On the complexity of multi-criteria scheduling problems for workflow applications

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

Kunal Agrawal, MIT, USA
Harald Kosch, University of Passau, Germany

Gotha/MAO meeting in Paris
January 9, 2009

Introduction and motivation

- Scheduling applications onto parallel platforms: difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms: even more difficult!
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Selectivity: some tasks filter data

Scheduling *workflow applications* onto *heterogeneous platforms*

Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Selectivity: some tasks filter data

Scheduling *workflow applications* onto *heterogeneous platforms*

Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Selectivity: some tasks filter data

Scheduling *workflow applications* onto *heterogeneous platforms*

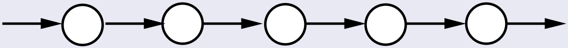
Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Selectivity: some tasks filter data

Scheduling *workflow applications* onto *heterogeneous platforms*

Multi-criteria scheduling of *workflow* applications

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

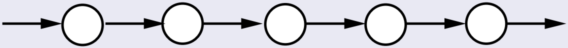
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

Multi-criteria scheduling of *workflow* applications

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

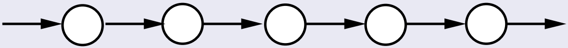
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

Multi-criteria scheduling of *workflow* applications

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

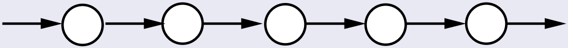
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

Multi-criteria scheduling of *workflow* applications

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

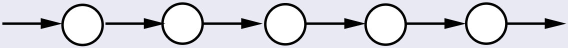
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

Multi-criteria scheduling of *workflow* applications

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

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

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

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

Major contributions

Definitions

- Workflow applications
- Computational platforms and communication models
- Multi-criteria mappings

Theory

- Problem complexity
- Linear programming formulation

Practice

- Heuristics for sub-problems
- Experiments: compare and evaluate heuristics
- Simulation of real applications (JPEG encoder)

In this talk: small examples to illustrate problem complexity

Major contributions

Definitions

- Workflow applications
- Computational platforms and communication models
- Multi-criteria mappings

Theory

- Problem complexity
- Linear programming formulation

Practice

- Heuristics for sub-problems
- Experiments: compare and evaluate heuristics
- Simulation of real applications (JPEG encoder)

In this talk: small examples to illustrate problem complexity

Major contributions

Definitions

- Workflow applications
- Computational platforms and communication models
- Multi-criteria mappings

Theory

- Problem complexity
- Linear programming formulation

Practice

- Heuristics for sub-problems
- Experiments: compare and evaluate heuristics
- Simulation of real applications (JPEG encoder)

In this talk: small examples to illustrate problem complexity

Major contributions

Definitions

- Workflow applications
- Computational platforms and communication models
- Multi-criteria mappings

Theory

- Problem complexity
- Linear programming formulation

Practice

- Heuristics for sub-problems
- Experiments: compare and evaluate heuristics
- Simulation of real applications (JPEG encoder)

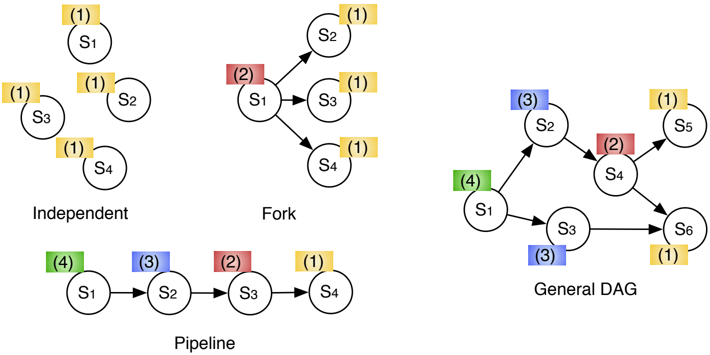
In this talk: small examples to illustrate problem complexity

Outline

1. Definitions: Application, Platform and Mappings
2. Working out examples
3. Summary of complexity results
4. Conclusion

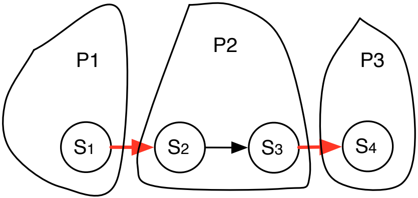
Application model

- Set of n application stages
- Workflow: each data set must be processed by all stages
- Computation cost of stage S_i: w_i
- Dependencies between stages



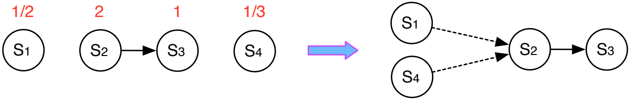
Application model: communication costs

- Two dependent stages $S_1 \to S_2$:
  data must be transferred from S_1 to S_2
- Fixed data size $\delta_{1,2}$, communication cost to pay only if S_1 and S_2 are mapped onto different processors
  (i.e., red arrows in the example)



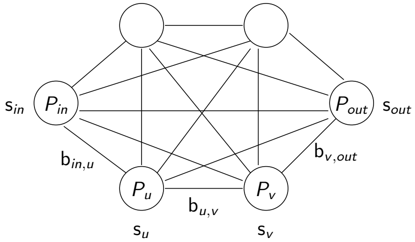
Application model: adding selectivity

- Stages with selectivity: stage S_i transforms (filters) data of size δ to size $\sigma_i \times \delta$ - σ_i: stage selectivity
- Computation cost depends on the data size (previous σ)
- May add dependencies to exploit selectivity



- S_1 and S_4 process file of initial size 1; S_1 removes even line numbers; S_2 removes two-third of the file
- Combined file of size $\frac{1}{2}.\frac{1}{3} = \frac{1}{6}$ (no cost for join)
- S_2 duplicates the file
- S_3 processes but does not alter the file

Platform model



- 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)
- P_{in}: input data – P_{out}: output data

Different platforms

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

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

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

Different platforms

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

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

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

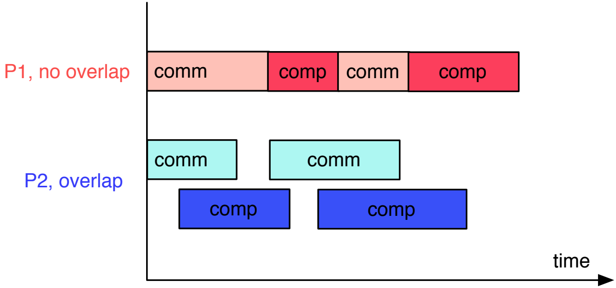
Fully Heterogeneous – Fully heterogeneous architectures, $s_u \neq s_v$ and $b_{u,v} \neq b_{u',v'}$: hierarchical platforms, grids

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

Platform model: communications

no overlap vs overlap

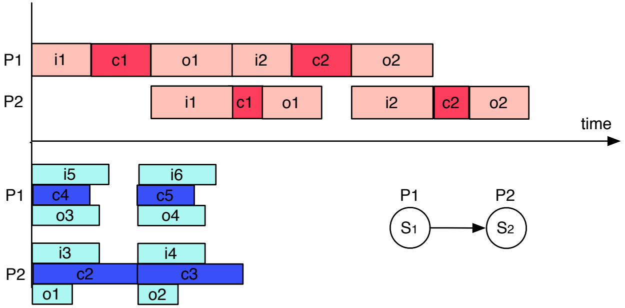
- no overlap: at each time step, either computation or communication
- overlap: a processor can simultaneously compute and communicate



Platform model: communications

one-port vs multi-port

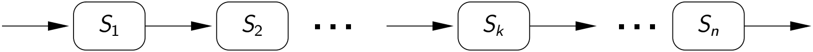
- one-port: each processor can either send or receive to/from a single other processor any time-step it is communicating
- bounded multi-port: simultaneous send and receive, but bound on the total outgoing/incoming communication (limitation of network card)



Mapping strategies: rule of the game

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

- The pipeline case: several mapping strategies



The pipeline application

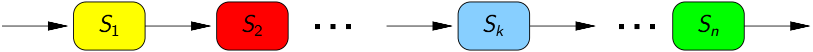
- Other applications: one-to-one and general always defined
- Define connected-subgraph mapping (instead of interval)

- Replication: independent sets of processors, instead of a single processor as above

Mapping strategies: rule of the game

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

- The pipeline case: several mapping strategies



ONE-TO-ONE MAPPING

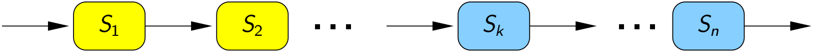
- Other applications: one-to-one and general always defined
- Define connected-subgraph mapping (instead of interval)

- Replication: independent sets of processors, instead of a single processor as above

Mapping strategies: rule of the game

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

- The pipeline case: several mapping strategies



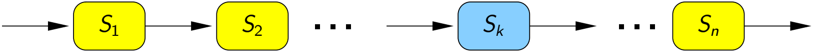
INTERVAL MAPPING

- Other applications: one-to-one and general always defined
- Define connected-subgraph mapping (instead of interval)

- Replication: independent sets of processors, instead of a single processor as above

Mapping strategies: rule of the game

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



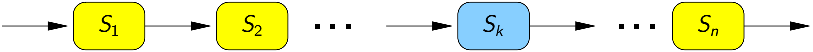
GENERAL MAPPING

- Other applications: one-to-one and general always defined
- Define connected-subgraph mapping (instead of interval)
- Replication: independent sets of processors, instead of a single processor as above

Mapping strategies: rule of the game

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

- The pipeline case: several mapping strategies



GENERAL MAPPING

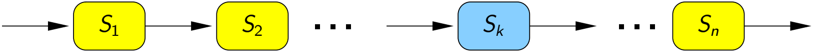
- Other applications: one-to-one and general always defined
- Define connected-subgraph mapping (instead of interval)

- Replication: independent sets of processors, instead of a single processor as above

Mapping strategies: rule of the game

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

- The pipeline case: several mapping strategies



General Mapping

- Other applications: one-to-one and general always defined
- Define connected-subgraph mapping (instead of interval)

- Replication: independent sets of processors, instead of a single processor as above

Mapping: replication and stage types

- Monolithic stages: must be mapped on one single processor since computation for a data set may depend on result of previous computation
- Replicable stages: can be replicated on several processors, but not parallel, *i.e.* a data set must be entirely processed on a single processor
- Data-parallel stages: inherently parallel stages, one data set can be computed in parallel by several processors
- Replication for reliability (also called duplication): one data set is processed several times on different processors.

Mapping: replication and stage types

- Monolithic stages: must be mapped on one single processor since computation for a data set may depend on result of previous computation
- Replicable stages: can be replicated on several processors, but not parallel, *i.e.* a data set must be entirely processed on a single processor
- Data-parallel stages: inherently parallel stages, one data set can be computed in parallel by several processors

- Replication for reliability (also called duplication): one data set is processed several times on different processors.

Mapping: objective function?

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{FP}$

Mapping: objective function?

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{FP}$

Multi-criteria

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

Mapping: objective function?

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{FP}$

Multi-criteria

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

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

Mapping: objective function?

Mono-criterion

- Minimize period $\mathcal{P}$ (inverse of throughput)
- Minimize latency $\mathcal{L}$ (time to process a data set)
- Minimize application failure probability $\mathcal{FP}$

Bi-criteria

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

- And so on...

An example of formal definitions

- Pipeline application, INTERVAL MAPPING
- Period/Latency problem with no replication
- *Communication Homogeneous*: one-port with no overlap

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

An example of formal definitions

- Pipeline application, INTERVAL MAPPING
- Period/Latency problem with no replication
- *Communication Homogeneous*: one-port with no overlap

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

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

An example of formal definitions

- Pipeline application, INTERVAL MAPPING
- Period/Latency problem with no replication
- *Communication Homogeneous*: multi-port with overlap

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

An example of formal definitions

- Pipeline application, INTERVAL MAPPING
- Period/Latency problem with no replication
- *Communication Homogeneous*: multi-port with overlap

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

$\mathcal{L}$ = the longest path of the mapping as without overlap, but does not necessarily respect previous period

$\mathcal{L} = (2K + 1).\mathcal{P}$, where K is the number of processor changes

Outline

1. Definitions: Application, Platform and Mappings
2. Working out examples
3. Summary of complexity results
4. Conclusion

Period - No communication, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

Period - No communication, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

Period - No communication, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Period - No communication, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

2 processors (P_1 and P_2) of speed 1

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

Period - No communication, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

P_1 of speed 2, and P_2 of speed 3

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

Heterogeneous platform?

Period - No communication, no replication

$$\begin{array}{ccccccc} \mathcal{S}_1 & \rightarrow & \mathcal{S}_2 & \rightarrow & \mathcal{S}_3 & \rightarrow & \mathcal{S}_4 \\ 2 & & 1 & & 3 & & 4 \end{array}$$

P_1 of speed 2, and P_2 of speed 3

Optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?

$\mathcal{P} = 6, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_4 \rightarrow P_2$ – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

Heterogeneous platform?

$\mathcal{P} = 2, \quad \mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_2, \mathcal{S}_4 \rightarrow P_1$

Heterogeneous chains-on-chains, NP-hard

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing both for computation and comm.

Optimal latency?

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing both for computation and comm.

Optimal latency?

With only one processor, $\mathcal{L} = 12$

No internal communication to pay

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5, \quad \mathcal{S}_1\mathcal{S}_3 \rightarrow P_1, \, \mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing both for computation and comm.

Optimal latency?

Same mapping as above: $\mathcal{L} = 21$ with no period constraint

$\mathcal{P} = 21$, no conflicts

$P_{in} \rightarrow P_1$	0 0 0
P_1	1 2 1 2/12 13 14
$P_1 \rightarrow P_2$	3 4 5 6 15
$P_2 \rightarrow P_1$	8 9 10 11
P_2	7 16 17 18 19
$P_2 \rightarrow P_{out}$	20

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With overlap: optimal period?

$\mathcal{P} = 5$, $\mathcal{S}_1\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_2\mathcal{S}_4 \rightarrow P_2$

Perfect load-balancing both for computation and comm.

Optimal latency? with $\mathcal{P} = 5$?

Progress step-by-step in the pipeline $\rightarrow$ no conflicts

$K = 4$ processor changes, $\mathcal{L} = (2K+1).\mathcal{P} = 9\mathcal{P} = 45$

	...	period k	period $k+1$	period $k+2$	...
$in \rightarrow P_1$	...	$ds^{(k)}$	$ds^{(k+1)}$	$ds^{(k+2)}$	...
P_1	...	$ds^{(k-1)}$, $ds^{(k-5)}$	$ds^{(k)}$, $ds^{(k-4)}$	$ds^{(k+1)}$, $ds^{(k-3)}$	...
$P_1 \rightarrow P_2$	...	$ds^{(k-2)}$, $ds^{(k-6)}$	$ds^{(k-1)}$, $ds^{(k-5)}$	$ds^{(k)}$, $ds^{(k-4)}$	...
$P_2 \rightarrow P_1$	...	$ds^{(k-4)}$	$ds^{(k-3)}$	$ds^{(k-2)}$	...
P_2	...	$ds^{(k-3)}$, $ds^{(k-7)}$	$ds^{(k-2)}$, $ds^{(k-6)}$	$ds^{(k-1)}$, $ds^{(k-5)}$	...
$P_2 \rightarrow out$	...	$ds^{(k-8)}$	$ds^{(k-7)}$	$ds^{(k-6)}$	...

Latency - No replication, different comm. models

$$\xrightarrow{1} \begin{array}{c}\mathcal{S}_1\\ 2\end{array} \xrightarrow{4} \begin{array}{c}\mathcal{S}_2\\ 1\end{array} \xrightarrow{4} \begin{array}{c}\mathcal{S}_3\\ 3\end{array} \xrightarrow{1} \begin{array}{c}\mathcal{S}_4\\ 4\end{array} \xrightarrow{1}$$

2 processors of speed 1

With no overlap: optimal period and latency?

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With no overlap: optimal period and latency?

General mappings too difficult to handle:
restrict to interval mappings

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With no overlap: optimal period and latency?

General mappings too difficult to handle:
restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \rightarrow P_1$, $\mathcal{S}_4 \rightarrow P_2$

Latency - No replication, different comm. models

$$\xrightarrow{1} \underset{2}{\mathcal{S}_1} \xrightarrow{4} \underset{1}{\mathcal{S}_2} \xrightarrow{4} \underset{3}{\mathcal{S}_3} \xrightarrow{1} \underset{4}{\mathcal{S}_4} \xrightarrow{1}$$

2 processors of speed 1

With no overlap: optimal period and latency?

General mappings too difficult to handle:
restrict to interval mappings

$\mathcal{P} = 8$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3 \to P_1$, $\mathcal{S}_4 \to P_2$

$\mathcal{L} = 12$: $\mathcal{S}_1\mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \to P_1$

Example with replication and data-parallelism

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

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Replicate interval $[S_u..\mathcal{S}_v]$ on $P_1, \ldots, P_q$

$$\ldots \mathcal{S} \;--\; \begin{array}{l} \mathcal{S}_u \ldots \mathcal{S}_v \text{ on } P_1\text{: data sets } \mathbf{1, 4, 7, \ldots} \\ \mathcal{S}_u \ldots \mathcal{S}_v \text{ on } P_2\text{: data sets } \mathbf{2, 5, 8, \ldots} \\ \mathcal{S}_u \ldots \mathcal{S}_v \text{ on } P_3\text{: data sets } \mathbf{3, 5, 9, \ldots} \end{array} \;--\; \mathcal{S} \ldots$$

$$\mathcal{P} = \frac{\sum_{k=u}^{v} w_k}{q \times \min_i(s_i)} \text{ and } \mathcal{L} = q \times \mathcal{P}$$

- ☺ Efficient with similar-speed processors
- ☺ Replicate intervals and save communications
- ☹ Bottleneck: slowest processor; no impact on latency

Example with replication and data-parallelism

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

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Data Parallelize single stage $\mathcal{S}_k$ on $P_1, \ldots, P_q$

$$\begin{array}{c} \mathcal{S}\ (w = 16) \\ \bullet\bullet\bullet\bullet \\ \bullet\bullet\bullet\bullet \\ \bullet\bullet\bullet\bullet \\ \bullet\bullet\bullet\bullet \end{array} \quad \Rightarrow \quad \begin{array}{l} P_1\ (s_1 = 2) : \bullet\bullet\bullet\bullet\bullet\bullet\bullet\bullet \\ P_2\ (s_2 = 1) : \bullet\bullet\bullet\bullet \\ P_3\ (s_3 = 1) : \bullet\bullet\bullet\bullet \end{array}$$

$$\mathcal{P} = \frac{w_k}{\sum_{i=1}^{q} s_i} \text{ and } \mathcal{L} = \mathcal{P}$$

- ☺ Perfect load-balance, no idle time of processors
- ☺ Decreases both period and latency
- ☹ Works only for a single stage: more communications to pay

Example with replication and data-parallelism

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

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

Example with replication and data-parallelism

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

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$$\mathcal{S}_1 \overset{\text{DP}}{\longrightarrow} P_1 P_2, \quad \mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \overset{\text{REP}}{\longrightarrow} P_3 P_4$$

$$\mathcal{P} = \max\left(\tfrac{14}{2+1}, \tfrac{4+2+4}{2\times 1}\right) = 5, \quad \mathcal{L} = 14.67$$

Optimal latency?

Example with replication and data-parallelism

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

Interval mapping, 4 processors, $s_1 = 2$ and $s_2 = s_3 = s_4 = 1$

Optimal period?

$$\mathcal{S}_1 \overset{\text{DP}}{\longrightarrow} P_1 P_2, \quad \mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \overset{\text{REP}}{\longrightarrow} P_3 P_4$$

$$\mathcal{P} = \max\left(\frac{14}{2+1}, \frac{4+2+4}{2\times 1}\right) = 5, \quad \mathcal{L} = 14.67$$

Optimal latency? $\mathcal{S}_1 \overset{\text{DP}}{\longrightarrow} P_2 P_3 P_4, \quad \mathcal{S}_2\mathcal{S}_3\mathcal{S}_4 \rightarrow P_1$

$$\mathcal{P} = \max\left(\frac{14}{1+1+1}, \frac{4+2+4}{2}\right) = 5, \quad \mathcal{L} = 9.67 \text{ (optimal)}$$

Outline

1. Definitions: Application, Platform and Mappings
2. Working out examples
3. Summary of complexity results
4. Conclusion

Filters: stages with selectivity

- One-to-one mappings
  - No communication, homogeneous processors: period, latency and bi-criteria problems polynomial (with precedence constraints)
  - With heterogeneous processors: all problems NP-hard, even for independent tasks. Inapproximability results both for period and latency minimization problems
  - With homogeneous communication, overlap or no-overlap: all problems become NP-hard
- General mappings: NP-hard already on fully homogeneous platforms with no communications and for independent tasks (reduction from 2-partition)

Filters: stages with selectivity

- One-to-one mappings
  - No communication, homogeneous processors: period, latency and bi-criteria problems polynomial (with precedence constraints)
  - With heterogeneous processors: all problems NP-hard, even for independent tasks. Inapproximability results both for period and latency minimization problems
  - With homogeneous communication, overlap or no-overlap: all problems become NP-hard
- General mappings: NP-hard already on fully homogeneous platforms with no communications and for independent tasks (reduction from 2-partition)

Pipeline: minimizing period or latency

	Period			Latency		
	o2o	int	gen	o2o	int	gen
noc hom	P(t)	P(DP)	NPC(2P)	P(t)		
het	P(g)	NPC(*)	NPC(-)	P(g)	P(t)	
noo fhom	P(t)	P(DP)	NPC(-)	P(t)		
chom	P(bs)	NPC(-)		P(g)	P(t)	
fhet	NPC(CT)	NPC(-)		NPC(T)	NPC(*)	P(DP)
wov fhom	P(t)	P(DP)	NPC(-)	similar		
chom	P(g)	NPC(-)		to		
fhet	NPC(TC)	NPC(-)		noo		

noc: No comm – noo: Comm, no overlap – wov: Comm, with overlap

P: Polynomial (t) trivial – (g) greedy algorithm – (DP) dynamic programming algorithm – (bs) binary search algorithm

NPC: NP-complete (-) comes from simpler case – (2P) 2-Partition – (CT) Chinese traveller – (T) TSP – (*) involved reduction

Pipeline: minimizing period and latency

		Bi-criteria	
	o2o	int	gen
noc hom	P(t)	P(DP)	NPC(-)
het	P(g)	NPC(-)	
noo fhom	P(t)	P(DP)	NPC(-)
chom	P(m)	NPC(-)	
fhet	NPC(-)		
wov fhom	P(t)	P(DP)	NPC(-)
chom	P(g)	NPC(-)	
fhet	NPC(-)		

noc: No comm – noo: Comm, no overlap – wov: Comm, with overlap

P: Polynomial (t) trivial – (g) greedy algorithm – (DP) dynamic programming algorithm – (m) matching+binary search algorithm

NPC: NP-complete (-) comes from mono-criterion

Complexity results....

- ... more cases I did not talk about
- period: rapidly NP-hard
- latency: difficult to define
- reliability: non-linear formula
- replication for period or reliability, data-parallelism, ...
- mix everything: even more exciting problems ☺
- *... please ask me for details and references ...*

Complexity results....

- ... more cases I did not talk about
- period: rapidly NP-hard
- latency: difficult to define
- reliability: non-linear formula
- replication for period or reliability, data-parallelism, ...
- mix everything: even more exciting problems ☺
- *... please ask me for details and references ...*

Complexity results....

- ... more cases I did not talk about
- period: rapidly NP-hard
- latency: difficult to define
- reliability: non-linear formula
- replication for period or reliability, data-parallelism, ...
- mix everything: even more exciting problems ☺
- ... *please ask me for details and references* ...

Complexity results....

- ... more cases I did not talk about
- period: rapidly NP-hard
- latency: difficult to define
- reliability: non-linear formula
- replication for period or reliability, data-parallelism, ...
- mix everything: even more exciting problems ☺
- *... please ask me for details and references ...*

Outline

1. Definitions: Application, Platform and Mappings
2. Working out examples
3. Summary of complexity results
4. Conclusion

Related work

Qishi Wu et al– Directed platform graphs (WAN); unbounded multi-port with overlap; mono-criterion problems

Subhlok and Vondran– Pipeline on hom platforms: extended

Chains-to-chains– Heterogeneous, replicate/data-parallelize

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

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

Conclusion

Definitions:

- Applications, platforms, and multi-criteria mappings

Theoretical side:

- Working out examples to show insight of problem complexity
- Full complexity study
- Linear program formulations for NP-hard instances

Practical side (not showed in this talk):

- Several polynomial heuristics and simulations
- JPEG application, good results of the heuristics (close to LP solution)

Future work

- Extend to other application graphs
- In particular, define latency for general DAGs (order communications)
- Multiple applications setting: even more criteria to optimize (fairness between applications)
- New heuristics for NP-hard cases, further experiments on practical applications.