Algorithms to handle uncertainties

Anne Benoit
Veronika Rehn-Sonigo, Yves Robert, Frederic Vivien
GRAAL team, LIP, École Normale Supérieure de Lyon, France
Arnold Rosenberg, Colorado State University, USA

ALEAE kick-off meeting in Paris
April 1st, 2009

Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
  dynamic platforms, change over time → uncertainties
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
  - need to model uncertainties
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Simple applications, but already challenging

Scheduling *simple applications* onto *dynamic platforms*

Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
  dynamic platforms, change over time → uncertainties
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
  - need to model uncertainties
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Simple applications, but already challenging

Scheduling *simple applications* onto *dynamic platforms*

Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
  dynamic platforms, change over time → uncertainties
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
  - need to model uncertainties
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Simple applications, but already challenging

Scheduling *simple applications* onto *dynamic platforms*

Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
  dynamic platforms, change over time → uncertainties
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
  - need to model uncertainties
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Simple applications, but already challenging

Scheduling *simple applications* onto *dynamic platforms*

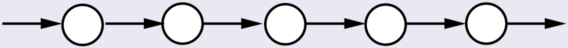
Introduction and motivation

- Scheduling applications onto parallel platforms:
  difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms:
  even more difficult!
  dynamic platforms, change over time → uncertainties
- Target platform
  - more or less heterogeneity
  - different communication models (overlap, one- vs multi-port)
  - need to model uncertainties
- Target application
  - Workflow: several data sets are processed by a set of tasks
  - Structured: independent tasks, linear chains, ...
  - Simple applications, but already challenging

Scheduling *simple applications* onto *dynamic platforms*

First problem: *Multi-criteria* scheduling of *workflows*

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

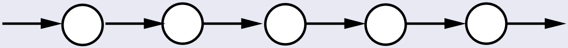
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

First problem: *Multi-criteria* scheduling of *workflows*

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

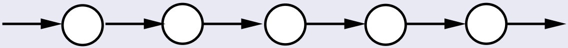
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

First problem: *Multi-criteria* scheduling of *workflows*

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

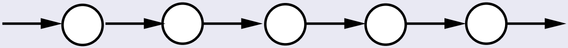
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

First problem: *Multi-criteria* scheduling of *workflows*

Workflow applications?



Several consecutive data sets enter the application graph.

Multi-criteria to optimize?

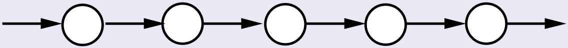
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data sets (inverse of throughput)

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

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

First problem: *Multi-criteria* scheduling of *workflows*

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

Second problem: *Divisible workload* with *failures*

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

Goal: maximize expected amount of work done

Major contributions

- Pb 1: Definition of workflow applications, computational platforms and communication models, multi-criteria mappings (including reliability issues)
  ⇒ Examples to illustrate problem complexity
- Pb 2: Definition of the failure model, the expected amount of work done, chunk sizes and replication
  ⇒ Optimality results for the one and two processor cases, inherent difficulties of this problem

Illustration through two problems of our algorithms and techniques to handle uncertainties

Proactive methods: replication for reliability

Major contributions

- Pb 1: Definition of workflow applications, computational platforms and communication models, multi-criteria mappings (including reliability issues)
  $\Rightarrow$ Examples to illustrate problem complexity
- Pb 2: Definition of the failure model, the expected amount of work done, chunk sizes and replication
  $\Rightarrow$ Optimality results for the one and two processor cases, inherent difficulties of this problem

Illustration through two problems of our algorithms and techniques to handle uncertainties

Proactive methods: replication for reliability

Major contributions

- Pb 1: Definition of workflow applications, computational platforms and communication models, multi-criteria mappings (including reliability issues)
  $\Rightarrow$ Examples to illustrate problem complexity
- Pb 2: Definition of the failure model, the expected amount of work done, chunk sizes and replication
  $\Rightarrow$ Optimality results for the one and two processor cases, inherent difficulties of this problem

Illustration through two problems of our algorithms and techniques to handle uncertainties

Proactive methods: replication for reliability

Major contributions

- Pb 1: Definition of workflow applications, computational platforms and communication models, multi-criteria mappings (including reliability issues)
  $\Rightarrow$ Examples to illustrate problem complexity
- Pb 2: Definition of the failure model, the expected amount of work done, chunk sizes and replication
  $\Rightarrow$ Optimality results for the one and two processor cases, inherent difficulties of this problem

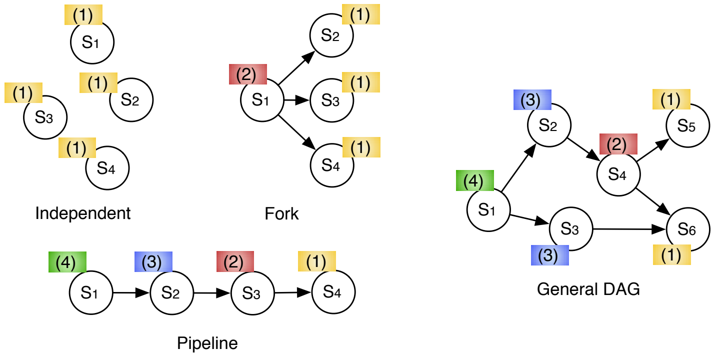
Illustration through two problems of our algorithms and techniques to handle uncertainties

Proactive methods: replication for reliability

Outline

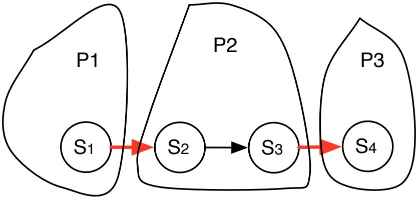
Application model

- 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

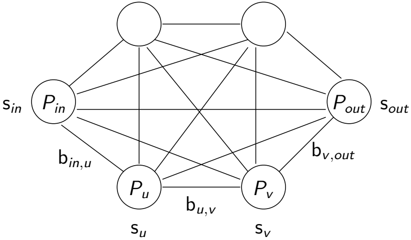


Application model: communication costs

- Two dependent stages $S_1 \rightarrow S_2$:
  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 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 $\text{link}_{u,v} : P_u \rightarrow P_v$, 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 - cycle-stealing scenario)
- 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

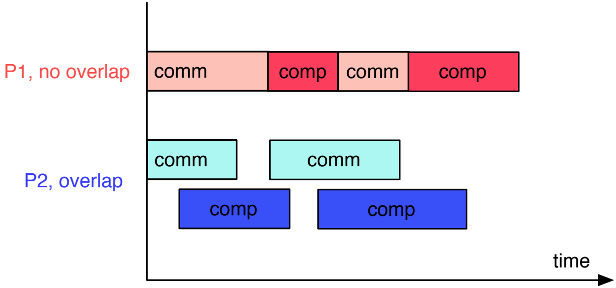
Fully Heterogeneous – Fully heterogeneous architectures, $s_u \neq s_v$ and $b_{u,v} \neq b_{u',v'}$: hierarchical platforms, grids

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

Platform model: communications

no overlap vs overlap

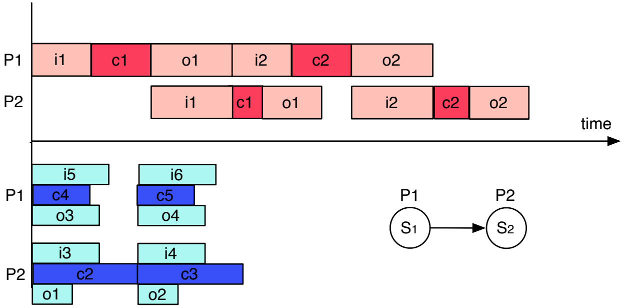
- no overlap: at each time step, either computation or communication
- overlap: a processor can simultaneously compute and communicate



Platform model: communications

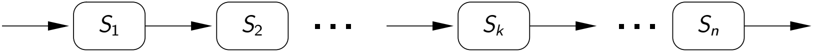
one-port vs multi-port

- 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

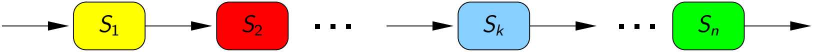


The pipeline application

- 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



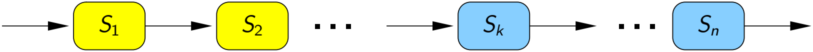
ONE-TO-ONE MAPPING

- 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

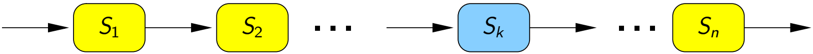


INTERVAL MAPPING

- 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

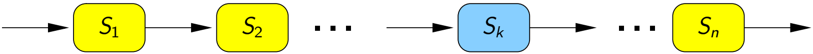


GENERAL MAPPING

- 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
- Dealable 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
- Dealable 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 $\mathcal{F}$

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

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L} + \gamma.\mathcal{F}$?
- 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 $\mathcal{F}$

Multi-criteria

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

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

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, m intervals
- Period/Latency/Reliability problem with replication only for reliability (monolithic stages)

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

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

$$\mathcal{P}^{(no)} = \max_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \left\{ \frac{\delta_{j-1}}{\min_{v \in \text{alloc}(j-1)} \mathsf{b}_{v,u}} + \frac{\sum_{i \in I_j} \mathsf{w}_i}{\mathsf{s}_u} + \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{\mathsf{b}_{u,v}} \right\}$$

An example of formal definitions

- Pipeline application, m intervals
- Period/Latency/Reliability problem with replication only for reliability (monolithic stages)

$$\mathcal{F} = 1 - \prod_{1 \leq j \leq m} \left(1 - \prod_{u \in \text{alloc}(j)} \mathsf{f}_u\right)$$

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

$$\mathcal{P}^{(no)} = \max_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \left\{ \frac{\delta_{j-1}}{\min_{v \in \text{alloc}(j-1)} \mathsf{b}_{v,u}} + \frac{\sum_{i \in I_j} \mathsf{w}_i}{\mathsf{s}_u} + \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{\mathsf{b}_{u,v}} \right\}$$

$$\mathcal{L} = \sum_{u \in \text{alloc}(1)} \frac{\delta_0}{\mathsf{b}_{in,u}} + \sum_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \left\{ \frac{\sum_{i \in I_j} \mathsf{w}_i}{\mathsf{s}_u} + \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{\mathsf{b}_{u,v}} \right\}$$

An example of formal definitions

- Pipeline application, m intervals
- Period/Latency/Reliability problem with replication only for reliability (monolithic stages)

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

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

$$\mathcal{P}^{(ov)} = \max_{1 \leq j \leq m} \max_{u \in \text{alloc}(j)} \max \left\{ \frac{\delta_{j-1}}{\min\limits_{v \in \text{alloc}(j-1)} \mathsf{b}_{v,u}}, \frac{\sum_{i \in I_j} \mathsf{w}_i}{\mathsf{s}_u}, \sum_{v \in \text{alloc}(j+1)} \frac{\delta_j}{\mathsf{b}_{u,v}} \right\}$$

An example of formal definitions

- Pipeline application, m intervals
- Period/Latency/Reliability problem with replication only for reliability (monolithic stages)

$$\mathcal{F} = 1 - \prod_{1 \leq j \leq m} (1 - \prod_{u \in \mathrm{alloc}(j)} \mathsf{f}_u)$$

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

$$\mathcal{P}^{(ov)} = \max_{1 \leq j \leq m} \max_{u \in \mathrm{alloc}(j)} \max \left\{ \frac{\delta_{j-1}}{\min_{v \in \mathrm{alloc}(j-1)} \mathsf{b}_{v,u}}, \frac{\sum_{i \in I_j} \mathsf{w}_i}{\mathsf{s}_u}, \sum_{v \in \mathrm{alloc}(j+1)} \frac{\delta_j}{\mathsf{b}_{u,v}} \right\}$$

$\mathcal{L}$ = 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

Complexity: working out examples

- Mono-criterion reliability: replicate the whole pipeline as a single interval on all processors
- Latency: one interval saves communication ☺
- Bi-criteria (reliability/latency) polynomial algorithm for *Communication Homogeneous-Failure Homogeneous* platforms
- Much more difficult with *Failure Heterogeneous*, open complexity (see following example)
- And think about adding replication for period into the story...

Complexity: working out examples

- Mono-criterion reliability: replicate the whole pipeline as a single interval on all processors
- Latency: one interval saves communication ☺
- Bi-criteria (reliability/latency) polynomial algorithm for *Communication Homogeneous-Failure Homogeneous* platforms
- Much more difficult with *Failure Heterogeneous*, open complexity (see following example)
- And think about adding replication for period into the story...

Complexity: working out examples

- Mono-criterion reliability: replicate the whole pipeline as a single interval on all processors
- Latency: one interval saves communication ☺
- Bi-criteria (reliability/latency) polynomial algorithm for *Communication Homogeneous-Failure Homogeneous* platforms
- Much more difficult with *Failure Heterogeneous*, open complexity (see following example)
- And think about adding replication for period into the story...

Complexity: working out examples

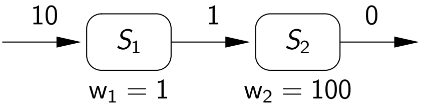
- Mono-criterion reliability: replicate the whole pipeline as a single interval on all processors
- Latency: one interval saves communication ☺
- Bi-criteria (reliability/latency) polynomial algorithm for *Communication Homogeneous-Failure Homogeneous* platforms
- Much more difficult with *Failure Heterogeneous*, open complexity (see following example)
- And think about adding replication for period into the story...

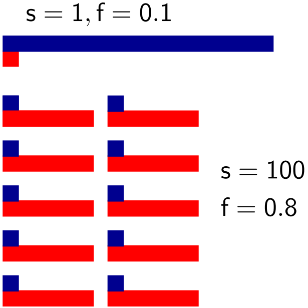
Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous

Fixed latency: 22





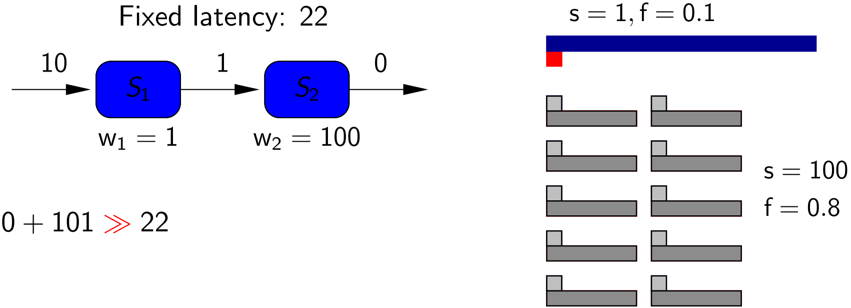
Open complexity

Impact of communication model on period and latency?

Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous



$10 + 101 \gg 22$

Open complexity

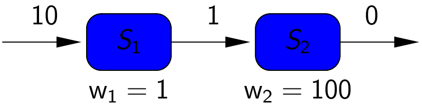
Impact of communication model on period and latency?

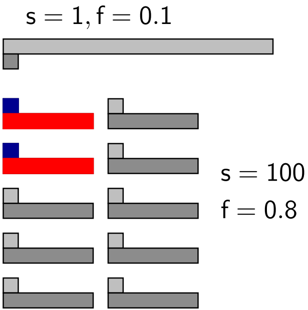
Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous

Fixed latency: 22





$20 + 101/100 < 22$

$\mathcal{F} = (1 - (1 - 0.8^2)) = 0.64$

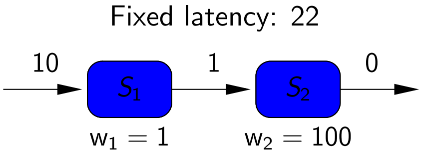
Open complexity

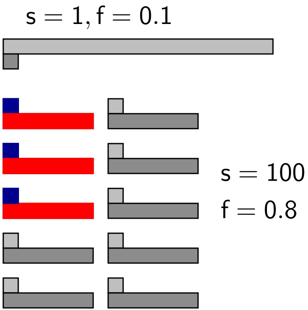
Impact of communication model on period and latency?

Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous





$30 + 101/100 > 22$

Open complexity

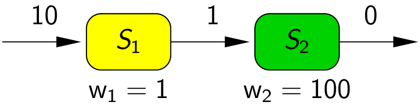
Impact of communication model on period and latency?

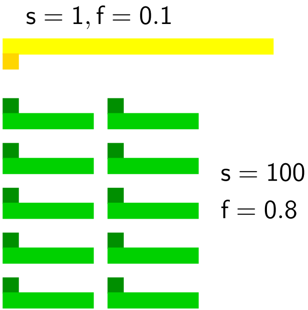
Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous

Fixed latency: 22





$10 + 1/1 + 10 \times 1 + 100/100 = 22$

$\mathcal{F} : 1 - (1 - 0.1) \times (1 - 0.8^{10}) < 0.2$

Open complexity

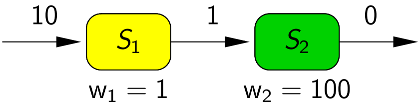
Impact of communication model on period and latency?

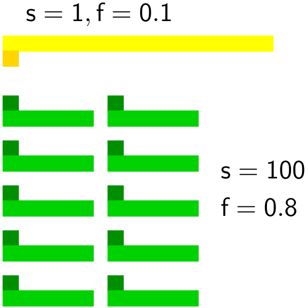
Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous

Fixed latency: 22





$$10 + 1/1 + 10 \times 1 + 100/100 = 22$$
$$\mathcal{F} : 1 - (1 - 0.1) \times (1 - 0.8^{10}) < 0.2$$

Open complexity

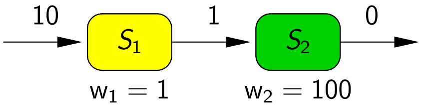
Impact of communication model on period and latency?

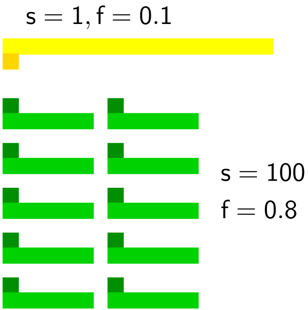
Bi-criteria reliability/latency - Interval mapping

Minimize $\mathcal{F}$ with fixed latency

Communication homogeneous - Failure heterogeneous

Fixed latency: 22





$10 + 1/1 + 10 \times 1 + 100/100 = 22$

$\mathcal{F} : 1 - (1 - 0.1) \times (1 - 0.8^{10}) < 0.2$

Open complexity

Impact of communication model on 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 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?

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

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

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

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

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?

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?

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

Perfect load-balancing both for computation and comm.

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

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

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

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

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

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

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

Complexity results for Pb 1

$\mathcal{F}$	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

Pb 2: Chunking

- Large divisible computational workload, to execute on p identical processors subject to unrecoverable interruptions

- 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

Pb 2: Chunking

- Large divisible computational workload, to execute on p identical processors subject to unrecoverable interruptions

- 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

Pb 2: Chunking

- Large divisible computational workload, to execute on p identical processors subject to unrecoverable interruptions
- 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

Pb 2: 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

Pb 2: 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

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

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

Goal: maximize expected work production

Interruption model

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

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

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 (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 (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 $\omega_1 + \omega_2 \leq W$

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

Charged-initiation model

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

Single chunk

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

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

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

Charged-initiation model

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

Single chunk

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

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

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

Some results

Theorem: Relating the two models

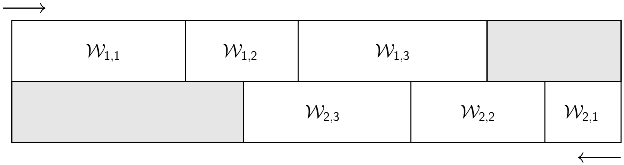
$$E^{(\mathrm{f})}(W,n) \ \geq \ E^{(\mathrm{c})}(W,n) \ \geq \ E^{(\mathrm{f})}(W,n) - n\varepsilon$$

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:

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

2 computers: general shape of optimal solution



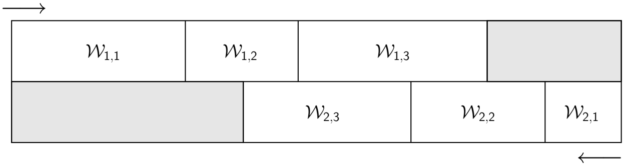
Theorem

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**

2 computers: general shape of optimal solution



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**

Lessons learnt from Problem 2

- Probability law to model interruptions → problem rapidly untractable
- Difficult to decide the size of chunks
- With more than one processor, difficult to decide which part of the work should be replicated
- Optimal out of reach: heuristics (structured solution), upper and lower bounds, experiments

- Proactive methods already turn out to be challenging, we did not investigate reactive methods so far

Outline

1. Problem 1
2. Problem 2
3. Conclusion

Related work

Problem 1:

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

Problem 2:

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

Conclusion

Problem 1:

- Definition of applications, platforms, multi-criteria mappings, failure models
- Working out examples to show insight of problem complexity, full complexity study, linear program formulations (NP-hard instances)
- Practical side: Several polynomial heuristics and simulations, JPEG application, good results of the heuristics (close to LP solution)

Problem 2:

- 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