Multi-criteria scheduling of 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

CS Colloquium, University of Memphis
September 19, 2008
Introduction and motivation

- Mapping 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)*

Mapping workflow applications onto heterogeneous platforms
Introduction and motivation

- Mapping 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)*
Introduction and motivation

- Mapping 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)

Mapping workflow applications onto heterogeneous platforms
Introduction and motivation

- Mapping 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)
Multi-criteria scheduling of workflow applications

Workflow:

Several consecutive data sets enter the application graph.

Criteria to optimize?

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

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

Reliability: inverse of $FP$, probability of failure of the application (i.e. some data sets will not be processed)
Multi-criteria scheduling of workflow applications

Workflow:

Several consecutive data sets enter the application graph.

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 $FP$, probability of failure of the application (i.e. some data sets will not be processed)
Multi-criteria scheduling of workflow applications

Workflow:

Several consecutive data sets enter the application graph.

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:

Several consecutive data sets enter the application graph.

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

![Diagram showing Workflow types: Independent, Fork, and Pipeline]
Application model: communication costs

- **Two dependent stages** $S_1 \rightarrow S_2$: 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 on different processors (i.e. red arrows in the example)
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 \( b_{u,v} \)
- **\( \text{fp}_u \)**: 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

![Diagram showing no overlap and overlap scenarios](image-url)
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

\[ S_1 \rightarrow S_2 \rightarrow \ldots \rightarrow S_k \rightarrow \ldots \rightarrow S_n \]

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

\[ S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_k \rightarrow \cdots \rightarrow S_n \]

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

\[ S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_k \rightarrow \cdots \rightarrow S_n \]

**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: 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: 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 $P$ (inverse of throughput)
- Minimize latency $L$ (time to process a data set)
- Minimize application failure probability $FP$
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 $P$ (inverse of throughput)
- Minimize latency $L$ (time to process a data set)
- Minimize application failure probability $FP$

Multi-criteria

- How to define it?
  - Minimize $\alpha P + \beta L + \gamma FP$?
- Values which are not comparable
- Minimize $P$ for a fixed latency and failure
- Minimize $L$ for a fixed period and failure
- Minimize $FP$ for a fixed period and latency
Mapping: objective function?

Mono-criterion

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

Bi-criteria

- Period and Latency:
  - Minimize $P$ for a fixed latency
  - Minimize $L$ 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

\[ P = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j-1}}{b} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{\text{alloc}(j)}} + \frac{\delta_{e_j}}{b} \right\} \]
An example of formal definitions

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

\[ P = \max_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j-1}}{b} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{alloc}(j)} + \frac{\delta_{e_j}}{b} \right\} \]

\[ L = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j-1}}{b} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{alloc}(j)} \right\} + \frac{\delta_n}{b} \]
An example of formal definitions

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

\[
P = \max_{1 \leq j \leq m} \left\{ \max \left\{ \frac{\sum_{i=d_j}^{e_j} W_i}{s_{\text{alloc}}(j)}, \frac{\delta_{d_j-1}}{b}, \frac{\delta_{d_j-1}}{B^i}, \frac{\delta_{e_j}}{b}, \frac{\delta_{e_j}}{B^o} \right\} \right\}
\]
An example of formal definitions

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

\[
P = \max_{1 \leq j \leq m} \left\{ \max \left\{ \frac{\sum_{i=d_j}^{e_j} w_i}{s_{alloc}(j)}, \frac{\delta_{d_j-1}}{b}, \frac{\delta_{e_j}}{B^i}, \frac{\delta_{e_j}}{B^o} \right\} \right\}
\]

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

\[
L = (2K + 1) \cdot P, \text{ where } K \text{ is the number of changes of processors}
\]
Outline

1. Definitions: Application, Platform and Mappings
2. Working out examples
3. Summary of complexity results
4. Conclusion
Period - No communication, no replication

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

2 processors of speed 1

Optimal period?
Period - No communication, no replication

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

2 processors of speed 1

Optimal period?
\[ P = 5, \quad S_1S_3 \rightarrow P_1, \quad S_2S_4 \rightarrow P_2 \]
Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?
Period - No communication, no replication

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

2 processors of speed 1

Optimal period?
\[ P = 5, \quad S_1S_3 \rightarrow P_1, \quad S_2S_4 \rightarrow P_2 \]
Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?
\[ P = 6, \quad S_1S_2S_3 \rightarrow P_1, \quad S_4 \rightarrow P_2 \quad - \quad \text{Polynomial algorithm?} \]
Period - No communication, no replication

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

2 processors of speed 1

Optimal period?

\[ P = 5, \quad S_1S_3 \rightarrow P_1, \quad S_2S_4 \rightarrow P_2 \]

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

Interval mapping?

\[ P = 6, \quad S_1S_2S_3 \rightarrow P_1, \quad S_4 \rightarrow P_2 \]

- Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works
## Period - No communication, no replication

\[
S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4
\]

\[
2 \rightarrow 1 \rightarrow 3 \rightarrow 4
\]

Speed of \( P_1 : 2 \mid P_2 : 3 \)

**Optimal period?**

\( \mathcal{P} = 5, \ S_1S_3 \rightarrow P_1, \ S_2S_4 \rightarrow P_2 \)

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

**Interval mapping?**

\( \mathcal{P} = 6, \ S_1S_2S_3 \rightarrow P_1, \ S_4 \rightarrow P_2 \) – Polynomial algorithm?

Classical chains-on-chains problem, dynamic programming works

**Heterogeneous platform?**
Period - No communication, no replication

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

\[ 2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \]

Speed of \( P_1 : 2 \mid P_2 : 3 \)

Optimal period?
\( \mathcal{P} = 5, \quad S_1S_3 \rightarrow P_1, \quad S_2S_4 \rightarrow P_2 \)
Perfect load-balancing in this case, but NP-hard (2-PARTITION)

Interval mapping?
\( \mathcal{P} = 6, \quad S_1S_2S_3 \rightarrow P_1, \quad S_4 \rightarrow P_2 \) – Polynomial algorithm?
Classical chains-on-chains problem, dynamic programming works

Heterogeneous platform?
\( \mathcal{P} = 2, \quad S_1S_2S_3 \rightarrow P_2, \quad S_4 \rightarrow P_1 \)
Heterogeneous chains-on-chains, NP-hard
Latency - No replication, different comm. models

\[
\begin{array}{cccccc}
1 & \to S_1 & 4 & \to S_2 & 4 & \to S_3 & 1 & \to S_4 & 1 \\
2 & & 1 & & 3 & & 4 & & \\
\end{array}
\]

2 processors of speed 1

With overlap: optimal period?
Latency - No replication, different comm. models

\[
\begin{align*}
&1 \rightarrow S_1 \quad 4 \rightarrow S_2 \quad 4 \rightarrow S_3 \quad 1 \rightarrow S_4 \quad 1 \\
&2 \quad 1 \quad 3 \quad 4
\end{align*}
\]

2 processors of speed 1

With overlap: optimal period?

\( P = 5, \ S_1 S_3 \rightarrow P_1, \ S_2 S_4 \rightarrow P_2 \)

Perfect load-balancing both for computation and comm.

Optimal latency?
Latency - No replication, different comm. models

\[ \begin{array}{cccc}
1 & S_1 & 4 & S_2 \\
2 & 1 & 4 & S_3 \\
& 3 & 1 & S_4 \\
& 4 & & 1
\end{array} \]

2 processors of speed 1

With overlap: optimal period?

\[ P = 5, \quad S_1S_3 \rightarrow P_1, \quad S_2S_4 \rightarrow P_2 \]

Perfect load-balancing both for computation and comm.

Optimal latency?

With only one processor, \( L = 12 \)

No internal communication to pay
Latency - No replication, different comm. models

\[
\begin{array}{c}
1 \rightarrow S_1 \quad 4 \rightarrow S_2 \quad 4 \rightarrow S_3 \quad 1 \rightarrow S_4 \quad 1 \rightarrow \\
2 \quad 1 \quad 3 \quad 4
\end{array}
\]

2 processors of speed 1

With overlap: optimal period?
\( P = 5, \quad S_1S_3 \rightarrow P_1, \ S_2S_4 \rightarrow P_2 \)

Perfect load-balancing both for computation and comm.

Optimal latency?

Same mapping as above: \( L = 21 \) with no period constraint
\( P = 21, \ no \ conflicts \)

\[
\begin{array}{c|cccc}
P_{in} \rightarrow P_1 & 0 & 0 & 0 \\
P_1 \rightarrow P_2 & 1 & 2 & 1 & 2/12 & 13 & 14 \\
P_1 \rightarrow P_1 & 3 & 4 & 5 & 6 \\
P_2 \rightarrow P_2 & 8 & 9 & 10 & 11 \\
P_2 \rightarrow P_{out} & 7 & 16 & 17 & 18 & 19 & 20
\end{array}
\]

Anne.Benoit@ens-lyon.fr  
Memphis - Sept 19  
Multi-criteria scheduling of workflow applications  
18/26
Latency - No replication, different comm. models

\[ \begin{array}{cccccc}
1 & \rightarrow & S_1 & 4 & \rightarrow & S_2 \\
2 & \rightarrow & 1 & 4 & \rightarrow & S_3 \\
1 & \rightarrow & S_4 & 1 & \rightarrow & \\
\end{array} \]

2 processors of speed 1

With overlap: optimal period?

\( P = 5, \ S_1 S_3 \rightarrow P_1, \ S_2 S_4 \rightarrow P_2 \)

Perfect load-balancing both for computation and comm.

Optimal latency? with \( P = 5 \)?

Progress step-by-step in the pipeline \( \rightarrow \) no conflicts

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

<table>
<thead>
<tr>
<th>in ( \rightarrow ) ( P_1 )</th>
<th>( \ldots )</th>
<th>period ( k )</th>
<th>period ( k + 1 )</th>
<th>period ( k + 2 )</th>
<th>( \ldots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( ds^{(k)} )</td>
<td>( ds^{(k+1)} )</td>
<td>( ds^{(k+2)} )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( P_1 )</td>
<td>( \ldots )</td>
<td>( ds^{(k-1)}, ds^{(k-5)} )</td>
<td>( ds^{(k)}, ds^{(k-4)} )</td>
<td>( ds^{(k+1)}, ds^{(k-3)} )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( ds^{(k-2)}, ds^{(k-6)} )</td>
<td>( ds^{(k-1)}, ds^{(k-5)} )</td>
<td>( ds^{(k)}, ds^{(k-4)} )</td>
<td>( \ldots )</td>
<td></td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( ds^{(k-4)} )</td>
<td>( ds^{(k-3)} )</td>
<td>( ds^{(k-2)} )</td>
<td>( \ldots )</td>
<td></td>
</tr>
<tr>
<td>( P_2 )</td>
<td>( \ldots )</td>
<td>( ds^{(k-3)}, ds^{(k-7)} )</td>
<td>( ds^{(k-2)}, ds^{(k-6)} )</td>
<td>( ds^{(k-1)}, ds^{(k-5)} )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( ds^{(k-8)} )</td>
<td>( ds^{(k-7)} )</td>
<td>( ds^{(k-6)} )</td>
<td>( \ldots )</td>
<td></td>
</tr>
</tbody>
</table>

Anne.Benoit@ens-lyon.fr
Memphis - Sept 19
Multi-criteria scheduling of workflow applications 18/26
Latency - No replication, different comm. models

\[ 2 \rightarrow S_1 \rightarrow 1 \rightarrow S_2 \rightarrow 4 \rightarrow S_3 \rightarrow 1 \rightarrow S_4 \rightarrow 1 \]

2 processors of speed 1

With no overlap: optimal period and latency?
Latency - No replication, different comm. models

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

\[ \begin{align*}
1 & \rightarrow S_1 \\
4 & \rightarrow S_2 \\
4 & \rightarrow S_3 \\
1 & \rightarrow S_4 \\
\end{align*} \]

2 processors of speed 1

With **no overlap**: optimal period and latency?

General mappings too difficult to handle:
restrict to **interval mappings**

\( P = 8: \ S_1, S_2, S_3 \rightarrow P_1, S_4 \rightarrow P_2 \)
Latency - No replication, different comm. models

\[
\begin{align*}
1 & \rightarrow S_1 \quad & 4 & \rightarrow S_2 \\
2 & \quad & 1 & \rightarrow S_3 \\
4 & \rightarrow S_4 \\
1 & \rightarrow \end{align*}
\]

2 processors of speed 1

With no overlap: optimal period and latency?

General mappings too difficult to handle: restrict to interval mappings

\[P = 8: \quad S_1, S_2, S_3 \rightarrow P_1, \quad S_4 \rightarrow P_2\]

\[L = 12: \quad S_1, S_2, S_3, S_4 \rightarrow P_1\]
Example with replication and data-parallelism

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

14 4 2 4

Interval mapping, 4 processors, \( s_1 = 2 \) and \( s_2 = s_3 = s_4 = 1 \)

**Replicate** interval \([S_u..S_v]\) on \( P_1, \ldots, P_q \)

\[
P = \sum_{k=u}^{v} w_k \quad \text{and} \quad L = q \times P
\]
Example with replication and data-parallelism

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

14 4 2 4

Interval mapping, 4 processors, \( s_1 = 2 \) and \( s_2 = s_3 = s_4 = 1 \)

**Data Parallelize** single stage \( S_k \) on \( P_1, \ldots, P_q \)

\[ S \ (w = 16) \]

\[ \begin{array}{ccccccc}
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\end{array} \]

\[ \Rightarrow \]

\[ \begin{array}{ccccccc}
P_1 \ (s_1 = 2) : & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
P_2 \ (s_2 = 1) : & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
P_3 \ (s_3 = 1) : & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\end{array} \]

\[ \mathcal{P} = \frac{w_k}{\sum_{i=1}^{q} s_i} \] and \( \mathcal{L} = \mathcal{P} \)
Example with replication and data-parallelism

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

14 4 2 4

Interval mapping, 4 processors, \( s_1 = 2 \) and \( s_2 = s_3 = s_4 = 1 \)

Optimal period?
Example with replication and data-parallelism

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]

14 4 2 4

Interval mapping, 4 processors, \( s_1 = 2 \) and \( s_2 = s_3 = s_4 = 1 \)

Optimal period?

\[ S_1 \overset{\text{DP}}{\rightarrow} P_1P_2, \; S_2S_3S_4 \overset{\text{REP}}{\rightarrow} P_3P_4 \]

\[ P = \max\left(\frac{14}{2+1}, \frac{4+2+4}{2 \times 1}\right) = 5, \; L = 14.67 \]

Optimal latency?
Example with replication and data-parallelism

\[ S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \]
\[ 14 \quad 4 \quad 2 \quad 4 \]

Interval mapping, 4 processors, \( s_1 = 2 \) and \( s_2 = s_3 = s_4 = 1 \)

Optimal period?

\[ S_1^{DP} \rightarrow P_1P_2, \quad S_2S_3S_4^{REP} \rightarrow P_3P_4 \]
\[ P = \max(\frac{14}{2+1}, \frac{4+2+4}{2\times1}) = 5, \quad \mathcal{L} = 14.67 \]

Optimal latency?

\[ S_1^{DP} \rightarrow P_2P_3P_4, \quad S_2S_3S_4 \rightarrow P_1 \]
\[ P = \max(\frac{14}{1+1+1}, \frac{4+2+4}{2}) = 5, \quad \mathcal{L} = 9.67 \quad \text{(optimal)} \]
Outline

1. Definitions: Application, Platform and Mappings
2. Working out examples
3. Summary of complexity results
4. Conclusion
## Pipeline: minimizing period or latency

<table>
<thead>
<tr>
<th></th>
<th>o2o</th>
<th>Period</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>int</td>
<td>gen</td>
<td>gen</td>
</tr>
<tr>
<td>noc</td>
<td>P(t)</td>
<td>P(DP)</td>
<td>NPC(2P)</td>
</tr>
<tr>
<td>hom</td>
<td>P(g)</td>
<td>NPC(*)</td>
<td>NPC(-)</td>
</tr>
<tr>
<td>het</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>noo</td>
<td>P(t)</td>
<td>P(DP)</td>
<td>P(t)</td>
</tr>
<tr>
<td>fhom</td>
<td>P(bs)</td>
<td>NPC(-)</td>
<td>NPC(-)</td>
</tr>
<tr>
<td>chom</td>
<td>NPC(CT)</td>
<td>NPC(-)</td>
<td>NPC(-)</td>
</tr>
<tr>
<td>fhet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wov</td>
<td>P(t)</td>
<td>P(DP)</td>
<td>similar</td>
</tr>
<tr>
<td>fhom</td>
<td>P(g)</td>
<td>NPC(-)</td>
<td>to</td>
</tr>
<tr>
<td>chom</td>
<td>NPC(TC)</td>
<td>NPC(-)</td>
<td>noo</td>
</tr>
<tr>
<td>fhet</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>noc hom fhet</th>
<th>o2o</th>
<th>Bi-criteria</th>
<th>int</th>
<th>gen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P(t)</td>
<td>P(DP)</td>
<td>NPC(-)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P(g)</td>
<td>NPC(-)</td>
<td>NPC(-)</td>
<td></td>
</tr>
<tr>
<td>noo fhom chom fhet</td>
<td>P(t)</td>
<td>P(DP)</td>
<td>NPC(-)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P(m)</td>
<td>NPC(-)</td>
<td>NPC(-)</td>
<td></td>
</tr>
<tr>
<td>wov fhom chom fhet</td>
<td>P(t)</td>
<td>P(DP)</td>
<td>NPC(-)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P(g)</td>
<td>NPC(-)</td>
<td>NPC(-)</td>
<td></td>
</tr>
</tbody>
</table>

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
... 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 ...
... 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 ...
... 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* …
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.]
**Definitions:** Applications, platforms, and multi-criteria mappings

**Theoretical side:** Working out examples to show insight of problem complexity, and full complexity study

**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)
- New heuristics for NP-hard cases, further experiments