-criteria: reliability in addition to period and latency

Practice

- Real experiments on heterogeneous clusters with bigger pipeline applications, using MPI
- Comparison of effective performance against theoretical performance