Mapping pipeline skeletons onto heterogeneous platforms

Anne Benoit and Yves Robert

GRAAL team, LIP
École Normale Supérieure de Lyon

May 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 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

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

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

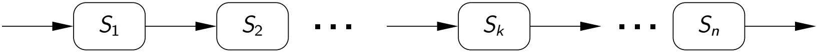
Rule of the game

- Map each pipeline stage on a single processor
- Goal: minimize execution time
- Several mapping strategies

The pipeline application

Rule of the game

- Map each pipeline stage on a single processor
- Goal: minimize execution time
- Several mapping strategies



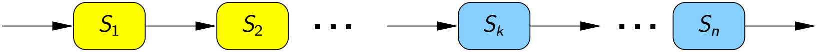
The pipeline application

Rule of the game

- Map each pipeline stage on a single processor
- Goal: minimize execution time
- Several mapping strategies

Rule of the game

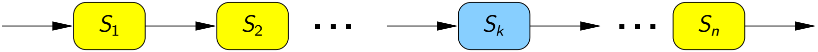
- Map each pipeline stage on a single processor
- Goal: minimize execution time
- Several mapping strategies



INTERVAL MAPPING

Rule of the game

- Map each pipeline stage on a single processor
- Goal: minimize execution time
- Several mapping strategies



General Mapping

Major contributions

Theory Formal approach to the problem

Problem complexity

Integer linear program for exact resolution

Practice Heuristics for INTERVAL MAPPING on clusters

Experiments to compare heuristics *and evaluate their absolute performance*

Major contributions

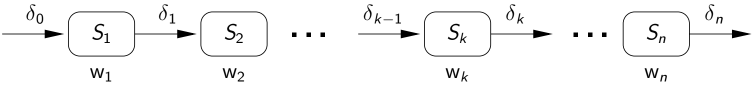
Theory
Formal approach to the problem
Problem complexity
Integer linear program for exact resolution

Practice
Heuristics for Interval Mapping on clusters
Experiments to compare heuristics *and evaluate their absolute performance*

Outline

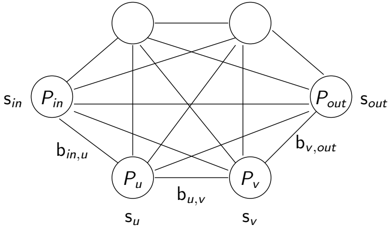
1. Framework
2. Complexity results
3. Heuristics
4. Experiments
5. Conclusion

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 $\mathcal{S}_{k+1}$

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

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

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

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 processor $P_{\mathrm{alloc}(j)}$
- Minimize the period:

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

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 processor $P_{\mathrm{alloc}(j)}$
- Minimize the period:

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

Outline

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

Complexity results

	Fully Hom.	Comm. Hom.
One-to-one Mapping		
Interval Mapping		
General Mapping		

Complexity results

	Fully Hom.	**Comm. Hom.**
One-to-one Mapping	polynomial	polynomial
Interval Mapping		
General Mapping		

- Binary search polynomial algorithm for ONE-TO-ONE MAPPING
-
-
-

Complexity results

	Fully Hom.	Comm. Hom.
One-to-one Mapping	polynomial	polynomial
Interval Mapping	polynomial	NP-complete
General Mapping		

- Binary search polynomial algorithm for One-to-one Mapping
- Dynamic programming algorithm for Interval Mapping on Hom. platforms (NP-hard otherwise)
-
-

Complexity results

	Fully Hom.	**Comm. Hom.**
One-to-one Mapping	polynomial	polynomial
Interval Mapping	polynomial	NP-complete
General Mapping	same complexity as Interval	

- Binary search polynomial algorithm for ONE-TO-ONE MAPPING
- Dynamic programming algorithm for INTERVAL MAPPING on Hom. platforms (NP-hard otherwise)
- General mapping: same complexity as INTERVAL MAPPING
-

Complexity results

	Fully Hom.	Comm. Hom.
One-to-one Mapping	polynomial	polynomial
Interval Mapping	polynomial	NP-complete
General Mapping	same complexity as Interval	

- Binary search polynomial algorithm for ONE-TO-ONE MAPPING
- Dynamic programming algorithm for INTERVAL MAPPING on Hom. platforms (NP-hard otherwise)
- General mapping: same complexity as INTERVAL MAPPING
- All problem instances NP-complete on *Fully Heterogeneous* platforms

One-to-one/Comm. Hom.: binary search algorithm

- Work with fastest n processors, numbered P_1 to P_n, where $\mathrm{s}_1 \leq \mathrm{s}_2 \leq \ldots \leq \mathrm{s}_\mathrm{n}$
- Mark all stages $\mathcal{S}_1$ to $\mathcal{S}_\mathrm{n}$ as free
- **For** $u = 1$ **to** n
  - Pick up any free stage $\mathcal{S}_k$ s.t. $\delta_{k-1}/\mathrm{b} + \mathrm{w}_k/\mathrm{s}_u + \delta_k/\mathrm{b} \leq T_\mathrm{period}$
  - Assign $\mathcal{S}_k$ to P_u, and mark $\mathcal{S}_k$ as already assigned
  - If no stage found return "failure"
- Proof: exchange argument

One-to-one/Comm. Hom.: binary search algorithm

- Work with fastest n processors, numbered P_1 to P_n, where $s_1 \leq s_2 \leq \ldots \leq s_n$
- Mark all stages $\mathcal{S}_1$ to $\mathcal{S}_n$ as free
- **For** $u = 1$ **to** n
  - Pick up any free stage $\mathcal{S}_k$ s.t. $\delta_{k-1}/b + w_k/s_u + \delta_k/b \leq T_{\text{period}}$
  - Assign $\mathcal{S}_k$ to P_u, and mark $\mathcal{S}_k$ as already assigned
  - If no stage found return "failure"

- Proof: exchange argument

Outline

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

Greedy heuristics

Target clusters: *Com. hom.* platforms and INTERVAL MAPPING

H1a-GR: random – fixed intervals

H1b-GRIL: random interval length

H2-GSW: biggest $\sum w$ – Place interval with most computations on fastest processor

H3-GSD: biggest $\delta_{in} + \delta_{out}$ – Intervals are sorted by communications $(\delta_{in} + \delta_{out})$
in: first stage of interval; $(out - 1)$: last one

H4-GP: biggest period on fastest processor – Balancing computation and communication: processors sorted by decreasing speed s_u; for current processor u, choose interval with biggest period $(\delta_{in} + \delta_{out})/\mathrm{b} + \sum_{i \in Interval} w_i/s_u$

Sophisticated heuristics

H5-BS121: binary search for ONE-TO-ONE MAPPING – optimal algorithm for ONE-TO-ONE MAPPING. When $p < n$, application cut in fixed intervals of length L.

H6-SPL: splitting intervals – Processors sorted by decreasing speed, all stages to first processor. At each step, select used proc j with largest period, split its interval (give fraction of stages to j'): minimize $max(period(j), period(j'))$ and split if maximum period improved.

H7a-BSL and H7b-BSC: binary search (longest/closest) – Binary search on period P: start with stage $s = 1$, build intervals (s, s') fitting on processors. For each u, and each $s' \geq s$, compute period $(s..s', u)$ and check whether it is smaller than P. **H7a**: maximizes s'; **H7b**: chooses the closest period.

Outline

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

Plan of experiments

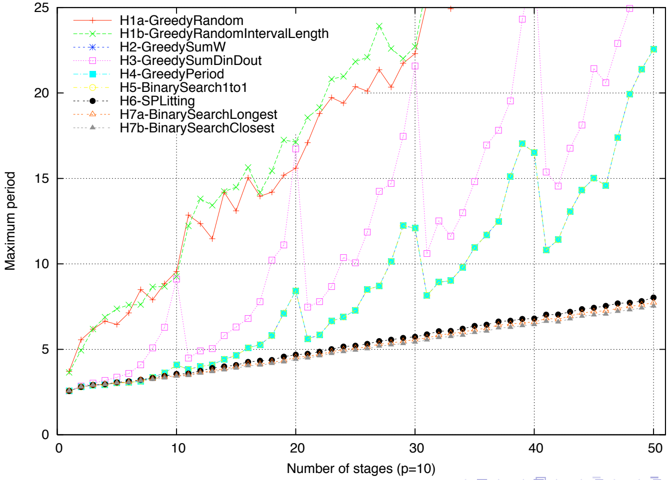
- Assess performance of polynomial heuristics
- Random applications, $n = 1$ to 50 stages
- Random platforms, $p = 10$ and $p = 100$ processors
- $b = 10$ (comm. hom.), proc. speed between 1 and 20
- Relevant parameters: ratios $\frac{\delta}{b}$ and $\frac{w}{s}$
- Average over 100 similar random appli/platform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

Plan of experiments

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

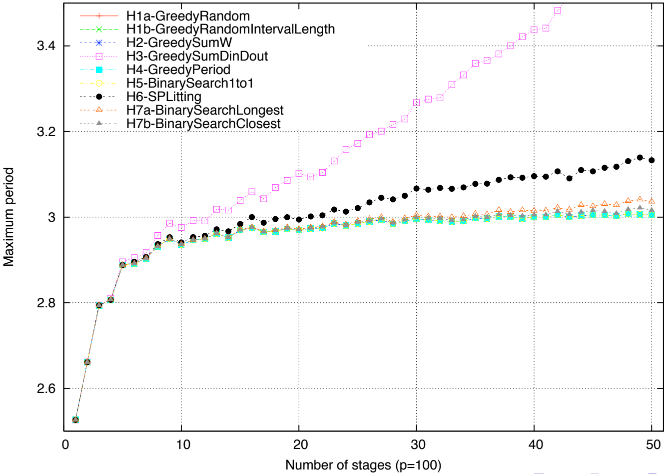
Experiment 1 - balanced comm/comp, hom comm

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



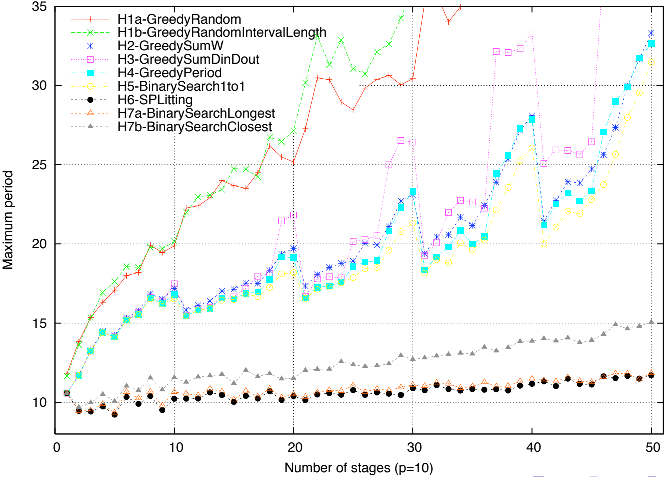
Experiment 1 - balanced comm/comp, hom comm

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



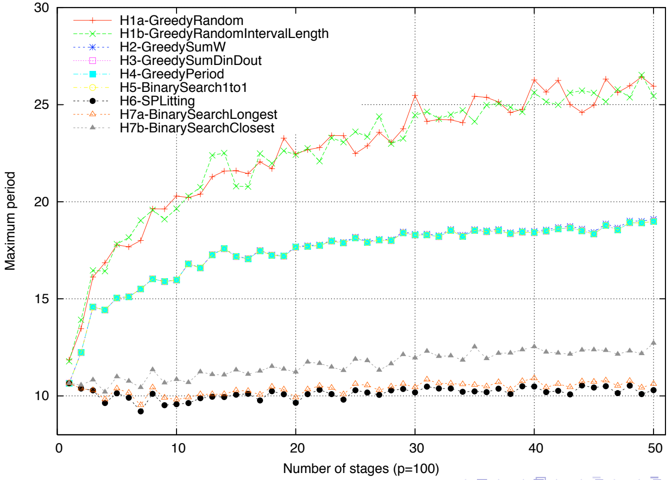
Experiment 2 - balanced comm/comp, het comm

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



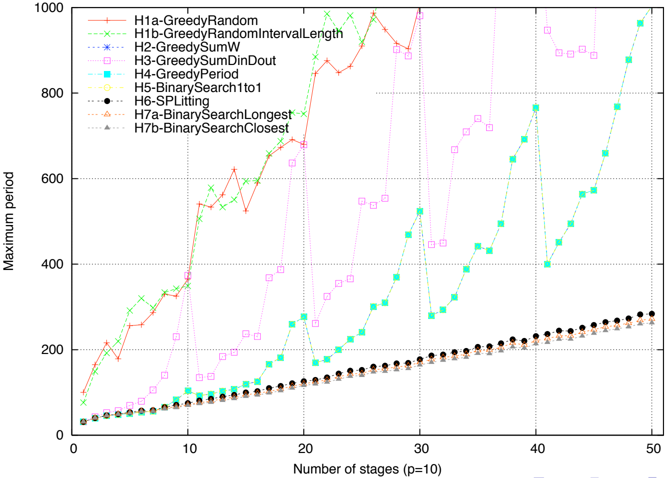
Experiment 2 - balanced comm/comp, het comm

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



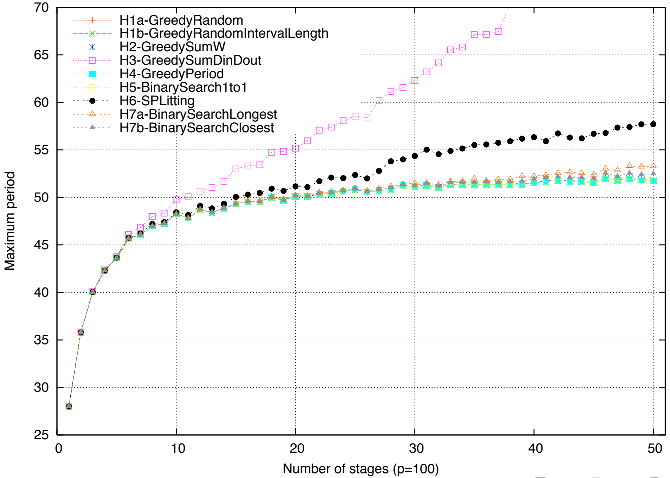
Experiment 3 - large computations

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



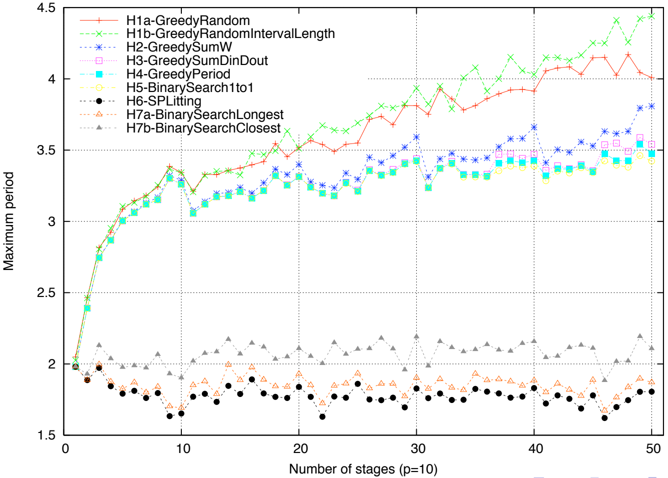
Experiment 3 - large computations

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



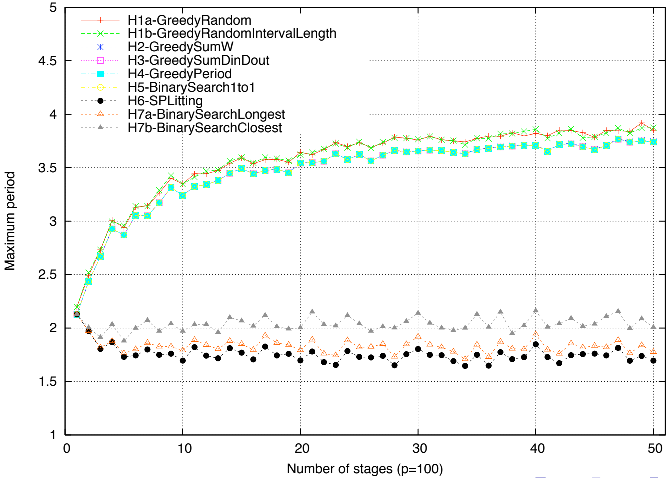
Experiment 4 - small computations

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



Experiment 4 - small computations

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



Summary of experiments

- Much more efficient than random mappings
- Three dominant heuristics for different cases
- Insignificant communications (hom. or small) and many processors: H5-BS121 (One-to-one Mapping)
- Insignificant communications (hom. or small) and few processors: H7b-BSC (binary search: clever choice where to split)
- Important communications (het. or big): H6-SPL (splitting choice relevant for any number of processors)

Summary of experiments

- Much more efficient than random mappings
- Three dominant heuristics for different cases

- Insignificant communications (hom. or small) and many processors: H5-BS121 (ONE-TO-ONE MAPPING)
- Insignificant communications (hom. or small) and few processors: H7b-BSC (binary search: clever choice where to split)
- Important communications (het. or big): H6-SPL (splitting choice relevant for any number of processors)

Outline

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

Related work

Scheduling task graphs on heterogeneous platforms– Acyclic task graphs scheduled on different speed processors [Topcuoglu et al.]. Communication contention: 1-port model [Beaumont et al.].

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

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

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

Conclusion

Theoretical side – Complexity for different mapping strategies and different platform types

Practical side

- Optimal polynomial algorithm for ONE-TO-ONE MAPPING
- Design of several heuristics for INTERVAL MAPPING on *Communication Homogeneous*
- Comparison of their performance
- Linear program to assess the absolute performance of the heuristics, which turns out to be quite good

Future work

Short term

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

Longer term

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