Scheduling Workflow Applications

Yves Robert

École Normale Supérieure de Lyon
Yves.Robert@ens-lyon.fr
http://graal.ens-lyon.fr/~yrobert

joint work with
Kunal Agrawal, Anne Benoit, Fanny Dufossé,
Veronika Rehn-Sonigo, Arnold Rosenberg, Frédéric Vivien

USC, Apr. 1, 2009

Introduction

- 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)
  - Reliability issues
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...

Scheduling *workflow applications* onto *heterogeneous platforms*

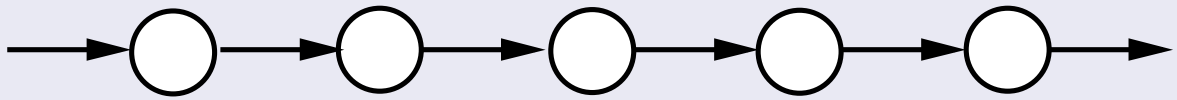
Introduction

- 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)
  - Reliability issues
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...

Scheduling *workflow applications* onto *heterogeneous platforms*

Multi-criteria scheduling

Workflow applications?



Several consecutive data sets enter the application graph.

Criteria to optimize?

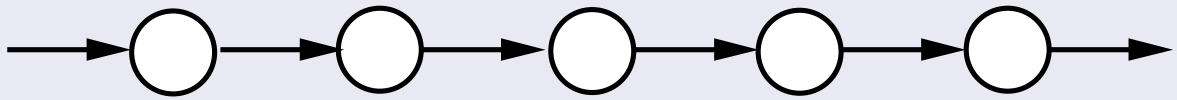
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

Multi-criteria scheduling

Workflow applications?



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

Our approach

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

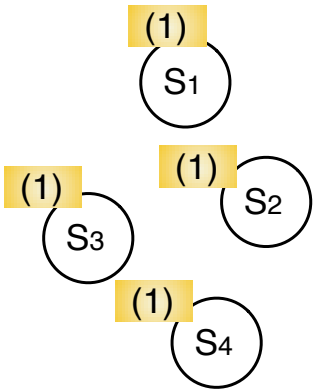
Examples to illustrate problem complexity

Outline

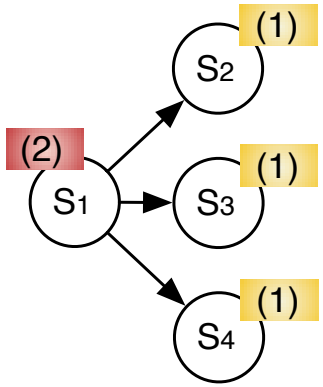
1. Framework
2. Working out examples
3. Conclusion
4. A problem for the road

Application model

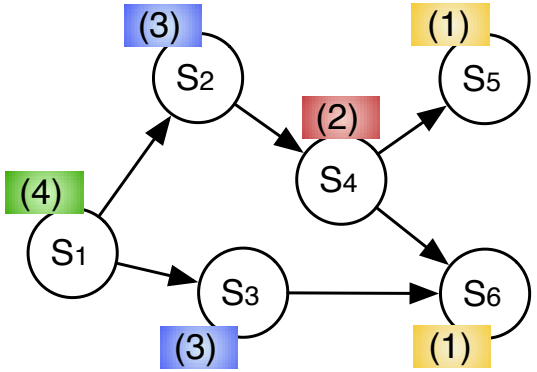
- Set of n application stages
- Workflow: each data set must be processed by all stages
- Computation cost of stage S_i: w_i
- Dependencies between stages



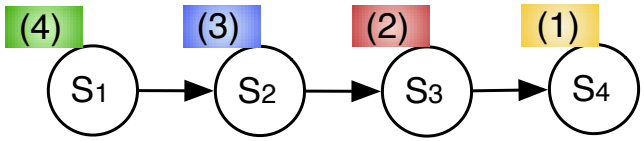
Independent



Fork



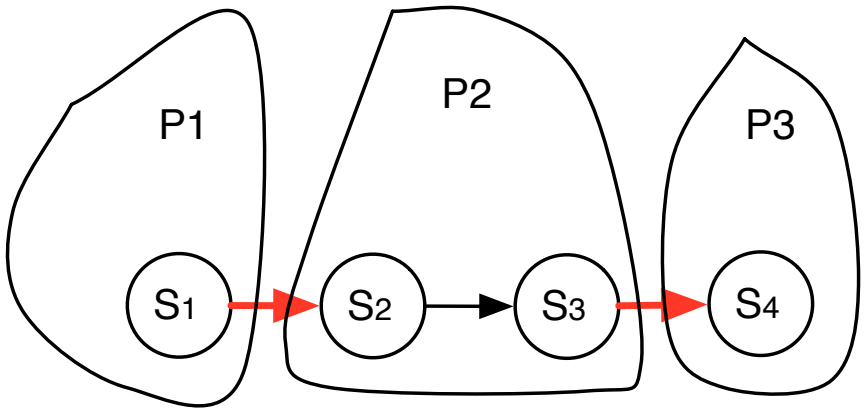
General DAG



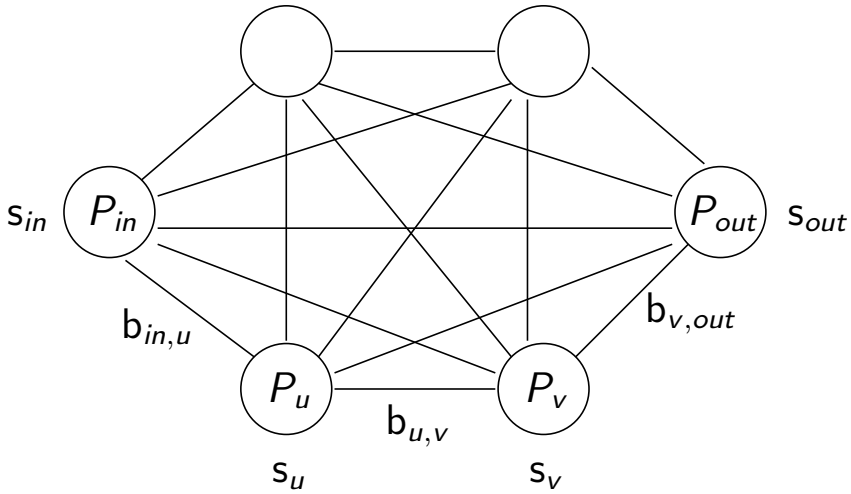
Pipeline

Application model: communication costs

- Two dependent stages $S_1 \longrightarrow S_2$:
  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)



Platform model



- p processors P_u, $1 \leq u \leq p$, fully interconnected
- s_u: speed of processor P_u
- bidirectional link $link_{u,v} : P_u \longrightarrow P_v$, bandwidth $b_{u,v}$
- f_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 ($f_u = f_v$)

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

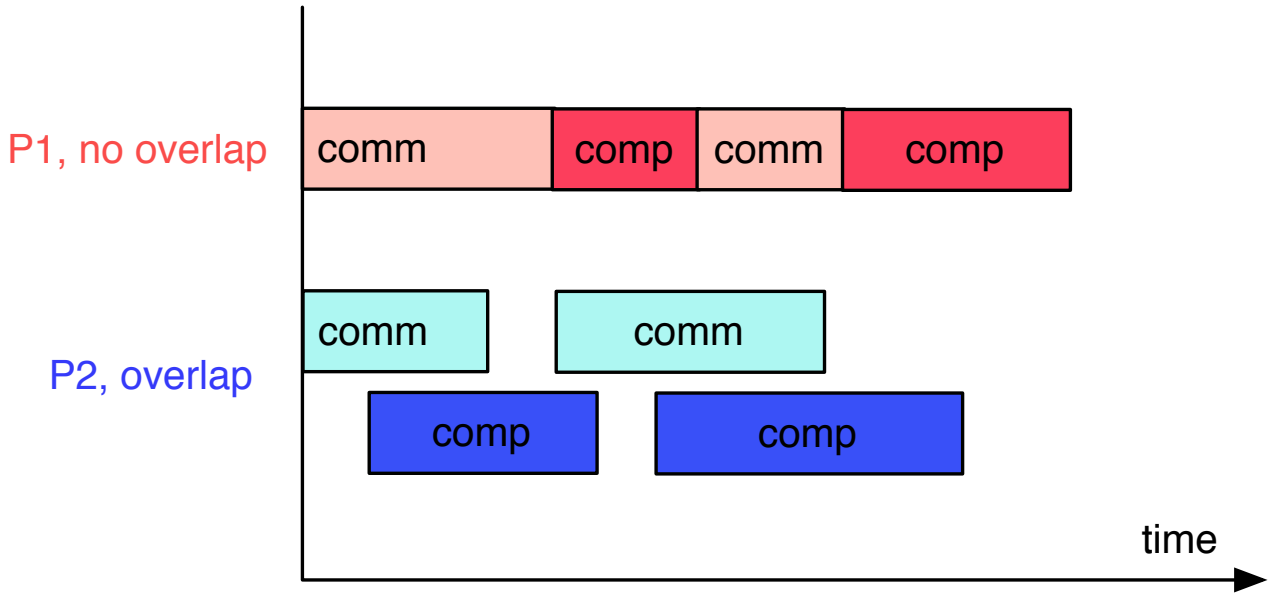
Fully Heterogeneous – Fully heterogeneous architectures, $s_u \neq s_v$ and $b_{u,v} \neq b_{u',v'}$: hierarchical platforms, grids

Failure Heterogeneous – Different failure probabilities ($f_u \neq f_v$)

Platform model: communications

no overlap vs overlap

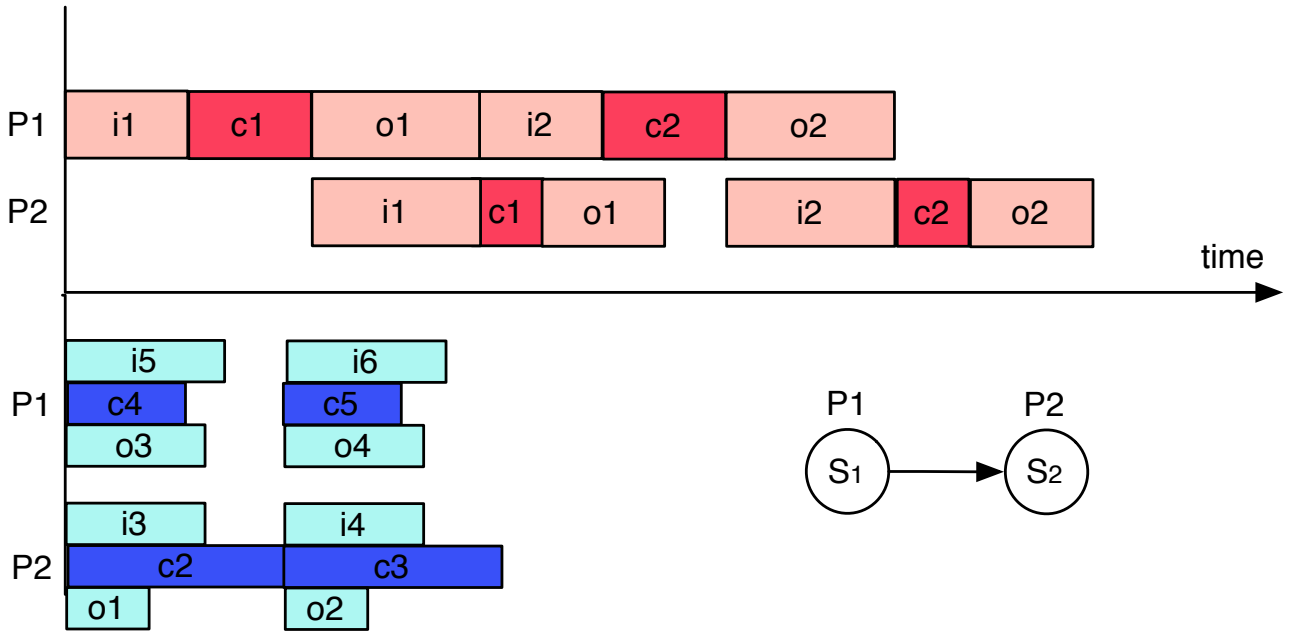
- no overlap: at each time step, either computation or communication
- overlap: a processor can simultaneously compute and communicate



Platform model: communications

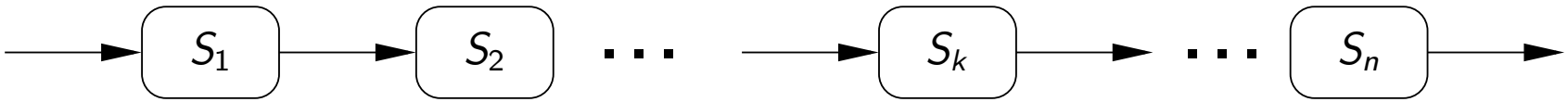
one-port vs multi-port

- 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
- Several mapping strategies

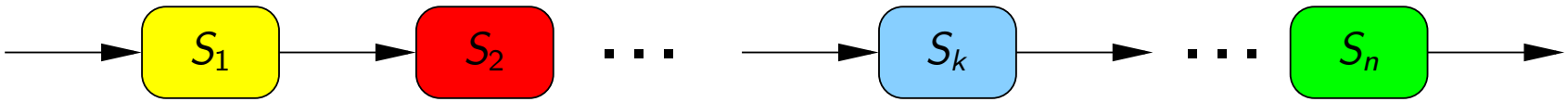


The pipeline application

- 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
- Several mapping strategies

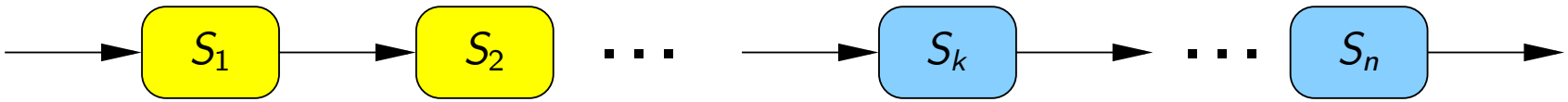


One-to-one Mapping

- 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
- Several mapping strategies

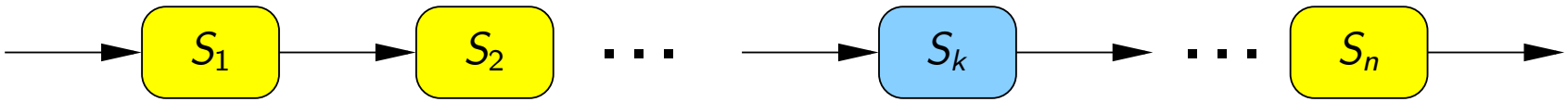


Interval Mapping

- 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
- Several mapping strategies

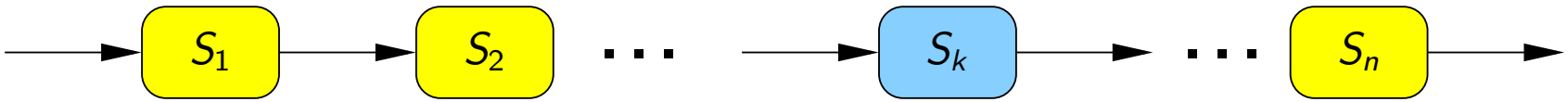


General Mapping

- 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
- Several mapping strategies



General Mapping

- 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 $\mathcal{F}$

Multi-criteria

- Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{F}$?
- Values are not comparable

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

Multi-criteria

- Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{F}$?
- Values are not comparable

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

Multi-criteria

- Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{F}$?
- Values are not comparable

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

Formal definitions (1/4)

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

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

$$\mathcal{L} = \sum_{1 \leq j \leq m} \left\{ \frac{\delta_{d_j - 1}}{b} + \frac{\sum_{i=d_j}^{e_j} w_i}{s_{alloc(j)}} \right\} + \frac{\delta_n}{b}$$

Formal definitions (2/4)

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

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

$\mathcal{L}$ = longest path of 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

Formal definitions (3/4)

- Pipeline application, INTERVAL MAPPING, m intervals
- Period/Latency/Reliability, monolithic stages

$$\mathcal{F} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \text{alloc}(j)} f_u)$$

Worst-case period and latency: one-port without overlap

$$\mathcal{P}^{(no)} = \max_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \left\{ \frac{\delta_{j-1}}{\min_{v \in \text{alloc}(j-1)} b_{v,u}} + \frac{\sum_{i \in I_j} w_i}{s_u} + \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{b_{u,v}} \right\}$$

$$\mathcal{L} = \sum_{u \in \text{alloc}(1)} \frac{\delta_0}{b_{in,u}} + \sum_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \left\{ \frac{\sum_{i \in I_j} w_i}{s_u} + \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{b_{u,v}} \right\}$$

Formal definitions (4/4)

- Pipeline application, INTERVAL MAPPING, m intervals
- Period/Latency/Reliability, monolithic stages

$$\mathcal{F} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \text{alloc}(j)} f_u)$$

Worst-case period and latency: multi-port with overlap

$$\mathcal{P}^{(ov)} = \max_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \max \left\{ \frac{\delta_{j-1}}{\min_{v \in \text{alloc}(j-1)} b_{v,u}}, \frac{\sum_{i \in I_j} w_i}{s_u}, \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{b_{u,v}} \right\}$$

$\mathcal{L}$ = longest path of 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 Framework

2 Working out examples

3 Conclusion

4 A problem for the road

Period - No communication, no replication

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

2 processors (P_1 and P_2) of speed 1

Optimal period?

Period - No communication, no replication

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

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

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

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

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

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

Digression: 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?

Digression: 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

$\mathcal{P} = 20$

NP-hard for different-speed processors, even without communications

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?

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

Perfect load-balancing both for computation and comm.

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

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

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?

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

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

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

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 $P_1, \ldots, P_q$

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

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

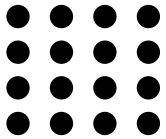
- ☺ Efficient with similar-speed processors
- ☺ Replicate intervals and save communications
- ☹ Bottleneck: slowest processor; no impact on latency

Example with replication and data-parallelism

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

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

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

$\mathcal{S}$ $(w = 16)$ 

$\Rightarrow$ P_1 $(s_1 = 2)$:

P_2 $(s_2 = 1)$:

P_3 $(s_3 = 1)$:

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

- ☺ 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

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

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

Optimal period?

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?

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

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

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?

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

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

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

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

Complexity results

$\mathcal{F}$	**Failure-Hom.**	**Failure-Het.**
One-to-one	polynomial	NP-hard
Interval	polynomial	
General	polynomial	

$\mathcal{L}$	**Fully Hom.**	**Comm. Hom.**	**Hetero.**
no DP, One-to-one	polynomial		NP-hard
no DP, Interval	polynomial		NP-hard
no DP, General	polynomial		
with DP, no coms	polynomial	NP-hard	

$\mathcal{P}$	**Fully Hom.**	**Comm. Hom.**	**Hetero.**
One-to-one	polynomial	polynomial, NP-hard (rep)	NP-hard
Interval	polynomial	NP-hard	NP-hard
General	NP-hard		

Outline

1 Framework

2 Working out examples

3 Conclusion

4 A problem for the road

Related work

Subhlok and Vondran– Pipeline on homogeneous platforms

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

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

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

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 [Bhat et al.]

Conclusion

Theoretical side

- Applications, platforms, and multi-criteria mappings
- Full complexity study
- Linear program formulations for NP-hard instances
- Mix replication for period and reliability, data-parallelism, . . . ☺
- Fairness issues for multiple applications

Practical side

- Several polynomial heuristics and simulations
- JPEG and MPEG application

Outline

1 Framework

2 Working out examples

3 Conclusion

4 A problem for the road

Divisible workload with *failures*

- Large divisible computational workload
- Assemblage of p identical computers
- Unrecoverable interruptions
- A-priori knowledge of risk (failure probability)

- Landmark paper by Bhatt, Chung, Leighton & Rosenberg on cycle stealing
- Hardware failures

Goal: maximize expected amount of work done

Chunking

- Sending each remote computer **large** amounts of work:
  - ☺ decrease message packaging overhead
  - ☹ maximize vulnerability to interruption-induced losses
- Sending each remote computer **small** amounts of work:
  - ☺ minimize vulnerability to interruption-induced losses
  - ☹ maximize message packaging overhead

Chunking

- Sending each remote computer **large** amounts of work:
  - ☺ decrease message packaging overhead
  - ☹ maximize vulnerability to interruption-induced losses
- Sending each remote computer **small** amounts of work:
  - ☺ minimize vulnerability to interruption-induced losses
  - ☹ maximize message packaging overhead

Replication

- Replicating tasks (same work sent to $q \geq 2$ remote computers):
  - ☺ lessen vulnerability to interruption-induced losses
  - ☹ minimize opportunities for "parallelism" and productivity

- Communication/control to/of remote computers **costly**
  - $\Rightarrow$ orchestrate task replication statically
  - ☹ duplicate work unnecessarily when few interruptions
  - ☺ prevent server from becoming bottleneck

Replication

- Replicating tasks (same work sent to $q \geq 2$ remote computers):
  - ☺ lessen vulnerability to interruption-induced losses
  - ☹ minimize opportunities for "parallelism" and productivity

- Communication/control to/of remote computers **costly**
  $\Rightarrow$ orchestrate task replication statically
  - ☹ duplicate work unnecessarily when few interruptions
  - ☺ prevent server from becoming bottleneck

Risk increases with time

	A	B	C	D
P_1	1	2	3	4

Risk increases with time

	A	B	C	D
P_1	1	2	3	4
P_2				

Risk increases with time

	A	B	C	D
P_1	1	2	3	4
P_2	4	3	2	1

Risk increases with time

	A	B	C	D
P_1	1	2	3	4
P_2	4	3	2	1
P_3				

Risk increases with time

	A	B	C	D
P_1	1	2	3	4
P_2	4	3	2	1
P_3	4	3	2	1

Risk increases with time

	A	B	C	D
P_1	1	2	3	4
P_2	4	3	2	1
P_3	4	3	2	1

	A	B	C	D
P_1	1	2	3	4
P_2	4	3	1	2
P_3	3	2	4	1

Interruption model

$$dPr = \begin{cases} \kappa dt & \text{for } t \in [0, 1/\kappa] \\ 0 & \text{otherwise} \end{cases}$$

$$Pr(w) = \min\left\{1, \int_0^w \kappa dt\right\} = \min\{1, \kappa w\}$$

Goal: maximize expected work production

Interruption model

$$dPr = \begin{cases} \kappa dt & \text{for } t \in [0, 1/\kappa] \\ 0 & \text{otherwise} \end{cases}$$

$$Pr(w) = \min\left\{1, \int_0^w \kappa dt\right\} = \min\{1, \kappa w\}$$

Goal: maximize expected work production

Free-initiation model (1/2)

Regimen Θ: allocate whole workload on a single computer

$$E^{(\mathrm{f})}(\text{jobdone}, \Theta) \;=\; \int_0^{\infty} Pr(\text{jobdone} \geq u \text{ under } \Theta)\; du$$

Single chunk

$$E^{(\mathrm{f})}(W, \Theta_1) \;=\; W\,(1 - Pr(W))$$

Two chunks with $\omega_1 + \omega_2 = W$

$$E^{(\mathrm{f})}(W, \Theta_2) = \omega_1(1 - Pr(\omega_1)) \;+\; \omega_2(1 - Pr(\omega_1 + \omega_2))$$

Free-initiation model (2/2)

With n chunks, maximize

$$E^{(\mathrm{f})}(W, n) = \omega_1(1 - Pr(\omega_1)) + \omega_2(1 - Pr(\omega_1 + \omega_2)) \\ \cdots + \omega_n(1 - Pr(\omega_1 + \cdots + \omega_n))$$

where

$$\omega_1 > 0, \ \omega_2 > 0, \ldots, \ \omega_n > 0$$

$$\omega_1 + \omega_2 + \cdots + \omega_n \leq W$$

Charged-initiation model

$$E^{(c)}(\text{jobdone}) = \int_0^{\infty} Pr(\text{jobdone} \geq u + \varepsilon)\ du.$$

Single chunk

$$E^{(c)}(W, 1) = W\,(1 - Pr(W + \varepsilon))$$

Two chunks with $\omega_1 + \omega_2 \leq W$

$$E^{(c)}(W, 2) = \omega_1(1 - Pr(\omega_1 + \varepsilon)) + \omega_2(1 - Pr(\omega_1 + \omega_2 + 2\varepsilon))$$

Results

Theorem: Relating the two models

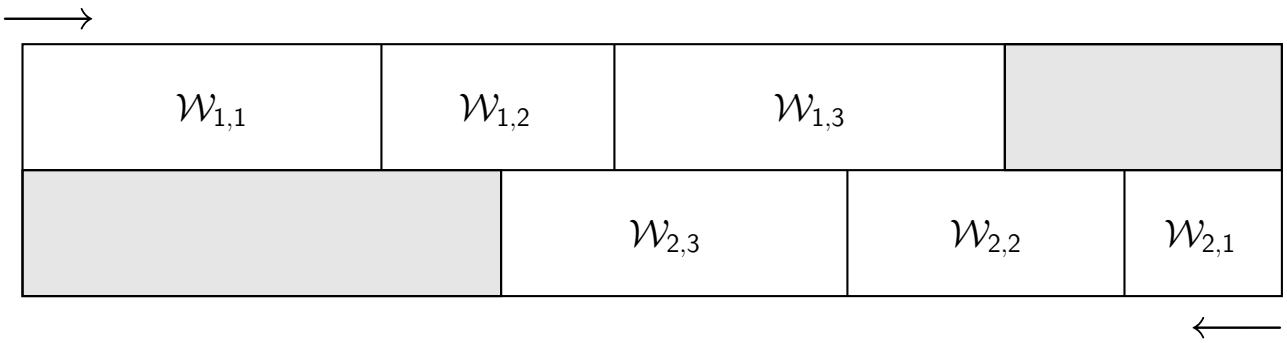
$$E^{(\mathrm{f})}(W,n) \;\geq\; E^{(\mathrm{c})}(W,n) \;\geq\; E^{(\mathrm{f})}(W,n) - n\varepsilon$$

Theorem: Free initiation model, 1 processor

Optimal schedule to deploy $W \in [0, \frac{1}{\kappa}]$ units of work in n chunks: use identical chunks of size Z/n:

$$Z \;=\; \min\left\{W, \frac{n}{n+1}\frac{1}{\kappa}\right\}, \quad E^{(\mathrm{f})}(W,n) \;=\; Z \;-\; \frac{n+1}{2n}Z^2\kappa$$

With 2 processors



Theorem

W_1 and $W2$ assigned workloads in optimal solution:

- 1. Either $W_1 \bigcap W2 = \emptyset$ or $W_1 \bigcup W2 = W$
- 2. P_1 processes $W_1 \setminus W2$ before $W_1 \bigcap W2$
- 3. P_1 and P_2 process $W_1 \bigcap W2$ in reverse order

☹ **Optimal out of reach even for 2 or 3 chunks per processor**

With p processors: Orchestrating

Chunk	1	2	3	4	5	6	7	8	9	10	11	12
P_1	1	6	9	12	2	5	8	11	3	4	7	10
P_2	12	1	6	9	11	2	5	8	10	3	4	7
P_3	9	12	1	6	8	11	2	5	7	10	3	4
P_4	6	9	12	1	5	8	11	2	4	7	10	3

Time-steps for execution of n = 12 chunks with g = 4 processors

With p processors: Group schedules

Chunk	1	2	3	4	5	6	7	8	9	10	11	12
P_1	1	6	9	12	2	5	8	11	3	4	7	10
P_2	12	1	6	9	11	2	5	8	10	3	4	7
P_3	9	12	1	6	8	11	2	5	7	10	3	4
P_4	6	9	12	1	5	8	11	2	4	7	10	3

Group 1 chunks 1-4	Group 2 chunks 5-8	Group 3 chunks 9-12
1	2	3
6	5	4
9	8	7
12	11	10

Time-steps for group execution

Group schedules

Group 1	Group 2	Group 3
1	2	3
6	5	4
9	8	7
12	11	10

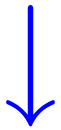
Group schedules

Group 1	Group 2	Group 3
1	2	3
6	5	4
9	8	7
12	11	10

All four executions fail with probability proportional to $1 \times 6 \times 9 \times 12$

Group schedules

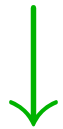
Group 1	Group 2	Group 3
1	2	3
6	5	4
9	8	7
12	11	10



All four executions fail with probability proportional to $2 \times 5 \times 8 \times 11$

Group schedules

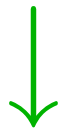
Group 1	Group 2	Group 3
1	2	3
6	5	4
9	8	7
12	11	10



All four executions fail with probability proportional to $3 \times 4 \times 7 \times 10$

Group schedules

Group 1	Group 2	Group 3
1	2	3
6	5	4
9	8	7
12	11	10



All four executions fail with probability proportional to $3 \times 4 \times 7 \times 10$

$$K = \sum_{j=1}^{\frac{n}{g}} \prod_{i=1}^{g} G_{i,j} = 1.6.9.12 + 2.5.8.11 + 3.4.7.10$$

Better performance for small K

Comparing group schedules for n = 20 and g = 4

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

$K_{cyclic} = 34104$

1	2	3	4	5
6	7	8	9	10
15	14	13	12	11
20	19	18	17	16

$K_{mirror} = 27284$

1	2	3	4	5
10	9	8	7	6
15	14	13	12	11
20	19	18	17	16

$K_{reverse} = 24396$

1	2	3	4	5
10	9	8	7	6
11	12	13	14	15
20	19	18	17	16

$K_{snake} = 25784$

1	2	3	4	5
14	12	10	8	6
15	13	11	9	7
16	17	18	19	20

$K_{worm} = 24276$

1	2	3	4	5
10	9	8	7	6
15	14	13	12	11
20	19	18	16	17

$K_{greedy} = 24390$

$K_{min} = 23780$

Conclusion

- Turned out much more difficult than expected (☺ or ☹?)
- Extension to resources with different risk functions
- Extension to resources with different computation capacities
- Master-slave approach with communication costs
- Comparison with dynamic approaches