Scheduling for Reliability: Complexity and Algorithms

Fanny Dufossé

Advisors:
Anne Benoit and Yves Robert

Roma team, LIP
Ecole Normale Supérieure de Lyon

6 September 2011

Context

Increasing need of computing power:

- simulations (weather, nuclear, ...)
- data processing

A solution to increase the computing power: parallelization

- split a computation into many smaller ones
- execute each small task on one computer

More computers → high probability of failure

Some areas are extremely sensitive to failures:

- critical transportations (airplanes,...)
- nuclear power plants

Context

Increasing need of computing power:

- simulations (weather, nuclear, ...)
- data processing

A solution to increase the computing power: parallelization

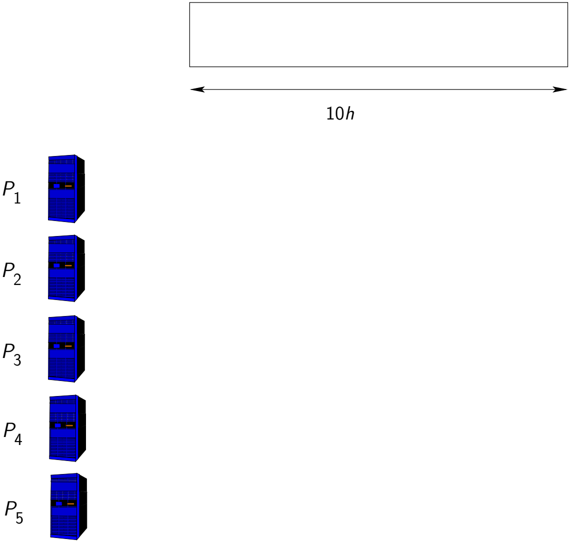
- split a computation into many smaller ones
- execute each small task on one computer

More computers → high probability of failure

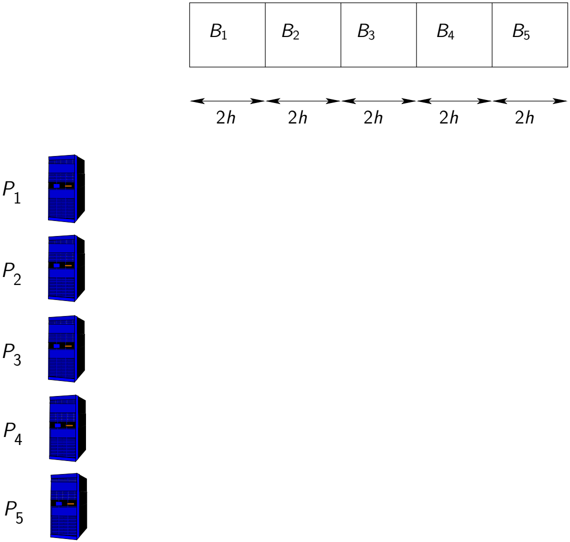
Some areas are extremely sensitive to failures:

- critical transportations (airplanes,...)
- nuclear power plants

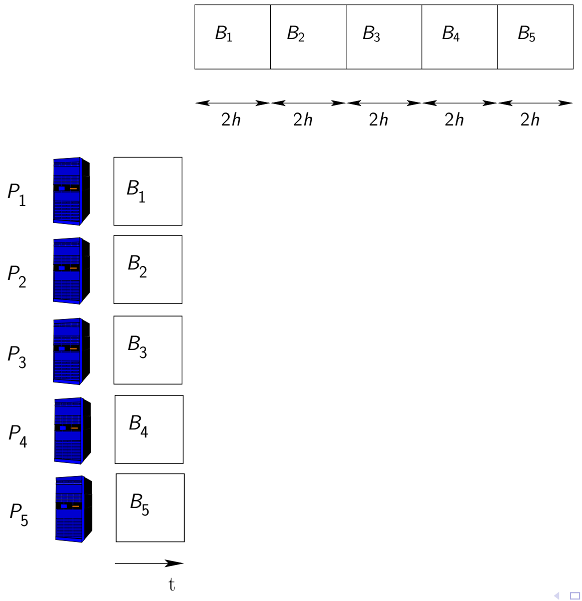
Idea



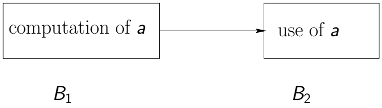
Idea



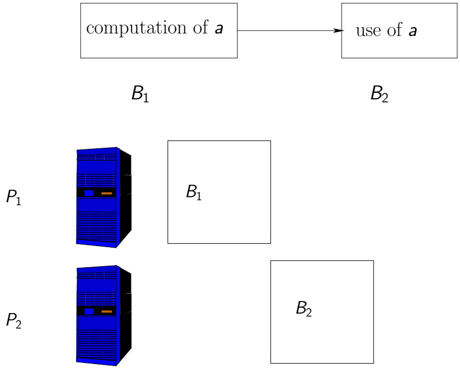
Idea



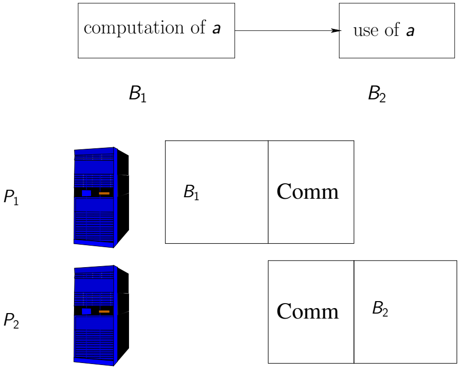
Precedence constraints and DAGs



Precedence constraints and DAGs

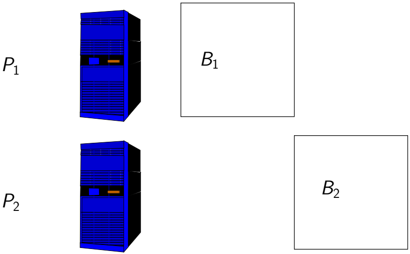


Precedence constraints and DAGs



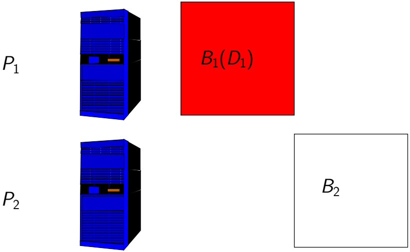
Objectives

- Latency



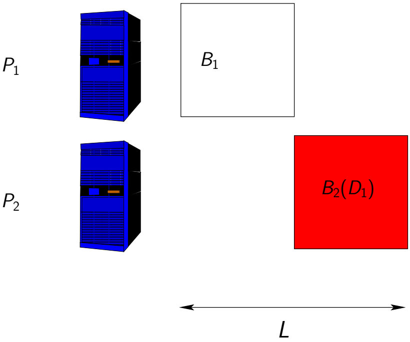
Objectives

- Latency



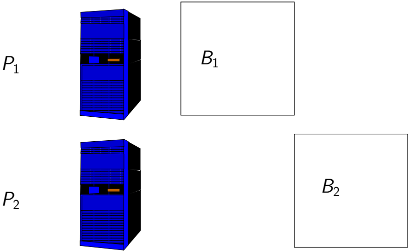
Objectives

- Latency



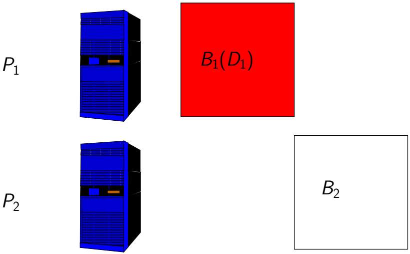
Objectives

- Latency
- Period



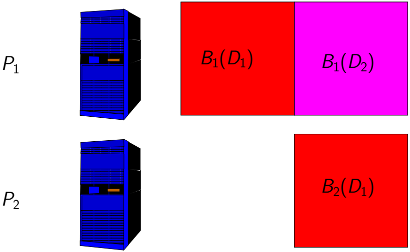
Objectives

- Latency
- Period



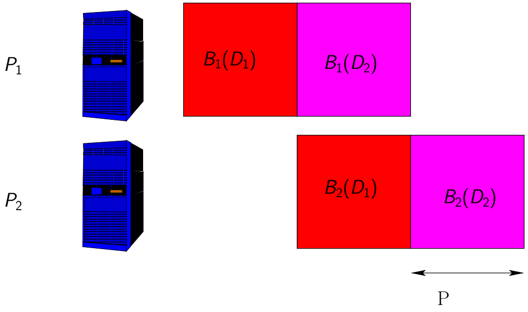
Objectives

- Latency
- Period



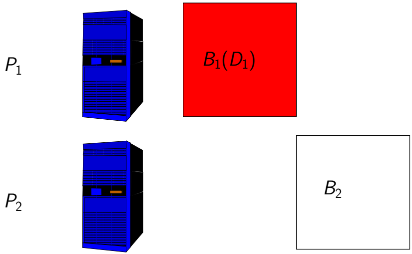
Objectives

- Latency
- Period



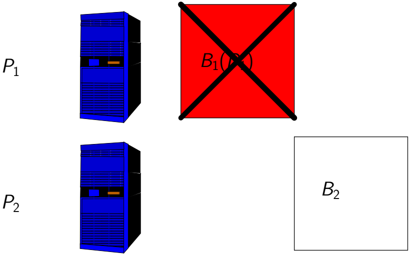
Objectives

- Latency
- Period
- Reliability



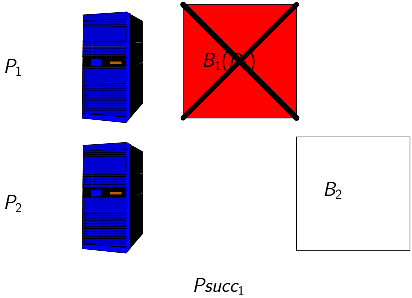
Objectives

- Latency
- Period
- Reliability



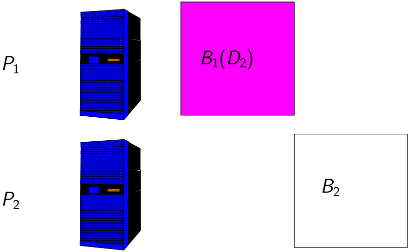
Objectives

- Latency
- Period
- Reliability



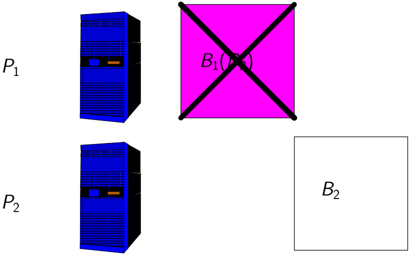
Objectives

- Latency
- Period
- Reliability



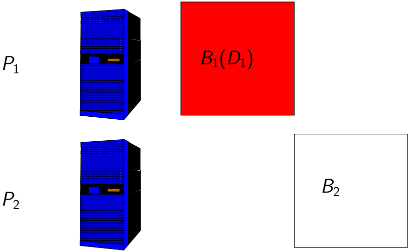
Objectives

- Latency
- Period
- Reliability



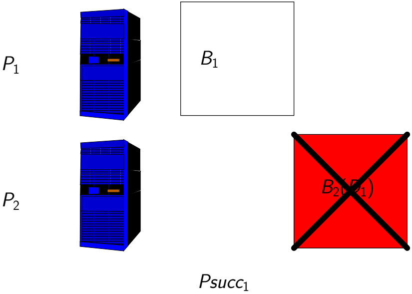
Objectives

- Latency
- Period
- Reliability



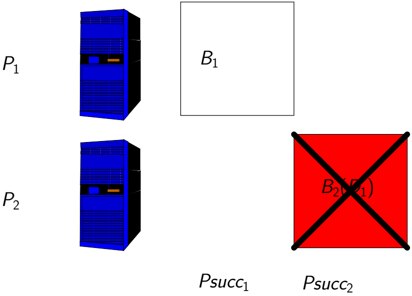
Objectives

- Latency
- Period
- Reliability



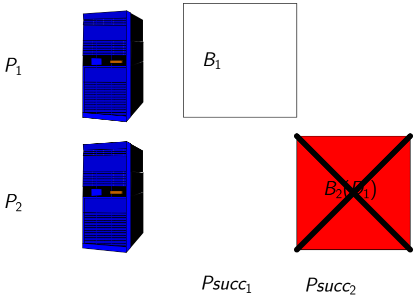
Objectives

- Latency
- Period
- Reliability



Objectives

- Latency
- Period
- Reliability



$$Psucc = Psucc_1 \times Psucc_2$$

Reliability of processors

Computers are subject to failures:

- transient failures
- fail-stop failures

Methods used to increase the reliability:

- Replication
- Checkpointing
- Migration

Reliability of processors

Computers are subject to failures:

- transient failures
- fail-stop failures

Methods used to increase the reliability:

- Replication
- Checkpointing
- Migration

Reliability of processors

Computers are subject to failures:

- transient failures
- fail-stop failures

Methods used to increase the reliability:

- Replication
- Checkpointing
- Migration

Approach

Roadmap:

- efficiently allocate tasks to processors
- design fast algorithms for these problems
- assess the complexity of these problems

For polynomial problems:

- provide optimal polynomial algorithms

For NP-complete problems:

- prove NP-completeness
- design optimal (exponential) algorithms (ILP,...)
- design approximation algorithms
- find bounds on approximation ratio
- design efficient heuristics and perform simulations

Approach

Roadmap:

- efficiently allocate tasks to processors
- design fast algorithms for these problems
- assess the complexity of these problems

For polynomial problems:

- provide optimal polynomial algorithms

For NP-complete problems:

- prove NP-completeness
- design optimal (exponential) algorithms (ILP,...)
- design approximation algorithms
- find bounds on approximation ratio
- design efficient heuristics and perform simulations

Approach

Roadmap:

- efficiently allocate tasks to processors
- design fast algorithms for these problems
- assess the complexity of these problems

For polynomial problems:

- provide optimal polynomial algorithms

For NP-complete problems:

- prove NP-completeness
- design optimal (exponential) algorithms (ILP,...)
- design approximation algorithms
- find bounds on approximation ratio
- design efficient heuristics and perform simulations

Plan

Outline

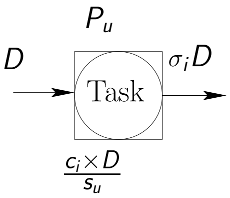
Application Model

Problem under study:

- streaming applications
- filtering tasks with selectivity σ_i and cost c_i (data bases, web services,...)
- servers with speed s_u
- possibility to add dependencies
- one-to-one mapping

Objective:

- minimize period
- minimize latency



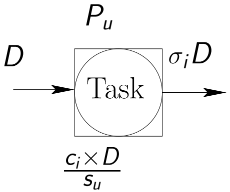
Application Model

Problem under study:

- streaming applications
- filtering tasks with selectivity σ_i and cost c_i (data bases, web services,...)
- servers with speed s_u
- possibility to add dependencies
- one-to-one mapping

Objective:

- minimize period
- minimize latency



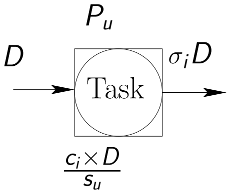
Application Model

Problem under study:

- streaming applications
- filtering tasks with selectivity σ_i and cost c_i (data bases, web services,...)
- servers with speed s_u
- possibility to add dependencies
- one-to-one mapping

Objective:

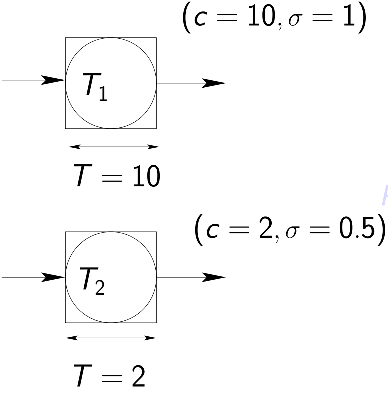
- minimize period
- minimize latency



Example

Instance with two independent tasks and two identical processors:

- $\sigma_1 = 1, \; c_1 = 10$
- $\sigma_2 = 0.5, \; c_2 = 2$
- $s = 1$
- $D = 1$

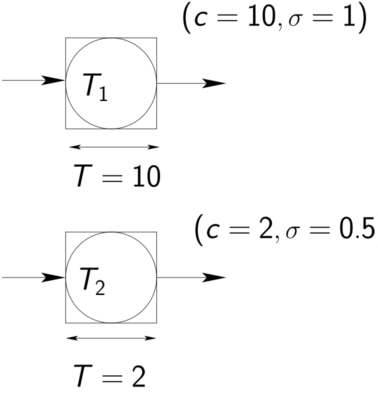


$P = 10, \; L = 10$

Example

Instance with two independent tasks and two identical processors:

- $\sigma_1 = 1,\ c_1 = 10$
- $\sigma_2 = 0.5,\ c_2 = 2$
- $s = 1$
- $D = 1$

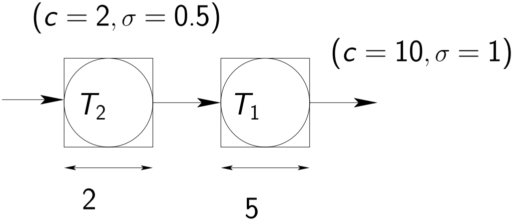


$P = 10,\ L = 10$

Example

Instance with two independent tasks and two identical processors:

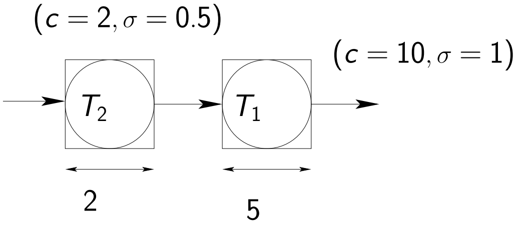
- $\sigma_1 = 1,\ c_1 = 10$
- $\sigma_2 = 0.5,\ c_2 = 2$
- $s = 1$
- $D = 1$



Example

Instance with two independent tasks and two identical processors:

- $\sigma_1 = 1,\ c_1 = 10$
- $\sigma_2 = 0.5,\ c_2 = 2$
- $s = 1$
- $D = 1$

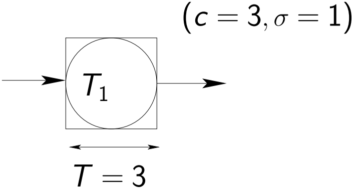


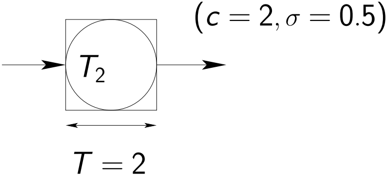
$P = 5,\ L = 7$

Example

Instance with two independent tasks and two identical processors:

- $\sigma_1 = 1$, $c_1 = 3$
- $\sigma_2 = 0.5$, $c_2 = 2$
- $s = 1$
- $D = 1$

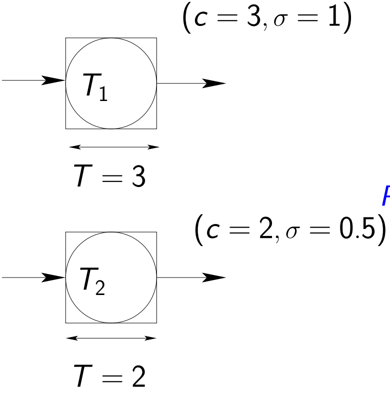




Example

Instance with two independent tasks and two identical processors:

- $\sigma_1 = 1$, $c_1 = 3$
- $\sigma_2 = 0.5$, $c_2 = 2$
- $s = 1$
- $D = 1$

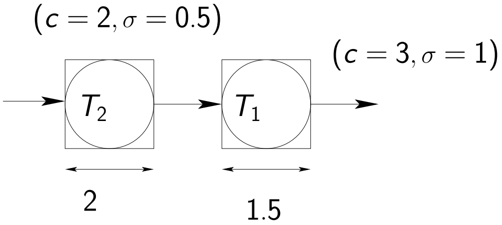


$P = 3,\ L = 3$

Example

Instance with two independent tasks and two identical processors:

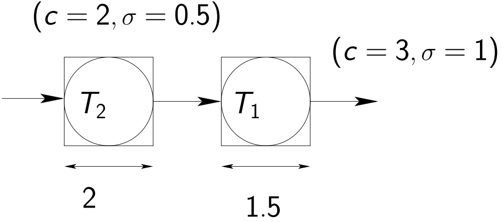
- $\sigma_1 = 1$, $c_1 = 3$
- $\sigma_2 = 0.5$, $c_2 = 2$
- $s = 1$
- $D = 1$



Example

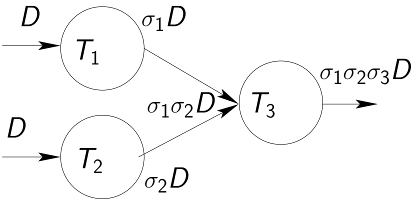
Instance with two independent tasks and two identical processors:

- $\sigma_1 = 1$, $c_1 = 3$
- $\sigma_2 = 0.5$, $c_2 = 2$
- $s = 1$
- $D = 1$



$P = 2$, $L = 3.5$

Example



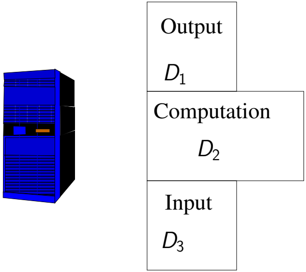
Combining selectivities

$$\mathcal{P} = \max\left(\frac{c_1}{s_1}, \frac{c_2}{s_2}, \frac{\sigma_1\sigma_2 c_3}{s_3}\right)$$

$$\mathcal{L} = \max\left(\frac{c_1}{s_1}, \frac{c_2}{s_2}\right) + \frac{\sigma_1\sigma_2 c_3}{s_3}$$

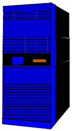
Communication models

- OVERLAP: overlap between communications and computations



Communication models

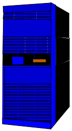
- OVERLAP: overlap between communications and computations
- INORDER: no overlap and FIFO execution of data sets by processors



Input	Computation	Output
D_1	D_1	D_1

Communication models

- OVERLAP: overlap between communications and computations
- INORDER: no overlap and FIFO execution of data sets by processors
- OUTORDER: no overlap and any possible executions order



Computation	Input	Output
D_1	D_2	D_1

General problem

Instance description:

- set of tasks
- dependence graph of these tasks
- set of processors
- communication model
- objective

The schedule:

- a plan (possibility to add dependencies)
- an allocation to processors (if they are heterogeneous)
- the execution times of computations and communications

General problem

Instance description:

- set of tasks
- dependence graph of these tasks
- set of processors
- communication model
- objective

The schedule:

- a plan (possibility to add dependencies)
- an allocation to processors (if they are heterogeneous)
- the execution times of computations and communications

General problem

Instance description:

- set of tasks
- dependence graph of these tasks
- set of processors
- communication model
- objective

The schedule:

- a plan (possibility to add dependencies)
- an allocation to processors (if they are heterogeneous)
- the execution times of computations and communications

General problem

Instance description:

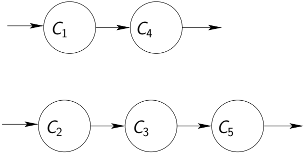
- set of tasks
- dependence graph of these tasks
- set of processors
- communication model
- objective

The schedule:

- a plan (possibility to add dependencies)
- an allocation to processors (if they are heterogeneous)
- the execution times of computations and communications

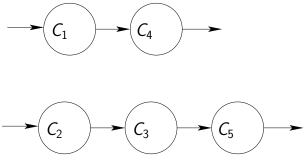
Example

Tasks and precedence constraints:

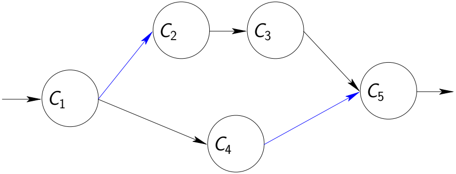


Example

Tasks and precedence constraints:

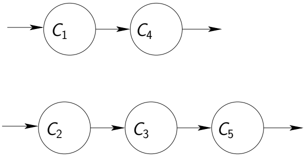


The plan:

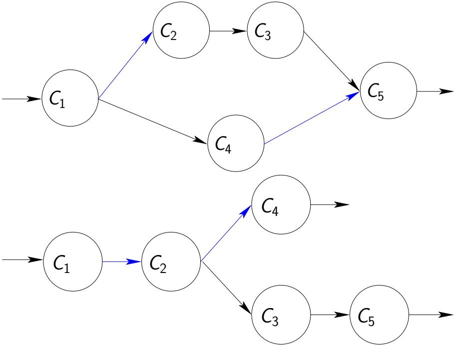


Example

Tasks and precedence constraints:

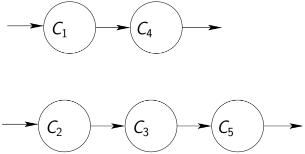


The plan:

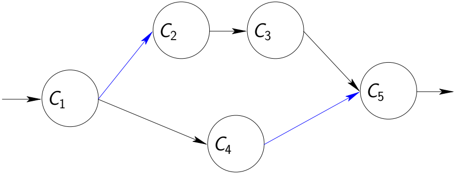


Example

Tasks and precedence constraints:

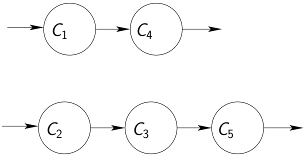


The plan:

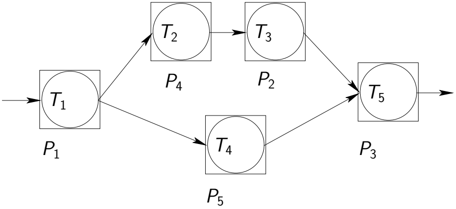


Example

Tasks and precedence constraints:

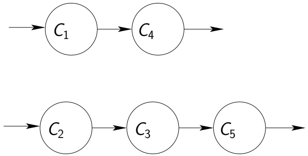


The allocation:

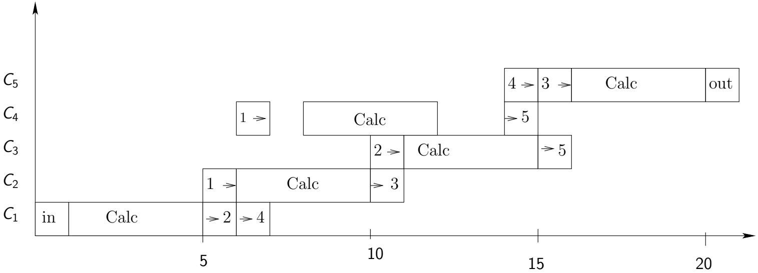


Example

Tasks and precedence constraints:



Execution times:



Complexity results

	Period	Latency
Hom. without comm.	Polynomial	Polynomial
Het. without comm.	Polynomial	Polynomial
Hom. with comm.	OVERLAP: Polynomial Other models: NP-hard	NP-hard

Complexity results for a given mapping

Complexity results

	Period	Latency
Hom. without comm.	Polynomial	Polynomial
Het. without comm.	NP-hard Inapproximable	NP-hard Inapproximable
Hom. with comm.	NP-hard	NP-hard

Complexity results for computing the optimal mapping

Conclusion for this study

Theoretical results

- a complete set of complexity results
- some approximation results
- integer linear program for some problems

Simulation results

- heuristics for the model with no communication costs
- simulations

Perspectives

- heuristics and simulations including communication costs
- approximation results for all NP-complete problems
- extend the model to replication

Outline

1. Introduction
2. Scheduling filtering applications (overview)
3. Reliability of pipelined real-time systems (overview)
4. Scheduling on volatile ressources
5. Conclusion and perspectives

Pipelined real-time systems

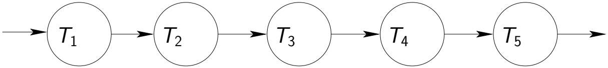
Real-time systems:

- jobs released
- deadline for each job

Pipelined real-time systems:

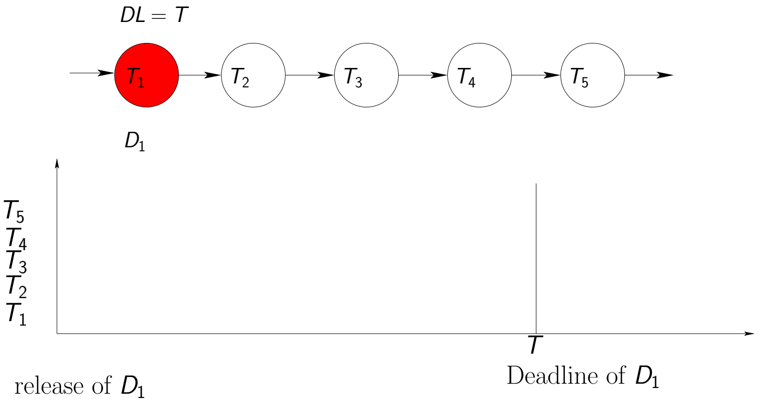
- a chain of tasks
- data sets are periodically released
- a deadline for each data set
- deadlines are periodic

Example



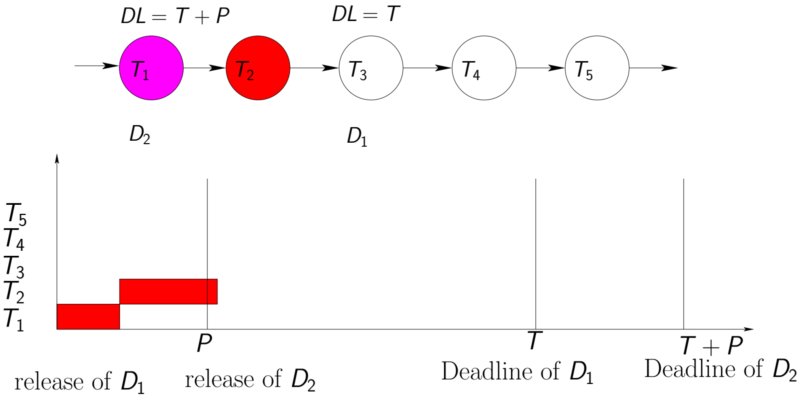
Example

$t = 0$



Example

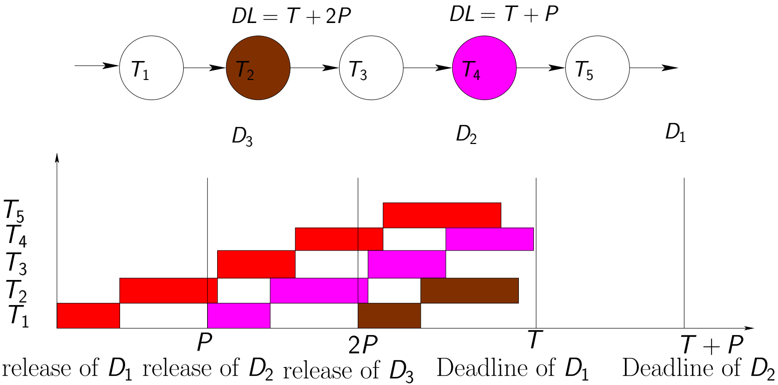
$t = P$



$DL = T + P$
$DL = T$
T_1
T_2
T_3
T_4
T_5
D_2
D_1
T_5
T_4
T_3
T_2
T_1
P
T
$T + P$
release of D_1
release of D_2
Deadline of D_1
Deadline of D_2

Example

$t = T$



$DL = T + 2P$
$DL = T + P$
T_1
T_2
T_3
T_4
T_5
D_3
D_2
D_1
T_5
T_4
T_3
T_2
T_1
P
$2P$
T
$T + P$
release of D_1
release of D_2
release of D_3
Deadline of D_1
Deadline of D_2

General model

Application model:

A chain of n tasks. Task T_k is characterized by:

- cost c_k
- output data size o_k (we suppose $o_n = 0$)

Platform:

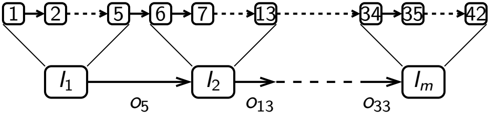
n processors. Processor P_u is characterized by:

- speed s_u
- failure rate per time unit λ_u

A processor can simultaneously execute some task and communicate data.

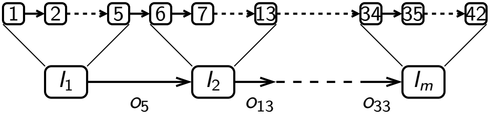
Interval mapping

The chain of tasks is divided into m intervals.

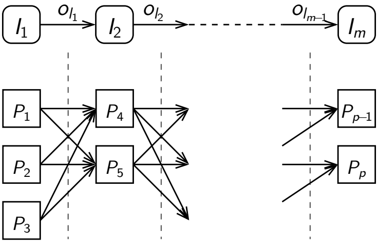


Interval mapping

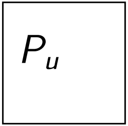
The chain of tasks is divided into m intervals.



The intervals are replicated on several processors.
Each processor executes only one interval.
Bound $\mathcal{K}$ on the number of replication of an interval.



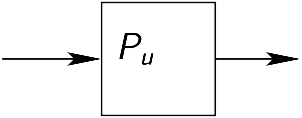
Failure model



Probability of success for computation of an interval I_i on P_u:

$$r_{u,i} = e^{-\lambda_u \times \frac{c_i}{s_u}}$$

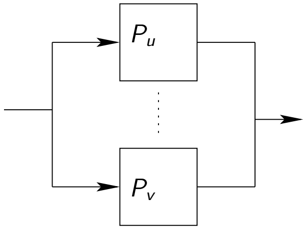
Failure model



Probability of success of an interval I_i of size W_i on P_u including communications:

$$r_{u,I_i} = r_{comm,i-1} \times e^{-\lambda_u \times \frac{c_i}{s_u}} \times r_{comm,i}$$

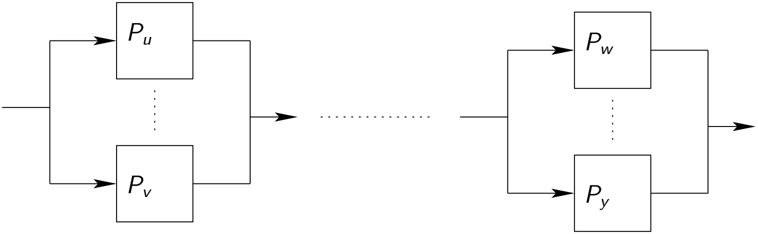
Failure model



Reliability of interval I_i on the set of processors $\mathcal{P}_i$ including communications:

$$1 - \prod_{P_u \in \mathcal{P}_i} (1 - r_{comm,i-1} \times r_{u,I_i} \times r_{comm,i})$$

Failure model



Reliability of a schedule:

$$r = \prod_{i=1}^{t} \left(1 - \prod_{P_u \in \mathcal{P}_i} (1 - r_{comm,i-1} \times r_{u,I_i} \times r_{comm,i}) \right)$$

Complexity results

	mono-criteria	bi-criteria	three-criteria
homogeneous case	period: Poly latency: Poly reliability: Poly	reliability-latency: NP-c reliability-period: Poly latency-period: Poly	NP-c
heterogeneous case	reliability: NP-c period: NP-c latency: Poly	NP-c	NP-c

Conclusion for this study

Theoretical results

- Realistic scenario for a classical model
- A complete theoretical study

Simulation results

- Heuristics for period and latency optimization
- Simulations

Perspectives

- More realistic probability distributions

Outline

Dealing with volatile resources

- Deploy applications on desktop grids (SETI@home,...)
- Iterative applications [Bahi07,Heddaya94]
- Resource availability: *UP*/ *DOWN* processes

The goal: **on-line** policies for resource selection:
- Which resources to enroll?
- How to compare configurations?

Dealing with volatile resources

- Deploy applications on desktop grids (SETI@home,...)
- Iterative applications [Bahi07,Heddaya94]
- Resource availability: *UP*/ *DOWN* processes

The goal: **on-line** policies for resource selection:
- Which resources to enroll?
- How to compare configurations?

Main assumptions

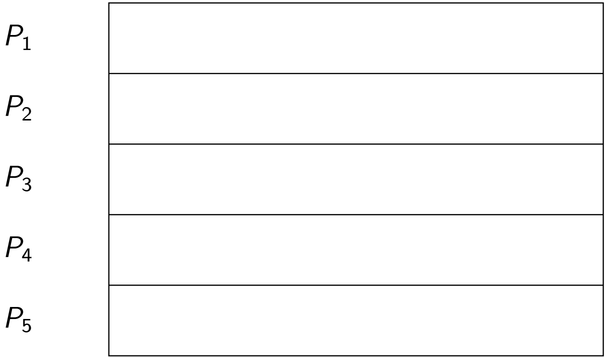
Problem

- Iterative application
- Master-worker paradigm
- Synchronization after each iteration
- Volatile platforms: transient failures & preemption
- Heterogeneous processors
- Limited available bandwidth from master to workers

Objective: Maximize expected number of iterations executed within limited time

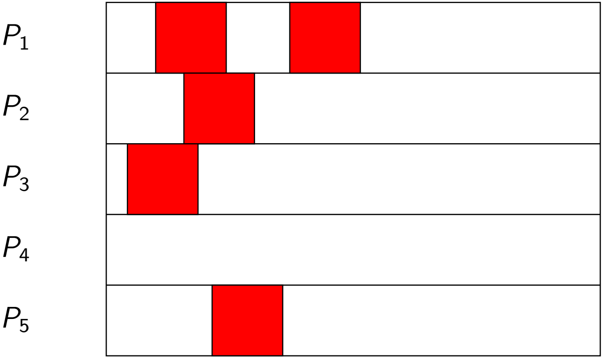
Example of iterative application

- 5 tasks
- 5 processors



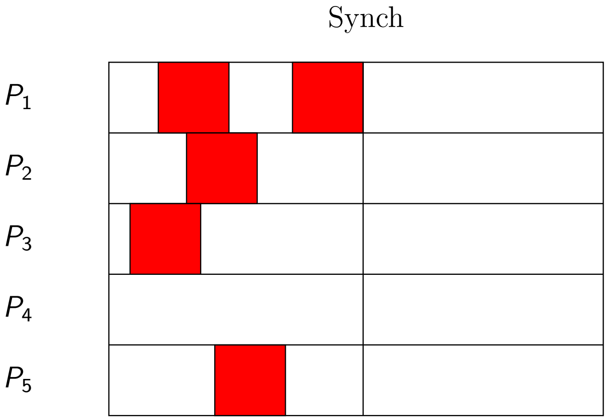
Example of iterative application

- 5 tasks
- 5 processors



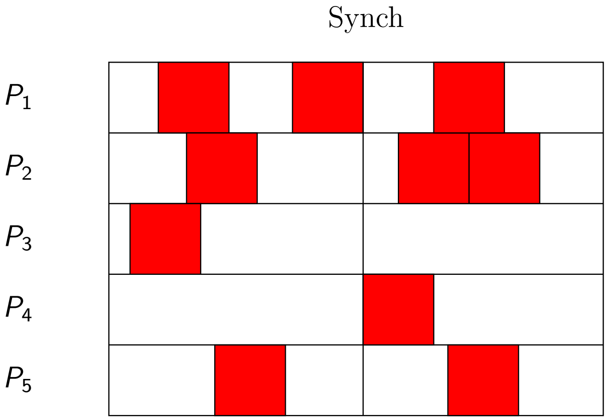
Example of iterative application

- 5 tasks
- 5 processors



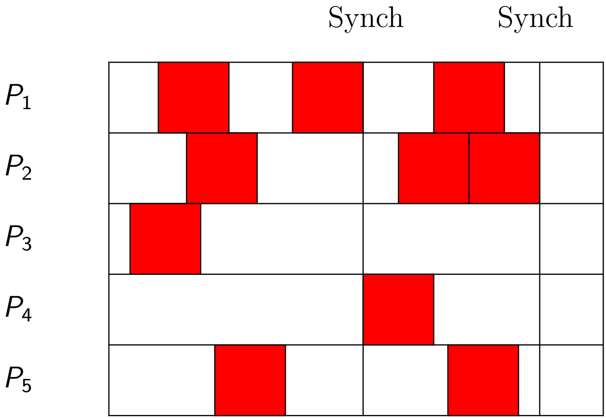
Example of iterative application

- 5 tasks
- 5 processors



Example of iterative application

- 5 tasks
- 5 processors



A realistic model (1/2)

Application

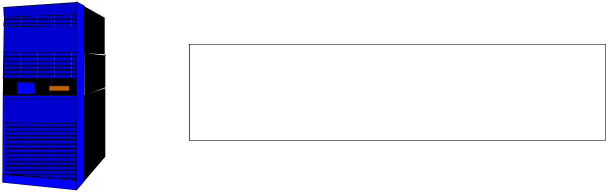
- Successive iterations
- Synchronization after each iteration
- For each iteration, m same-size tasks
- Two scenarios

TIGHTLY-COUPLED	continuously interacting tasks
INDEPENDENT	independent tasks

- Same program of size V_{prog} for all iterations
- Data set of size V_{data} for each task

Independent vs Tightly-Coupled scenarios

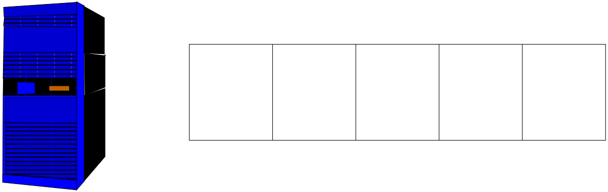
- 5 tasks
- 1 processor
- $T_{prog} = T_{data} = 0$



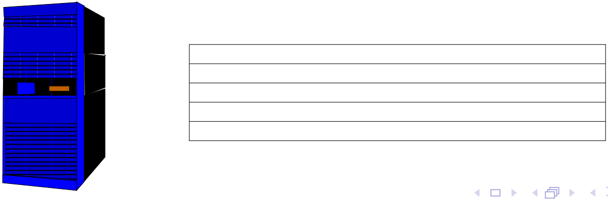
Independent vs Tightly-Coupled scenarios

- 5 tasks
- 1 processor
- $T_{prog} = T_{data} = 0$

Independent:



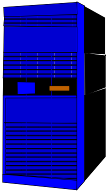
Tightly-Coupled:

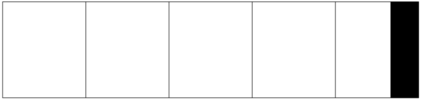


Independent vs Tightly-Coupled scenarios

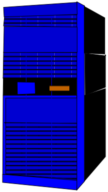
- 5 tasks
- 1 processor
- $T_{prog} = T_{data} = 0$

Independent:





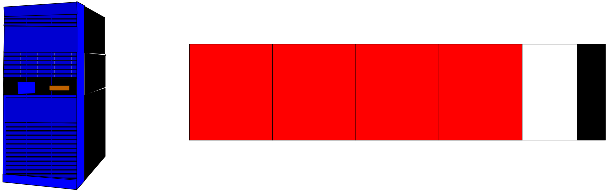
Tightly-Coupled:



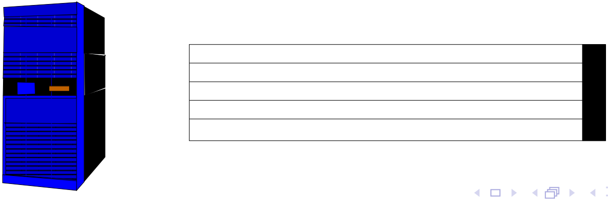
Independent vs Tightly-Coupled scenarios

- 5 tasks
- 1 processor
- $T_{prog} = T_{data} = 0$

Independent:

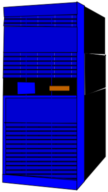


Tightly-Coupled:



Independent vs Tightly-Coupled scenarios

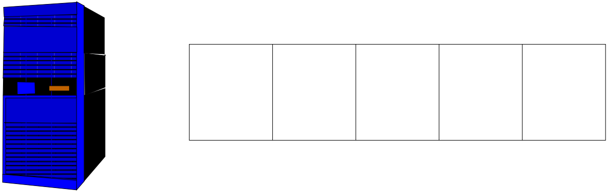
- 5 tasks
- 1 processor
- $T_{prog} = W = 0$



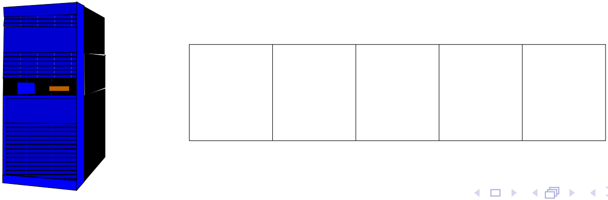
INDEPENDENT vs TIGHTLY-COUPLED scenarios

- 5 tasks
- 1 processor
- $T_{prog} = W = 0$

INDEPENDENT:



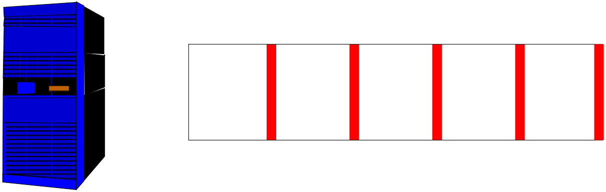
TIGHTLY-COUPLED:



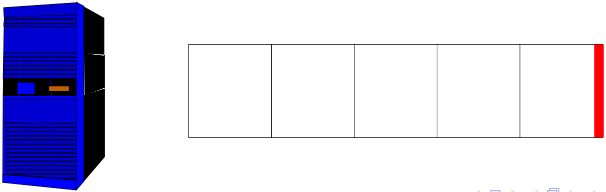
Independent vs Tightly-Coupled scenarios

- 5 tasks
- 1 processor
- $T_{prog} = W = 0$

Independent:



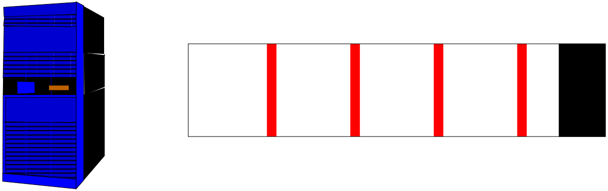
Tightly-Coupled:



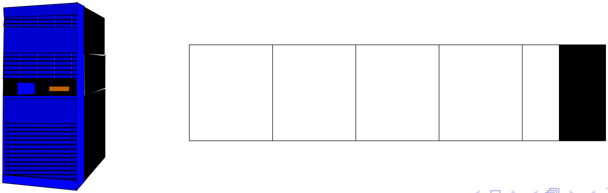
Independent vs Tightly-Coupled scenarios

- 5 tasks
- 1 processor
- $T_{prog} = W = 0$

Independent:



Tightly-Coupled:

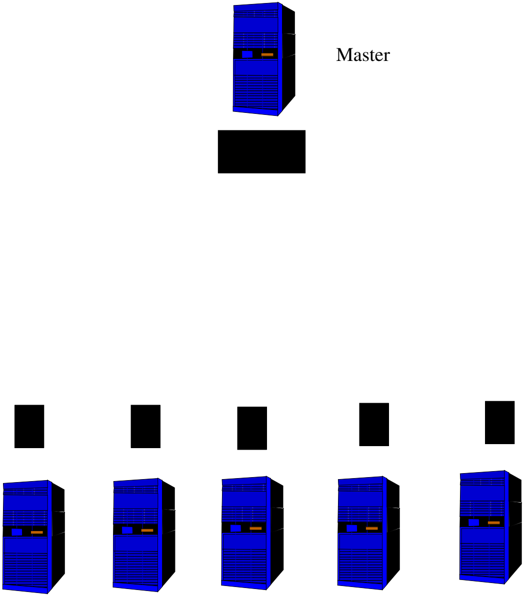


A realistic model (2/2)

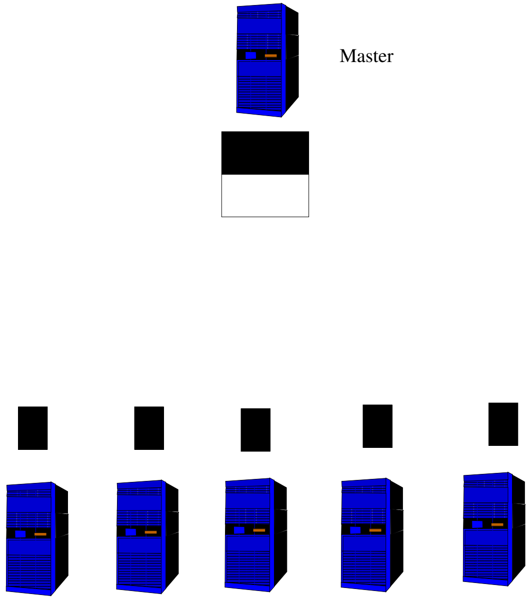
Platform

- Master-worker execution
- p heterogeneous resources/workers
  $\Rightarrow$ W_u cost of a task on processor P_u
- Limited bandwidth
  $\Rightarrow$ BW for master and bw for workers
  $\Rightarrow$ $n_{com} = \lfloor \frac{\text{BW}}{\text{bw}} \rfloor$ max number of simultaneous comms
- Overlap between computation and communication

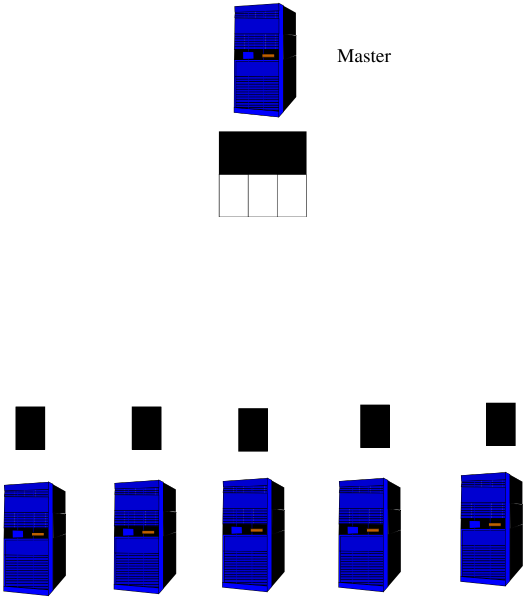
Communication model



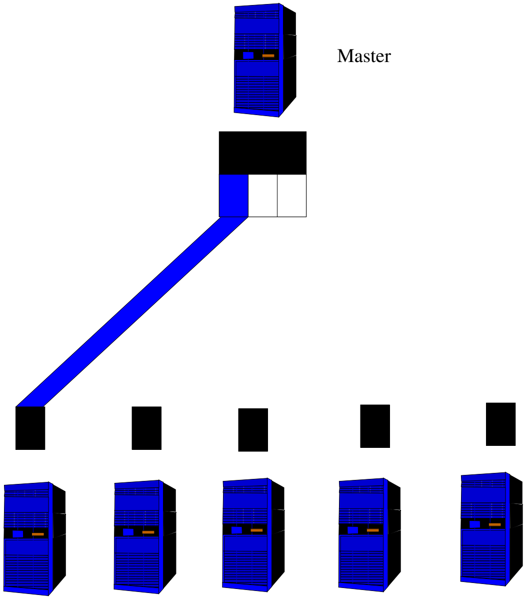
Communication model



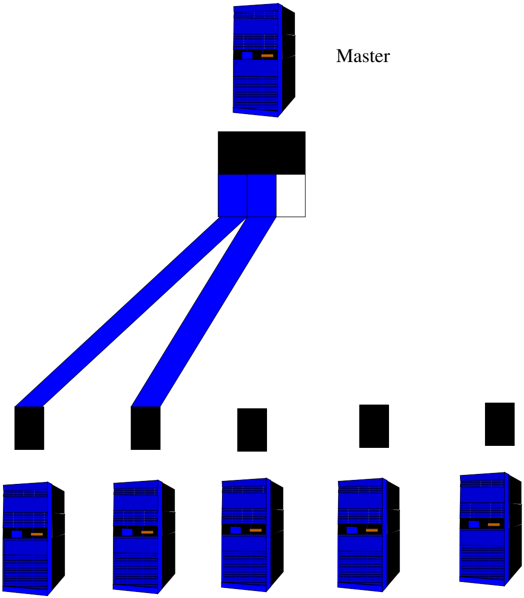
Communication model



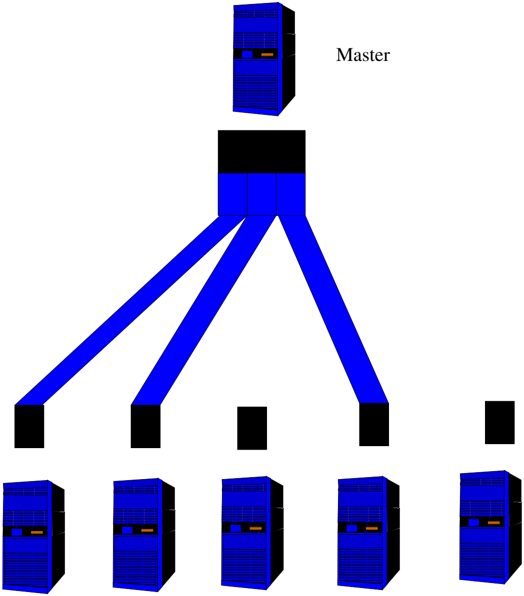
Communication model



Communication model



Communication model



Resource availability

Three possible processor states: *UP*, *RECLAIMED*, *DOWN*

- Preemption delays current operations
- Failure ⇒ current communications and computations are lost
- Program and data need be received again

On-line study

Availability modeled by (independent) 3-state Markov chains

Off-line study

For each processor P_u, state array T_u:

- $T_u[t] = -1$: processor *DOWN* at time slot t
- $T_u[t] = 0$: processor *RECLAIMED* at time slot t
- $T_u[t] = 1$: processor *UP* and available for comms and/or computing

Resource availability

Three possible processor states: *UP*, *RECLAIMED*, *DOWN*

- Preemption delays current operations
- Failure ⇒ current communications and computations are lost
- Program and data need be received again

On-line study

Availability modeled by (independent) 3-state Markov chains

Off-line study

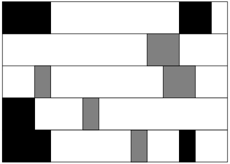
For each processor P_u, state array T_u:

- $T_u[t] = -1$: processor *DOWN* at time slot t
- $T_u[t] = 0$: processor *RECLAIMED* at time slot t
- $T_u[t] = 1$: processor *UP* and available for comms and/or computing

Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2,\ T_{data} = 1$

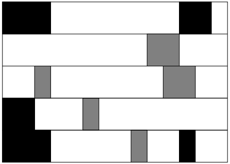


State array

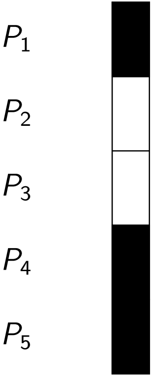
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



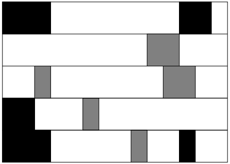
State array



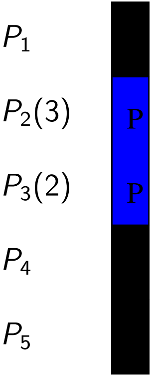
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



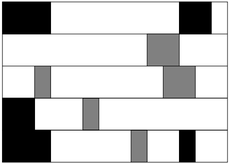
State array



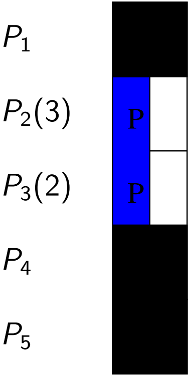
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



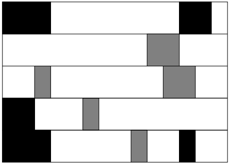
State array



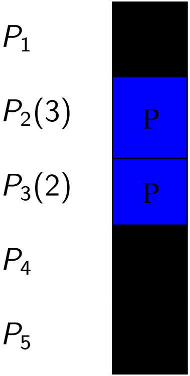
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2, \; T_{data} = 1$



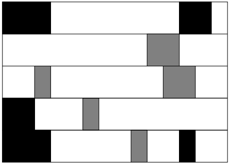
State array



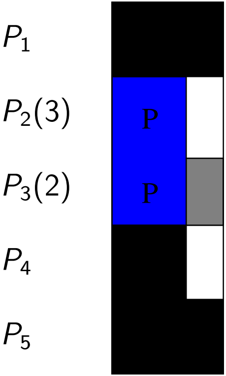
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2,\ T_{data} = 1$



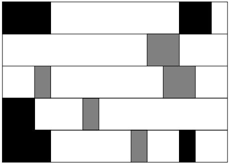
State array



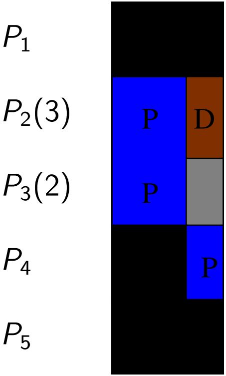
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



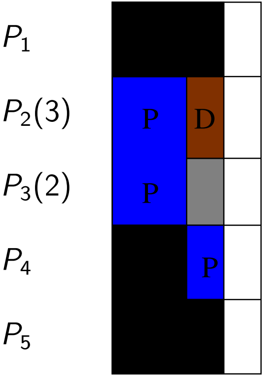
State array

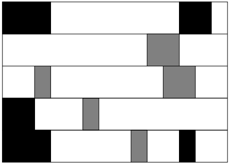


Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



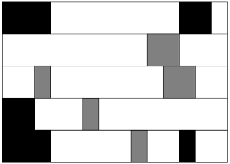


State array

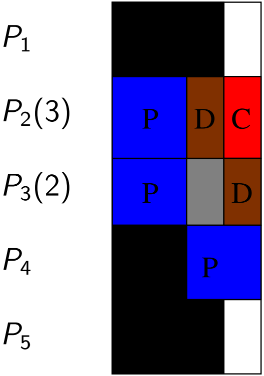
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



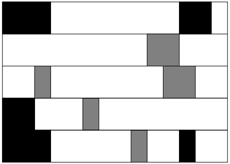
State array



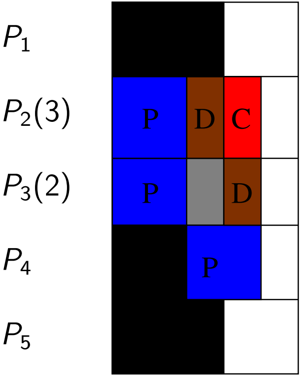
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



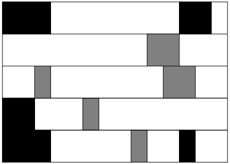
State array



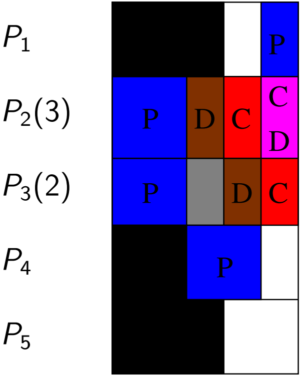
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2,\ T_{data} = 1$



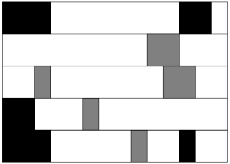
State array



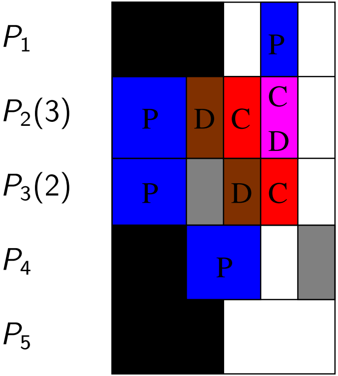
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



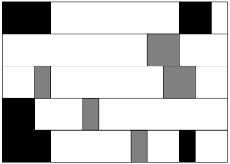
State array

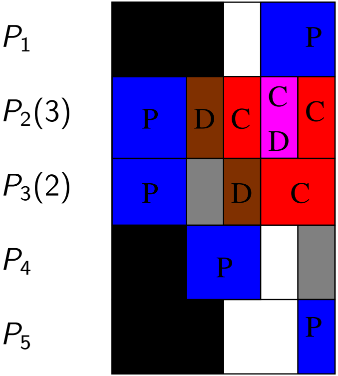


Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$

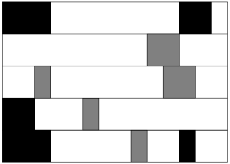




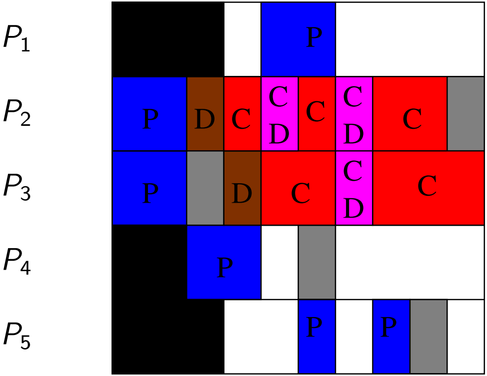
Example: INDEPENDENT scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



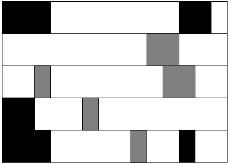
State array



Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$

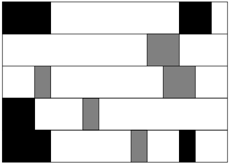


State array

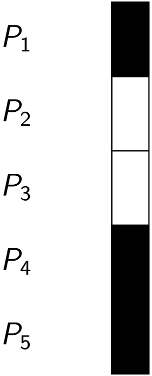
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



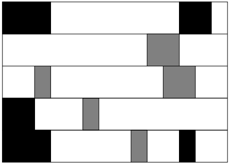
State array



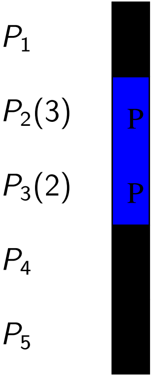
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



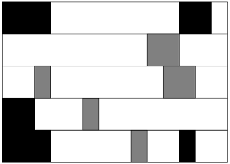
State array



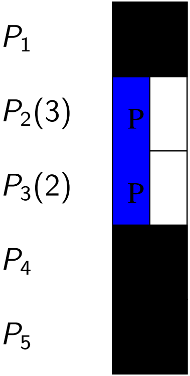
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



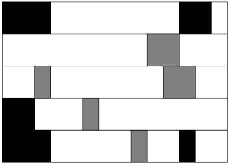
State array



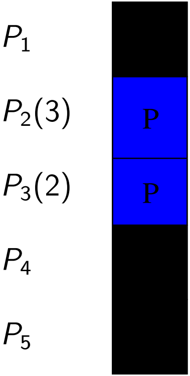
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



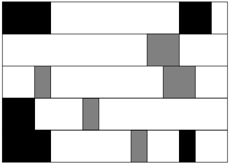
State array



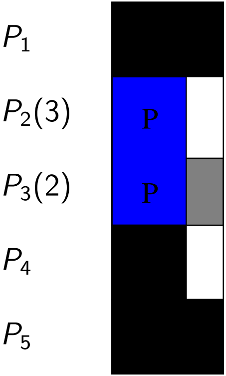
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



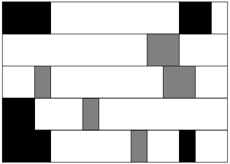
State array



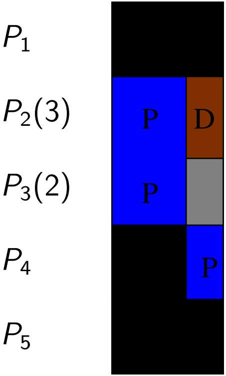
Example: Tightly-Coupled scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



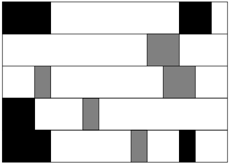
State array



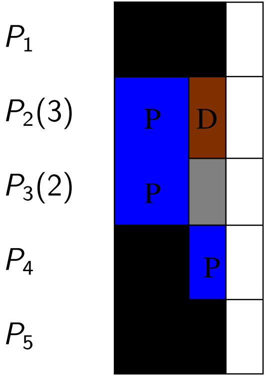
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



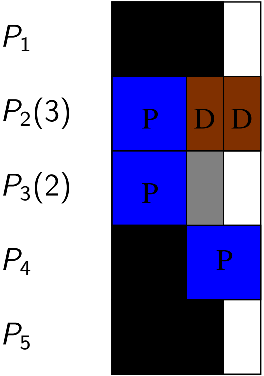
State array

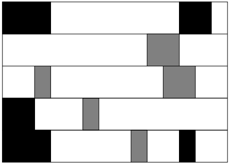


Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



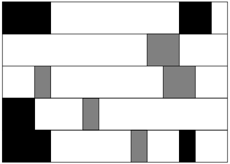


State array

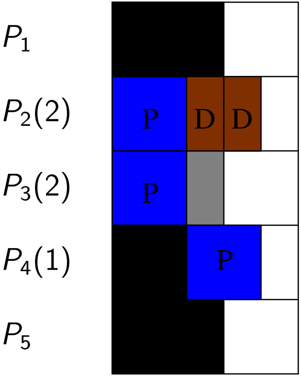
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



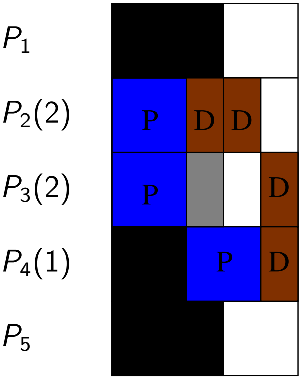
State array

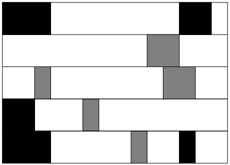


Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



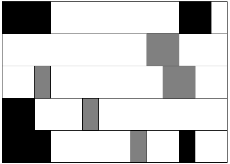


State array

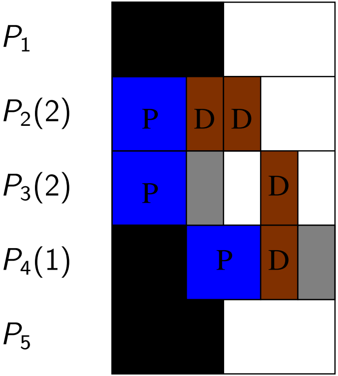
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



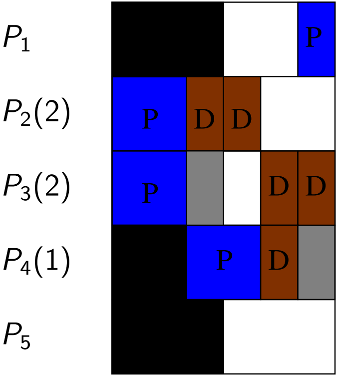
State array

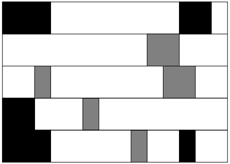


Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



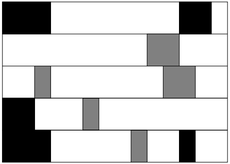


State array

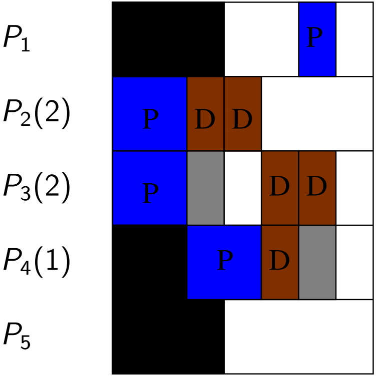
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



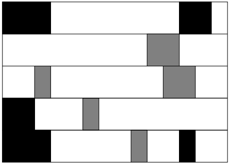
State array



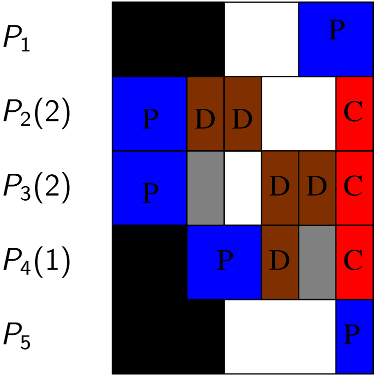
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



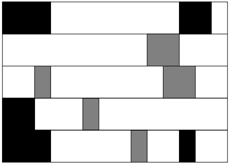
State array



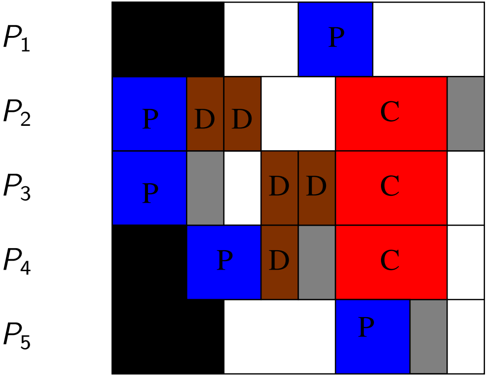
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



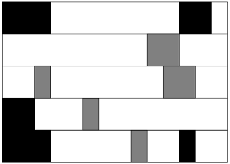
State array



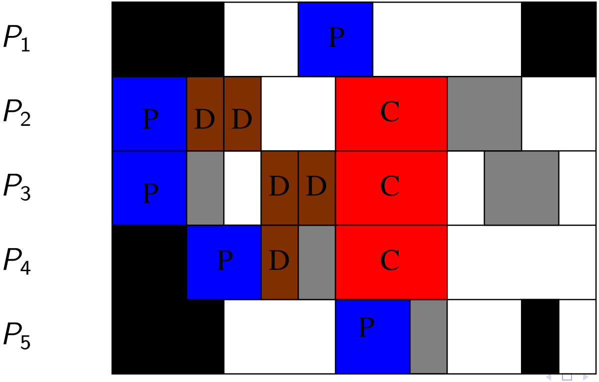
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



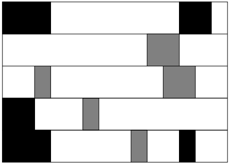
State array



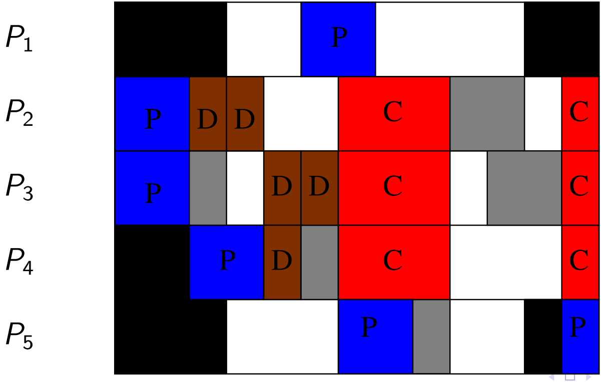
Example: TIGHTLY-COUPLED scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



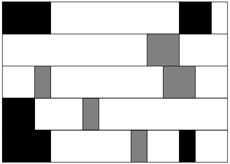
State array



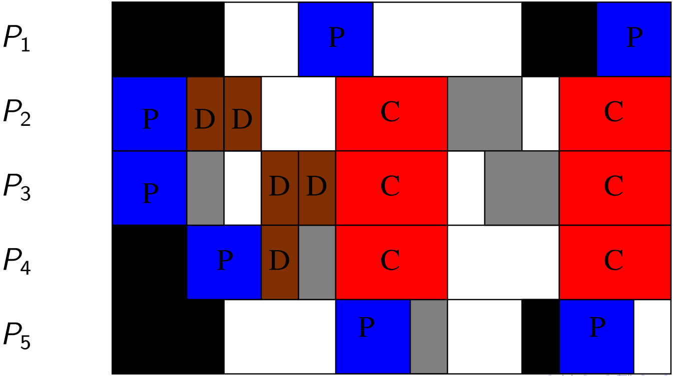
Example: Tightly-Coupled scenario

Instance with $m = 5$ tasks

- $p = 5$ processors, $w_i = i$
- $n_{com} = 2$
- $T_{prog} = 2$, $T_{data} = 1$



State array



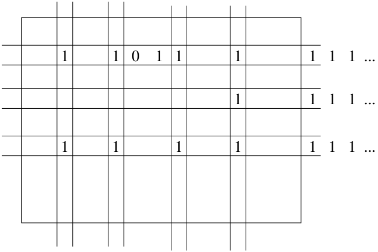
NP-completeness of TIGHTLY-COUPLED (1/2)

Theorem

TIGHTLY-COUPLED *is NP-complete even with uniform resources and no communications* ($T_{prog} = Tdata = 0$)

Reduction from ENCD (Exact Node Cardinality Decision problem): Given a bipartite graph $G = (U \cup V, E)$ and two integers a and b, does there exist a **bi-clique** with exactly a nodes in U and b nodes in V?

NP-completeness of TIGHTLY-COUPLED (2/2)



$p = |U|$ processors, $N = 2|V| + 1$ time slots
$T_i[j] = 1 \iff (u_i, v_j) \in E$ or $j \geq |V| + 1$, otherwise $T_i[j] = 0$
$m = a$ tasks of cost $W = b + |V| + 1$

Not enough time to compute two tasks on same processor
Need a processors available during **the same** W time slots
$\Rightarrow$ need b time slots of computation before time slot $|V|$

NP-completeness of INDEPENDENT (1/2)

Theorem

INDEPENDENT *is NP-complete even with uniform resources*

Reduction from 3SAT:
Given a set $U = \{x_1, ..., x_n\}$ of variables and a collection $\{C_1, ..., C_m\}$ of clauses, does there exist a truth assignment for U?

NP-completeness of INDEPENDENT (2/2)

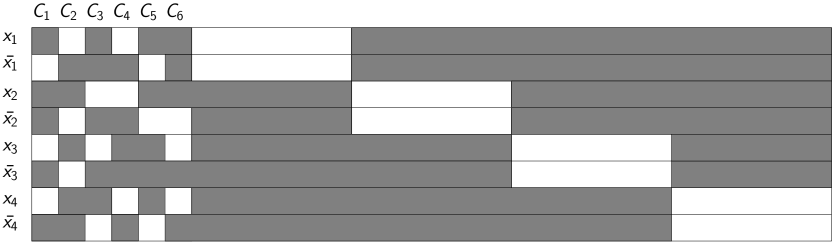
$(\bar{x_1} \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (\bar{x_2} \vee x_3 \vee x_4)$

m tasks, $2n$ procs, $n_{com} = 1$, $T_{prog} = m$, $T_{data} = 0$, $W = 1$
Two processors x and $\bar{x}$ per variable x

Processors x and $\bar{x}$ cannot both compute tasks
$\#$ executed tasks $= \#$ computation time slots $t \leq m$
m tasks executed $\Rightarrow$ enrolled processors validate 3SAT instance

NP-completeness of INDEPENDENT (2/2)

$(\bar{x}_1 \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge$
$(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee x_3 \vee x_4)$

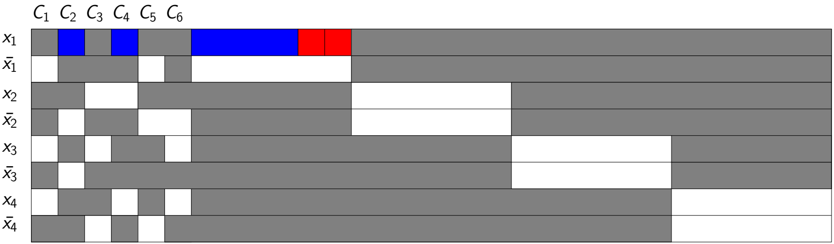


m tasks, $2n$ procs, $n_{com} = 1$, $T_{prog} = m$, $T_{data} = 0$, $W = 1$
Two processors x and $\bar{x}$ per variable x

Processors x and $\bar{x}$ cannot both compute tasks
$\#$ executed tasks $= \#$ computation time slots $t \leq m$
m tasks executed $\Rightarrow$ enrolled processors validate 3SAT instance

NP-completeness of INDEPENDENT (2/2)

$(\bar{x_1} \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (\bar{x_2} \vee x_3 \vee x_4)$

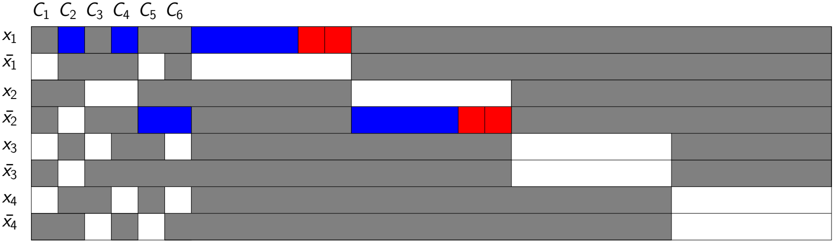


m tasks, $2n$ procs, $n_{com} = 1$, $T_{prog} = m$, $T_{data} = 0$, $W = 1$
Two processors x and $\bar{x}$ per variable x

Processors x and $\bar{x}$ cannot both compute tasks
\# executed tasks = \# computation time slots $t \leq m$
m tasks executed $\Rightarrow$ enrolled processors validate 3SAT instance

NP-completeness of INDEPENDENT (2/2)

$(\bar{x_1} \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (\bar{x_2} \vee x_3 \vee x_4)$

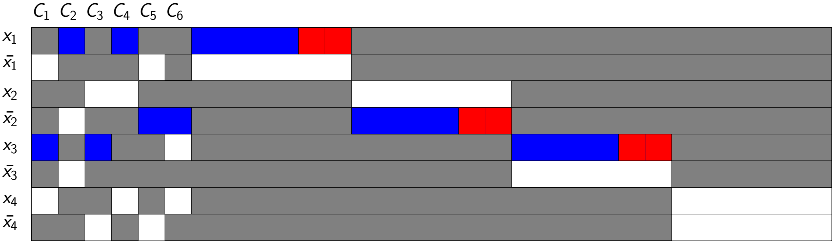


m tasks, $2n$ procs, $n_{com} = 1$, $T_{prog} = m$, $T_{data} = 0$, $W = 1$
Two processors x and $\bar{x}$ per variable x

Processors x and $\bar{x}$ cannot both compute tasks
$\#$ executed tasks $= \#$ computation time slots $t \leq m$
m tasks executed $\Rightarrow$ enrolled processors validate 3SAT instance

NP-completeness of INDEPENDENT (2/2)

$(\bar{x_1} \vee x_3 \vee x_4) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge (x_1 \vee x_2 \vee x_4) \wedge$
$(\bar{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (\bar{x_2} \vee x_3 \vee x_4)$



m tasks, $2n$ procs, $n_{com} = 1$, $T_{prog} = m$, $T_{data} = 0$, $W = 1$
Two processors x and $\bar{x}$ per variable x

Processors x and $\bar{x}$ cannot both compute tasks
\# executed tasks = \# computation time slots $t \leq m$
m tasks executed $\Rightarrow$ enrolled processors validate 3SAT instance

Markov chain

Transition matrix for processor P_i:

$$\begin{vmatrix} P_{u,u}(i) & P_{r,u}(i) & P_{d,u}(i) \\ P_{u,r}(i) & P_{r,r}(i) & P_{d,r}(i) \\ P_{u,d}(i) & P_{r,d}(i) & P_{d,d}(i) \end{vmatrix}$$

Probabilities:

- $\pi_u^{(i)}$ for P_i being *UP*
- $\pi_r^{(i)}$ for P_i being *RECLAIMED*
- $\pi_d^{(i)}$ for P_i being *DOWN*

Probability of success of a computation in INDEPENDENT

Probability of another time slot of computation:

$$
\begin{aligned}
P_{+}^{(q)} &= P_{u,u}^{(q)} + \sum_t P_{u,r} P_{r,r}^{t} P_{r,u} \\
&= P_{u,u}^{(q)} + \frac{P_{u,r}^{(q)} P_{r,u}^{(q)}}{1 - P_{r,r}^{(q)}}
\end{aligned}
$$

Probability of success of a computation of W time-slots:

$$P_{+}^{(q)}(W) = (P_{+}^{(q)})^{W-1}$$

Probability of success of a computation in INDEPENDENT

Probability of another time slot of computation:

$$P_+^{(q)} = P_{u,u}^{(q)} + \sum_t P_{u,r} P_{r,r}^t P_{r,u}$$
$$= P_{u,u}^{(q)} + \frac{P_{u,r}^{(q)} P_{r,u}^{(q)}}{1 - P_{r,r}^{(q)}}$$

Probability of success of a computation of W time-slots:

$$P_+^{(q)}(W) = (P_+^{(q)})^{W-1}$$

Probability of success of a computation in INDEPENDENT

Probability of another time slot of computation:

$$\begin{aligned} P_+^{(q)} &= P_{u,u}^{(q)} + \sum_t P_{u,r} P_{r,r}^t P_{r,u} \\ &= P_{u,u}^{(q)} + \frac{P_{u,r}^{(q)} P_{r,u}^{(q)}}{1 - P_{r,r}^{(q)}} \end{aligned}$$

Probability of success of a computation of W time-slots:

$$P_+^{(q)}(W) = (P_+^{(q)})^{W-1}$$

Probability of success of a computation in INDEPENDENT

Probability of another time slot of computation:

$$P_+^{(q)} = P_{u,u}^{(q)} + \sum_t P_{u,r} P_{r,r}^t P_{r,u}$$
$$= P_{u,u}^{(q)} + \frac{P_{u,r}^{(q)} P_{r,u}^{(q)}}{1 - P_{r,r}^{(q)}}$$

Probability of success of a computation of W time-slots:

$$P_+^{(q)}(W) = (P_+^{(q)})^{W-1}$$

Expected completion time in INDEPENDENT

Expected time of the next time slot of computation:

$$E^{(q)}(up) = 1 + \frac{P_{u,r}^{(q)} P_{r,u}^{(q)}}{1 - P_{r,r}^{(q)}} \times \frac{1}{P_{u,u}^{(q)}(1 - P_{r,r}^{(q)}) + P_{u,r}^{(q)} P_{r,u}^{(q)}}$$

Expected time of the completion of a computation of W time-slots:

$$E^{(q)}(W) = 1 + (W - 1)E^{(q)}(up)$$

Probability of success in TIGHTLY-COUPLED

A set S of processors with W time slots of workload
All processors of S are UP at time 0

Probability that all processors of S will be UP at time t:

$$P^{(S)}(t)$$

Let $E_u(S) = \sum_{t>0} P^{(S)}(t)$

Probability of another time slot of computation:

$$P_+^{(S)} = \frac{E_u(S)}{1 + E_u(S)}$$

Probability of success in TIGHTLY-COUPLED

A set S of processors with W time slots of workload
All processors of S are UP at time 0

Probability that all processors of S will be UP at time t:

$$P^{(S)}(t)$$

Let $E_u(S) = \sum_{t>0} P^{(S)}(t)$

Probability of another time slot of computation:

$$P_+^{(S)} = \frac{E_u(S)}{1 + E_u(S)}$$

Probability of success in TIGHTLY-COUPLED

A set S of processors with W time slots of workload
All processors of S are UP at time 0

Probability that all processors of S will be UP at time t:

$$P^{(S)}(t)$$

Let $E_u(S) = \sum_{t>0} P^{(S)}(t)$

Probability of another time slot of computation:

$$P_+^{(S)} = \frac{E_u(S)}{1 + E_u(S)}$$

Expected computation time in TIGHTLY-COUPLED

Let $E_u(S) = \sum_{t>0} P^{(S)}(t)$
Let $A(S) = \sum_{t>0} t \times P^{(S)}(t)$

Expected time of the next time slot of computation:

$$E_c^{(S)} = \frac{A(S)\left(1 - P_+^{(S)}\right)}{1 + E_u(S)}$$

General description of heuristics

Three classes of heuristics:

- Passive: the configuration may change only when one of the hosts in it goes to the DOWN state
- Dynamic: the configuration may change if a "better" processor becomes *UP*, but no ongoing communication/computation is terminated
- Proactive: like dynamic, but aggressive termination of ongoing communication/computation

General description of heuristics

Three classes of heuristics:

- Passive: the configuration may change only when one of the hosts in it goes to the DOWN state
- Dynamic: the configuration may change if a "better" processor becomes *UP*, but no ongoing communication/computation is terminated
- Proactive: like dynamic, but aggressive termination of ongoing communication/computation

General description of heuristics

Three classes of heuristics:

- Passive: the configuration may change only when one of the hosts in it goes to the DOWN state
- Dynamic: the configuration may change if a "better" processor becomes *UP*, but no ongoing communication/computation is terminated
- Proactive: like dynamic, but aggressive termination of ongoing communication/computation

Proactive criteria for TIGHTLY-COUPLED

After t time slots on a same iteration:

- Success probability: P
- Expected completion time: E
- Expected yield: $\frac{P}{E+t}$
- Apparent yield: $\frac{P}{E}$

Random heuristics

RANDOM: randomly select processor for next task

Weighted random:

- RANDOM1: weight $P_{u,u}^{(q)}$ for P_q
- RANDOM2: weight $P_{+}^{(q)}$ for P_q
- RANDOM3: weight $\pi_u^{(q)}$ for P_q
- RANDOM4: weight $1 - \pi_d^{(q)}$ for P_q

variants RANDOMXW: weight divided by w_q

Greedy heuristics for INDEPENDENT

- MCT: Minimum completion time

$$CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0) \max(T_{\text{data}}, w_q) + w_q$$

- EMCT: Expected MCT
- LW: Likely to work

$$\left(P_+^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down
$P_{UD}^{(q)}(k)$ probability to not fail *DOWN* during k time slots

$$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics for INDEPENDENT

- MCT: Minimum completion time

$$CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0)\max(T_{\text{data}}, w_q) + w_q$$

- EMCT: Expected MCT
- LW: Likely to work

$$\left(P_{+}^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down
$P_{UD}^{(q)}(k)$ probability to not fail *DOWN* during k time slots

$$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics for INDEPENDENT

- MCT: Minimum completion time

$$CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0)\max(T_{\text{data}}, w_q) + w_q$$

- EMCT: Expected MCT
- LW: Likely to work

$$\left(P_+^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down

$P_{UD}^{(q)}(k)$ probability to not fail *DOWN* during k time slots

$$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics for INDEPENDENT

- MCT: Minimum completion time

  $CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0) \max(T_{\text{data}}, w_q) + w_q$

- EMCT: Expected MCT
- LW: Likely to work

$$\left(P_{+}^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down

  $P_{UD}^{(q)}(k)$ probability to not fail *DOWN* during k time slots

$$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics for INDEPENDENT

- MCT: Minimum completion time

  $$CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0)\max(T_{\text{data}}, w_q) + w_q$$

- EMCT: Expected MCT
- LW: Likely to work

  $$\left(P_{+}^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down
  $P_{UD}^{(q)}(k)$ probability to not fail *DOWN* during k time slots

  $$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics for Tightly-Coupled

First possibility: Compute workload independently on each processor

- MCT: Minimum completion time

$$CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0) \max(T_{\text{data}}, w_q) + w_q$$

- EMCT: Expected MCT
- LW: Likely to work

$$\left(P_+^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down
  $P_{UD}^{(q)}(k)$ probability to not fail down during k time slots

$$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics for TIGHTLY-COUPLED

First possibility: Compute workload independently on each processor

- MCT: Minimum completion time

  $CT(P_q, n_q) = \text{Delay}(q) + T_{\text{data}} + \max(n_q - 1, 0)\max(T_{\text{data}}, w_q) + w_q$

- EMCT: Expected MCT
- LW: Likely to work

$$\left(P_+^{(q)}\right)^{CT(P_q, n_q+1)}$$

- UD: Unlikely Down
  $P_{UD}^{(q)}(k)$ probability to not fail down during k time slots

$$\left(P_{UD}^{(q)}\right)^{(E^{(q)}(CT(P_q, n_q+1)))}$$

variant *: $T_{\text{data}} \rightarrow \left\lceil \frac{n_{active}}{n_{com}} \right\rceil T_{\text{data}}$

Greedy heuristics designed for TIGHTLY-COUPLED

- IP: Incremental: Probability of success

$$q_0 = \text{ArgMax}\left\{P^S(q)\right\}$$

- IE: Incremental: Expected completion time

$$q_0 = \text{ArgMin}\left\{T^q_{comm} + T^q_{comp}\right\}$$

- IY: Incremental: Expected yield

$$q_0 = \text{ArgMax}\left\{\frac{P^S(q)}{t + T^S(q)}\right\}$$

- IAY: Incremental: Expected apparent yield

$$q_0 = \text{ArgMax}\left\{\frac{P^S(q)}{T^S(q)}\right\}$$

Instances for INDEPENDENT

Parameter values for Markov simulations

parameter	values
p	20
m	5, 10, 20, 40
n_{com}	5, 10, 20
w_{min}	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- $0.9 \leq P_{x,x} \leq 0.99$
- $P_{x,y} = \frac{1}{2}(1 - P_{x,x})$
- $w_{min} \leq w_q \leq 10 * w_{min}$
- $T_{data} = w_{min}$ and $T_{prog} = 5 * w_{min}$

Comparison of average dfb (degradation from best, percentage)

Instances for Tightly-Coupled

Parameter values for Markov simulations

parameter	values
p	20
m	5, 10
n_{com}	5, 10, 20
w_{min}	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- $0.9 \leq P_{x,x} \leq 0.99$
- $P_{x,y} = \frac{1}{2}(1 - P_{x,x})$
- $w_{min} \leq w_q \leq 10 * w_{min}$
- $T_{data} = w_{min}$ and $T_{prog} = 5 * w_{min}$

Comparison of average dfb (degradation from best, percentage)

Instances for TIGHTLY-COUPLED

Parameter values for Markov simulations

parameter	values
p	20
m	5, 10
n_{com}	5, 10, 20
w_{min}	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

- $0.9 \leq P_{x,x} \leq 0.99$
- $P_{x,y} = \frac{1}{2}(1 - P_{x,x})$
- $w_{min} \leq w_q \leq 10 * w_{min}$
- $T_{data} = w_{min}$ and $T_{prog} = 5 * w_{min}$

Comparison of average dfb (degradation from best, percentage)

Results over all problem instances for INDEPENDENT

Algorithm	Average *dfb*	#wins
EMCT	4.77	80320
EMCT*	4.81	78947
MCT	5.35	73946
MCT*	5.46	70952
UD*	7.06	42578
UD	8.09	31120
LW*	11.15	28802
LW	12.74	19529
RANDOM1W	28.42	259
RANDOM2W	28.43	301
RANDOM4W	28.51	278
RANDOM3W	31.49	188
RANDOM3	44.01	87
RANDOM4	47.33	88
RANDOM1	47.44	36
RANDOM2	47.53	73
RANDOM	47.87	45

Results over all problem instances for INDEPENDENT

Algorithm	Average *dfb*	#wins
EMCT	4.77	80320
EMCT*	4.81	78947
MCT	5.35	73946
MCT*	5.46	70952
UD*	7.06	42578
UD	8.09	31120
LW*	11.15	28802
LW	12.74	19529
RANDOM1W	28.42	259
RANDOM2W	28.43	301
RANDOM4W	28.51	278
RANDOM3W	31.49	188
RANDOM3	44.01	87
RANDOM4	47.33	88
RANDOM1	47.44	36
RANDOM2	47.53	73
RANDOM	47.87	45

Results over all problem instances for INDEPENDENT

Algorithm	Average *dfb*	#wins
EMCT	4.77	80320
EMCT*	4.81	78947
MCT	5.35	73946
MCT*	5.46	70952
UD*	7.06	42578
UD	8.09	31120
LW*	11.15	28802
LW	12.74	19529
RANDOM1W	28.42	259
RANDOM2W	28.43	301
RANDOM4W	28.51	278
RANDOM3W	31.49	188
RANDOM3	44.01	87
RANDOM4	47.33	88
RANDOM1	47.44	36
RANDOM2	47.53	73
RANDOM	47.87	45

Results over all problem instances for INDEPENDENT

Algorithm	Average *dfb*	#wins
EMCT	4.77	80320
EMCT*	4.81	78947
MCT	5.35	73946
MCT*	5.46	70952
UD*	7.06	42578
UD	8.09	31120
LW*	11.15	28802
LW	12.74	19529
RANDOM1W	28.42	259
RANDOM2W	28.43	301
RANDOM4W	28.51	278
RANDOM3W	31.49	188
RANDOM3	44.01	87
RANDOM4	47.33	88
RANDOM1	47.44	36
RANDOM2	47.53	73
RANDOM	47.87	45

Results with higher communication costs

Table: Results for contention-prone simulations

Communication times ×5

Algorithm	Average *dfb*
EMCT*	3.87
MCT*	4.10
UD*	5.23
EMCT	6.13
UD	6.42
MCT	7.70
LW*	8.76
LW	10.11

Communication times ×10

Algorithm	Average *dfb*
UD*	2.76
UD	3.20
EMCT*	3.66
LW*	4.02
MCT*	4.22
LW	4.46
EMCT	8.02
MCT	15.50

Results with higher communication costs

Table: Results for contention-prone simulations

Communication times ×5

Algorithm	Average *dfb*
EMCT*	3.87
MCT*	4.10
UD*	5.23
EMCT	6.13
UD	6.42
MCT	7.70
LW*	8.76
LW	10.11

Communication times ×10

Algorithm	Average *dfb*
UD*	2.76
UD	3.20
EMCT*	3.66
LW*	4.02
MCT*	4.22
LW	4.46
EMCT	8.02
MCT	15.50

Results with higher communication costs

Table: Results for contention-prone simulations

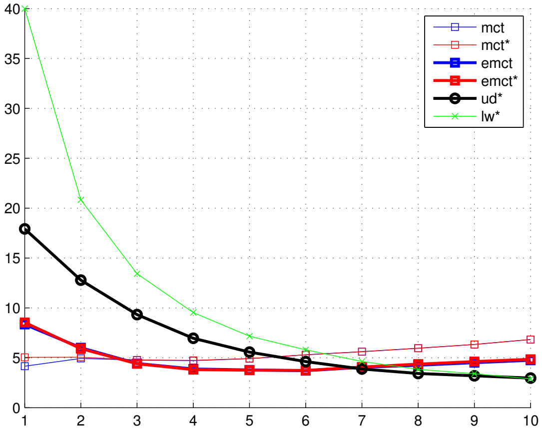
Communication times ×5

Algorithm	Average *dfb*
EMCT*	3.87
MCT*	4.10
UD*	5.23
EMCT	6.13
UD	6.42
MCT	7.70
LW*	8.76
LW	10.11

Communication times ×10

Algorithm	Average *dfb*
UD*	2.76
UD	3.20
EMCT*	3.66
LW*	4.02
MCT*	4.22
LW	4.46
EMCT	8.02
MCT	15.50

Influence of w_{min} for INDEPENDENT



Averaged dfb results vs. w_{min}

Results for best 10 heuristics for TIGHTLY-COUPLED

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
Y-IE	33.06	17.76	69.71	40.33
P-IE	34.48	16.66	68.02	41.34
E-IAY	35.26	24.83	71.37	55.80
E-IY	45.44	20.38	64.22	69.38
IE	51.40	10.81	69.71	59.83
IAY	59.32	8.46	70.12	93.12
IY	74.69	6.02	63.08	106.61
E-IP	77.08	15.41	49.51	103.19
E-EMCT*	92.23	8.63	43.14	164.71
E-LW*	92.80	12.65	44.65	123.30

General results for the best 10 heuristics

Results for best 10 heuristics for TIGHTLY-COUPLED

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
Y-IE	33.06	17.76	69.71	40.33
P-IE	34.48	16.66	68.02	41.34
E-IAY	35.26	24.83	71.37	55.80
E-IY	45.44	20.38	64.22	69.38
IE	51.40	10.81	69.71	59.83
IAY	59.32	8.46	70.12	93.12
IY	74.69	6.02	63.08	106.61
E-IP	77.08	15.41	49.51	103.19
E-EMCT*	92.23	8.63	43.14	164.71
E-LW*	92.80	12.65	44.65	123.30

General results for the best 10 heuristics

Results for best 10 heuristics for TIGHTLY-COUPLED

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
Y-IE	33.06	17.76	69.71	40.33
P-IE	34.48	16.66	68.02	41.34
E-IAY	35.26	24.83	71.37	55.80
E-IY	45.44	20.38	64.22	69.38
IE	51.40	10.81	69.71	59.83
IAY	59.32	8.46	70.12	93.12
IY	74.69	6.02	63.08	106.61
E-IP	77.08	15.41	49.51	103.19
E-EMCT*	92.23	8.63	43.14	164.71
E-LW*	92.80	12.65	44.65	123.30

General results for the best 10 heuristics

Results for best 10 heuristics with 5 tasks

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
Y-IE	32.30	17.69	70.44	40.21
P-IE	33.83	16.36	68.67	41.19
E-IAY	35.34	23.74	70.99	56.98
E-IY	44.10	20.48	64.86	67.89
IE	47.59	11.33	70.44	50.82
IAY	57.57	9.18	69.66	96.75
IY	71.02	6.63	63.67	108.21
E-IP	73.82	15.68	50.17	98.56
E-EMCT*	87.23	9.32	44.80	162.71
E-LW	90.68	12.82	45.07	119.19

Results with 5 tasks for the best 10 heuristics

Results for best 10 heuristics with 5 tasks

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
Y-IE	32.30	17.69	70.44	40.21
P-IE	33.83	16.36	68.67	41.19
E-IAY	35.34	23.74	70.99	56.98
E-IY	44.10	20.48	64.86	67.89
IE	47.59	11.33	70.44	50.82
IAY	57.57	9.18	69.66	96.75
IY	71.02	6.63	63.67	108.21
E-IP	73.82	15.68	50.17	98.56
E-EMCT*	87.23	9.32	44.80	162.71
E-LW	90.68	12.82	45.07	119.19

Results with 5 tasks for the best 10 heuristics

Results for best 10 heuristics with 10 tasks

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
E-IAY	34.83	31.20	73.60	48.23
Y-IE	37.48	18.20	65.40	40.98
P-IE	38.31	18.40	64.20	42.21
E-IY	53.32	19.80	60.40	77.59
IAY	69.62	4.20	72.80	67.90
IE	73.74	7.80	65.40	97.15
E-IP	96.20	13.80	45.60	127.04
IY	96.29	2.40	59.60	96.62
E-LW	105.30	11.60	42.20	145.12
E-EMCT*	121.64	4.60	33.40	175.99

Results with 10 tasks for the best 10 heuristics

Results for best 10 heuristics with 10 tasks

Algorithm	Average *dfb*	#wins	#good rate	*stdv*
E-IAY	34.83	31.20	73.60	48.23
Y-IE	37.48	18.20	65.40	40.98
P-IE	38.31	18.40	64.20	42.21
E-IY	53.32	19.80	60.40	77.59
IAY	69.62	4.20	72.80	67.90
IE	73.74	7.80	65.40	97.15
E-IP	96.20	13.80	45.60	127.04
IY	96.29	2.40	59.60	96.62
E-LW	105.30	11.60	42.20	145.12
E-EMCT*	121.64	4.60	33.40	175.99

Results with 10 tasks for the best 10 heuristics

Conclusion for this study

Theoretical results

- Realistic models
- Complexity results for off-line problems
- Probability computations

Simulation results

- Large set of efficient heuristics
- Simulations

Perspectives

- Real life traces are not Markovian (Weibull or Pareto distributions)
- The yield is not used in INDEPENDENT heuristics

Outline

1. Introduction
2. Scheduling filtering applications (overview)
3. Reliability of pipelined real-time systems (overview)
4. Scheduling on volatile ressources
5. Conclusion and perspectives

Conclusion

Application models:

- Filtering tasks
- Pipelined real-time systems
- Iterative applications

Failure models:

- Transient failures
- Desktop grids

Results:

- Complete sets of complexity results
- Some approximation results and ILP formulations
- Heuristics and simulations

Perspectives

- More approximation results
- Design more heuristics for some models
- Consider additional criteria (power consumption,...)
- Study other methods to increase reliability (checkpointing, migration)
- More realistic probability distribution for failures

Bibliography

Journal paper

Kunal Agrawal, Anne Benoit, Fanny Dufossé, and Yves Robert. Mapping filtering streaming applications. *Algorithmica*, 2010.

International conferences

Anne Benoit, Fanny Dufossé, and Yves Robert. Filter placement on a pipelined architecture. In *11th Workshop on Advances in Parallel and Distributed Computational Models APDCM 2009*. IEEE Computer Society Press, 2009.

Anne Benoit, Fanny Dufossé, and Yves Robert. On the complexity of mapping pipelined filtering services on heterogeneous platforms. In *IPDPS'2009, the 23rd IEEE International Parallel and Distributed Processing Symposium*. IEEE Computer Society Press, 2009.

Kunal Agrawal, Anne Benoit, Fanny Dufossé, and Yves Robert. Mapping filtering streaming applications with communication costs. In *21st ACM Symposium on Parallelism in Algorithms and Architectures SPAA 2009*. ACM Press, 2009.

Anne Benoit, Bruno Gaujal, Fanny Dufossé, Matthieu Gallet, and Yves Robert. Computing the throughput of probabilistic and replicated streaming applications. In *22nd ACM Symposium on Parallelism in Algorithms and Architectures SPAA 2010*. ACM Press, 2010.

Anne Benoit, Fanny Dufossé, Alain Girault, and Yves Robert. Reliability and performance optimization of pipelined real-time systems. In *International Conference on Parallel Processing*, page 20, 2010.

Henri Casanova, Fanny Dufossé, Yves Robert, and Frédéric Vivien. Scheduling parallel iterative applications on volatile resources. In *IPDPS'2011, the 25th IEEE International Parallel and Distributed Processing Symposium*. IEEE Computer Society Press, 2011.

Guillaume Aupy, Anne Benoit, Fanny Dufossé, and Yves Robert. Brief announcement: Reclaiming the energy of a schedule, models and algorithms. In *23rd ACM Symposium on Parallelism in Algorithms and Architectures SPAA 2011*. ACM Press, 2011.