Multi-criteria Scheduling of Pipeline Workflows

Anne Benoit Veronika Rehn-Sonigo Yves Robert

GRAAL team, LIP
École Normale Supérieure de Lyon
France

Heteropar'2007

Introduction and motivation

Mapping pipeline skeletons onto
communication homogeneous platforms

- Previous talk: theoretical complexity results with no communications
- Now, more realistic platforms, but no replication nor data-parallelism
- Heuristics and experiments

Introduction and motivation

Mapping pipeline skeletons onto communication homogeneous platforms

- Previous talk: theoretical complexity results with no communications
- Now, more realistic platforms, but no replication nor data-parallelism
- Heuristics and experiments

Introduction and motivation

Mapping pipeline skeletons onto communication homogeneous platforms

- Previous talk: theoretical complexity results with no communications
- Now, more realistic platforms, but no replication nor data-parallelism
- Heuristics and experiments

Why restrict to pipelines?

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

- Extensions (done)
  - with communications
  - with heterogeneous processors/links
  - with different optimization criteria
  - goal: assess complexity, design heuristics

- Extensions (current work)
  - deal with DAGs

Why restrict to pipelines?

- Chains-on-chains partitioning problem
  - no communications
  - identical processors
- Extensions (done)
  - with communications
  - with heterogeneous processors/links
  - with different optimization criteria
  - goal: assess complexity, design heuristics
- Extensions (current work)
  - deal with DAGs

Why restrict to pipelines?

- Chains-on-chains partitioning problem
  - no communications
  - identical processors
- Extensions (done)
  - with communications
  - with heterogeneous processors/links
  - with different optimization criteria
  - goal: assess complexity, design heuristics
- Extensions (current work)
  - deal with DAGs

Chains-on-chains

Load-balance **contiguous** tasks

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

Chains-on-chains

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?

Chains-on-chains

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

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

Chains-on-chains

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

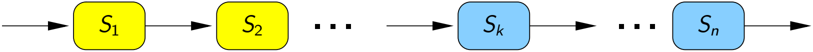
$$T_{\text{period}} = 20$$

- Back to Bokhari and Iqbal partitioning papers
- See survey by Pinar and Aykanat, JPDC 64, 8 (2004)
- If processors have different speeds?

Rule of the game

- Map each pipeline stage on a single processor
- Goal: minimize execution time AND minimize latency

- Interval Mapping



Major contributions

Theory Definition of bi-criteria mapping
Problem complexity

Practice Heuristics for INTERVAL MAPPING on clusters
Experiments to compare heuristics and evaluate their performance

Major contributions

Theory	Definition of bi-criteria mapping Problem complexity
Practice	Heuristics for Interval Mapping on clusters Experiments to compare heuristics and evaluate their performance

Outline

1 Framework

2 Complexity results

3 Heuristics

4 Experiments

5 Conclusion

Framework

- Application: n-stages pipeline
- Platform: p processors fully interconnected
- s_u: speed of processor P_u
- bidirectional link $\mathrm{link}_{u,v} : P_u \rightarrow P_v$, bandwidth $\mathrm{b}_{u,v}$
- one-port model: each processor can either send, receive or compute at any time-step

Different platforms

Fully Homogeneous – Identical processors ($\mathsf{s}_u = \mathsf{s}$) and links ($\mathsf{b}_{u,v} = \mathsf{b}$): typical parallel machines

Communication Homogeneous – Different-speed processors ($\mathsf{s}_u \neq \mathsf{s}_v$), identical links ($\mathsf{b}_{u,v} = \mathsf{b}$): networks of workstations, clusters

Fully Heterogeneous – Fully heterogeneous architectures, $\mathsf{s}_u \neq \mathsf{s}_v$ and $\mathsf{b}_{u,v} \neq \mathsf{b}_{u',v'}$: hierarchical platforms, grids

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

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

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

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

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

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

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. Heuristics
4. Experiments
5. 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 $a_1, a_2, \dots, a_n$, 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 $d_k \le e_k$ for $1 \le k \le p$, $d_1 = 1$, $d_{k+1} = e_k + 1$ for $1 \le k \le p - 1$ and $e_p = n$, and a permutation σ of $\{1, 2, \dots, p\}$, such that

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

Complexity results

Minimize the period?

Chains-on-chains problem with different speed processors!

Definition (Hetero-1D-Partition-Dec)

Given n elements $a_1, a_2, \ldots, a_\mathrm{n}$, p values $\mathrm{s}_1, \mathrm{s}_2, \ldots, \mathrm{s}_\mathrm{p}$ and a bound K, can we find a partition of $[1..\mathrm{n}]$ into p intervals $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_\mathrm{p}$, 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 $e_p = \mathrm{n}$, and a permutation σ of $\{1, 2, \ldots, \mathrm{p}\}$, such that

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

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

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

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

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

Outline

1. Framework
2. Complexity results
3. Heuristics
4. Experiments
5. Conclusion

Heuristics

- Target clusters: *Communication Homogeneous* platforms and INTERVAL MAPPING
- *n* stages, *p* processors
- Minimizing period NP-complete → bi-criteria problems NP-complete

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 all stages to 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 (splitting reduces period but increases latency).

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. Heuristics
4. Experiments
5. 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

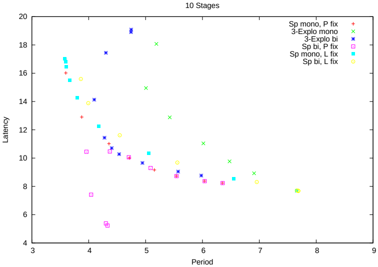
- Assess performance of polynomial heuristics

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

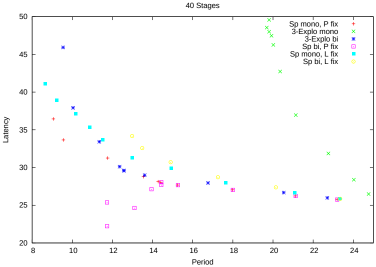
- $b = 10$, proc. speed between 1 and 20
- Relevant parameters: ratios $\frac{\delta}{b}$ and $\frac{w}{s}$
- Average over 50 similar random appli/platform pairs

Experiment 1 - balanced comm/comp, hom comm

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



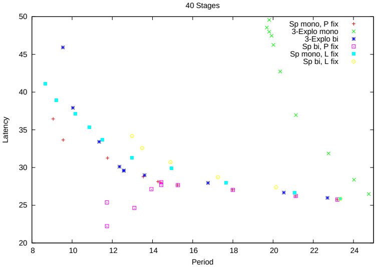
10 stages.
☺ Sp bi P
☹ 3-Explo mono



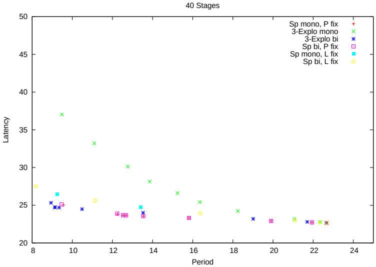
40 stages.
☺ Sp mono P
☹ 3-Explo mono

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



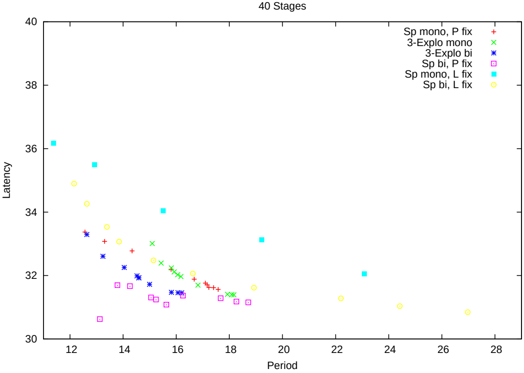
40 stages, 100 procs.
☺ 3 Explo bi
☹ 3-Explo mono

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

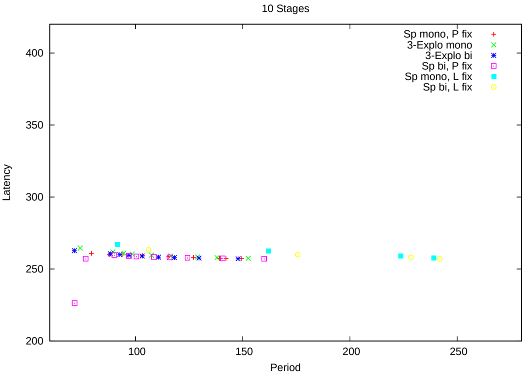
- communication time between 1 and 20
- computation time between 10 and 1000

100 processors.

5 stages.

☺ Sp bi P

☹ Sp mono L

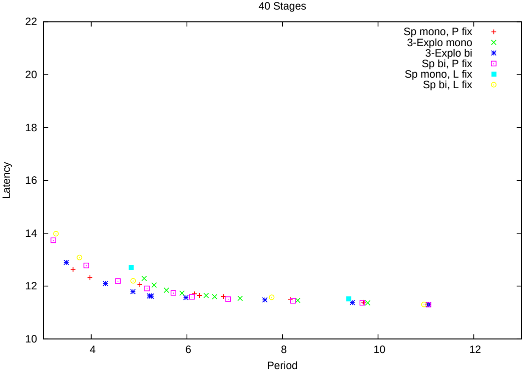


Experiment 4 - small computations

- 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 procesoors:
  - 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 procesoors:
  - 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 procesoors:
  - Sp bi P and Sp bi L

Outline

1. Framework
2. Complexity results
3. Heuristics
4. Experiments
5. 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

Practical side

- Design of several polynomial heuristics
- Extensive simulations to compare their performance

Future work

Short term

- Heuristics for *Fully Heterogeneous* platforms, with stage replication
- Extension to DAG-trees (a DAG which is a tree when un-oriented)

Longer term

- Real experiments on heterogeneous clusters, using an already-implemented skeleton library and MPI
- Comparison of effective performance against theoretical performance

Open problems

- Replication for fault-tolerance vs replication for parallelism
  - compute several time the same data-set in case of failure
  - uses more resources and does not decrease period or latency
  - increases robustness
- Energy savings
  - processors that can run at different frequencies
  - trade-off between energy consumption and speed
- Simultaneous execution of several (concurrent) workflows
  - competition for CPU and network resources
  - fairness between applications (stretch)
  - sensitivity to application/platform parameter changes