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

Alpage Meeting
January 31, 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

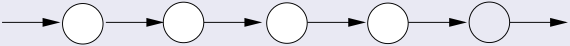
Introduction and motivation

- 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: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

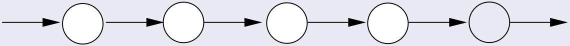
Latency: maximal time elapsed between beginning and end of execution of a data set

Reliability: 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: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

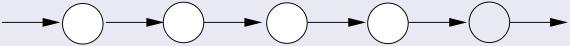
Latency: maximal time elapsed between beginning and end of execution of a data set

Reliability: 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: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

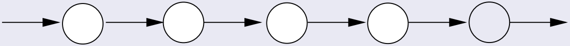
Latency: maximal time elapsed between beginning and end of execution of a data set

Reliability: 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: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

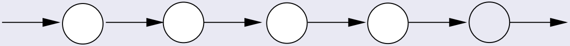
Latency: maximal time elapsed between beginning and end of execution of a data set

Reliability: 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: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

Reliability: probability of failure of the application (i.e. some data-sets will not be processed)

Multi-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? Problem: NP-hard

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? Problem: NP-hard

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? Problem: NP-hard

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

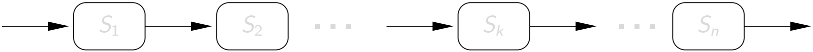
$T_{\text{period}} = 20$

If processors have different speeds? Problem: NP-hard

Rule of the game

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

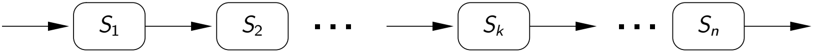
- Several mapping strategies



The pipeline application

Rule of the game

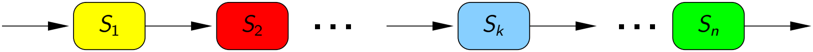
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



The pipeline application

Rule of the game

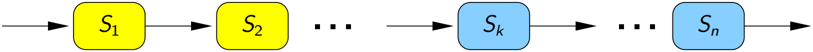
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



ONE-TO-ONE MAPPING

Rule of the game

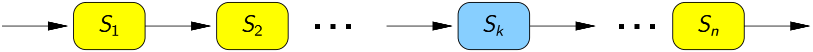
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



INTERVAL MAPPING

Rule of the game

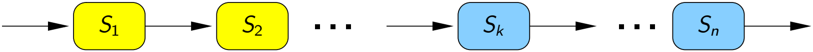
- Map each pipeline stage onto one or more processors
- Goal: minimize period/latency and maximize reliability
- Several mapping strategies



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 to compare heuristics and evaluate their performance
Simulation of a real world application (JPEG encoder)

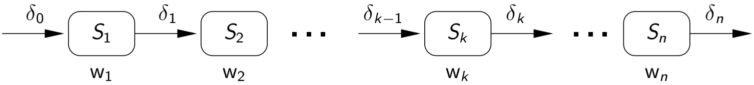
Major contributions

Theory
- Definition of multi-criteria mappings
- 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 (JPEG encoder)

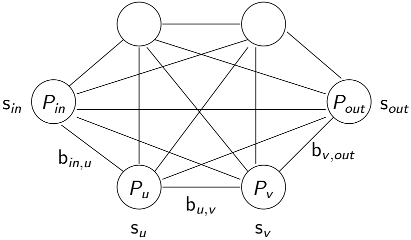
Outline

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

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 $b_{u,v}$
- 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: 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

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

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

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

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

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

Mapping problem: INTERVAL MAPPING

- Several consecutive stages onto the same processor
- 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 $e_m = \mathrm{n}$)
- Interval I_j mapped onto set of processors $P_{\mathrm{alloc}(j)}$
- $k_j = |\mathrm{alloc}(j)|$ processors executing I_j, $k_j \geq 1$.

Objective function?

Mono-criterion

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

Objective function?

Mono-criterion

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

Multi-criteria

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

Objective function?

Mono-criterion

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

Multi-criteria

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

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

Objective function?

Mono-criterion

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

Bi-criteria

- Period and Latency:
- Minimize T_{period} for a fixed latency
- Minimize T_{latency} for a fixed period

Objective function?

Mono-criterion

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

Bi-criteria

- Failure and Latency:
- Minimize T_{failure} for a fixed latency
- Minimize T_{latency} 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)

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

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

Interval Mapping problem - Period/Latency

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

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

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

Interval Mapping problem - Period/Latency

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

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

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

Interval Mapping problem - 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)

$$T_{\text{latency}} = \sum_{1 \le j \le p} \left\{ k_j \times \frac{\delta_{d_j - 1}}{\mathrm{b}} + \frac{\sum_{i=d_j}^{e_j} \mathrm{w}_i}{\min_{u \in \text{alloc}(j)} (\mathrm{s}_u)} \right\} + \frac{\delta_n}{\mathrm{b}}$$

$$T_{\text{failure}} = 1 - \prod_{1 \le j \le p} \left(1 - \prod_{u \in \text{alloc}(j)} \mathrm{fp}_u\right)$$

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)

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

$$T_{\text{failure}} = 1 - \prod_{1 \leq j \leq p} \left(1 - \prod_{u \in \text{alloc}(j)} \mathsf{fp}_u\right)$$

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)

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

$$T_{\text{failure}} = 1 - \prod_{1 \le j \le p} (1 - \prod_{u \in \text{alloc}(j)} \mathsf{fp}_u)$$

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 communications, identical for each mapping.

Complexity results: Latency - Com Hom

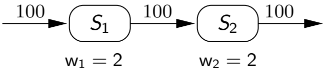
Lemma

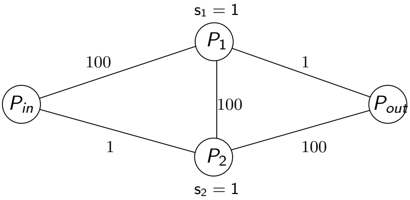
On *Fully Homogeneous* and *Communication Homogeneous* platforms, the optimal interval mapping which minimizes latency can be determined in polynomial time.

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

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

Lemma

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

Reduction from the Traveling Salesman Problem TSP

Still an open problem for interval mappings
(but we conjecture it is NP-hard)

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

Lemma

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

Reduction from the Traveling Salesman Problem TSP

Still an open problem for interval mappings
(but we conjecture it is NP-hard)

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

Lemma

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

Reduction from the Traveling Salesman Problem TSP

Still an open problem for interval mappings
(but we conjecture it is NP-hard)

Complexity results: Period

Minimize the period?

Chains-on-chains problem for *Fully Hom.* platforms: polynomial

Com. Hom.: Chains-on-chains with different speed processors!

Definition (HETERO-1D-PARTITION-DEC)

Given n elements $a_1, a_2, \dots, a_n$, p values $s_1, s_2, \dots, s_p$ and a bound K, can we find a partition of [1..n] into p intervals $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_p$, 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 $\{1, 2, \dots, p\}$, 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: Period

Minimize the period?

Chains-on-chains problem for *Fully Hom.* platforms: polynomial

Com. Hom.: Chains-on-chains 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: Period

Minimize the period?

Chains-on-chains problem for *Fully Hom.* platforms: polynomial

Com. Hom.: Chains-on-chains 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: Period

Minimize the period?

Chains-on-chains problem for *Fully Hom.* platforms: polynomial

Com. Hom.: Chains-on-chains 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: Period

Theorem 1

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

Involved reduction

Theorem 2

The period minimization problem for interval mapping of pipeline graphs on *Communication Homogeneous* platforms is NP-complete.

Direct consequence from Theorem 1

Complexity results: Period

Theorem 1

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

Involved reduction

Theorem 2

The period minimization problem for interval mapping of pipeline graphs on *Communication Homogeneous* platforms is NP-complete.

Direct consequence from Theorem 1

Complexity results: Period

Theorem 1

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

Involved reduction

Theorem 2

The period minimization problem for interval mapping of pipeline graphs on *Communication Homogeneous* platforms is NP-complete.

Direct consequence from Theorem 1

Complexity results: Period

Theorem 1

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

Involved reduction

Theorem 2

The period minimization problem for interval mapping of pipeline graphs on *Communication Homogeneous* platforms is NP-complete.

Direct consequence from Theorem 1

Complexity results: Reliability

Lemma

Minimizing the failure probability can be done in polynomial time.

- Formula computing global failure probability
- 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
- 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

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

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

- 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 $\mathcal{FP}$

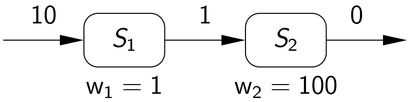
- Find k minimum, such that

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

- 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
- $T_{\text{latency}} \leq 22$, minimize T_{failure}

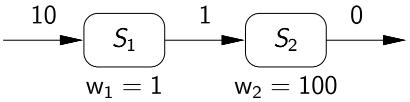


- One interval: $T_{\text{failure}} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $T_{\text{failure}} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$

- Open complexity (probably NP-hard)

Complexity results - Latency/Failure

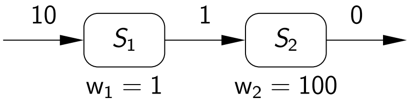
- *Communication Homogeneous-Failure Heterogeneous*
- Lemma NoSplit not true: example
- One slow and reliable processor, $s = 1$, $fp = 0.1$
- Ten fast and unreliable processors, $s = 100$, $fp = 0.8$
- $T_{\text{latency}} \leq 22$, minimize T_{failure}



- One interval: $T_{\text{failure}} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $T_{\text{failure}} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$
- 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, $s = 100$, $fp = 0.8$
- $T_{\text{latency}} \leq 22$, minimize T_{failure}

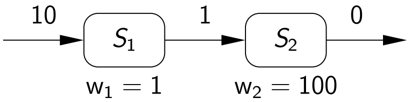


- One interval: $T_{\text{failure}} = (1 - (1 - 0.8^2)) = 0.64$
- Two intervals: $T_{\text{failure}} = 1 - (1 - 0.1).(1 - 0.8^{10}) < 0.2$

- 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, $s = 100$, $fp = 0.8$
- $T_{\text{latency}} \leq 22$, minimize T_{failure}



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

- 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 $\text{b}_{j,out} = 1$

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 $\mathrm{b}_{j,out} = 1$

Outline

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:

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

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 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

Heuristics

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

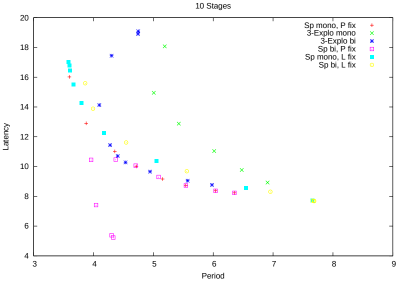
Two sets of heuristics

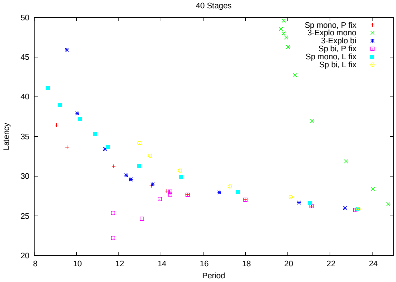
- Minimizing latency for a fixed period
- Minimizing period for a fixed latency

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

Heuristics comparison

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



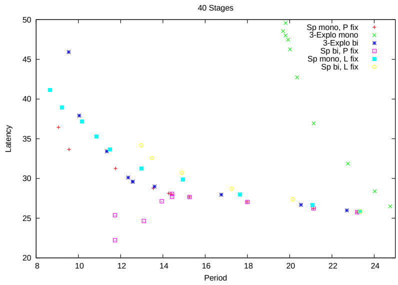


10 stages
☺ Sp bi P
☹ 3-Explo mono

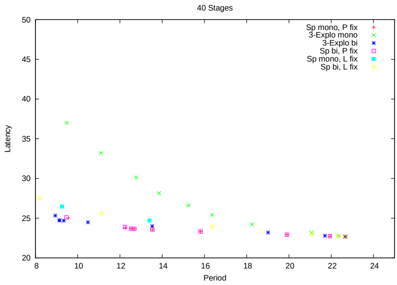
40 stages
☺ Sp mono P
☹ 3-Explo mono

Heuristics comparison

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



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

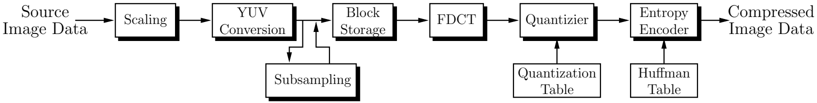


40 stages, 100 procs
☺ 3 Explo bi
☹ 3-Explo mono

Real World Application

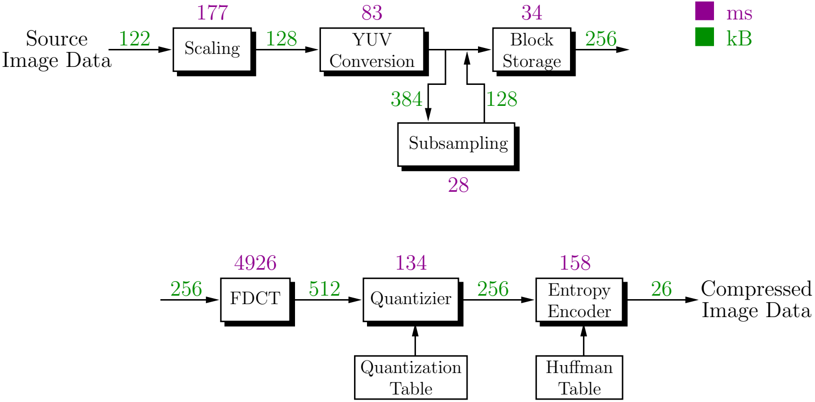
The JPEG encoder

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



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

JPEG Encoder



Simulation environment

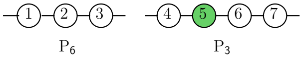
- MPI application
- Message passing + sleep()
- Homogeneous processors (Salle Europe)
- Simulation of heterogeneity
- Mapping 7 stages on 10 processors

Influence of the fixed parameter on the solution

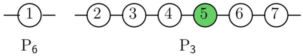
LP solutions:

minimize latency

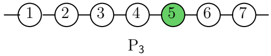
$P_{fix} = 310$ $L_{opt} = 337,575$



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

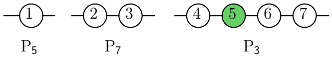


$P_{fix} = 330$ $L_{opt} = 322,700$

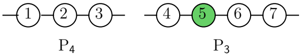


minimize period

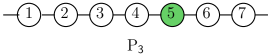
$L_{fix} = 370$ $P_{opt} = 307,319$



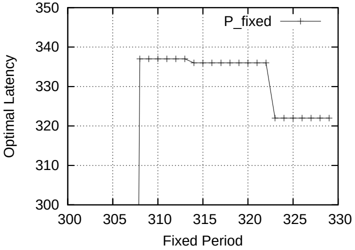
$L_{fix} = 340$ $P_{opt} = 307,319$



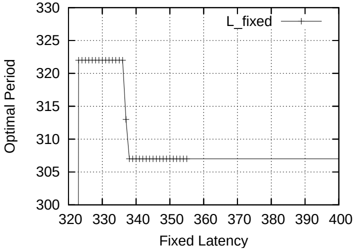
$L_{fix} = 330$ $P_{opt} = 322,700$



Bucket behavior of LP solutions

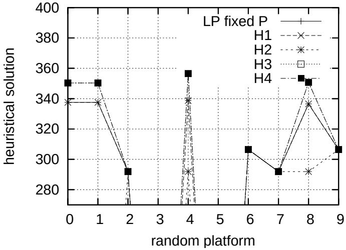


(a) Fixed P.

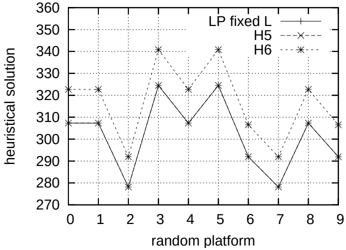


(b) Fixed L.

Behavior of heuristics (compared to LP)

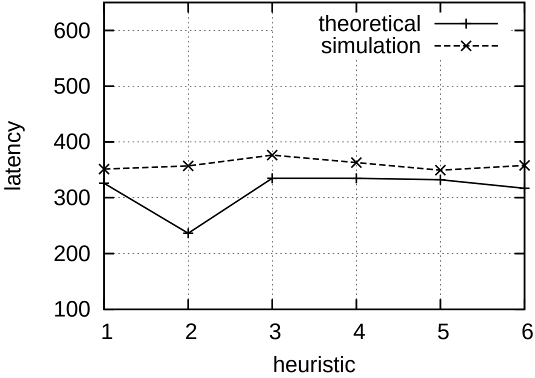


(c) Fixed P = 310.



(d) Fixed L = 370.

Comparison theory/experience



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
- 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

Practice

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