Mapping and Scheduling Streaming Applications using SMT Solvers

Pranav Tendulkar

Supervisors:
Dr. Oded Maler Dr. Peter Poplavko

Verimag, FRANCE

13 October 2014

Multi-core Processors Everywhere

Tablets

Laptops

Space-shuttle

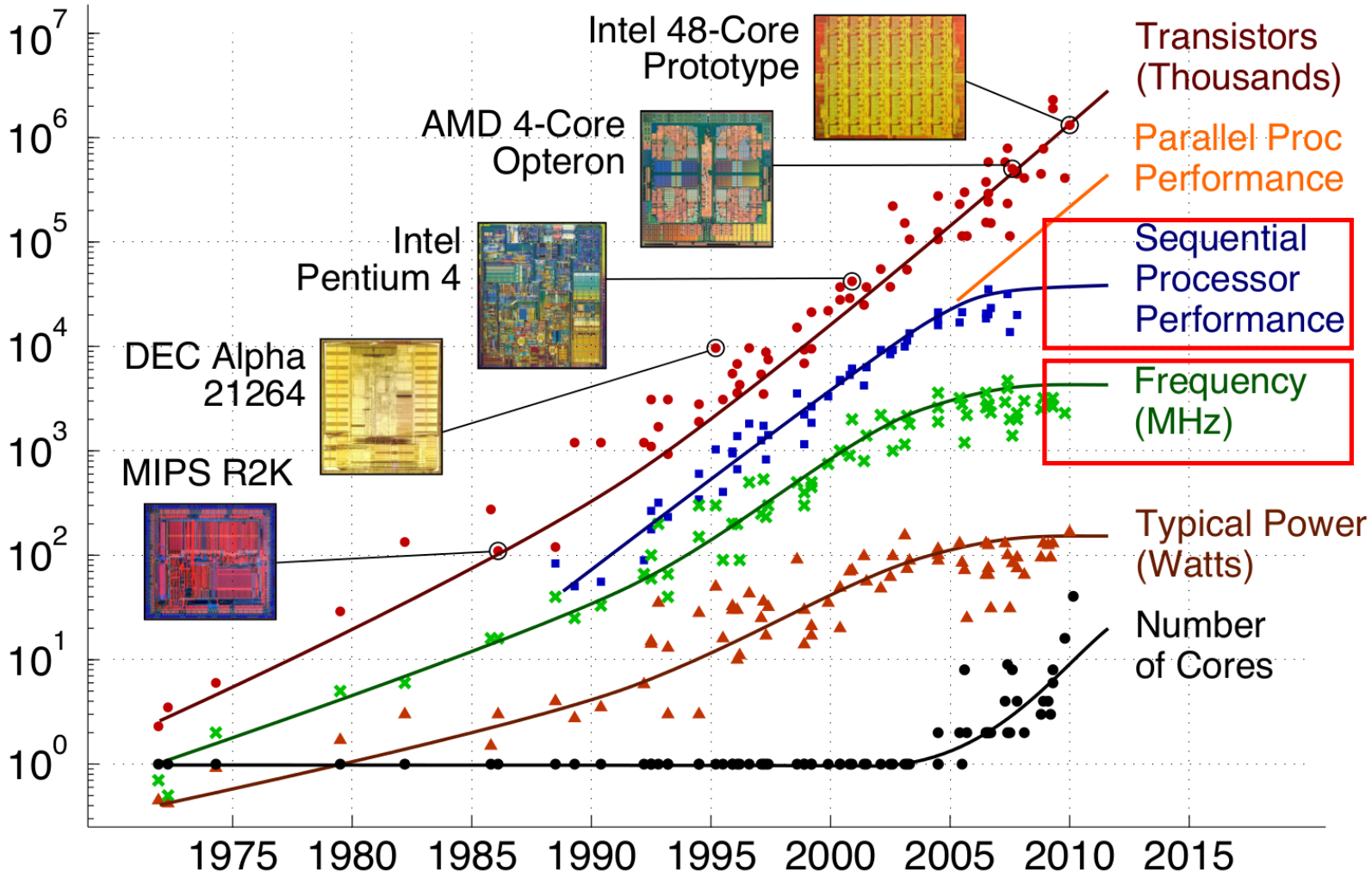
Phones

Cars

Smart-TV

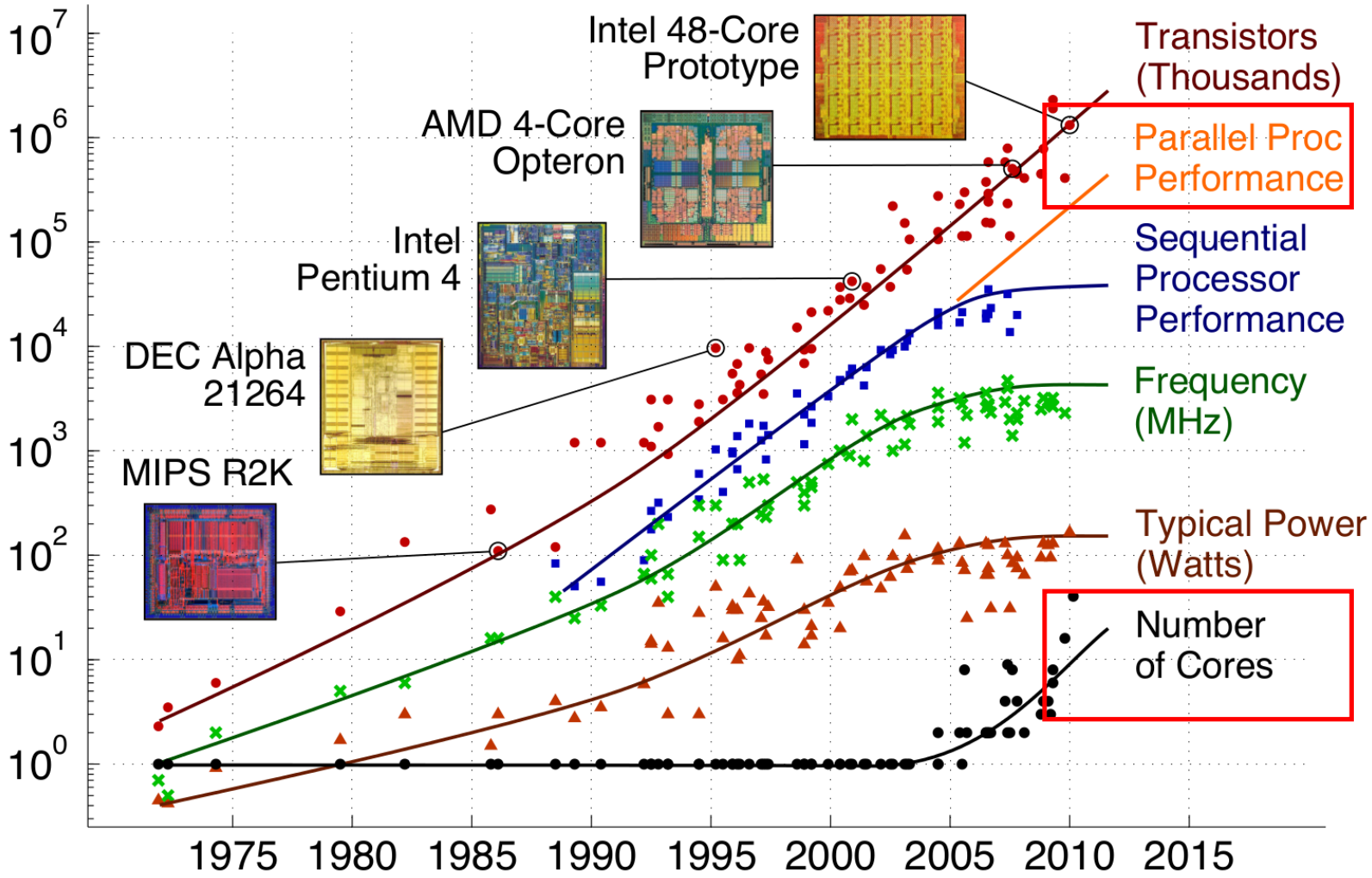
Cameras

Multi-core Processors Everywhere



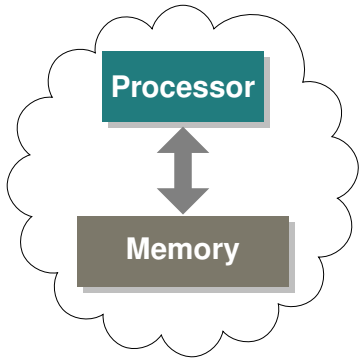
source : http://www.csl.cornell.edu/courses/ece5745/handouts.html

Multi-core Processors Everywhere

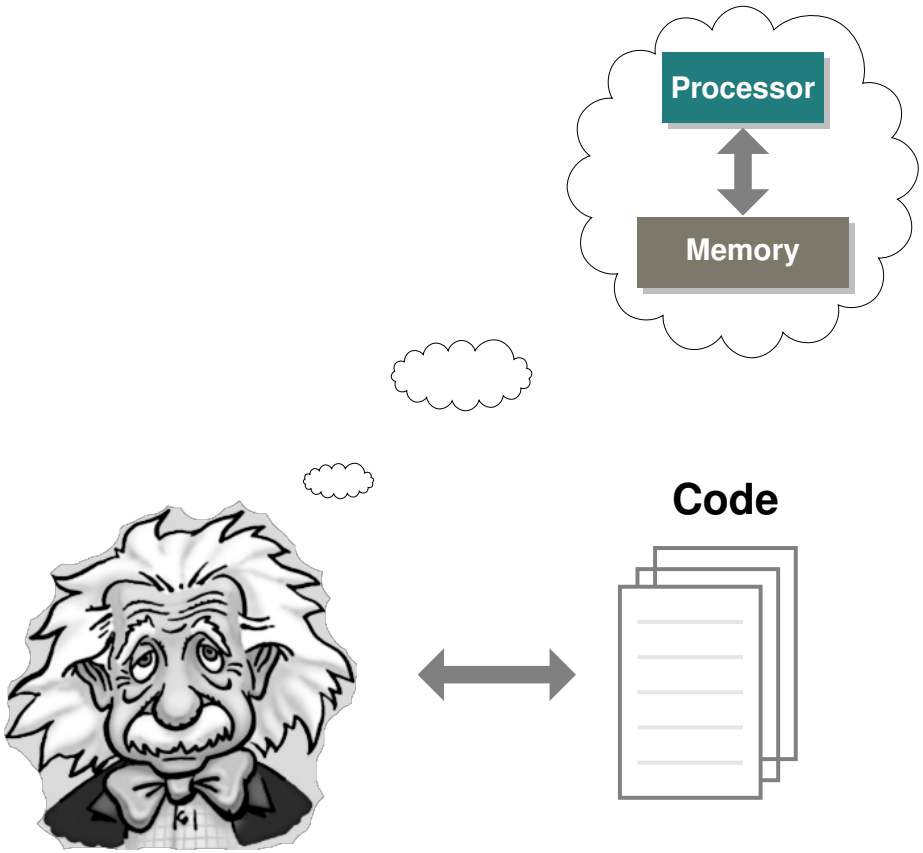


source : http://www.csl.cornell.edu/courses/ece5745/handouts.html

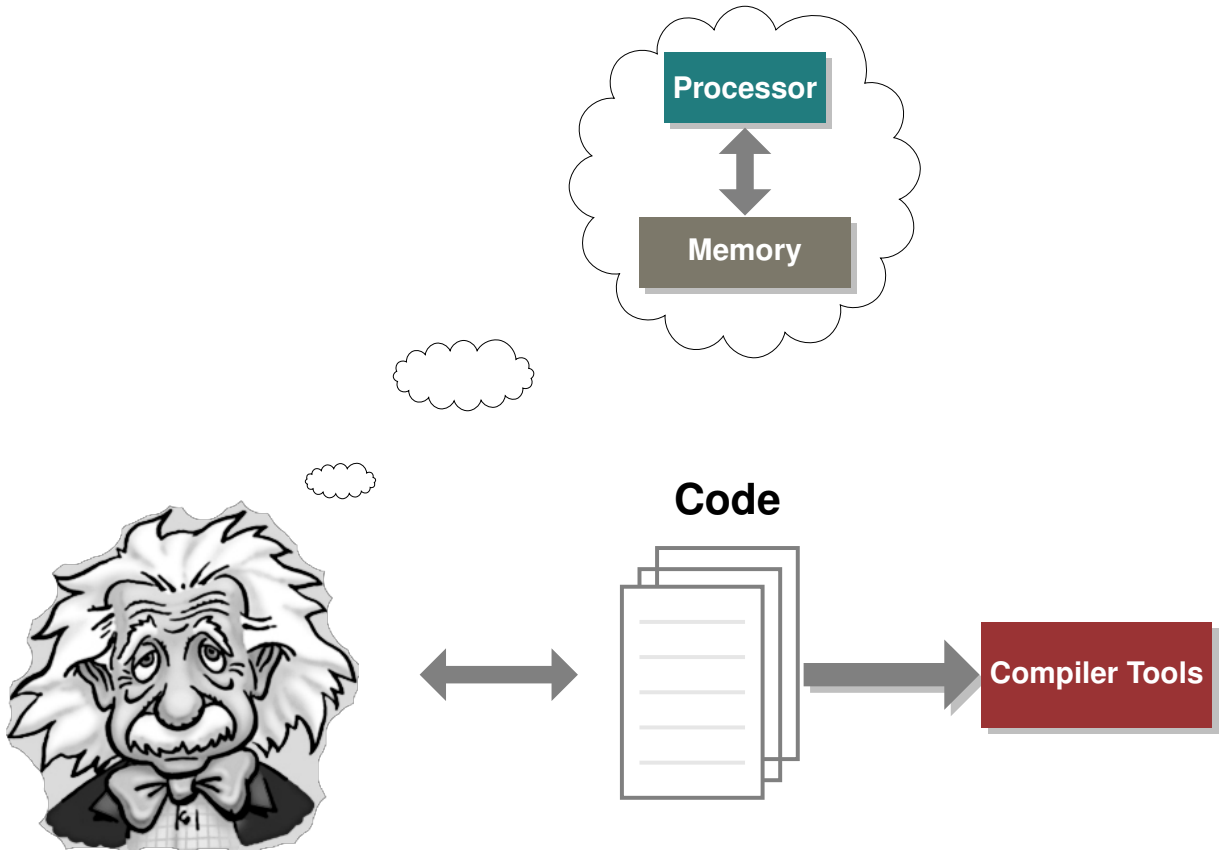
Software on Single Processor System



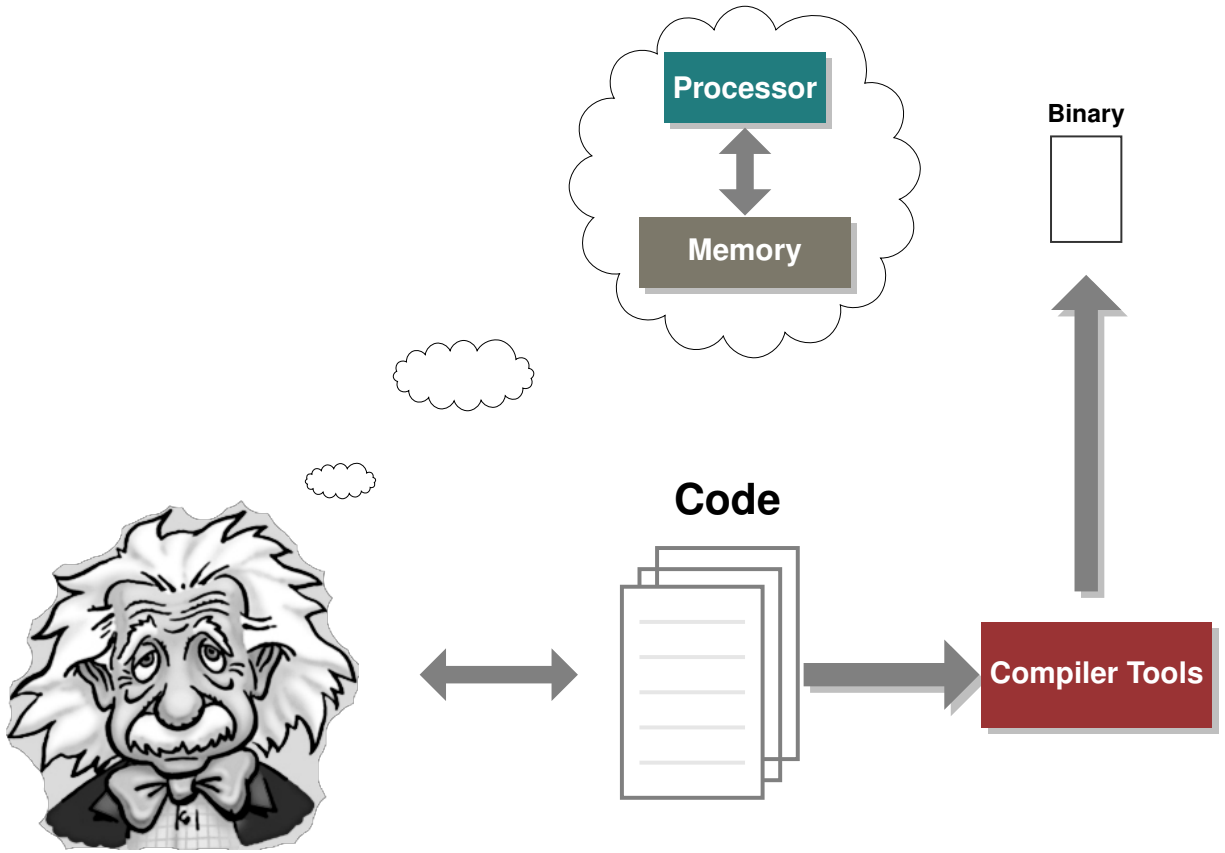
Software on Single Processor System



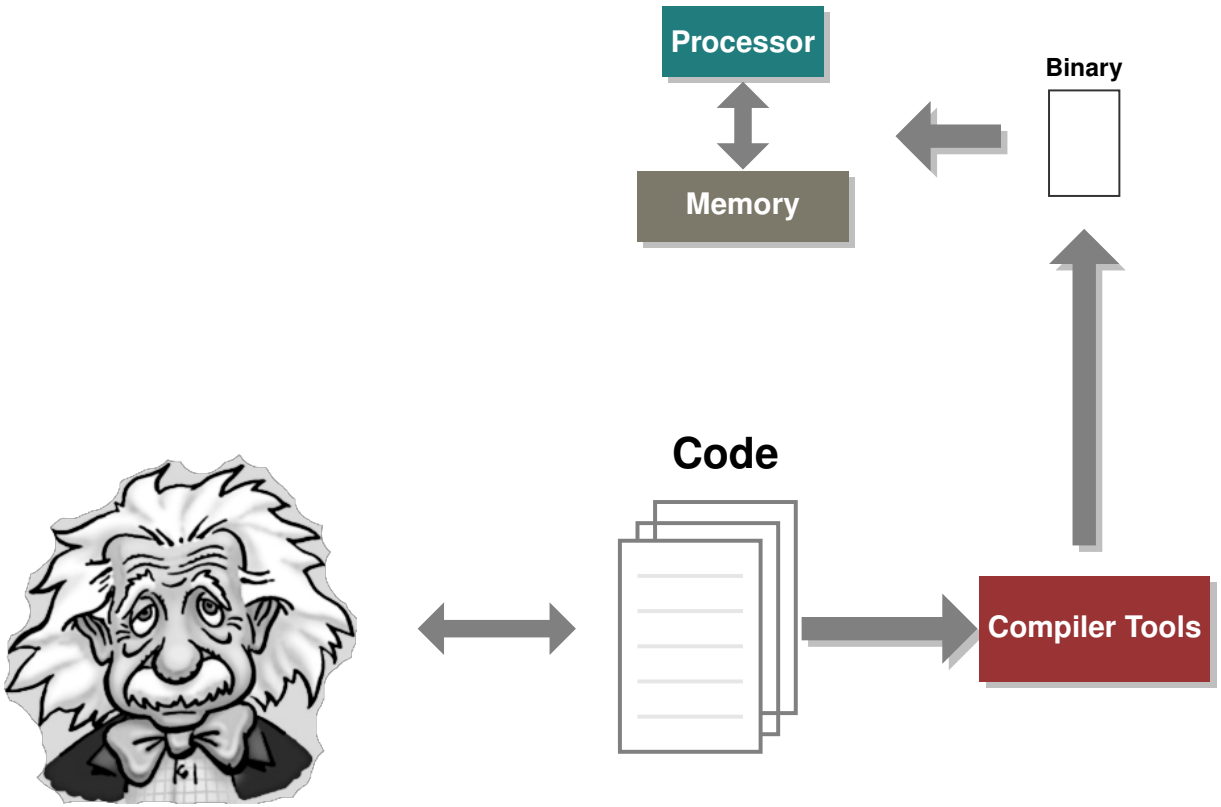
Software on Single Processor System



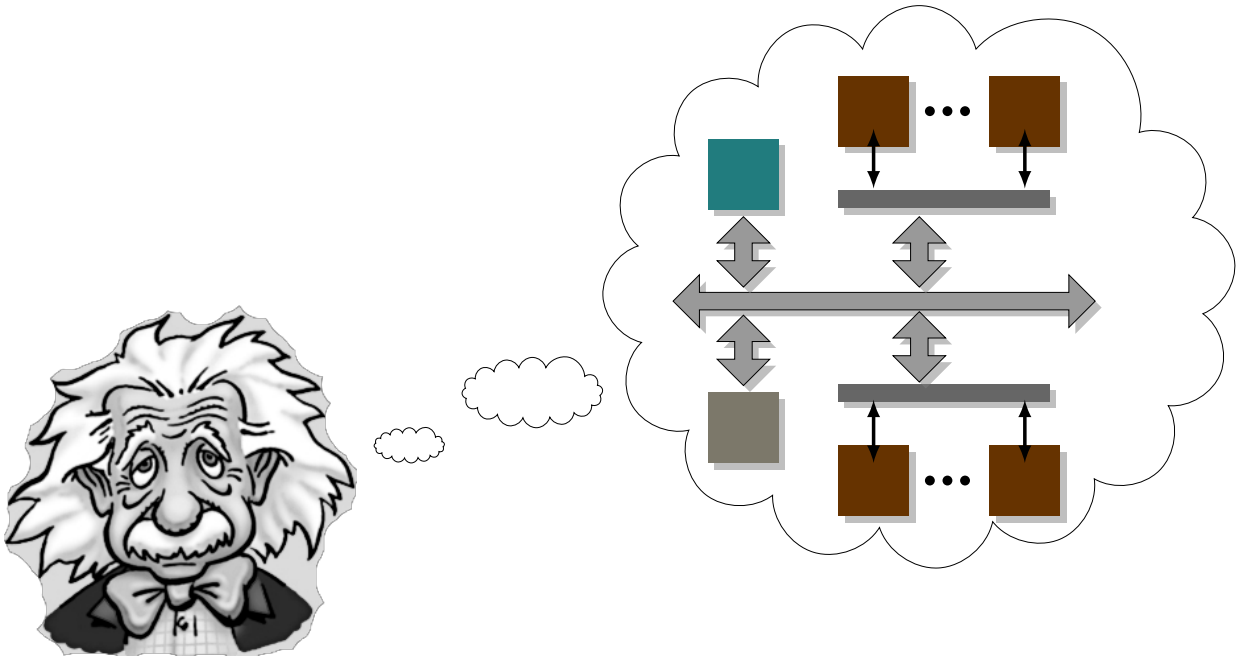
Software on Single Processor System



Software on Single Processor System

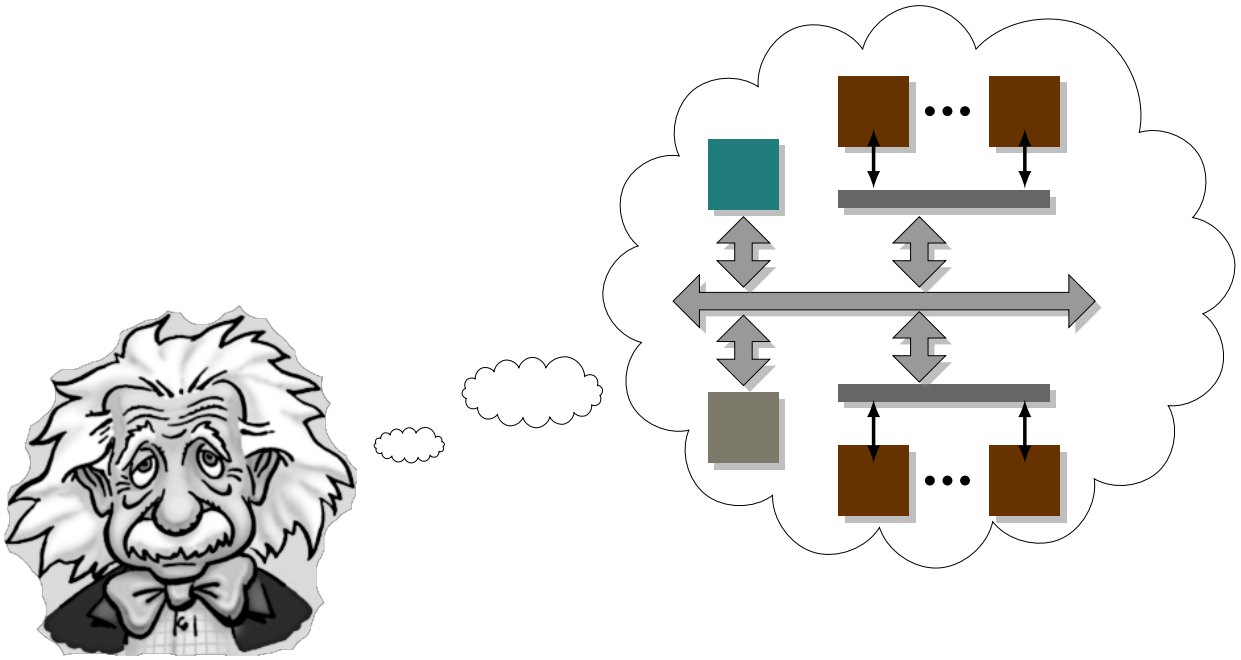


Multi-core systems



How To:

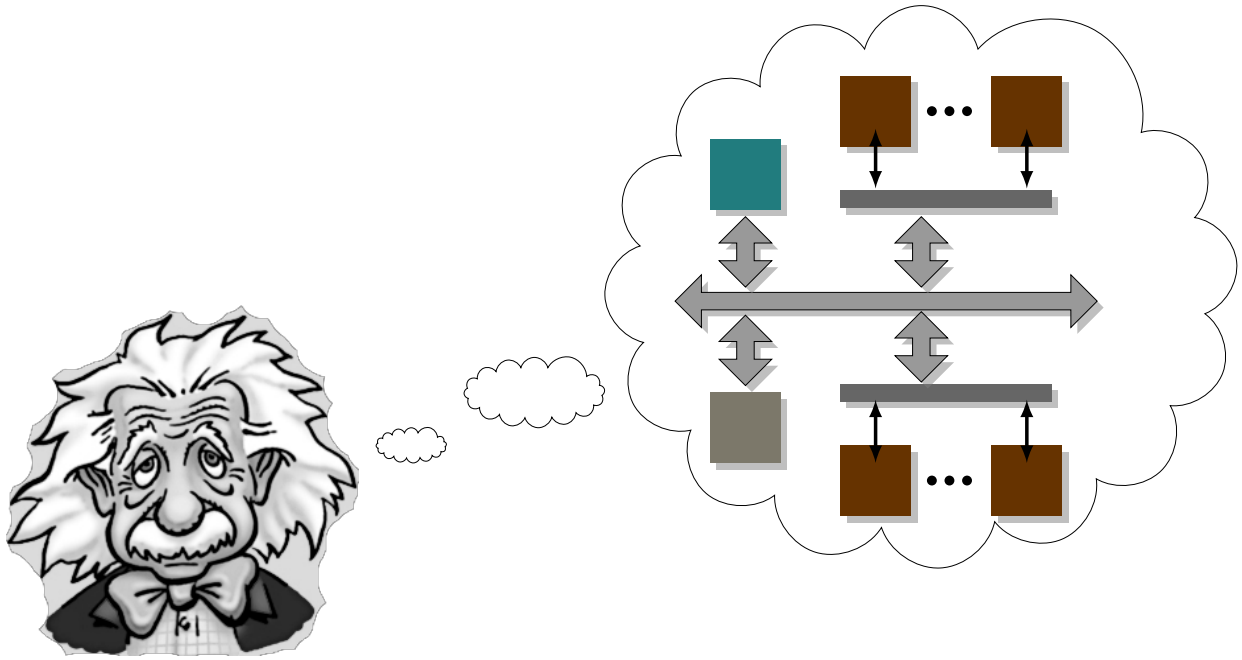
Multi-core systems



How To:

- **Deploy** the application to the platform

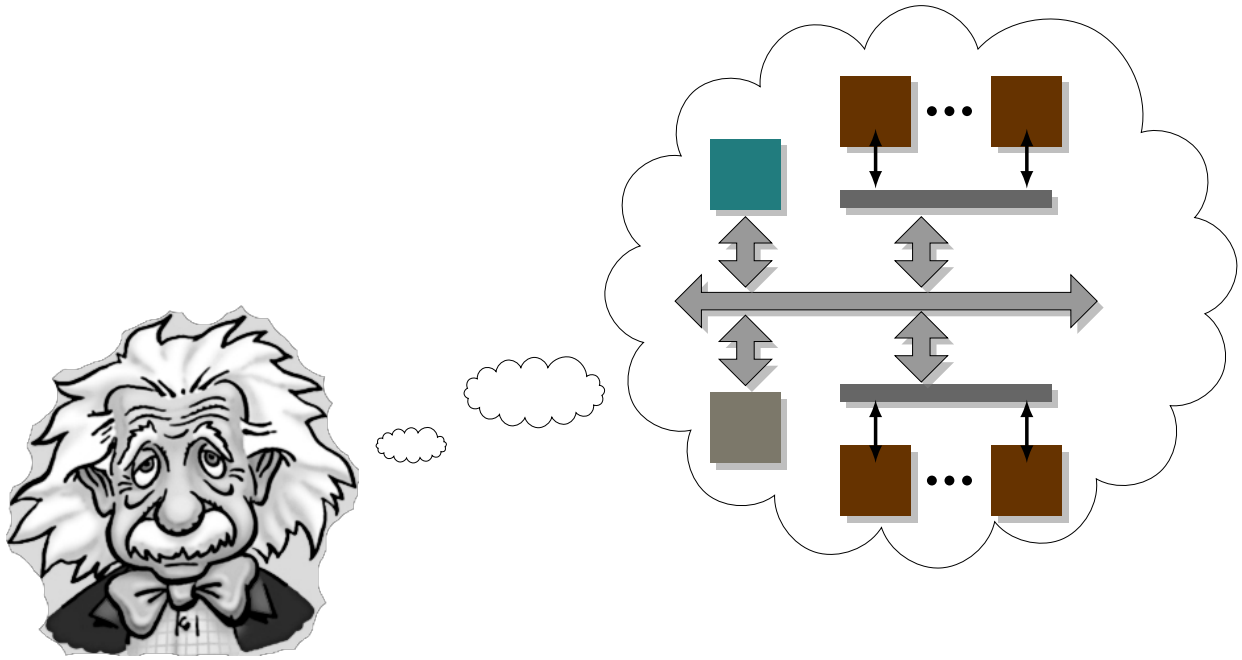
Multi-core systems



How To:

- **Deploy** the application to the platform
- Decide number of **processors** to use?

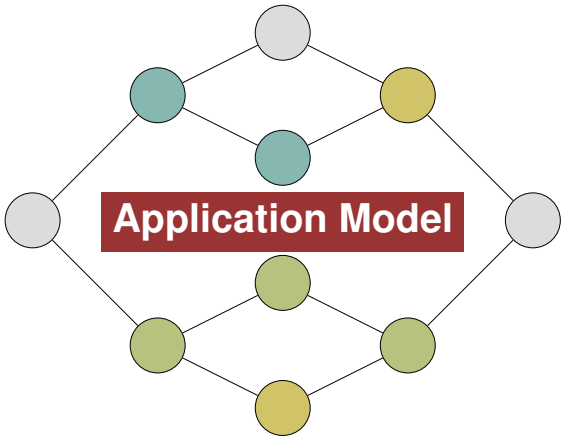
Multi-core systems



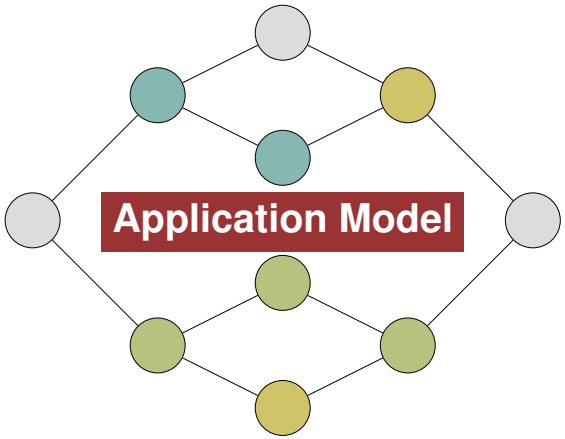
How To:

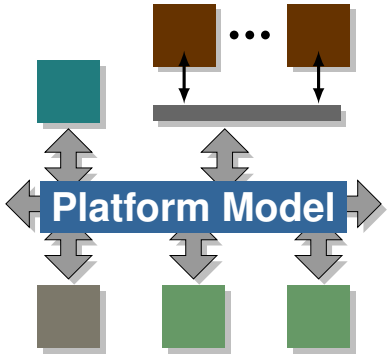
- **Deploy** the application to the platform
- Decide number of **processors** to use?
- **Allocate** tasks to processors and **schedule** them

Our Deployment Framework

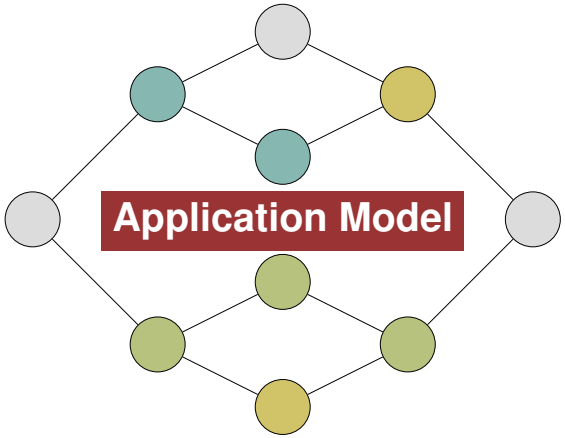


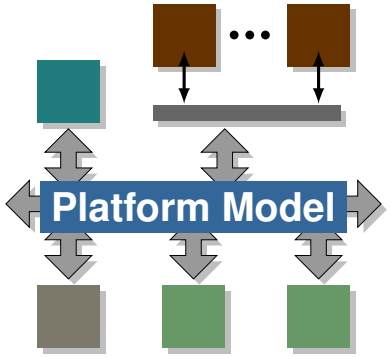
Our Deployment Framework

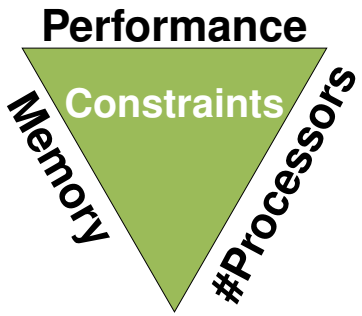




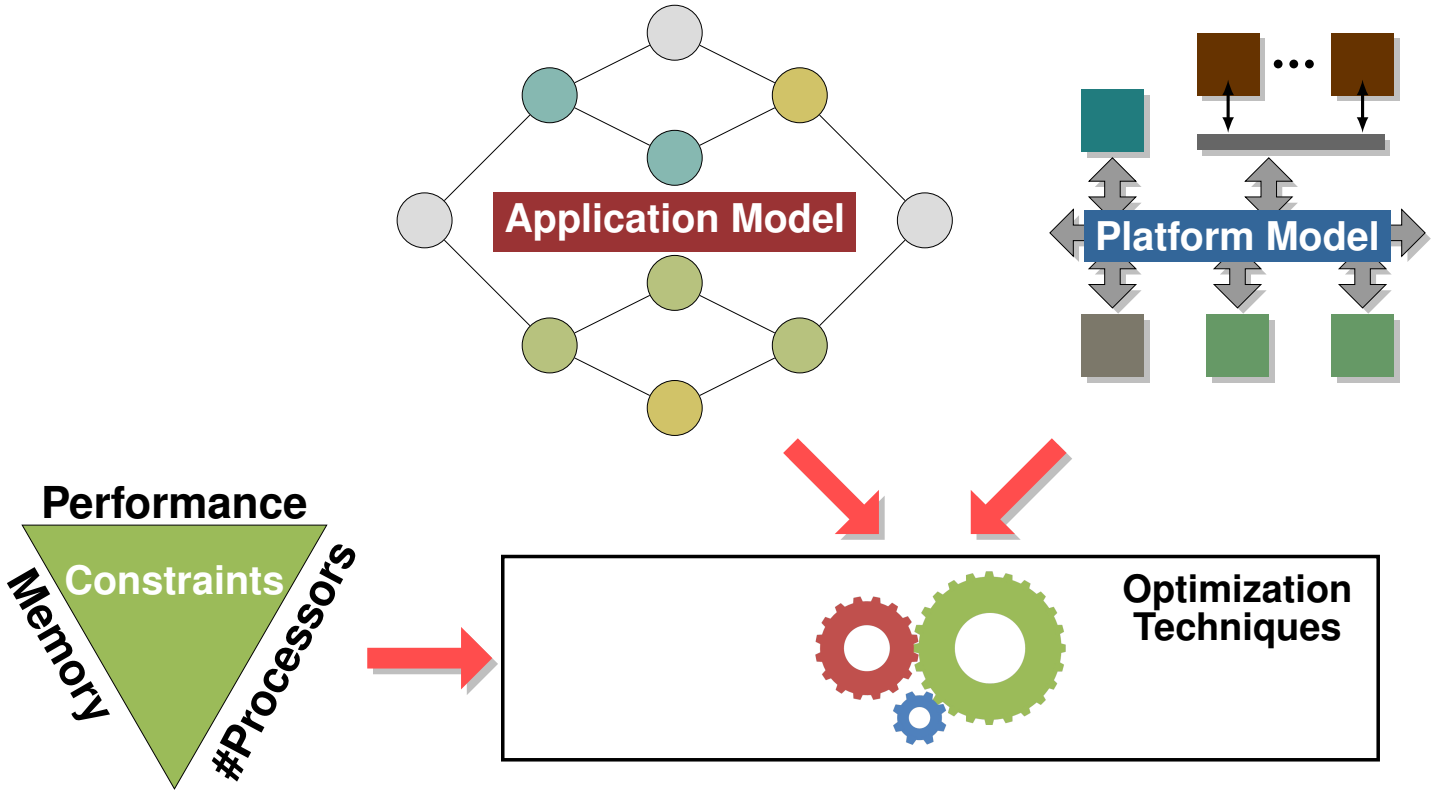
Our Deployment Framework



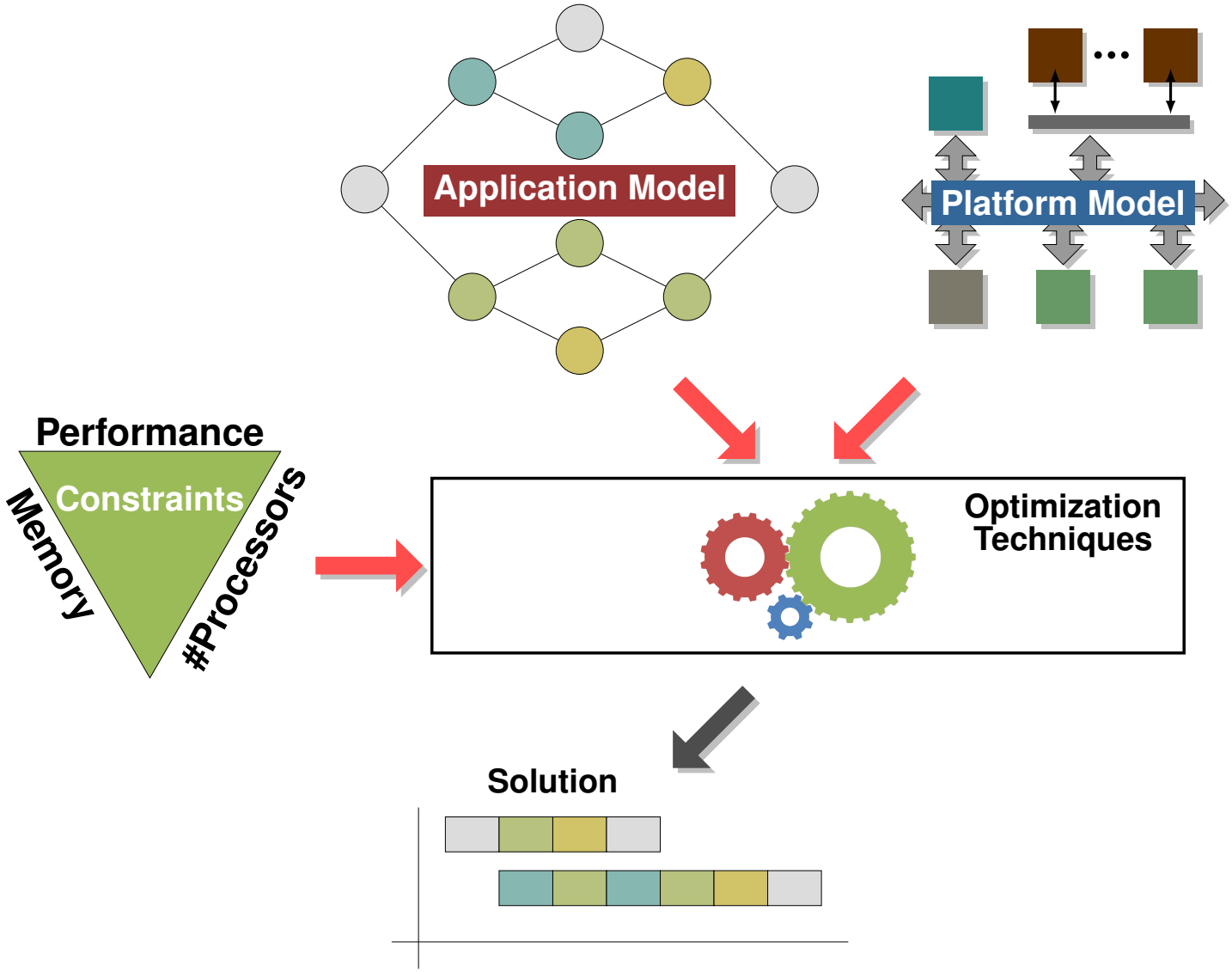




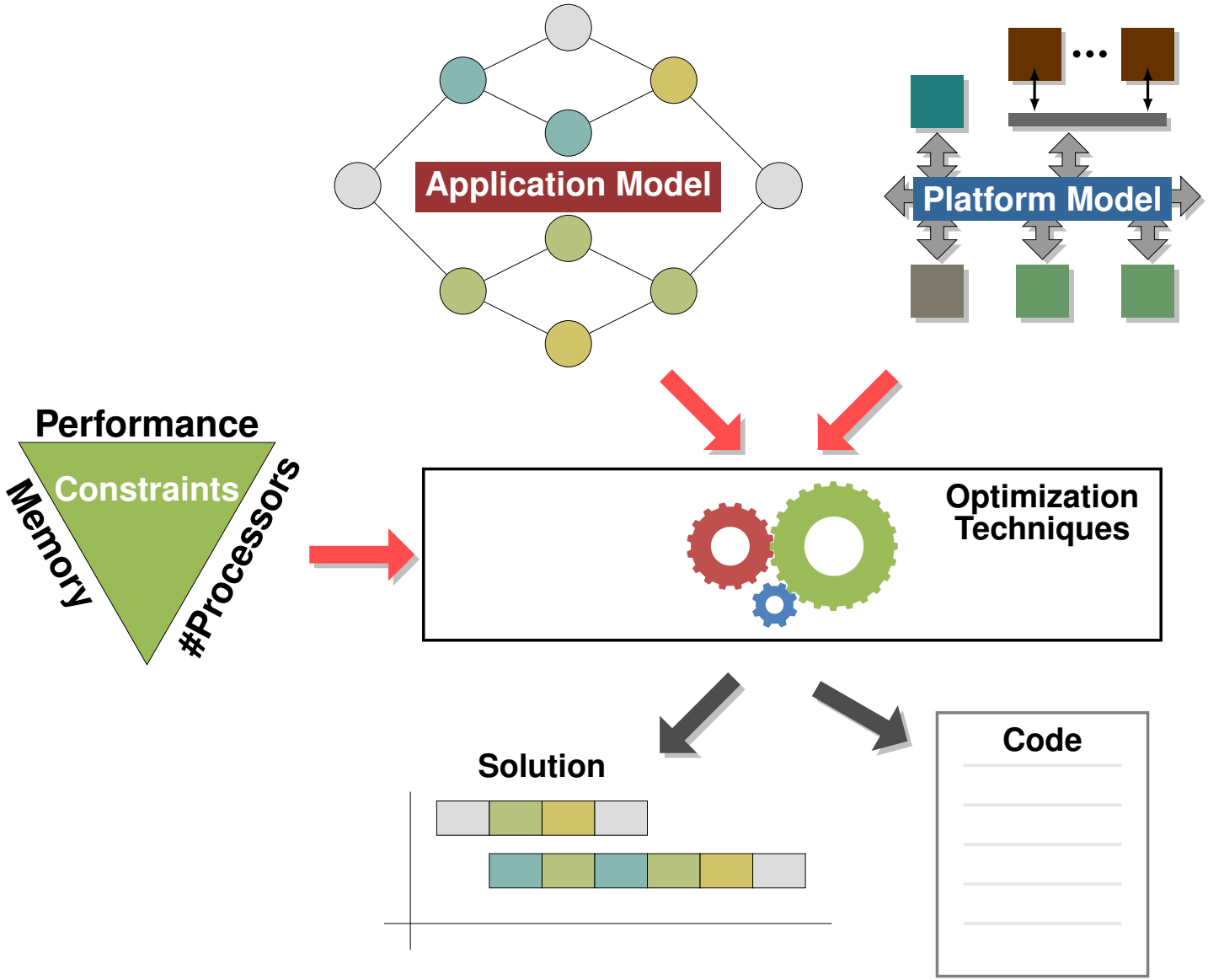
Our Deployment Framework



Our Deployment Framework

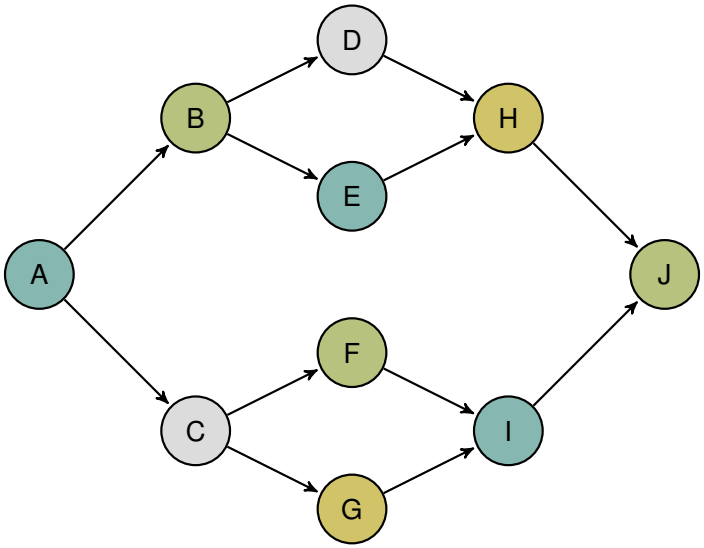


Our Deployment Framework



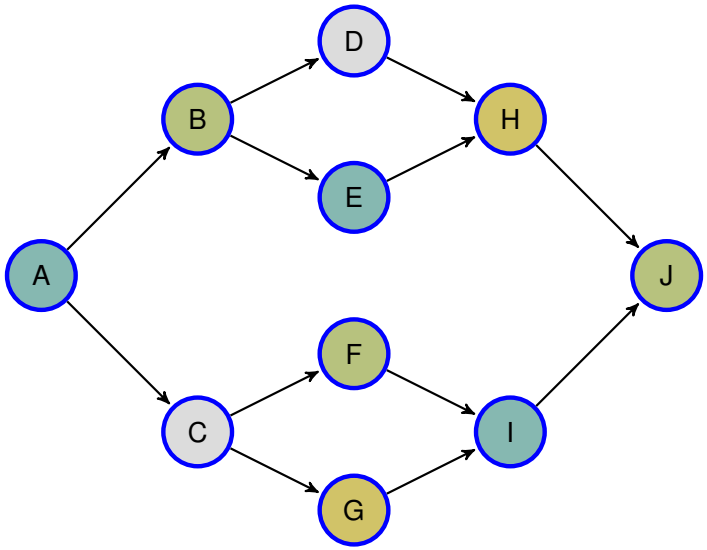
Application Model

Task Graph



Application Model

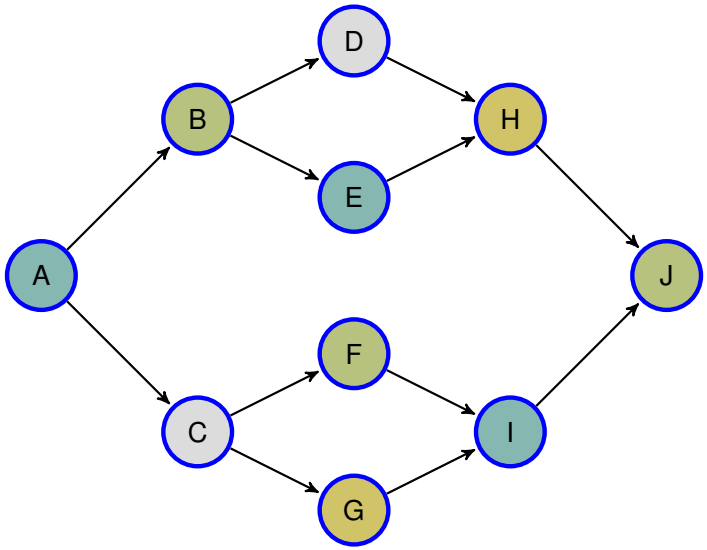
Task Graph



- **Tasks** : Software procedure

Application Model

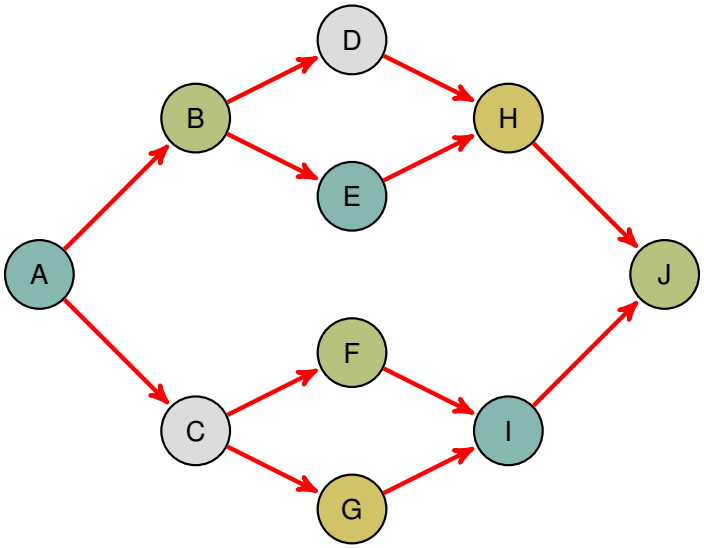
Task Graph



- **Tasks** : Software procedure
  - annotated with execution time

Application Model

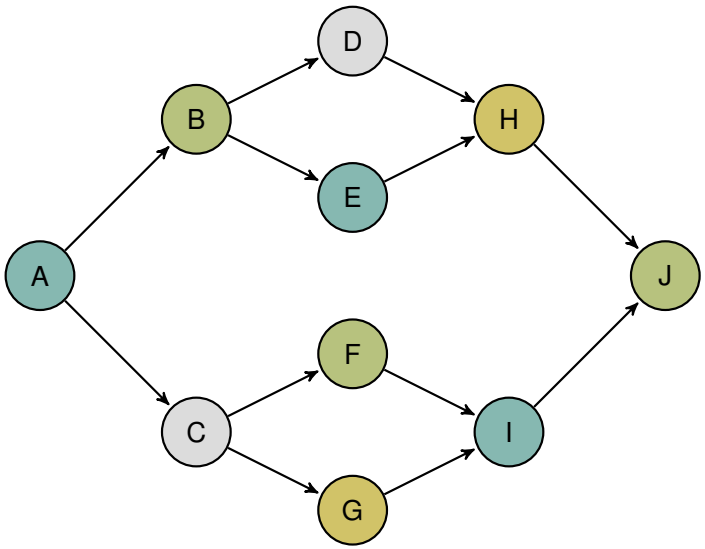
Task Graph



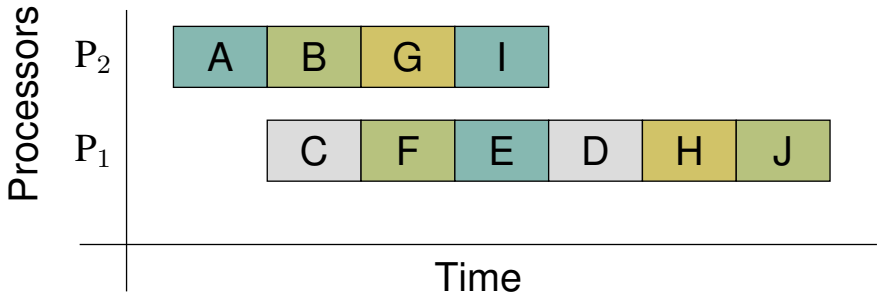
- **Tasks** : Software procedure
- **Edges** : Precedence relations

Deployment Problem

Task Graph



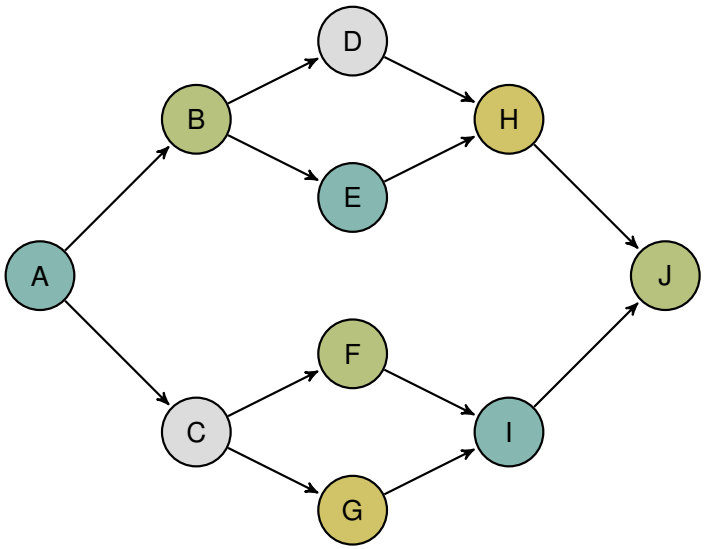
Deployment Solution



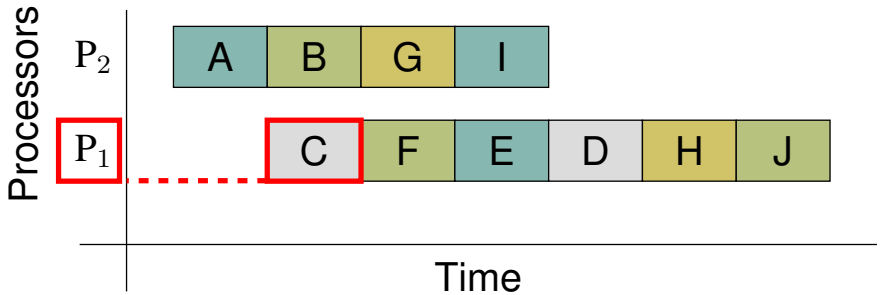
- **Tasks** : Software procedure
- **Edges** : Precedence relations

Deployment Problem

Task Graph



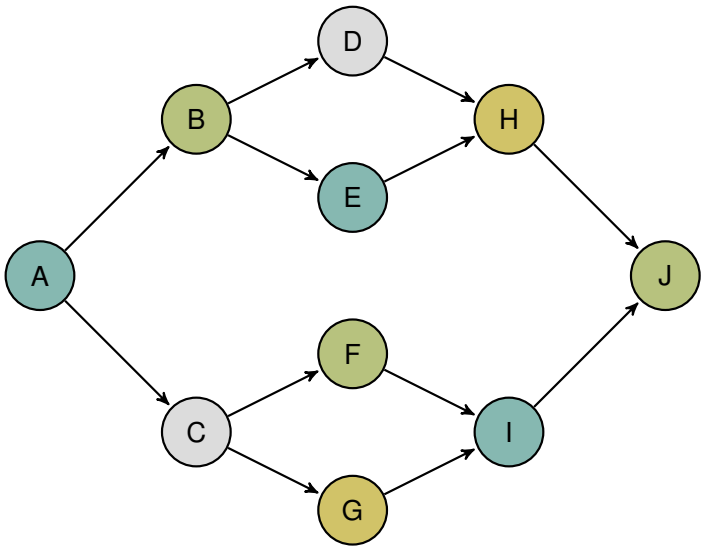
Deployment Solution



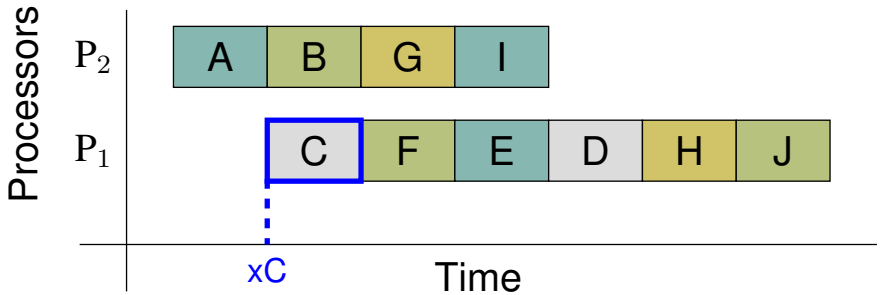
- **Tasks** : Software procedure
- **Edges** : Precedence relations
- **Mapping** : Task $\Rightarrow$ Processor

Deployment Problem

Task Graph

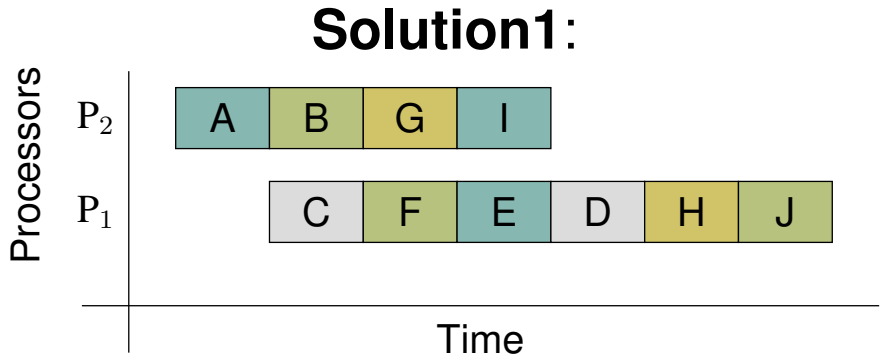


Deployment Solution



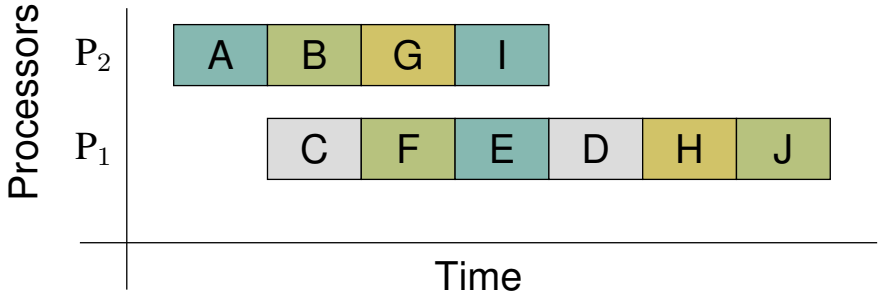
- **Tasks** : Software procedure
- **Edges** : Precedence relations
- **Mapping** : Task $\Rightarrow$ Processor
- **Scheduling** : Task $\Rightarrow$ Time

Deployment Problem

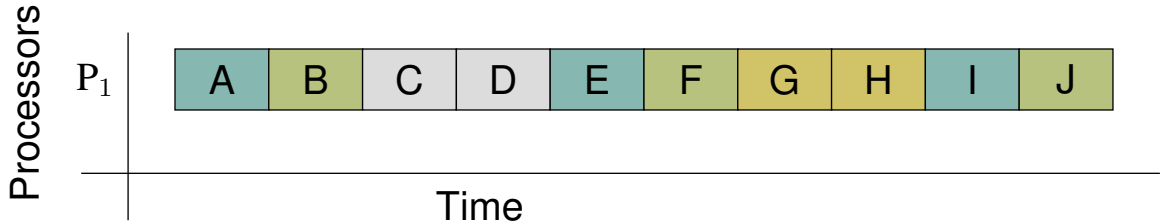


Deployment Problem

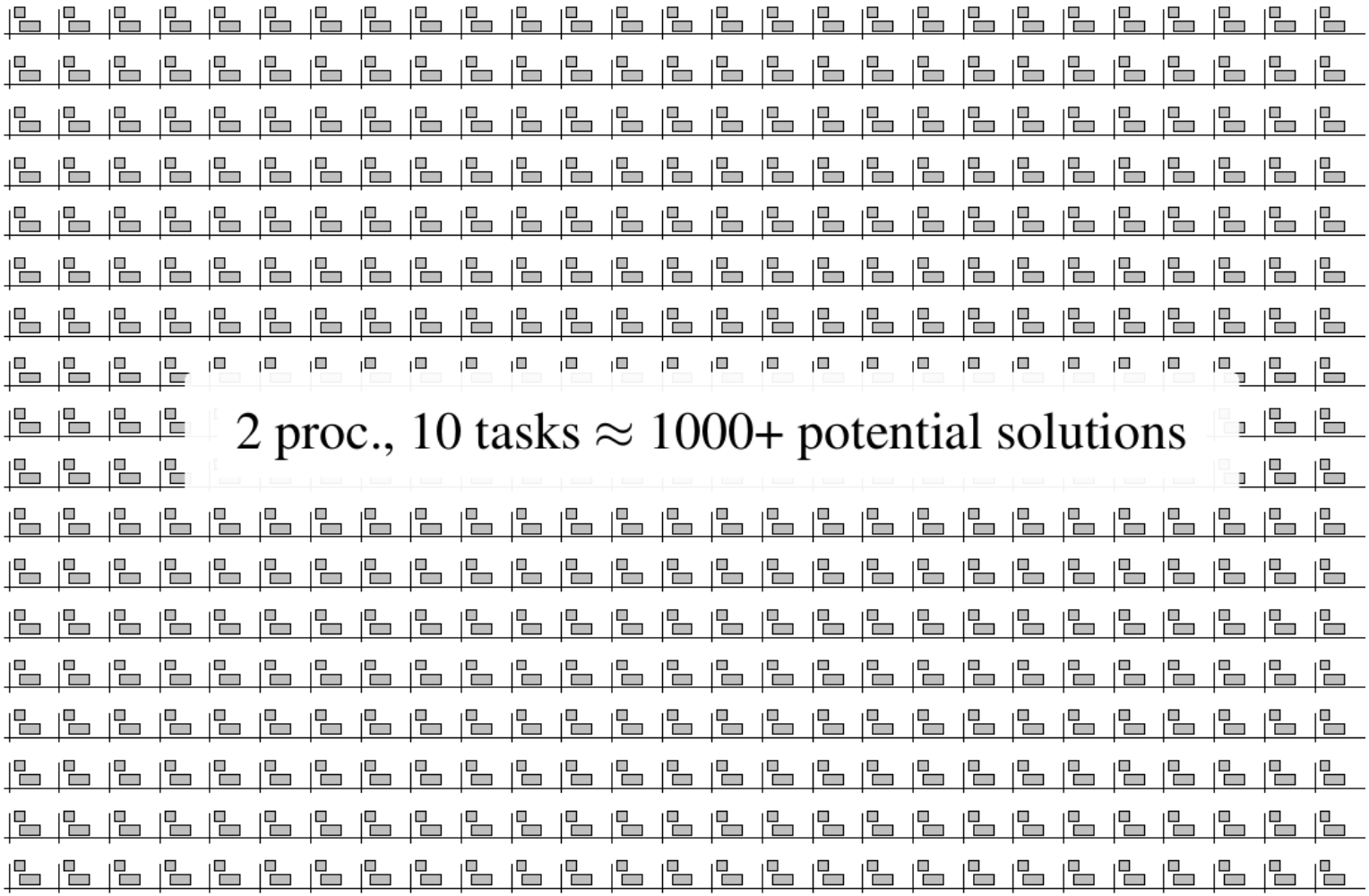
Solution1:



Solution2:



Solution space is large



Deployment problem

- The difficultly in the deployment is that the design space is exponential

Deployment problem

- The difficultly in the deployment is that the design space is exponential
- One needs to model **complex hardware** : Processors, Network, DMA

Deployment problem

- The difficultly in the deployment is that the design space is exponential
- One needs to model **complex hardware** : Processors, Network, DMA
- Multiple **Evaluation Criteria**
  - Latency
  - Memory used
  - Processors used
  - ...

Research Questions

How to:

- model the **software**

Research Questions

How to:

- model the **software**
- model the **hardware** (Processors, Network, DMA)

Research Questions

How to:

- model the **software**
- model the **hardware** (Processors, Network, DMA)
- Optimize deployment while dealing with design space **explosion**

Outline

Overview

Model of Computation

Synchronous Dataflow graphs (SDF)

by Edward Lee and David Messerschmitt in 1987

Model of Computation

Synchronous Dataflow graphs (SDF)

by Edward Lee and David Messerschmitt in 1987

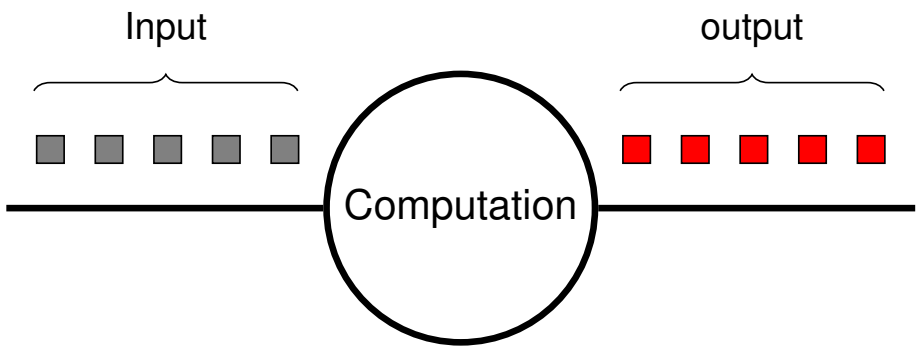
represents **Streaming Applications**

Model of Computation

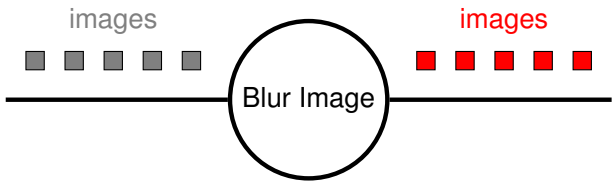
Synchronous Dataflow graphs (SDF)

by Edward Lee and David Messerschmitt in 1987

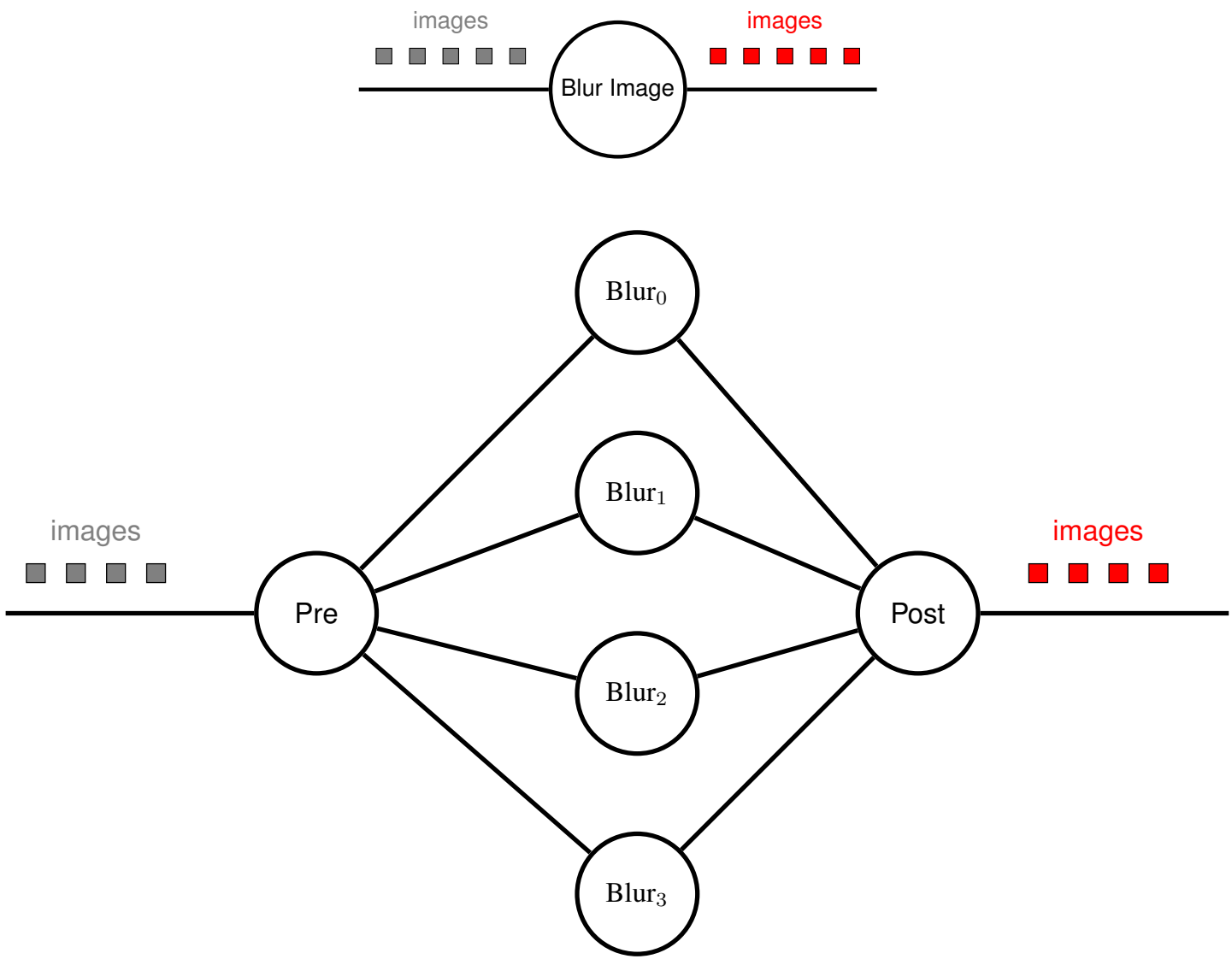
represents **Streaming Applications**



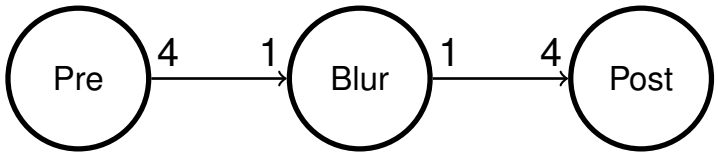
Synchronous DataFlow



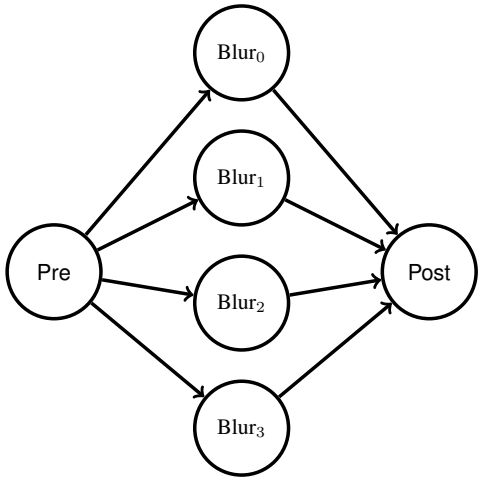
Synchronous DataFlow



Synchronous DataFlow

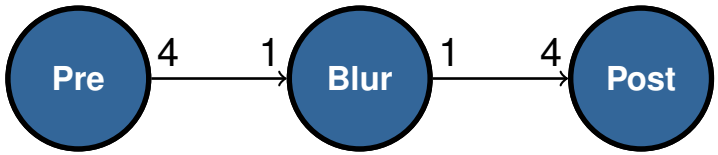


SDF Graph

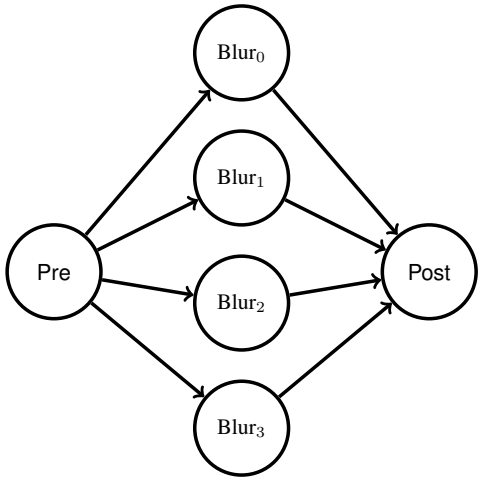


Task Graph

Synchronous DataFlow



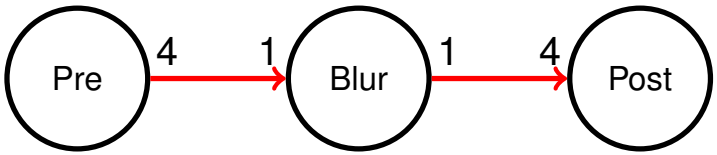
SDF Graph



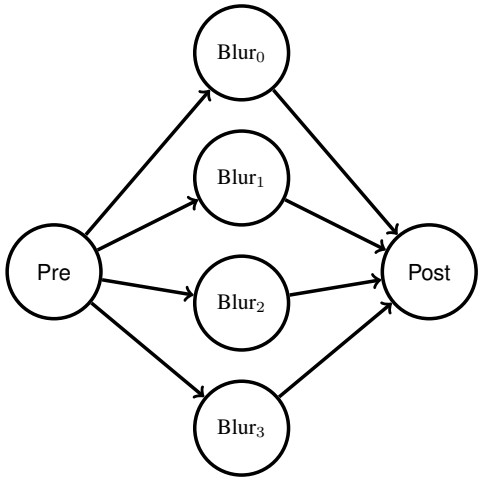
Task Graph

- **Actors**
  - Pre-processing
  - Blur
  - Post-processing

Synchronous DataFlow



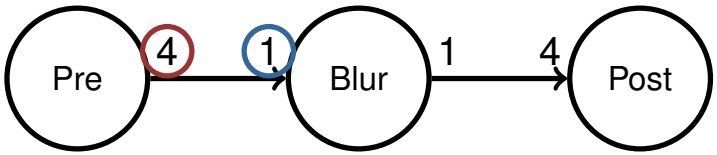
SDF Graph



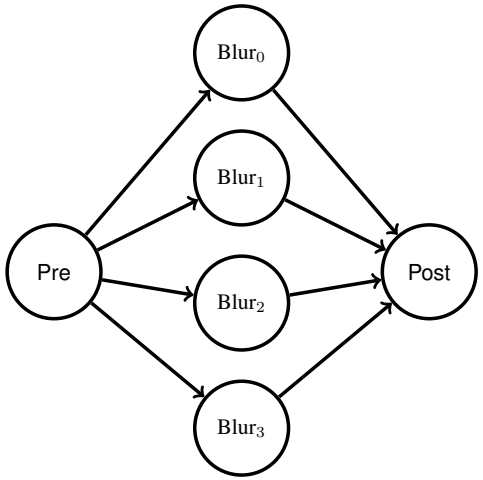
Task Graph

- **Actors**
  - Pre-processing
  - Blur
  - Post-processing
- **Edges**
  - Blur executes only after Pre-processing finishes

Synchronous DataFlow



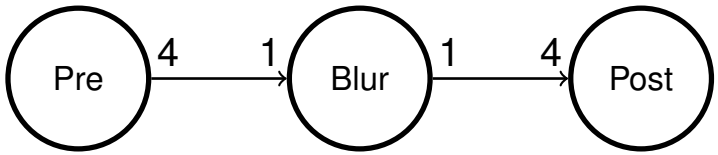
SDF Graph



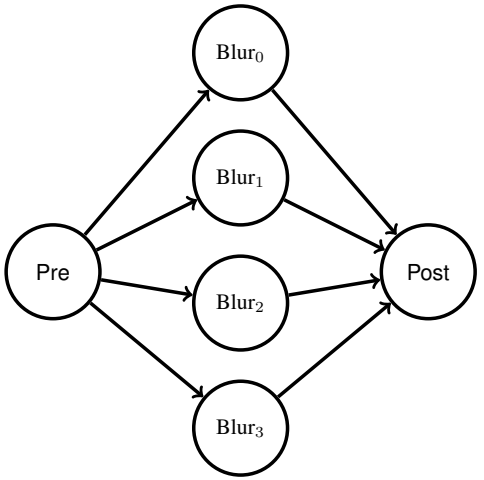
Task Graph

- **Actors**
  - Pre-processing
  - Blur
  - Post-processing
- **Edges**
  - Blur executes only after Pre-processing finishes
- **Rates**
  - Pre-processing produces **4 pieces** of an image (tokens)
  - Each Blur consumes **1 piece**

Synchronous DataFlow



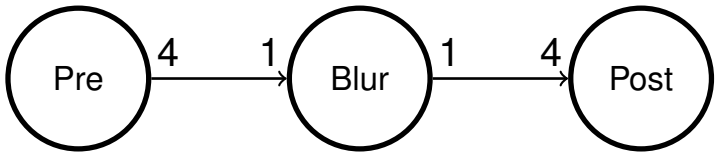
SDF Graph



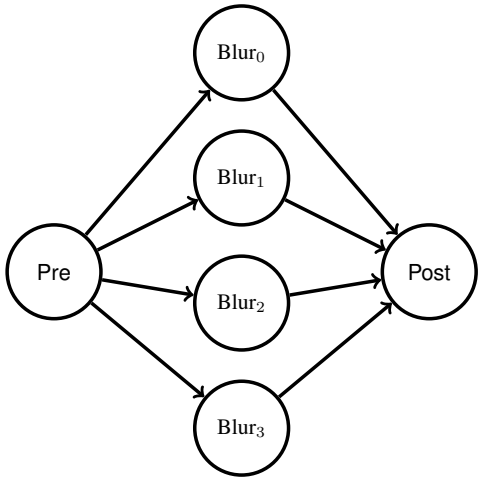
Task Graph

- Actor Blur is **compact representation** of data parallel tasks.

Synchronous DataFlow



SDF Graph



Task Graph

- Actor Blur is **compact representation** of data parallel tasks.
- All Blur tasks have **same properties** such as execution time.

Split-Join Graphs

we use **split-join graphs** : restriction of SDF

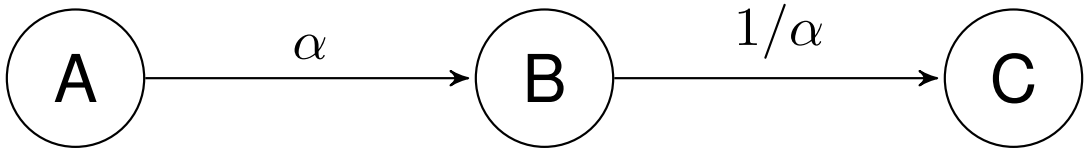
still covering perhaps 90% of use cases in the literature

Split-Join Graphs

we use **split-join graphs** : restriction of SDF

still covering perhaps 90% of use cases in the literature

a simple example:



α : spawn and split

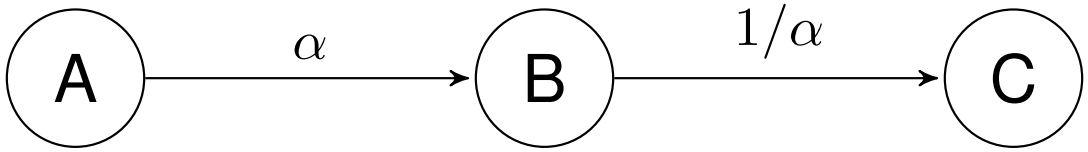
$1/\alpha$: wait and join

Split-Join Graphs

we use **split-join graphs** : restriction of SDF

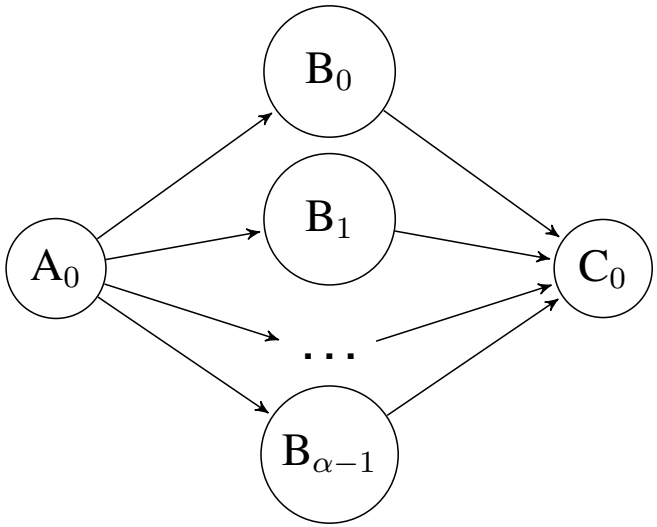
still covering perhaps 90% of use cases in the literature

a simple example:

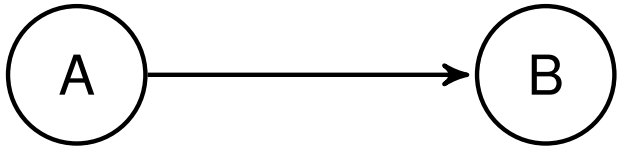


α : spawn and split

$1/\alpha$: wait and join



Restrictions compared to general SDF



Restrictions compared to general SDF

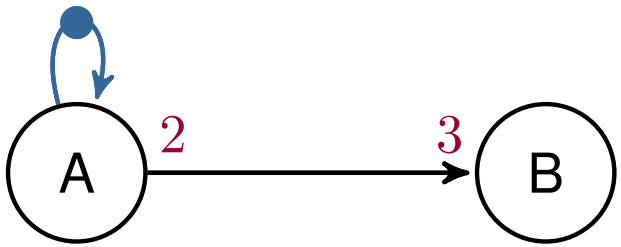
Split-join does not support:

- **Stateful** actors

Restrictions compared to general SDF

Split-join does not support:

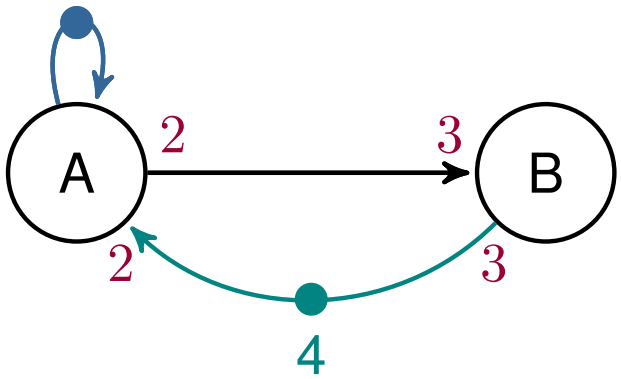
- **Stateful** actors
- **Non-proportional** rates



Restrictions compared to general SDF

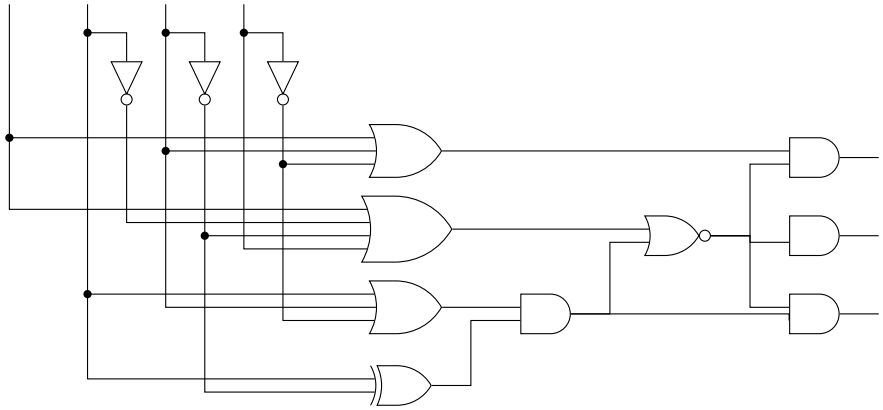
Split-join does not support:

- **Stateful** actors
- **Non-proportional** rates
- **Initial tokens** and **cyclic** paths

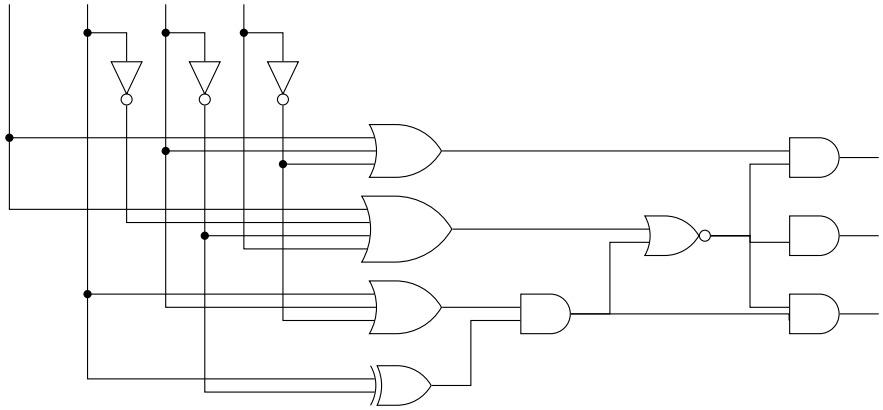


Overview

SATisfiability solver (SAT / SMT)

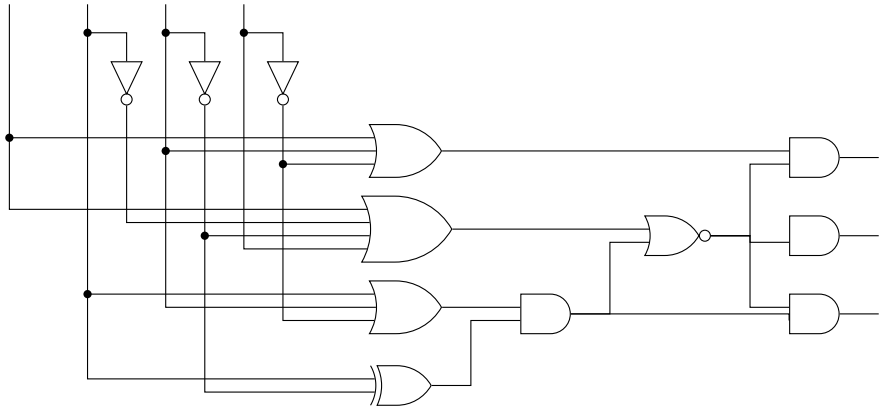


SATisfiability solver (SAT / SMT)



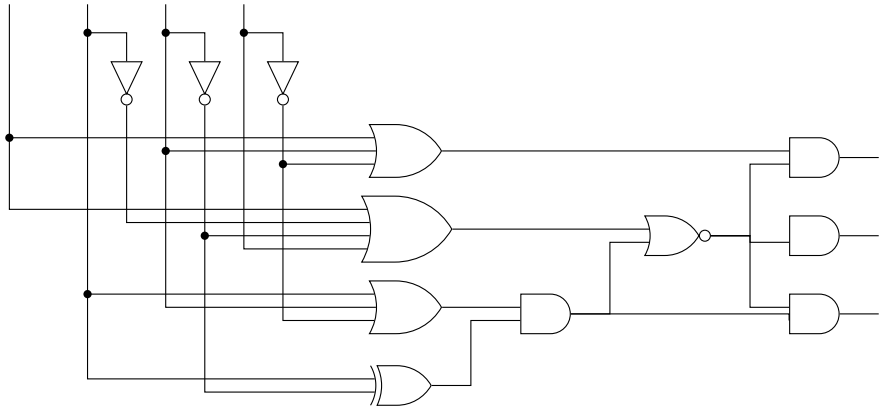
- Boolean variables
  - in_0, in_1, in_2 ...
  - out_0, out_1, out_2 ...

SATisfiability solver (SAT / SMT)



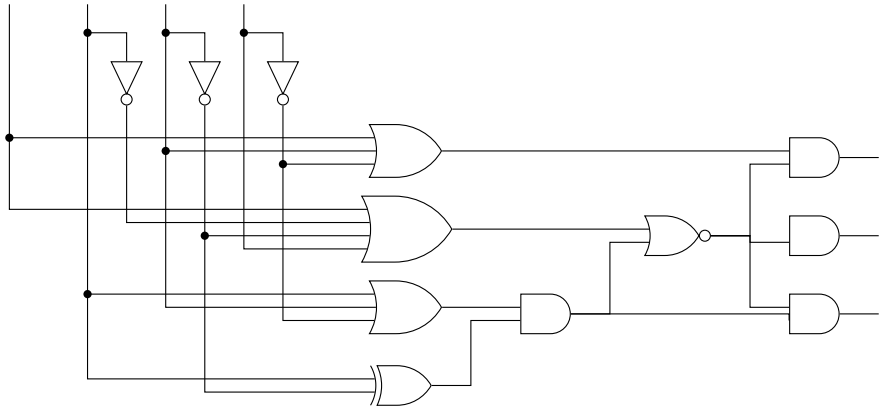
- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2$ ...
  - $\text{out}_0, \text{out}_1, \text{out}_2$ ...
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2$ ...

SATisfiability solver (SAT / SMT)

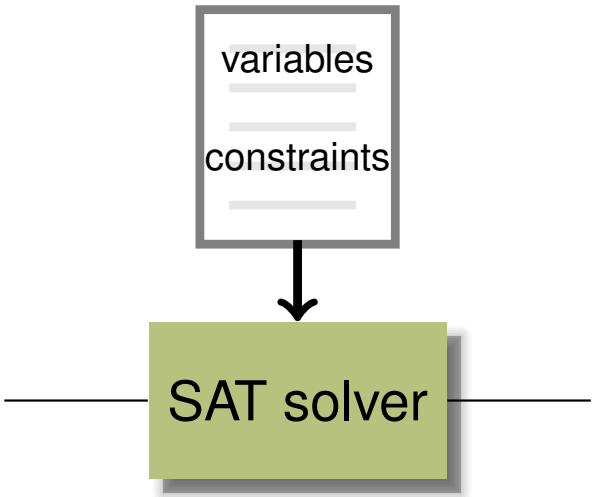


- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2$ ...
  - $\text{out}_0, \text{out}_1, \text{out}_2$ ...
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2$ ...

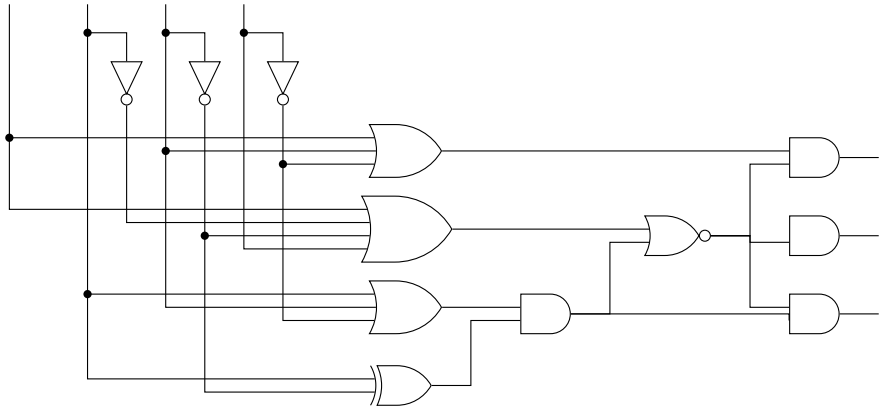
SATisfiability solver (SAT / SMT)



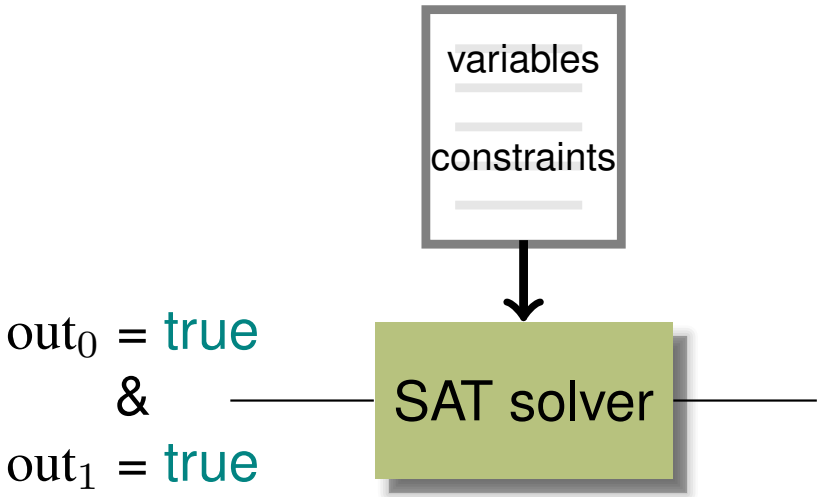
- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2$ ...
  - $\text{out}_0, \text{out}_1, \text{out}_2$ ...
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2$ ...



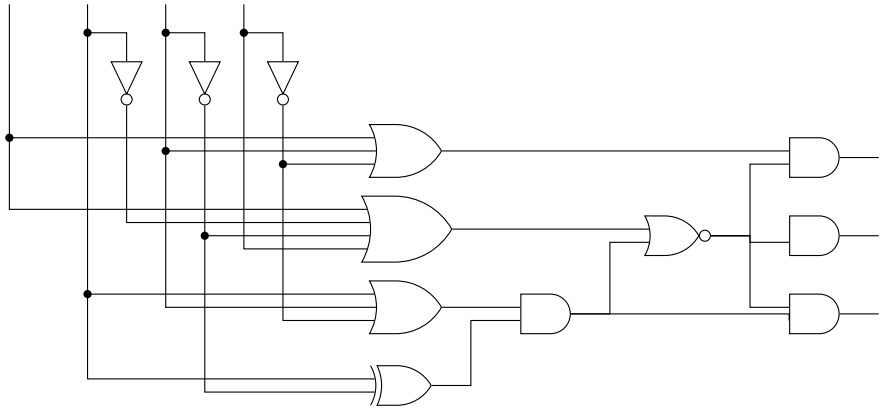
SATisfiability solver (SAT / SMT)



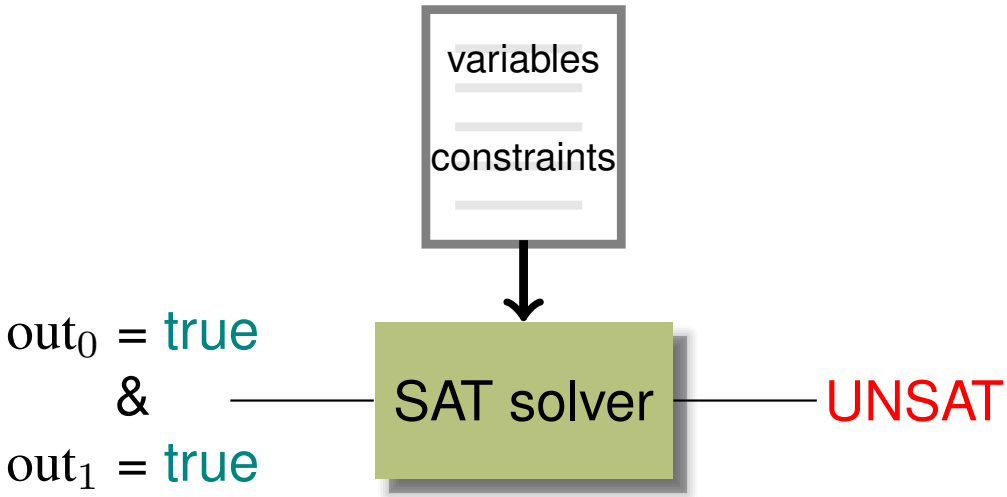
- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2$ ...
  - $\text{out}_0, \text{out}_1, \text{out}_2$ ...
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2$ ...



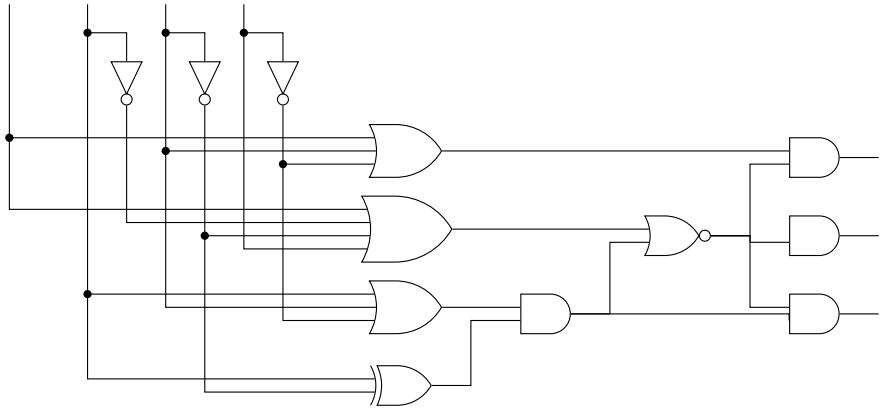
SATisfiability solver (SAT / SMT)



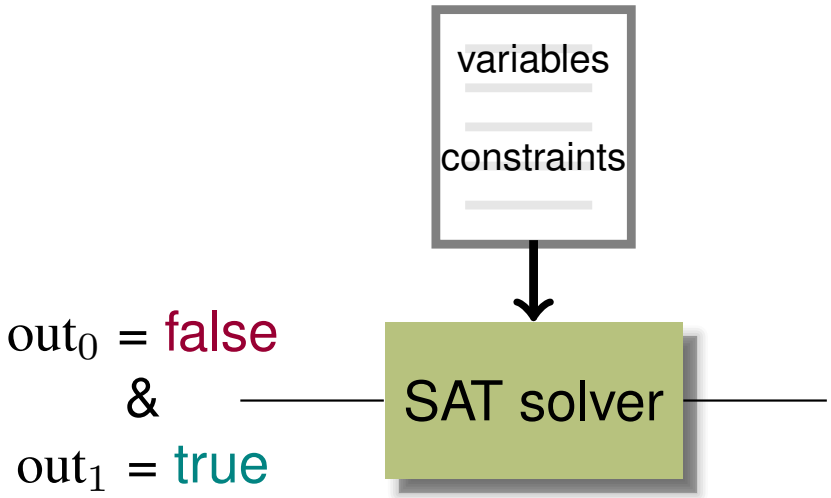
- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2 \ldots$
  - $\text{out}_0, \text{out}_1, \text{out}_2 \ldots$
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2 \ldots$



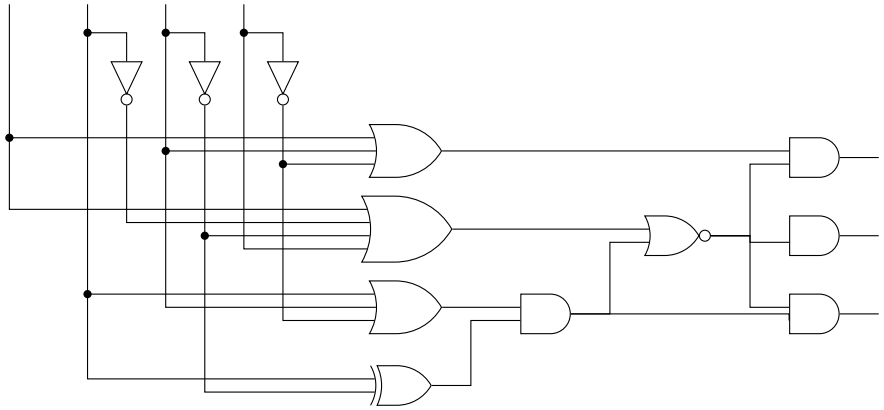
SATisfiability solver (SAT / SMT)



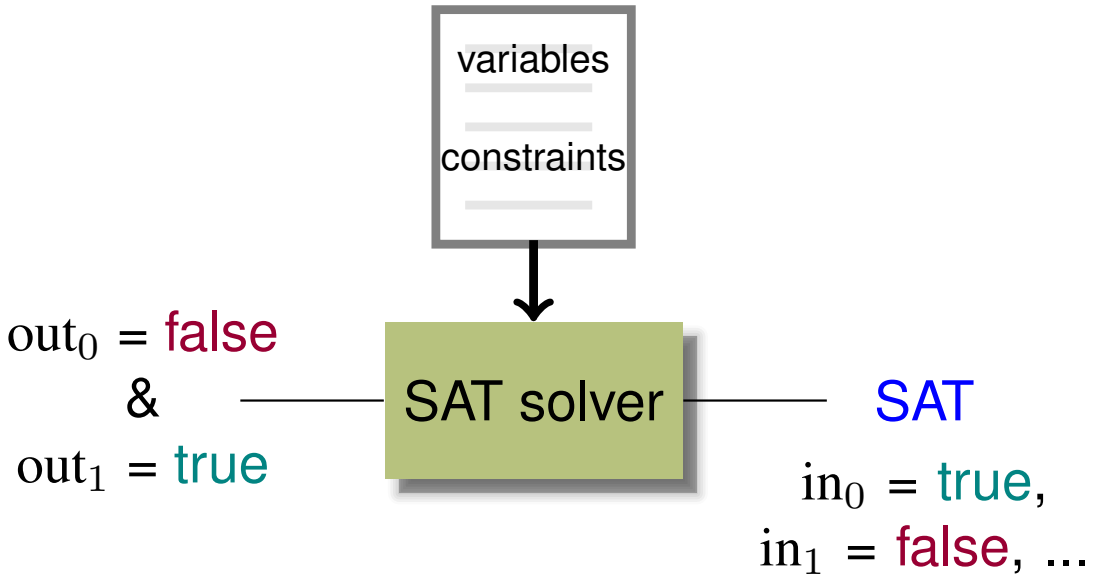
- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2 \ldots$
  - $\text{out}_0, \text{out}_1, \text{out}_2 \ldots$
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2 \ldots$



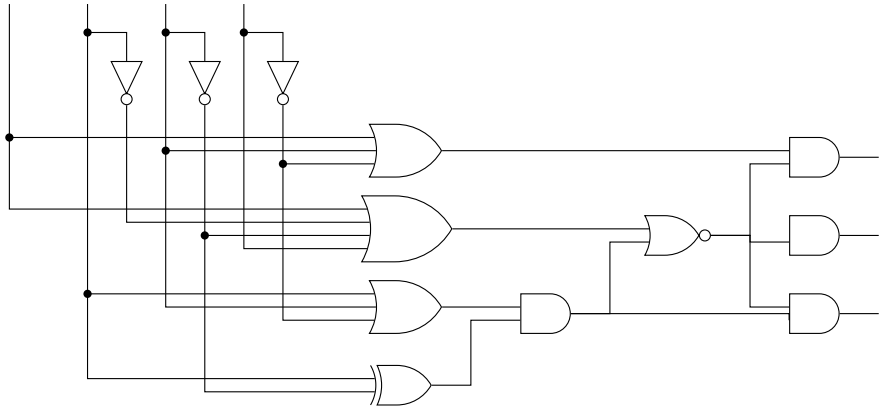
SATisfiability solver (SAT / SMT)



- Boolean variables
  - in_0, in_1, in_2 ...
  - out_0, out_1, out_2 ...
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2$ ...

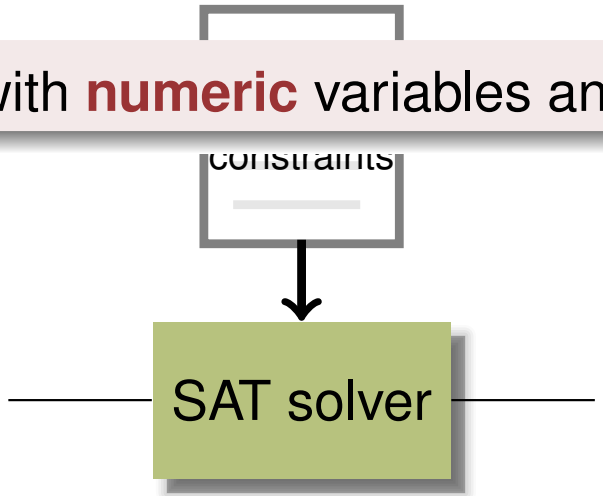


SATisfiability solver (SAT / SMT)

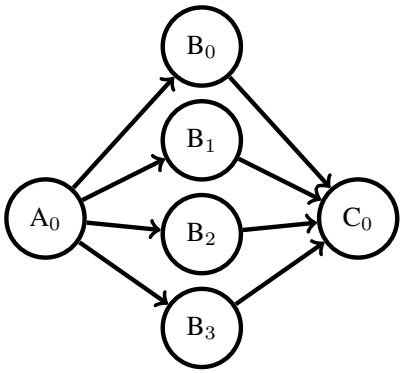


- Boolean variables
  - $\text{in}_0, \text{in}_1, \text{in}_2$ ...
  - $\text{out}_0, \text{out}_1, \text{out}_2$ ...
- Constraints
  - $\text{out}_0 = \text{in}_0 \vee \text{in}_1 \oplus \text{in}_2$ ...

SMT deals with **numeric** variables and constants



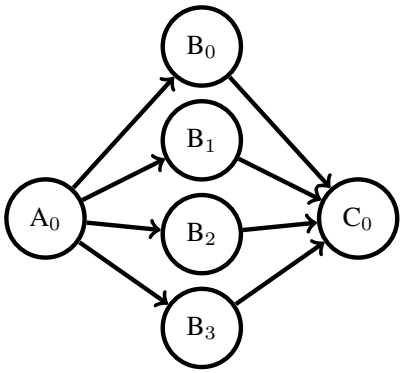
Encoding deployment with constraints



Task Graph

Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					

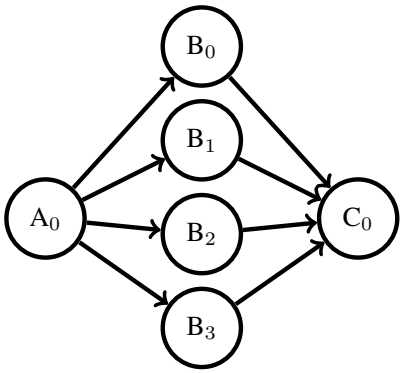
Encoding deployment with constraints



Task Graph

Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					
Start time	xA_0	xB_0	xB_1	xB_2	xB_3	xC_0

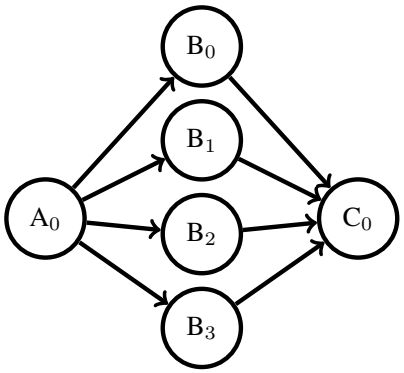
Encoding deployment with constraints



Task Graph

Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					
Start time	xA_0	xB_0	xB_1	xB_2	xB_3	xC_0
Allocated proc.	pA_0	pB_0	pB_1	pB_2	pB_3	pC_0

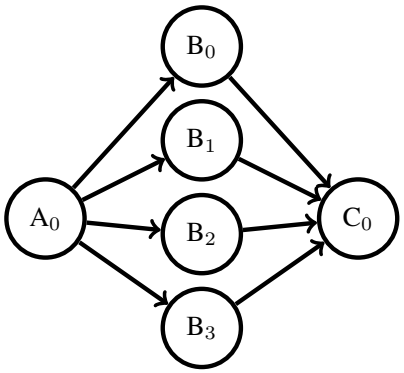
Encoding deployment with constraints



Task Graph

Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					
Start time	xA_0	xB_0	xB_1	xB_2	xB_3	xC_0
Allocated proc.	pA_0	pB_0	pB_1	pB_2	pB_3	pC_0
Duration	dA	dB				dC

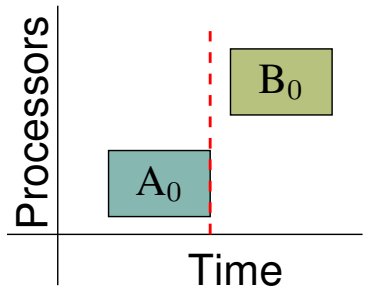
Encoding deployment with constraints



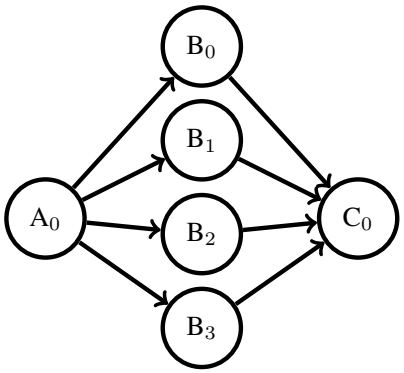
Task Graph

Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					
Start time	xA_0	xB_0	xB_1	xB_2	xB_3	xC_0
Allocated proc.	pA_0	pB_0	pB_1	pB_2	pB_3	pC_0
Duration	dA	dB				dC

- Precedence Constraints
  - $xB_0 \geq (xA_0 + dA)$



Encoding deployment with constraints

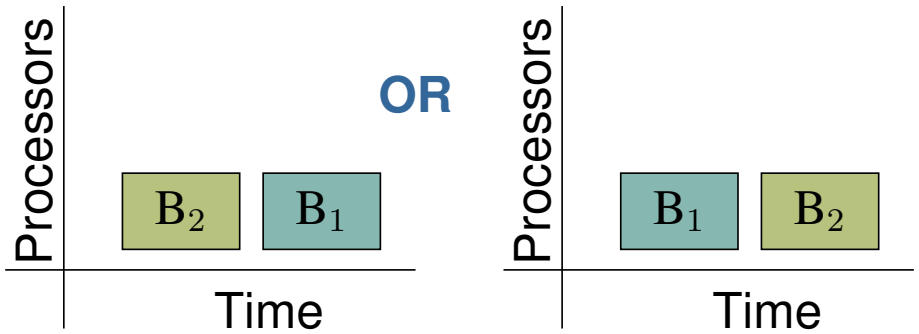


Task Graph

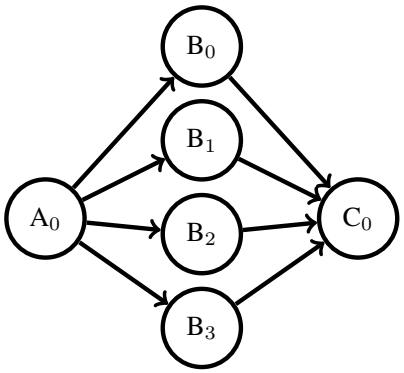
Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					
Start time	xA_0	xB_0	xB_1	xB_2	xB_3	xC_0
Allocated proc.	pA_0	pB_0	pB_1	pB_2	pB_3	pC_0
Duration	dA	dB				dC

- Precedence Constraints
  - $xB_0 \geq (xA_0 + dA)$
- Mutual Exclusion Constraints
  - if $(pB_1 = pB_2)$ then

$$xB_1 \geq (xB_2 + dB) \ \vee \ xB_2 \geq (xB_1 + dB)$$

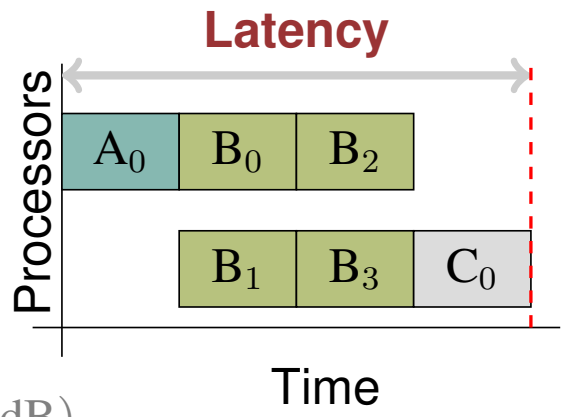


Encoding deployment with constraints



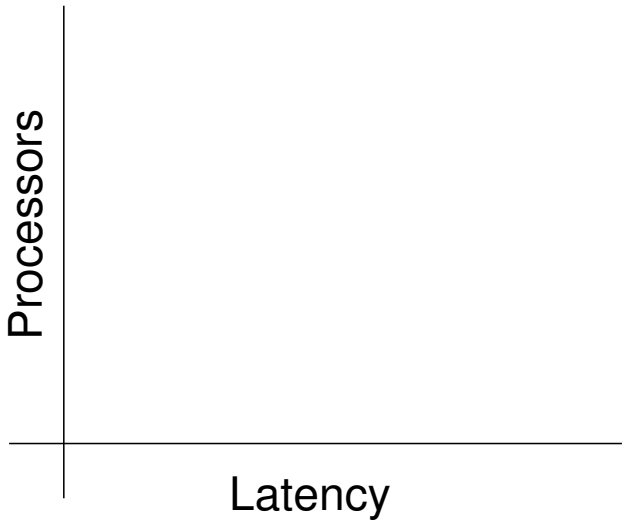
Task Graph

Actor	**A**	**B**				**C**
Tasks	A_0	B_0	B_1	B_2	B_3	C_0
Description	**Variables**					
Start time	xA_0	xB_0	xB_1	xB_2	xB_3	xC_0
Allocated proc.	pA_0	pB_0	pB_1	pB_2	pB_3	pC_0
Duration	dA	dB				dC

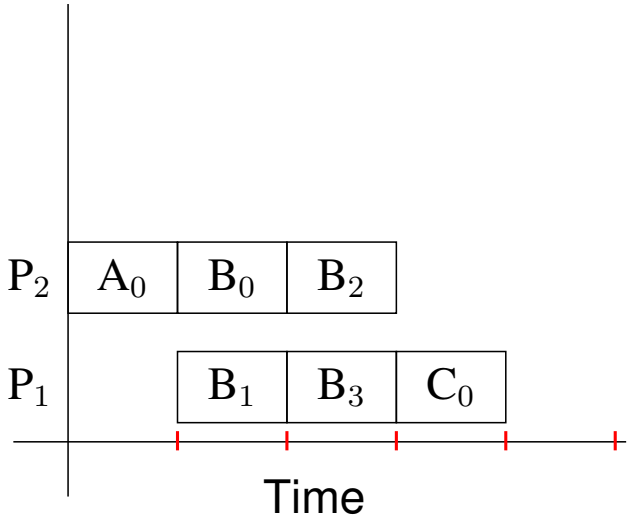


- Precedence Constraints
  - $xB_0 \geq (xA_0 + dA)$
- Mutual Exclusion Constraints
  - if $(pB_1 = pB_2)$ then
    $xB_1 \geq (xB_2 + dB) \ \vee \ xB_2 \geq (xB_1 + dB)$
- Latency Cost
  - $\text{Latency} = (xC_0 + dC)$

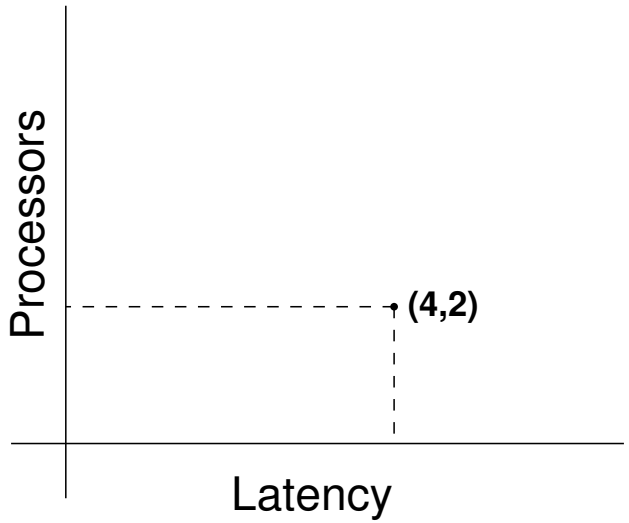
Multi-criteria Problem



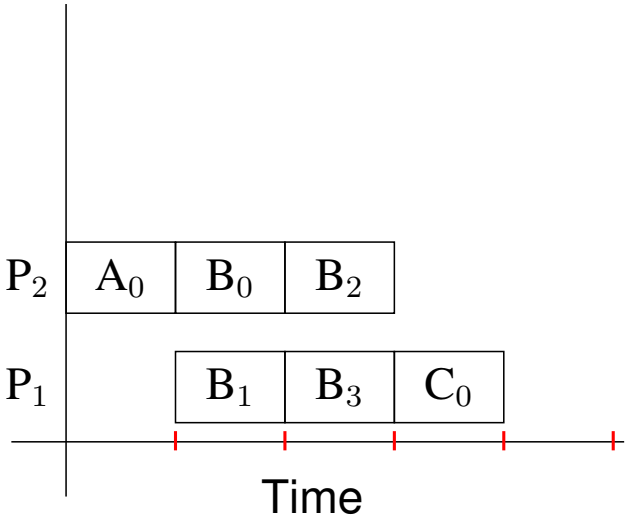
Multi-criteria Problem



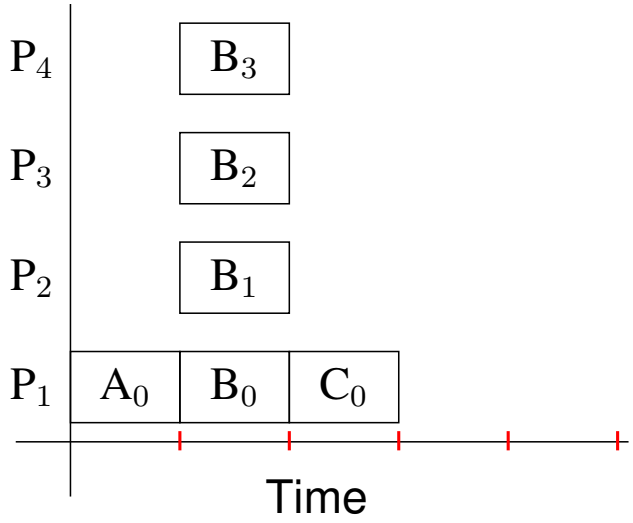
Latency = 4
#Proc = 2



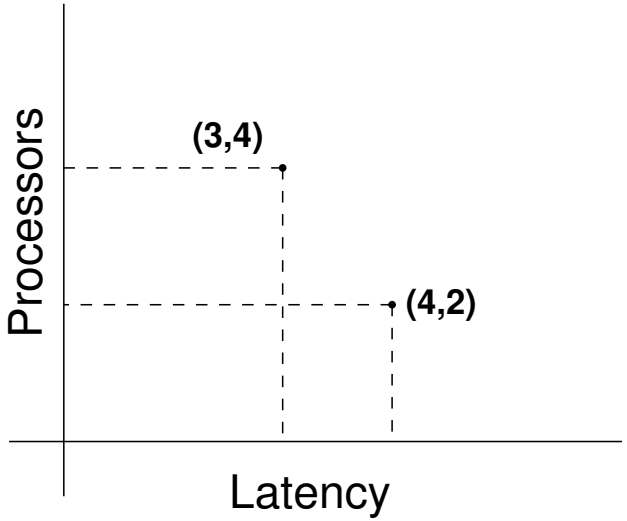
Multi-criteria Problem



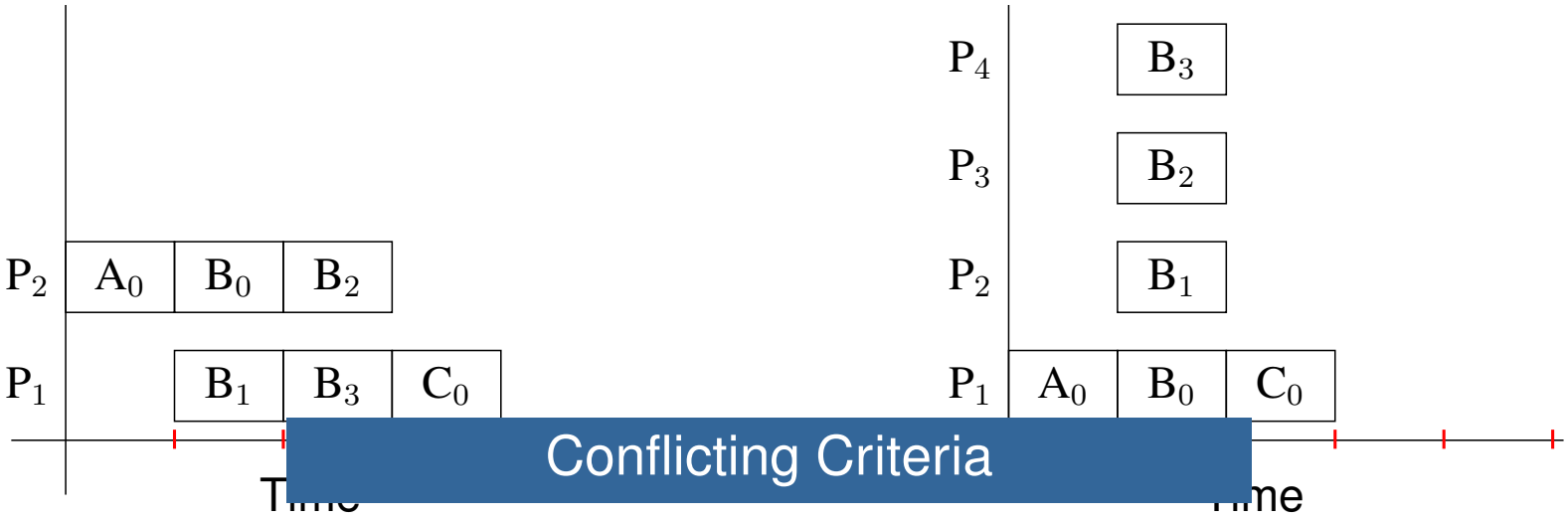
Latency = 4
#Proc = 2



Latency = 3
#Proc = 4

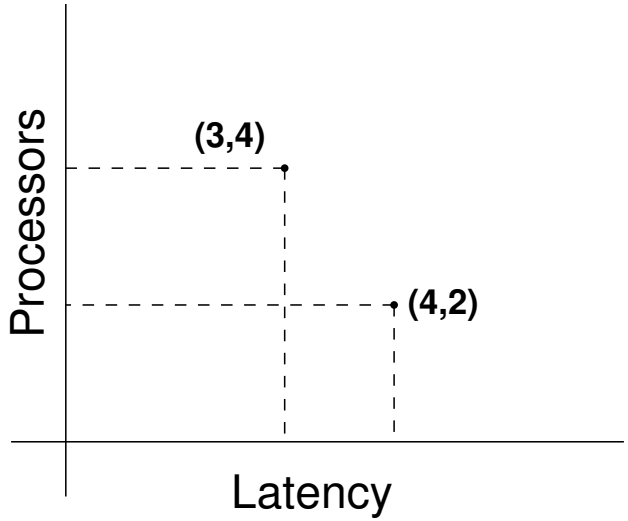


Multi-criteria Problem

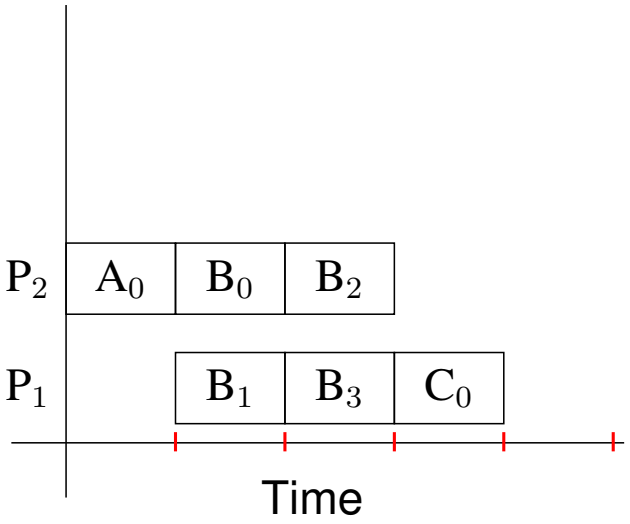


Latency = 4
#Proc = 2

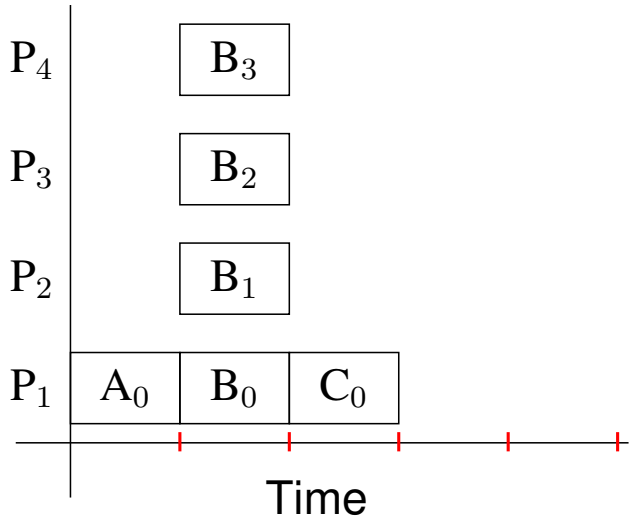
Latency = 3
#Proc = 4



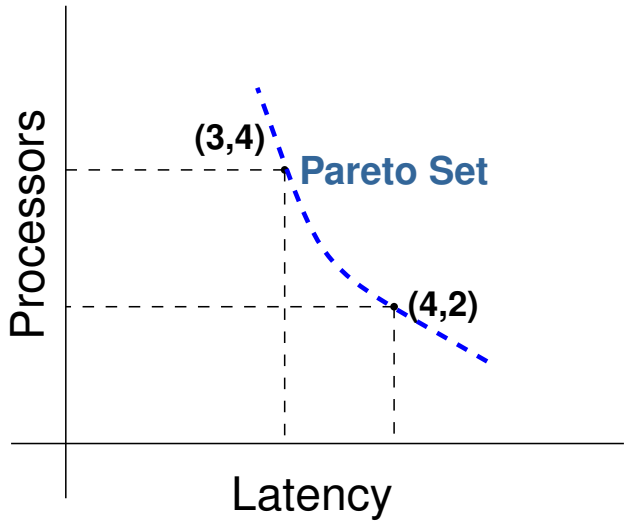
Multi-criteria Problem



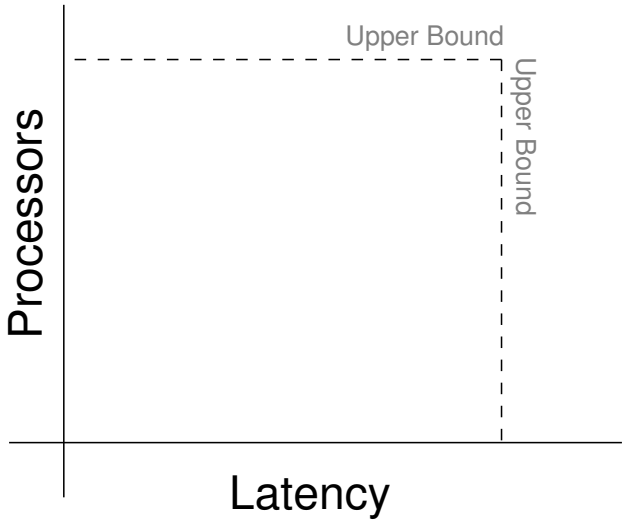
Latency = 4
#Proc = 2



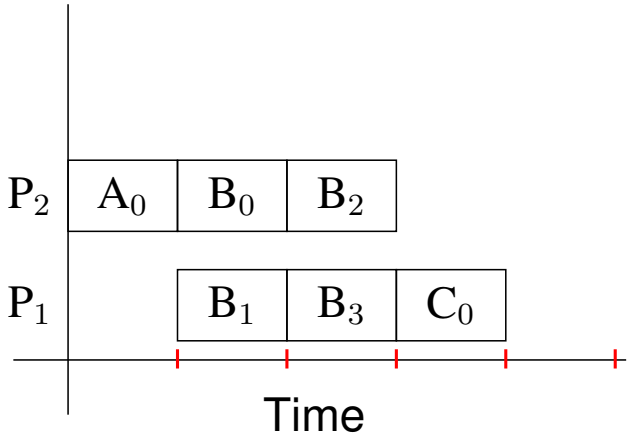
Latency = 3
#Proc = 4



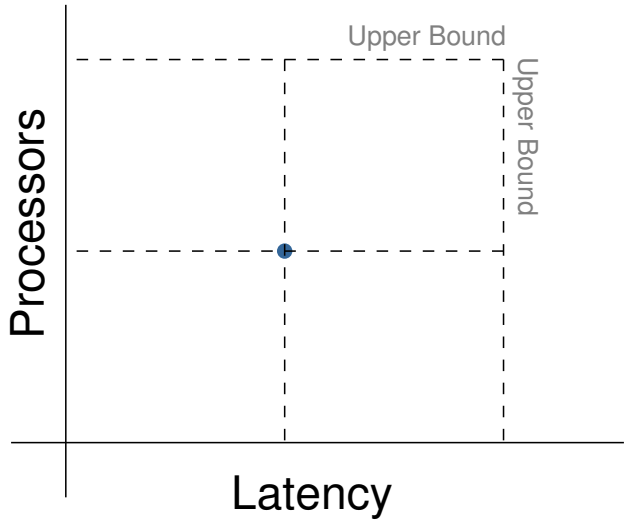
Problem Monotonicity



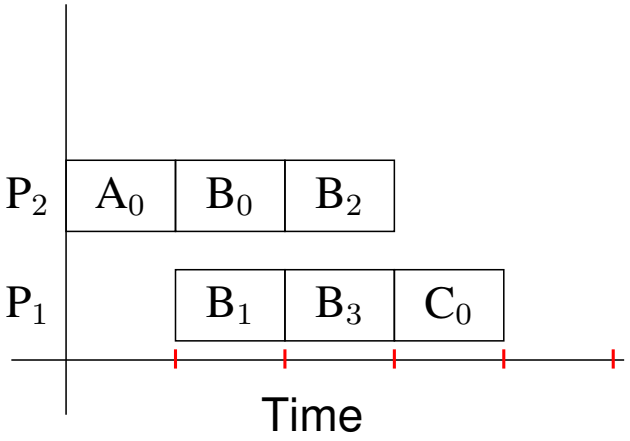
Problem Monotonicity



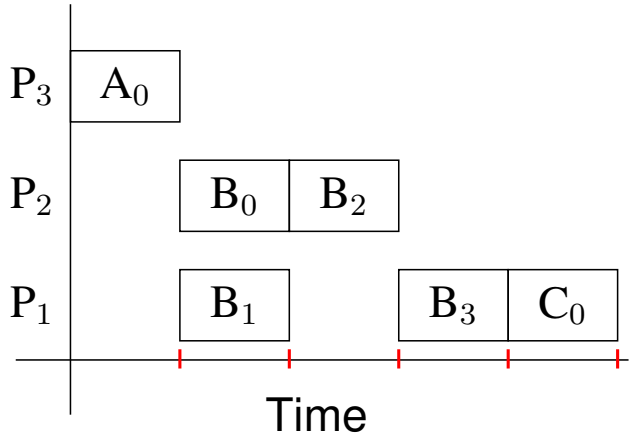
Latency = 4
#Proc = 2



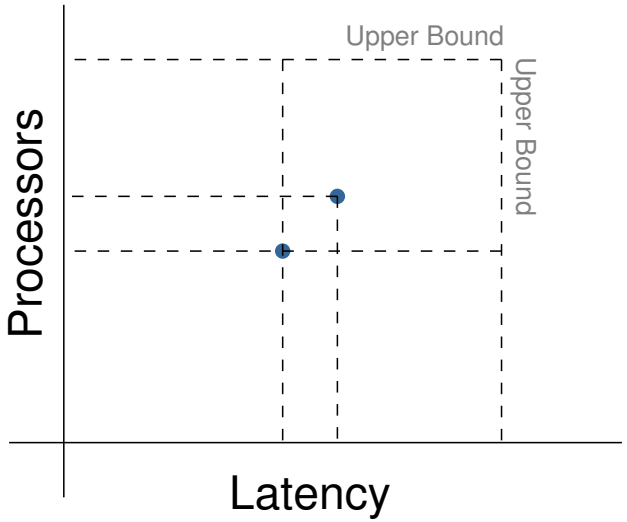
Problem Monotonicity



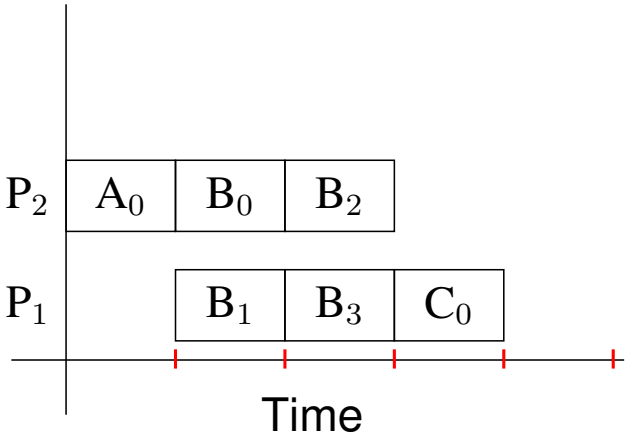
Latency = 4
#Proc = 2



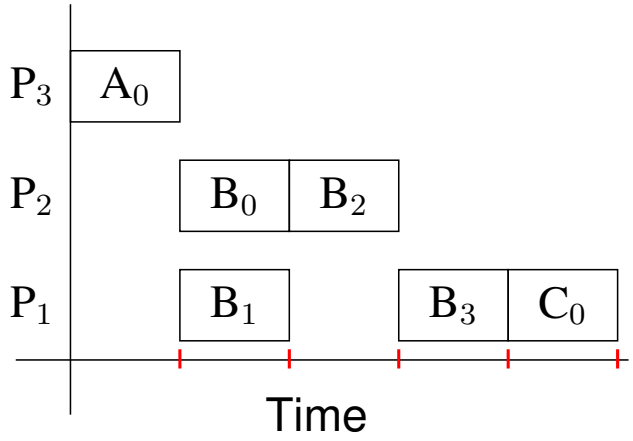
Latency = 5
#Proc = 3



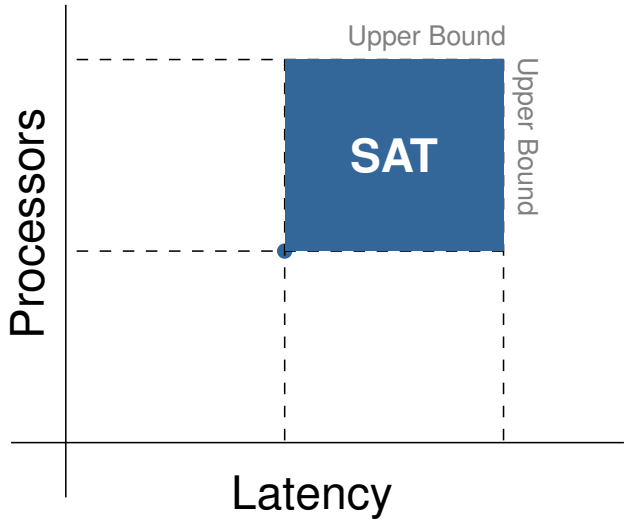
Problem Monotonicity



Latency = 4
#Proc = 2

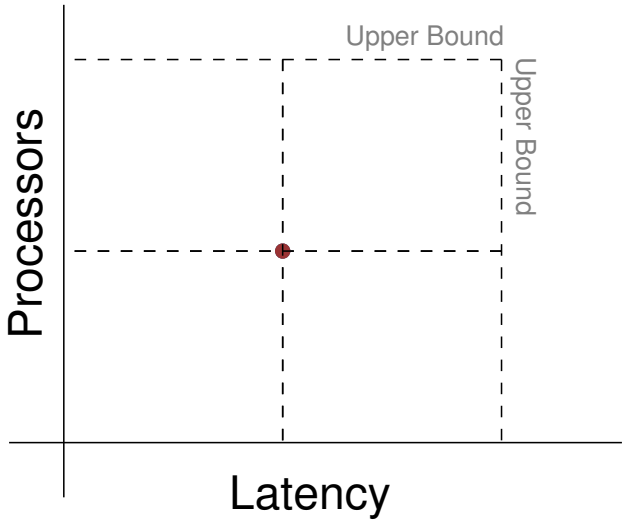


Latency = 5
#Proc = 3



Problem Monotonicity

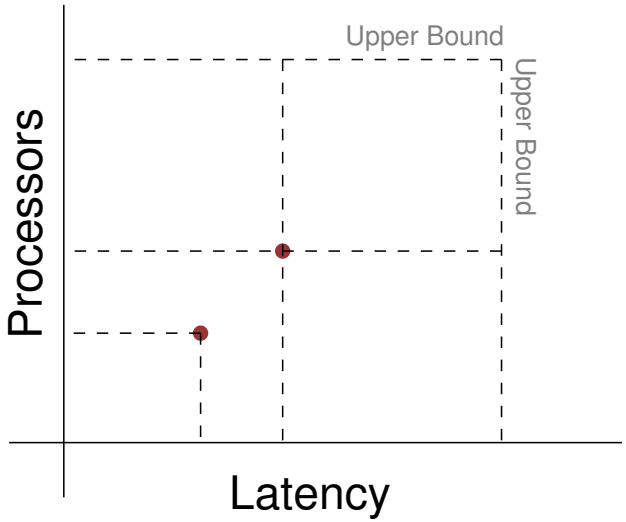
Latency ≤ 4
#Proc ≤ 2
Not Possible



Problem Monotonicity

Latency ≤ 4
#Proc ≤ 2
Not Possible

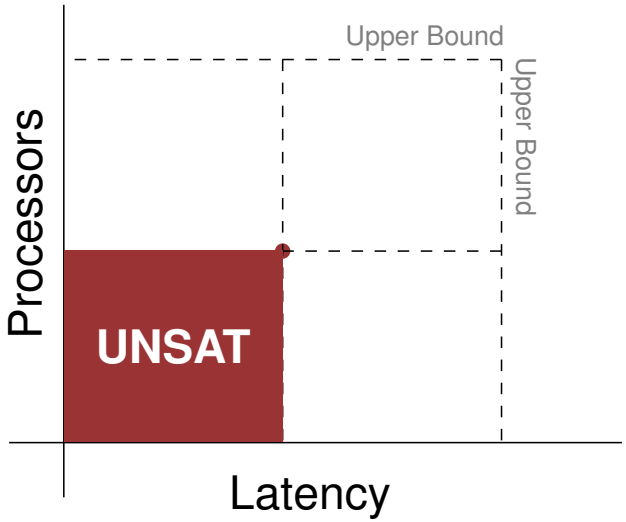
Latency = 2
#Proc = 1
Also Not Possible



Problem Monotonicity

Latency ≤ 4
#Proc ≤ 2
Not Possible

Latency = 2
#Proc = 1
Also Not Possible

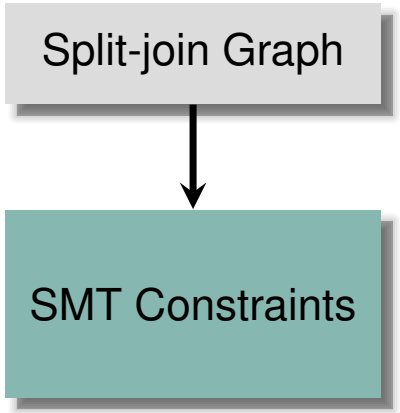


Design Space Exploration

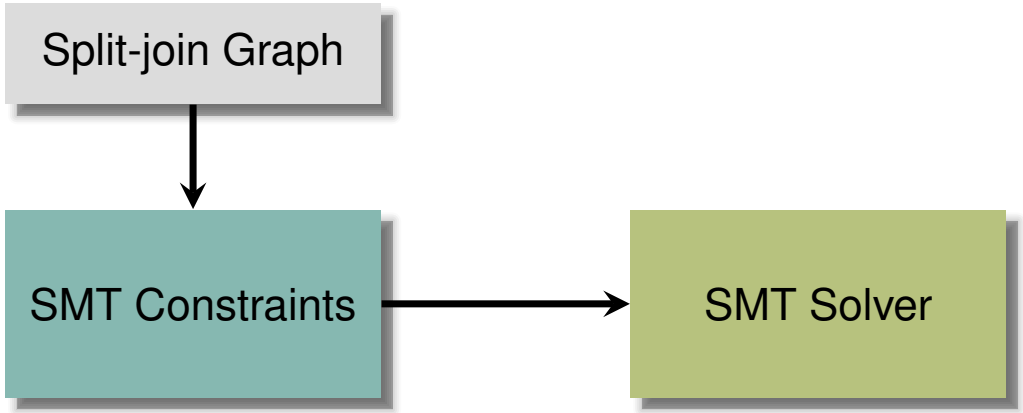
Design Space Exploration

Split-join Graph

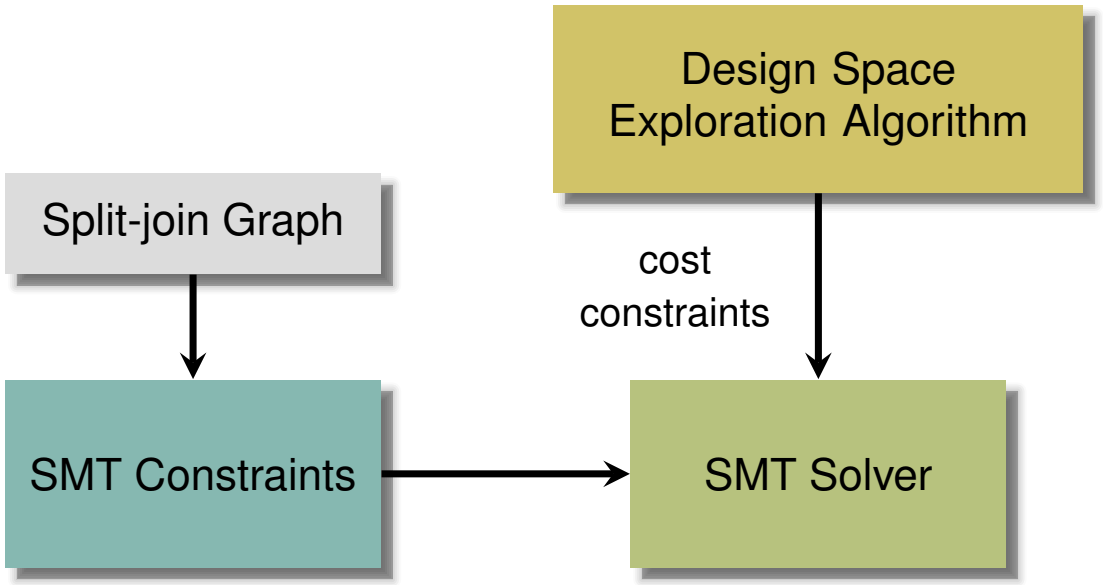
Design Space Exploration



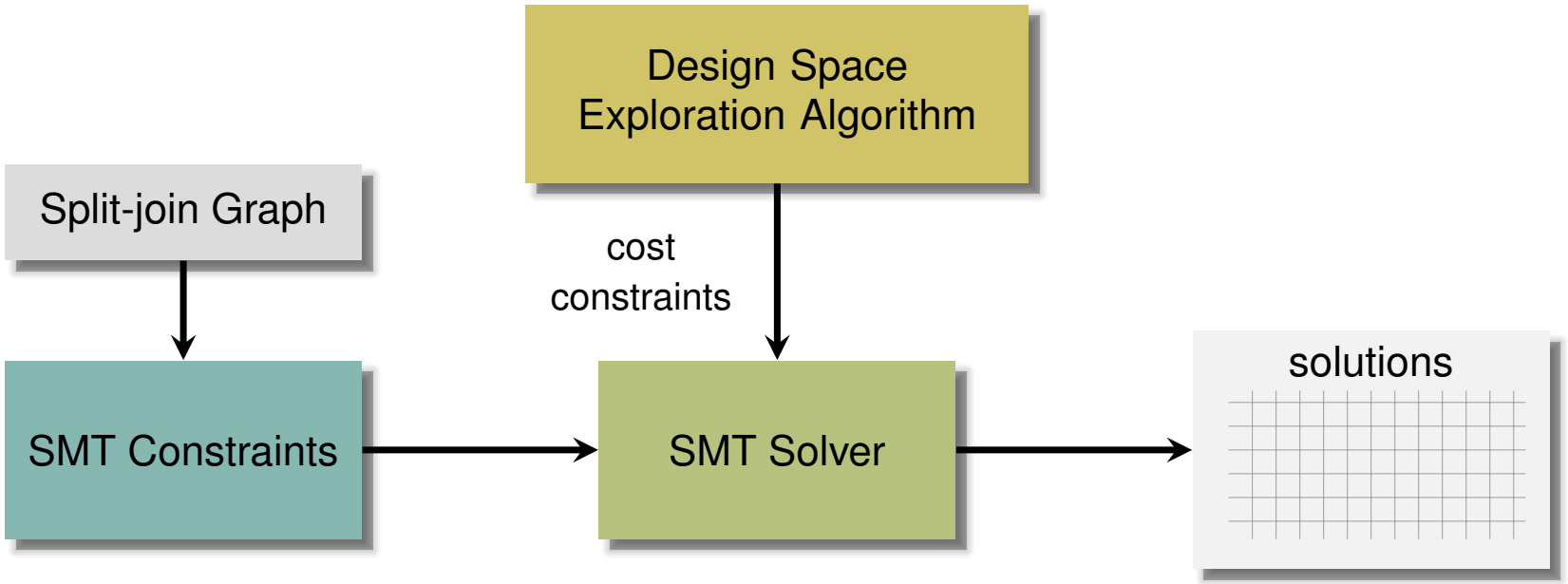
Design Space Exploration



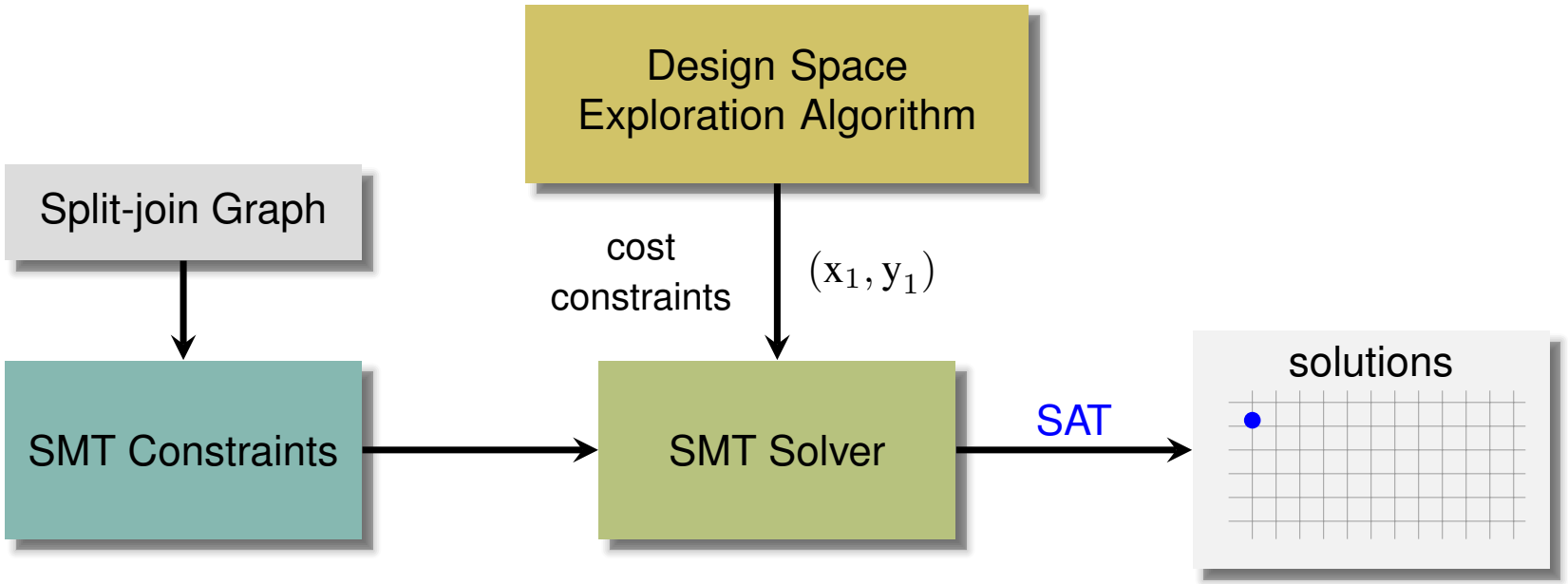
Design Space Exploration



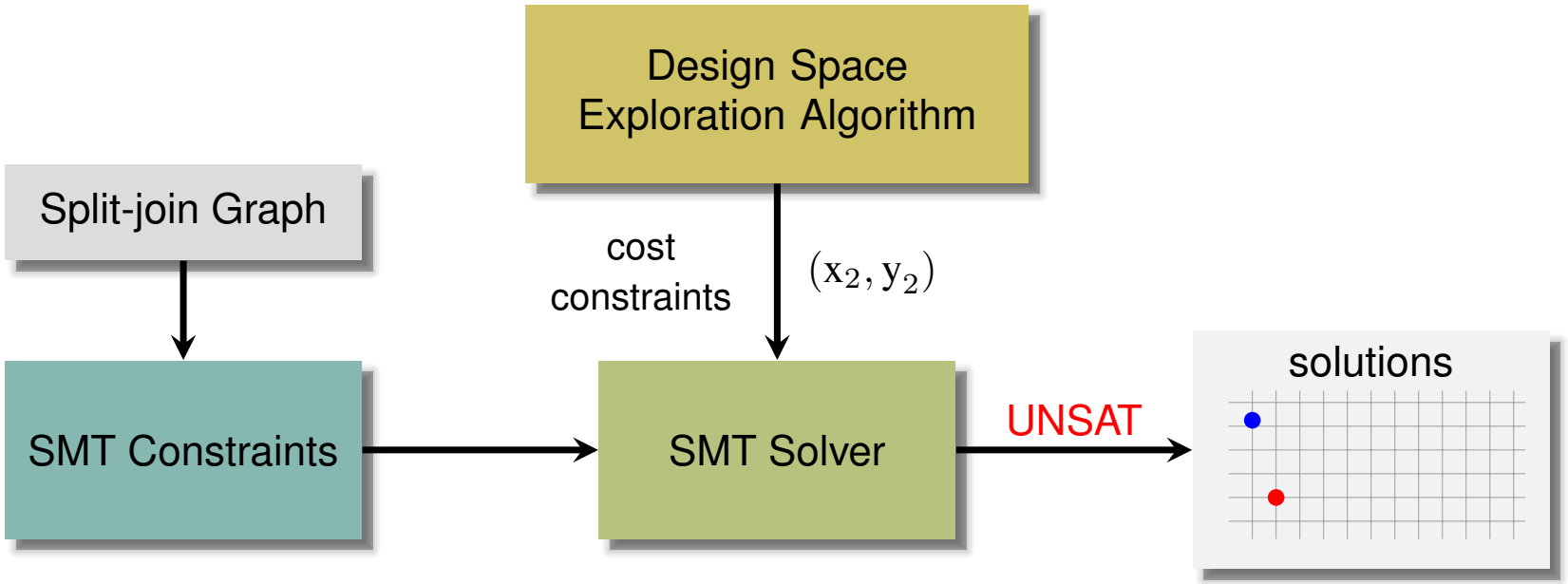
Design Space Exploration



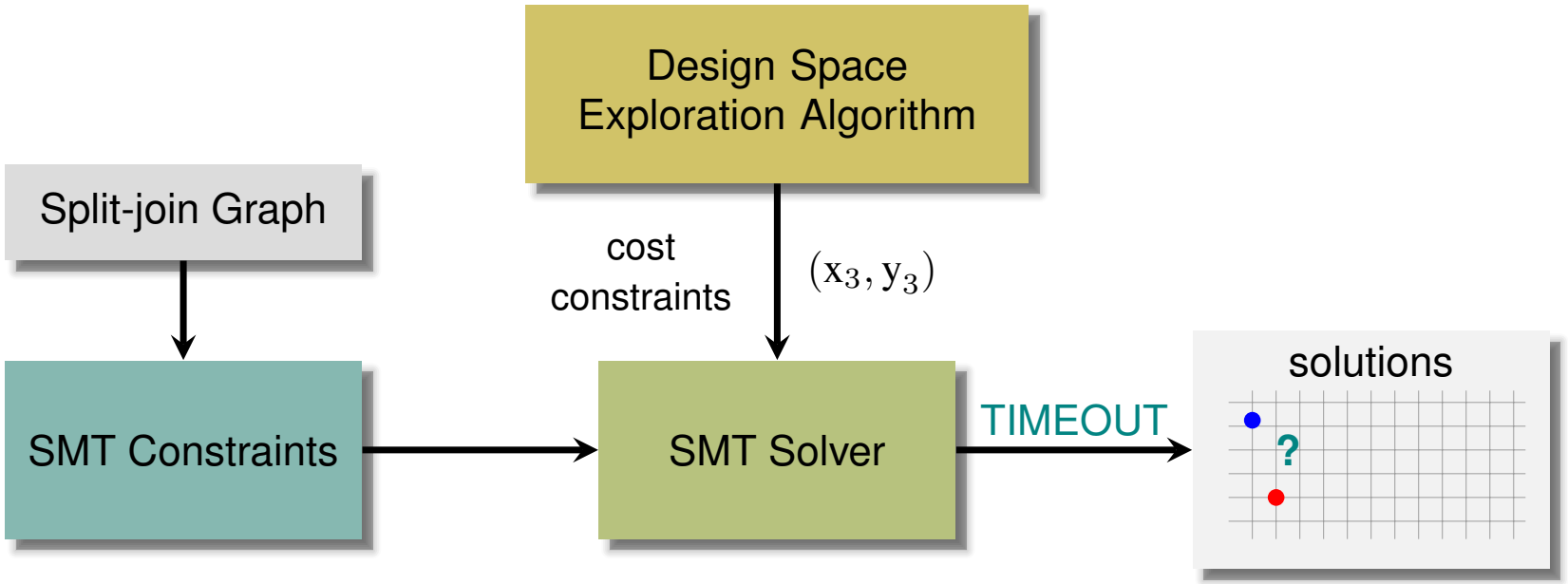
Design Space Exploration



Design Space Exploration

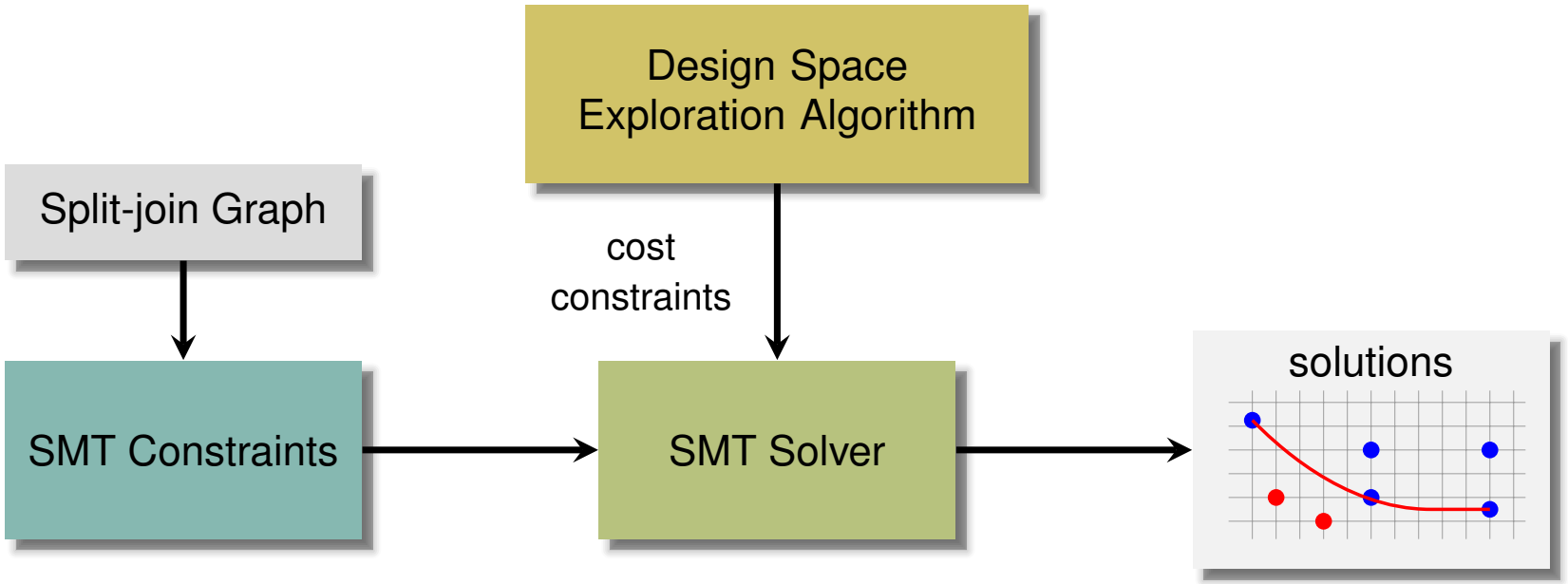


Design Space Exploration



Timeout:
Cannot decide SAT / UNSAT in a given **TIME-BUDGET**.

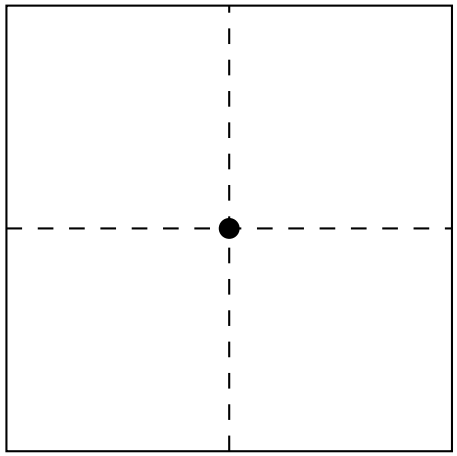
Design Space Exploration



Exploration Algorithm

Exploration Algorithm

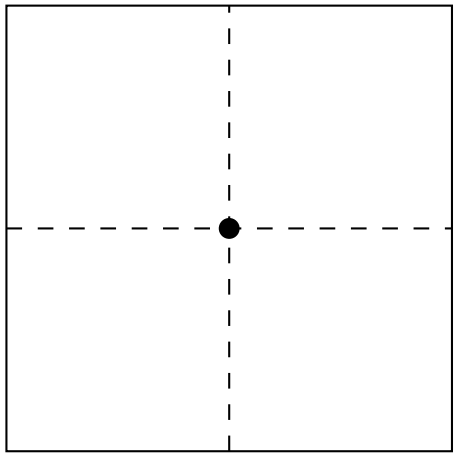
- **Divide cost space** using grids



sat points unsat points not yet explored points

Exploration Algorithm

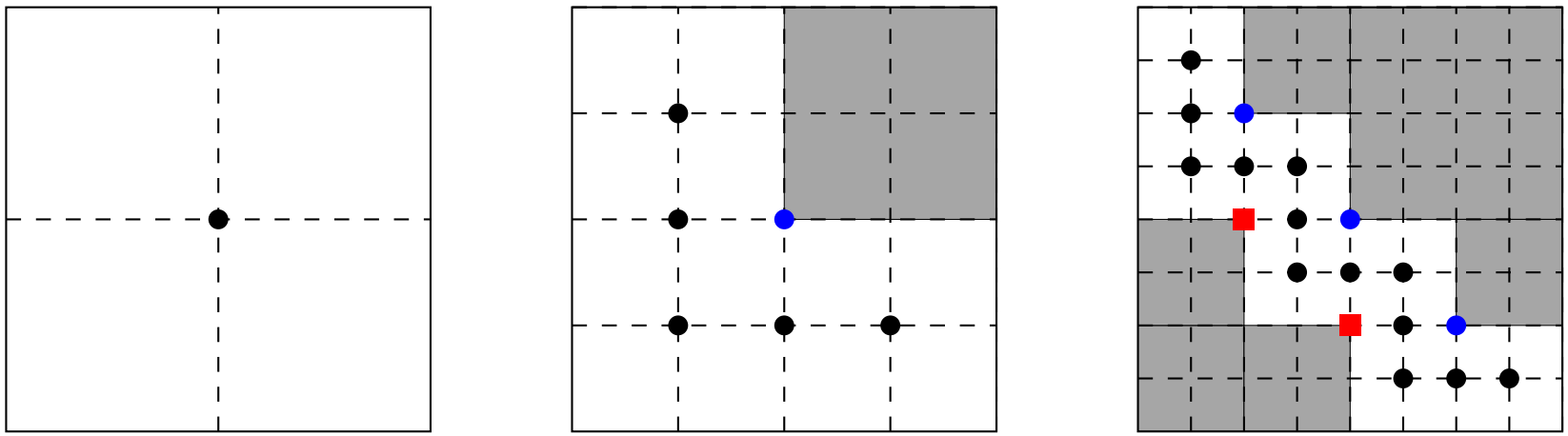
- **Divide cost space** using grids
- **One SMT query** per point on the grid



sat points unsat points not yet explored points

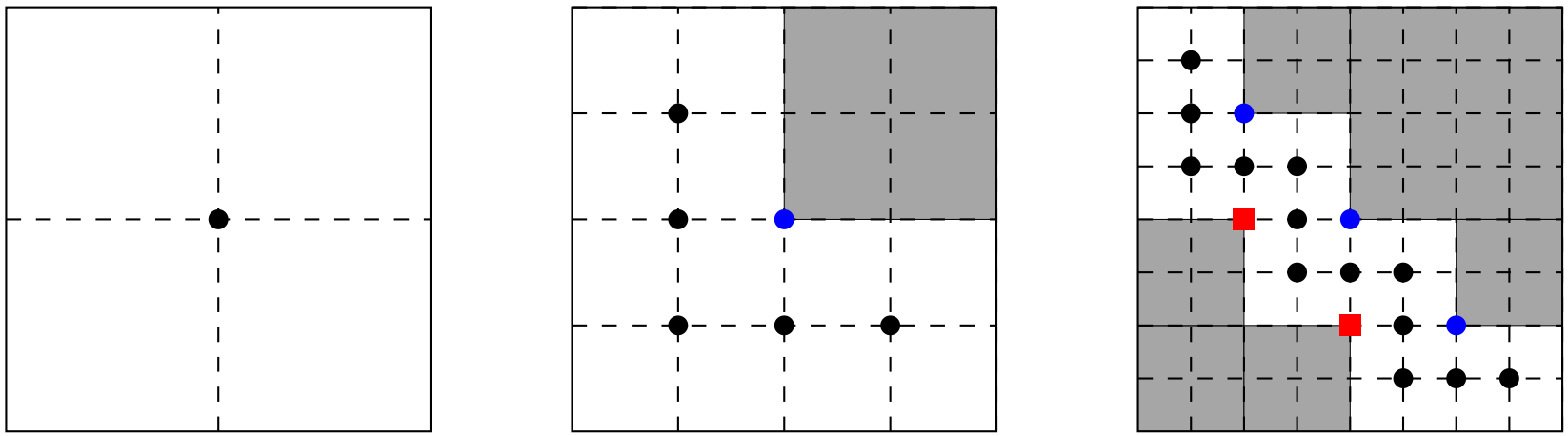
Exploration Algorithm

- **Divide cost space** using grids
- **One SMT query** per point on the grid
- **Finer** grid after every iteration



Exploration Algorithm

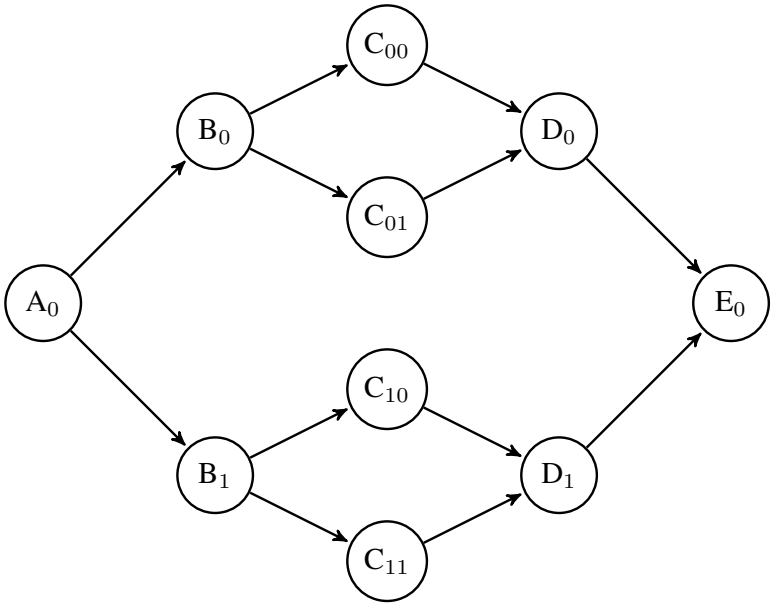
- **Divide cost space** using grids
- **One SMT query** per point on the grid
- **Finer** grid after every iteration
- Don't query in **known area**



Overview

Task Symmetry

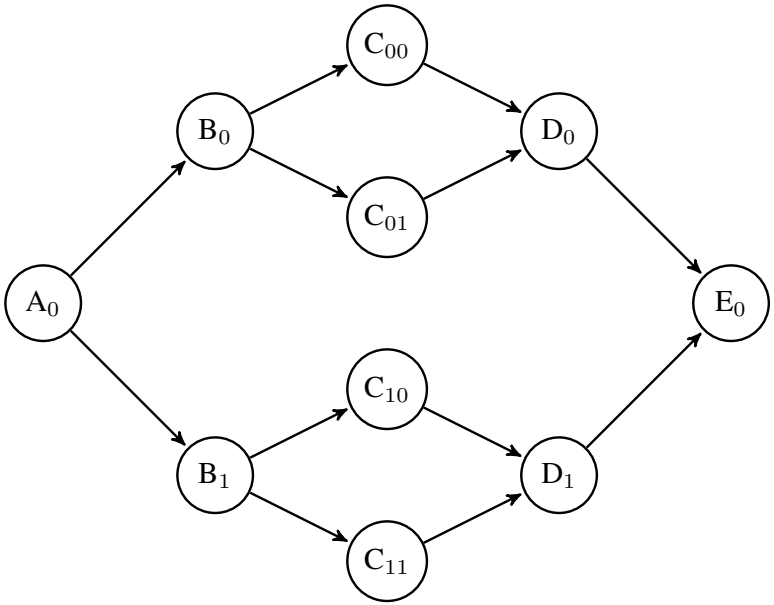
Task Symmetry



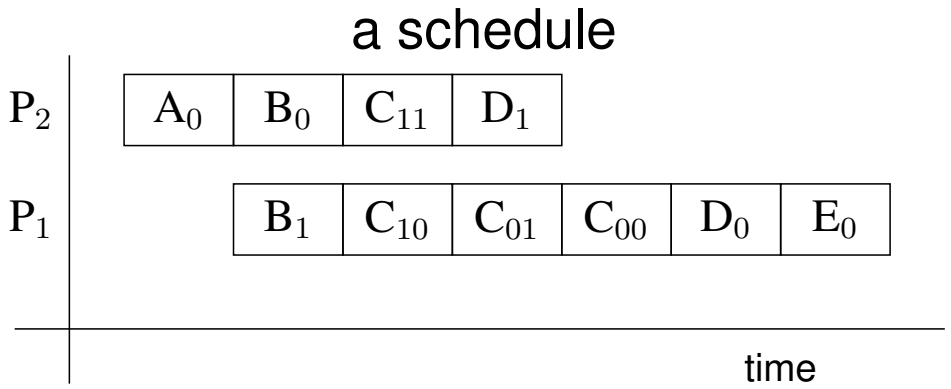
task graph

- all instances of actor C are similar (**symmetric**)

Task Symmetry

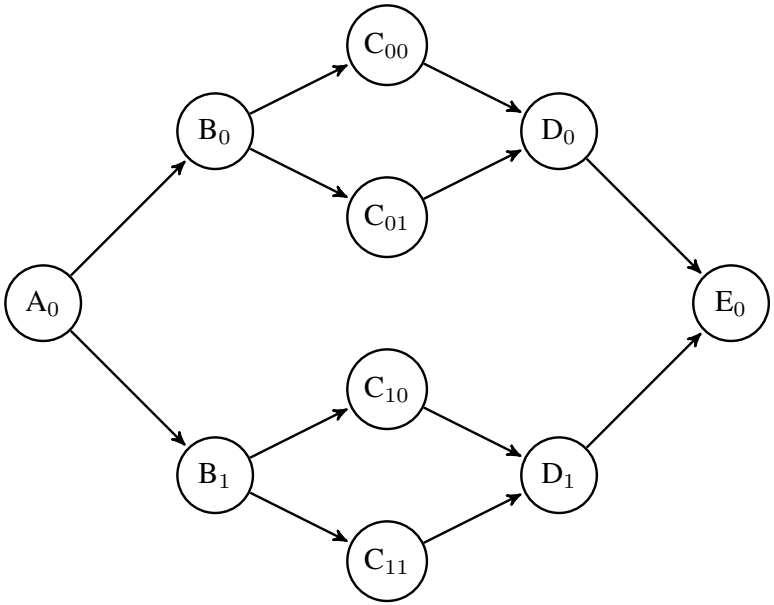


task graph



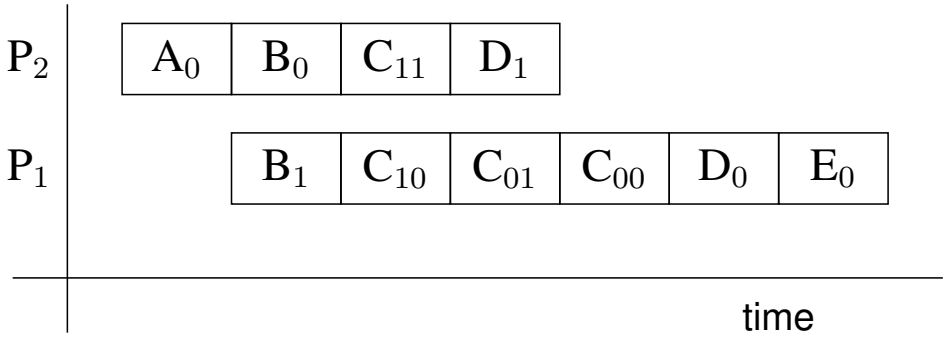
- all instances of actor C are similar (**symmetric**)

Task Symmetry

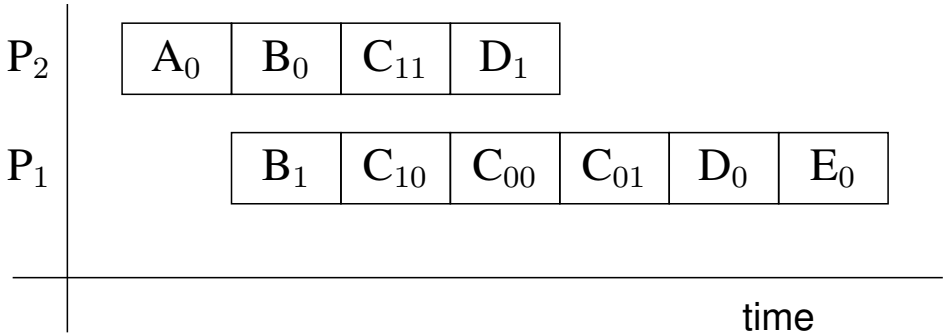


task graph

a schedule

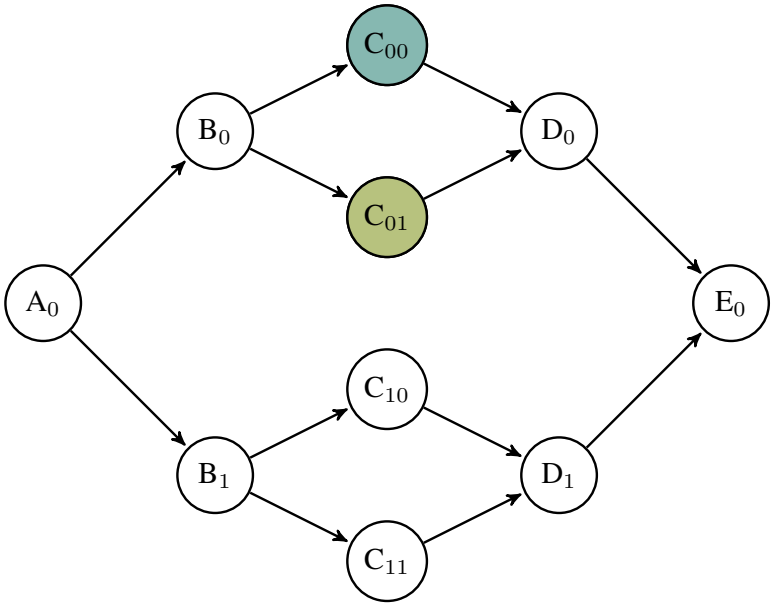


a **permuted** schedule



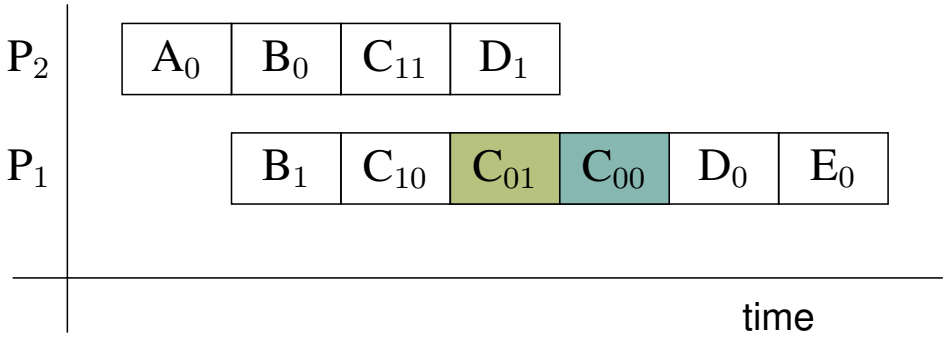
- all instances of actor C are similar (**symmetric**)

Task Symmetry

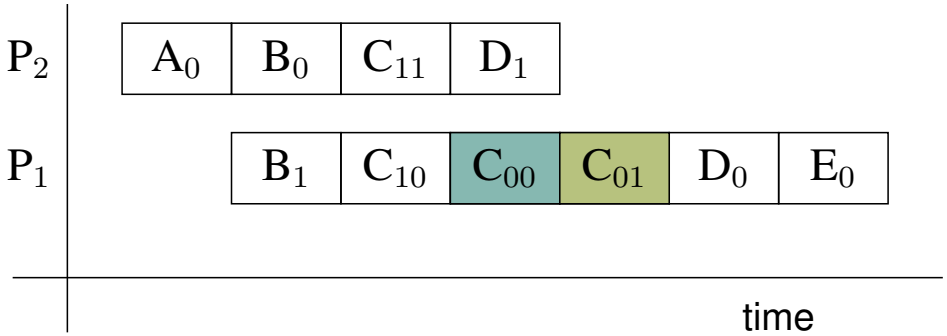


task graph

a schedule

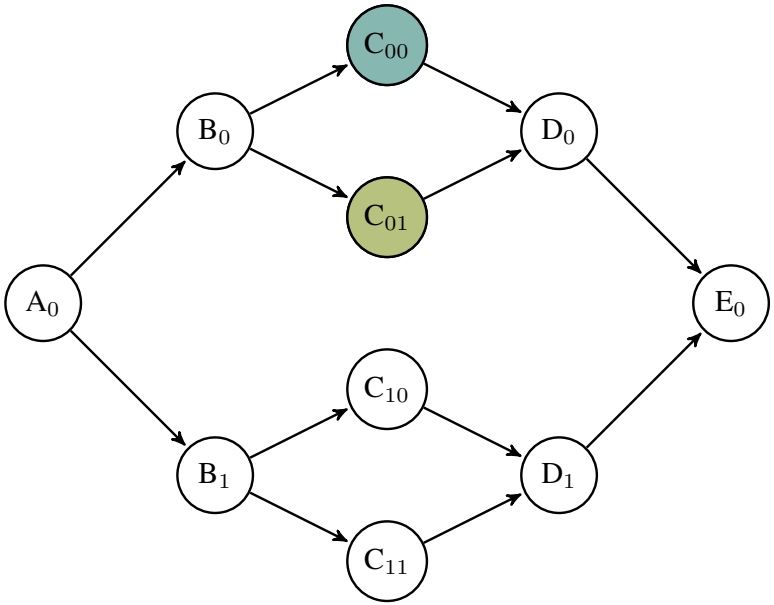


a **permuted** schedule

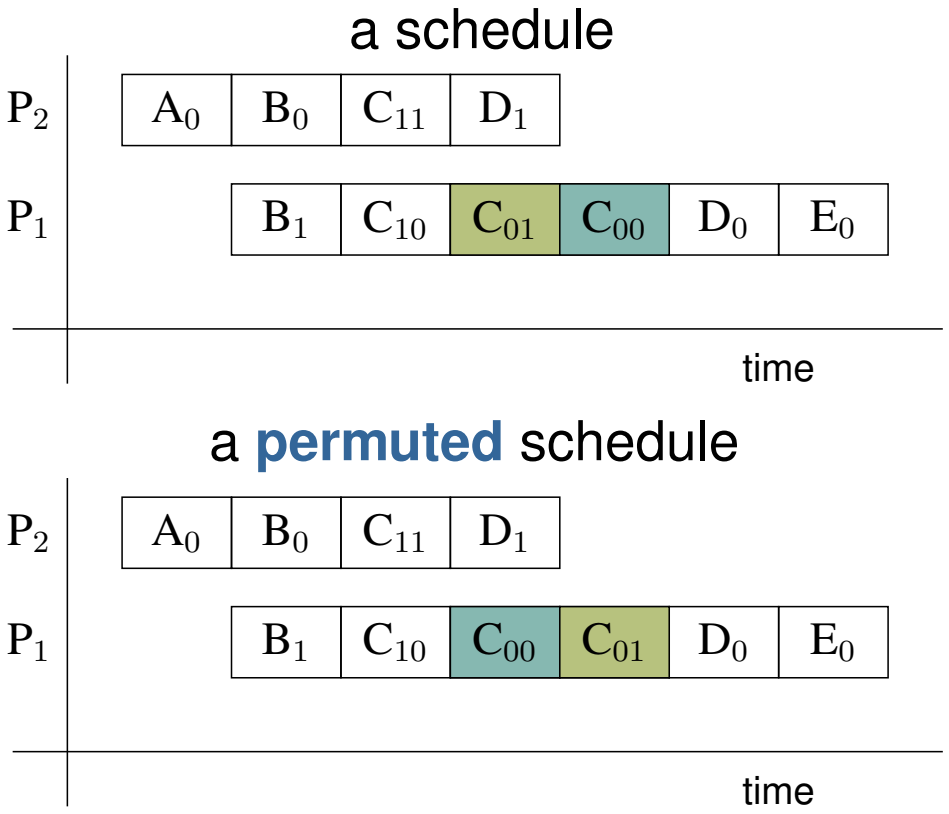


- all instances of actor C are similar (**symmetric**)

Task Symmetry

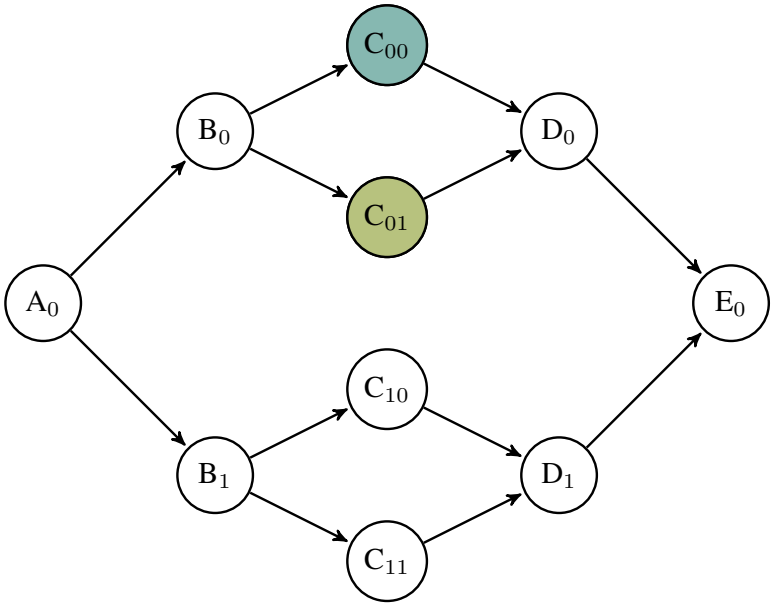


task graph

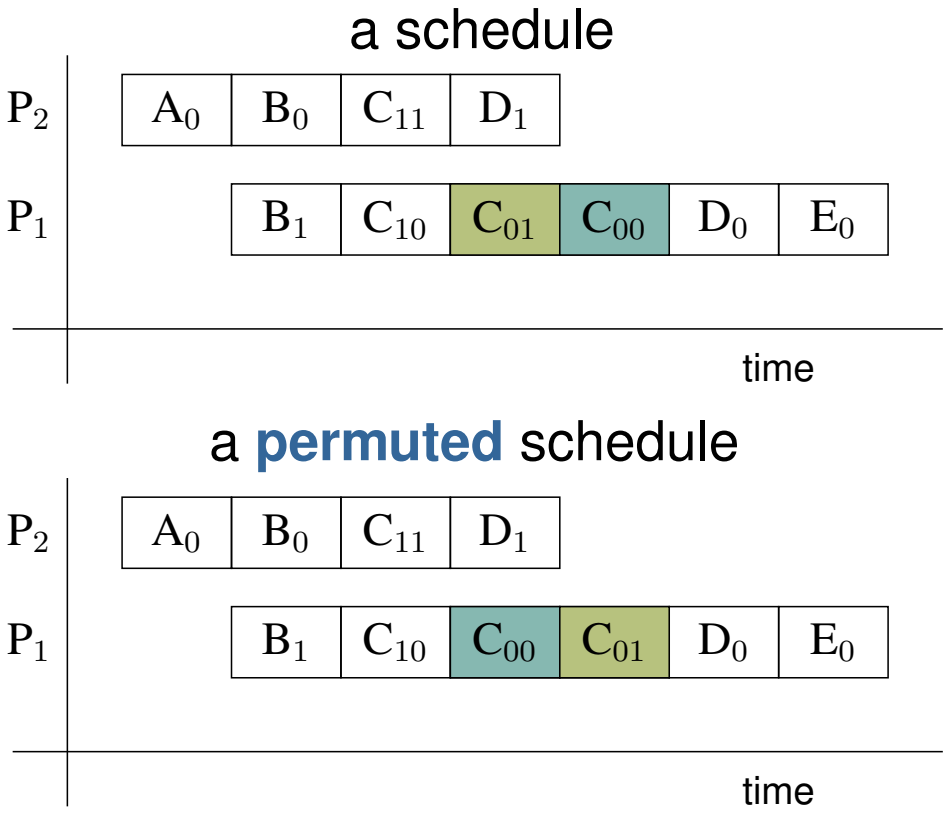


- all instances of actor C are similar (**symmetric**)
- **No change** in latency !

Task Symmetry

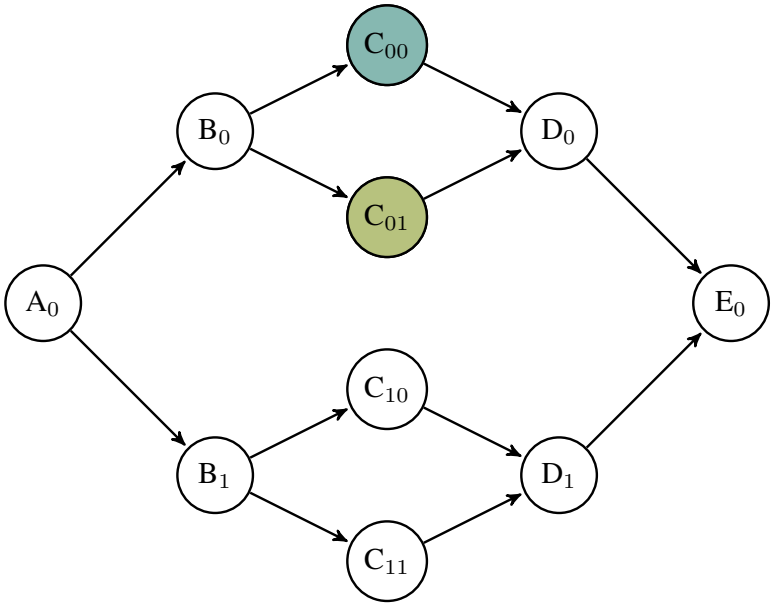


task graph

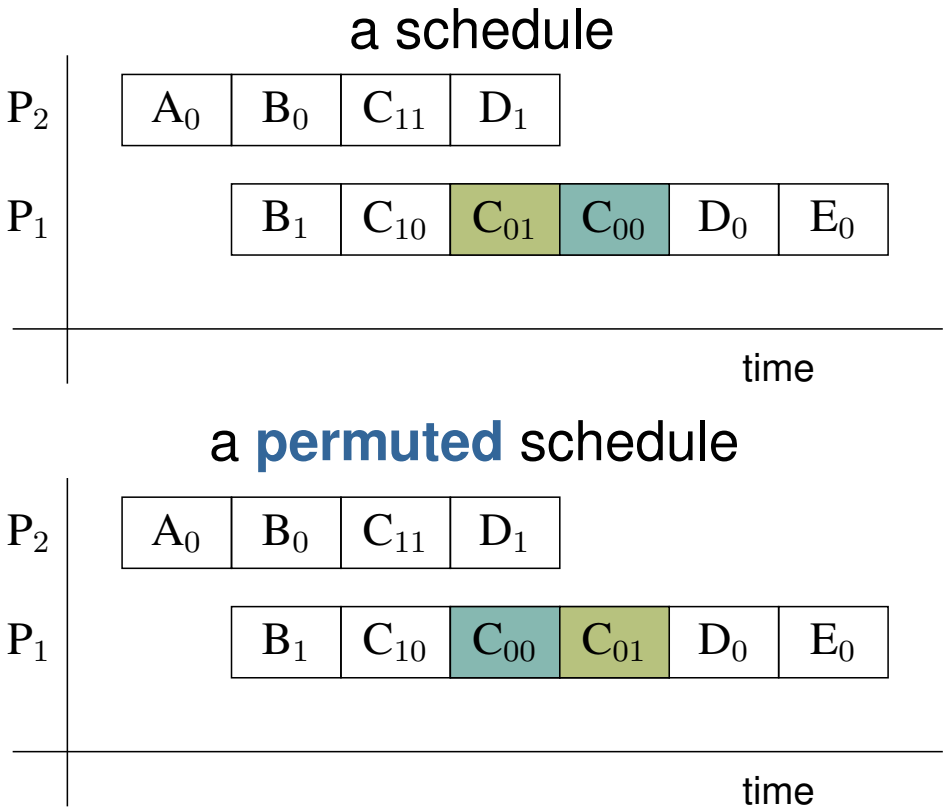


- all instances of actor C are similar (**symmetric**)
- **No change** in latency !
- **Huge number** of such symmetric solutions

Task Symmetry

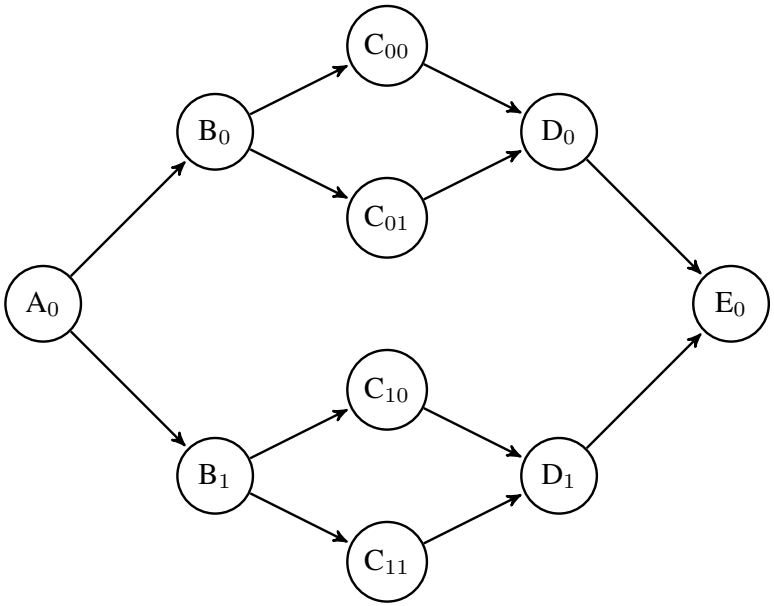


task graph

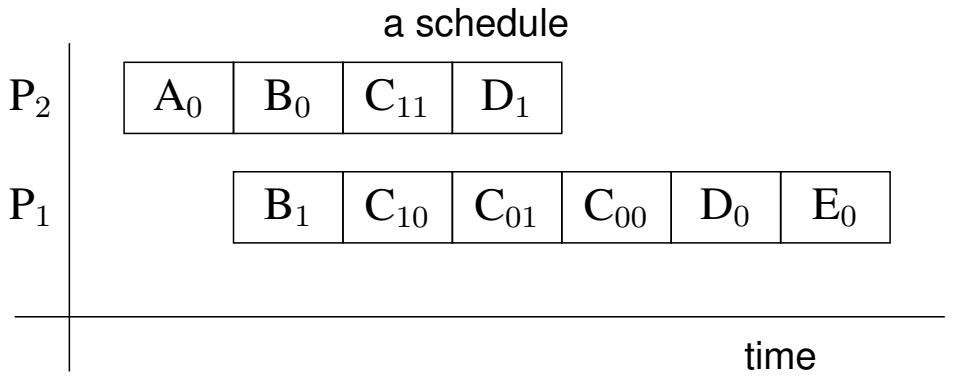


- all instances of actor C are similar (**symmetric**)
- **No change** in latency !
- **Huge number** of such symmetric solutions
- Add constraints to **eliminate** all but one

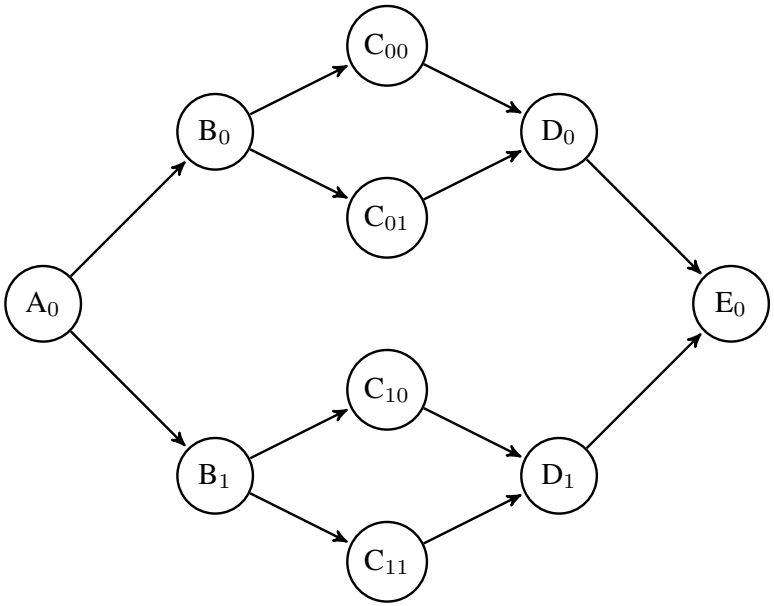
Task Symmetry



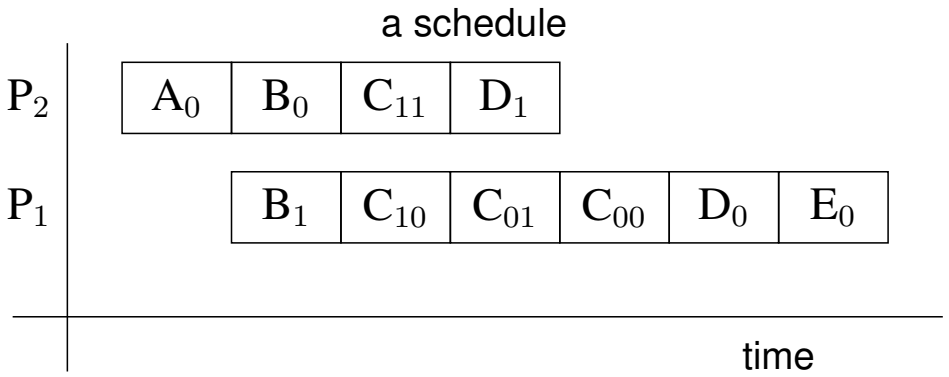
task graph



Task Symmetry

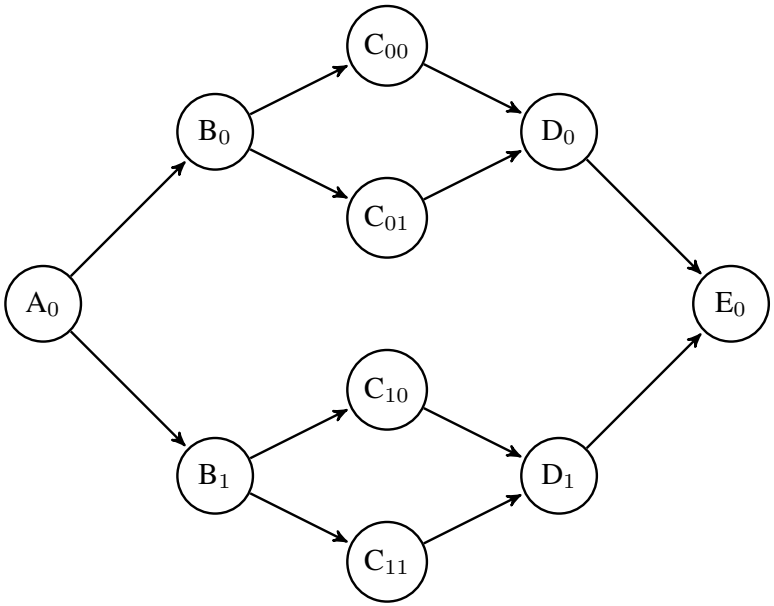


task graph

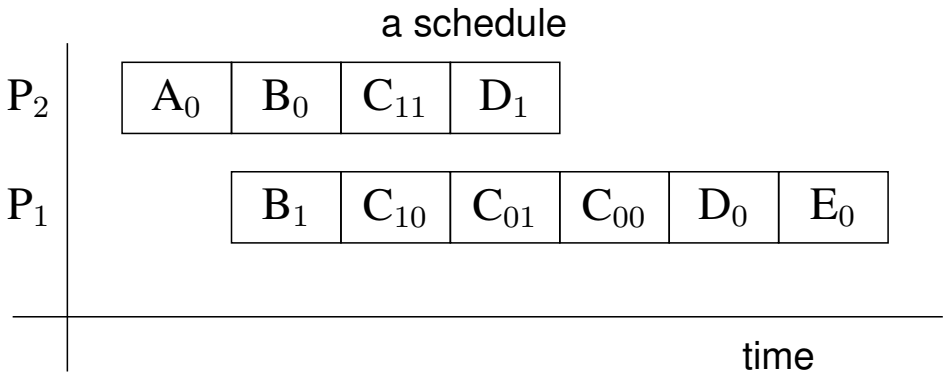


- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$

Task Symmetry

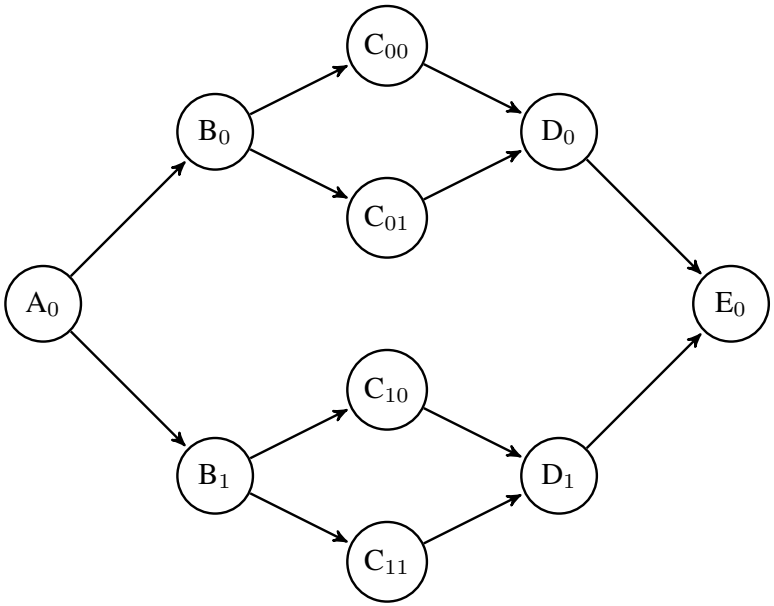


task graph

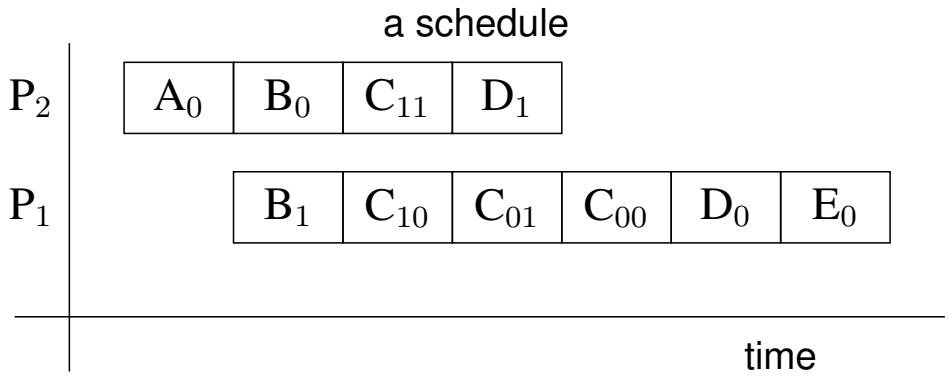


- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$
- enforce lexicographic order in schedule: $s(u) \leq s(u')$ for $u \ll u'$

Task Symmetry

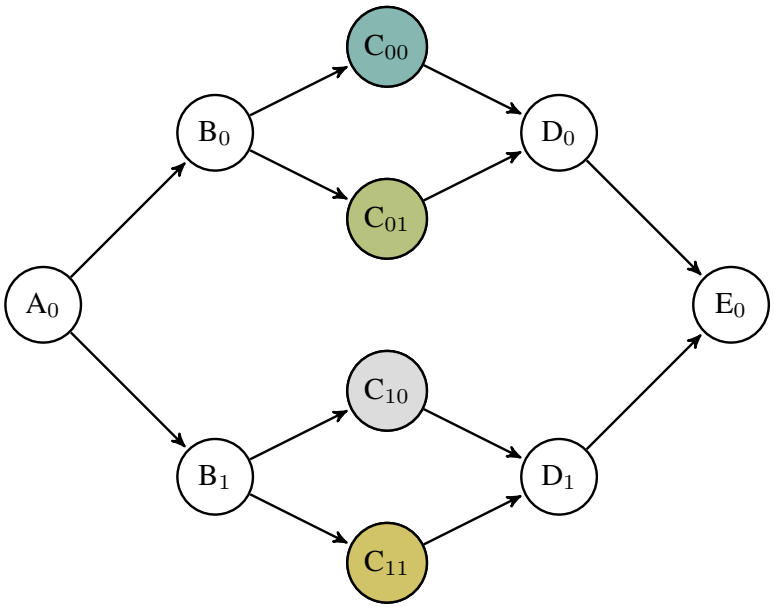


task graph



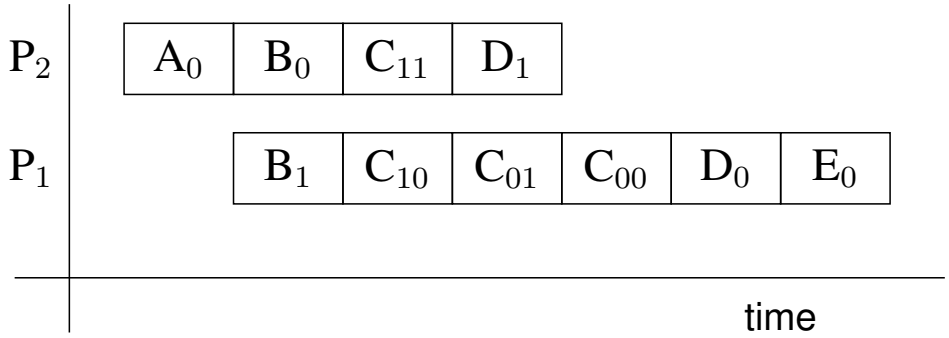
- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$
- enforce lexicographic order in schedule: $s(u) \leq s(u')$ for $u \ll u'$
- $s(C_{00}) \leq s(C_{01}) \leq s(C_{10}) \leq s(C_{11})$

Task Symmetry

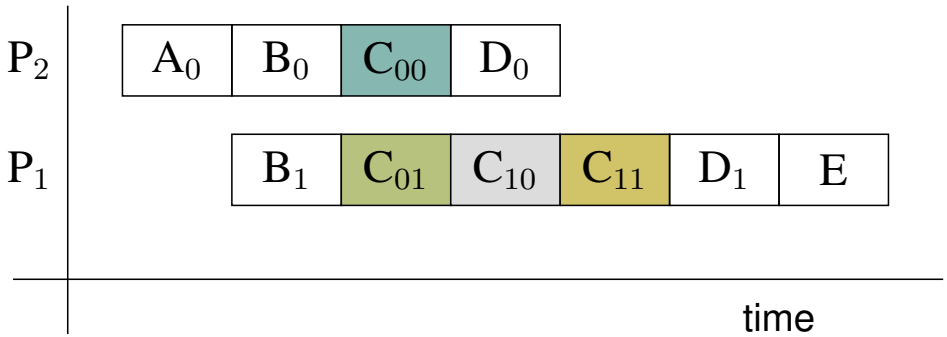


task graph

a schedule

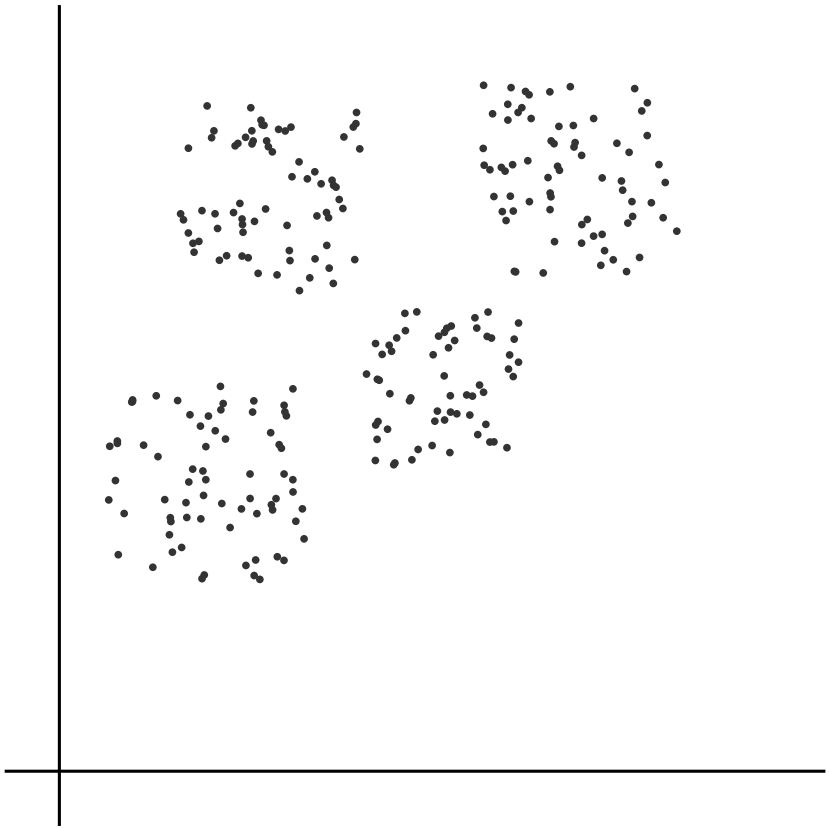


a **lexicographic** schedule

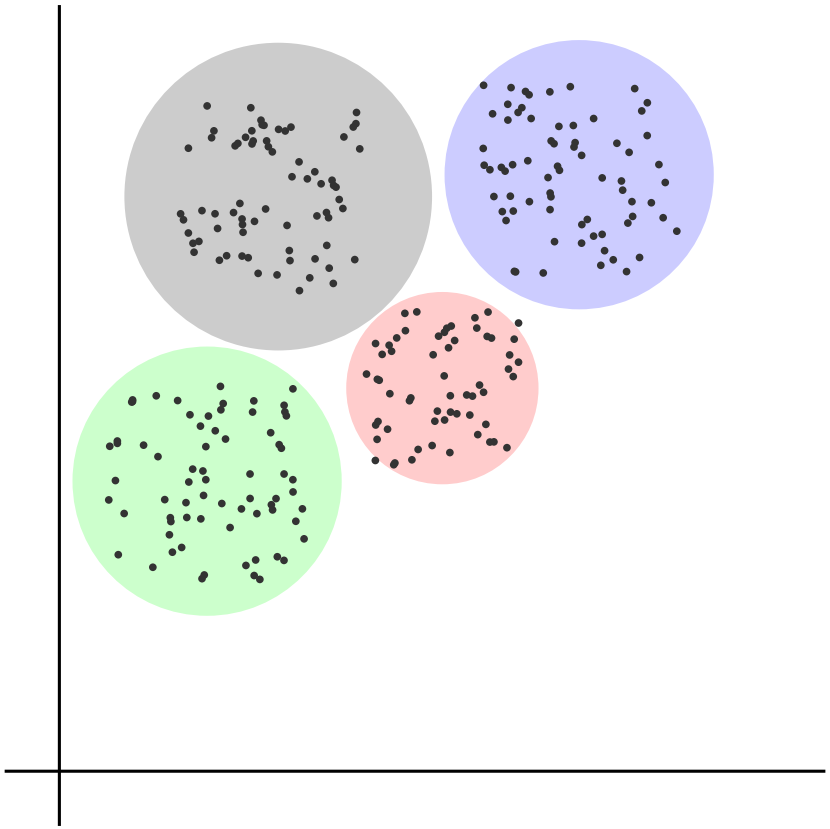


- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$
- enforce lexicographic order in schedule: $s(u) \leq s(u')$ for $u \ll u'$
- $s(C_{00}) \leq s(C_{01}) \leq s(C_{10}) \leq s(C_{11})$

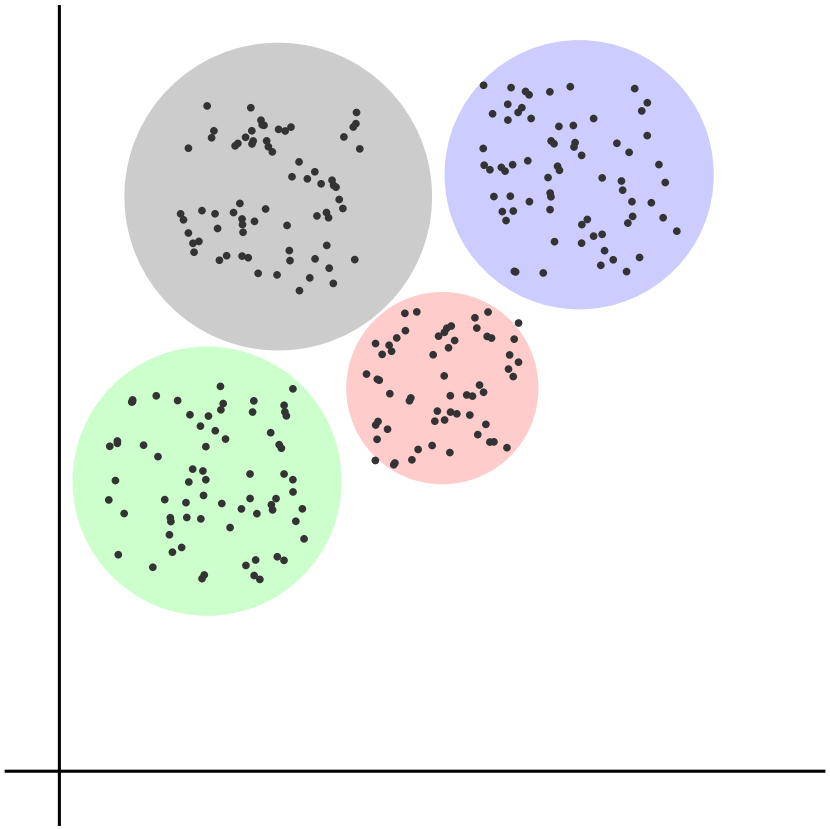
Task Symmetry : Theorem



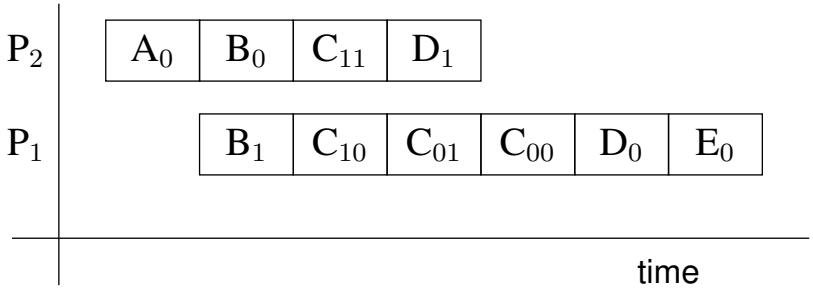
Task Symmetry : Theorem



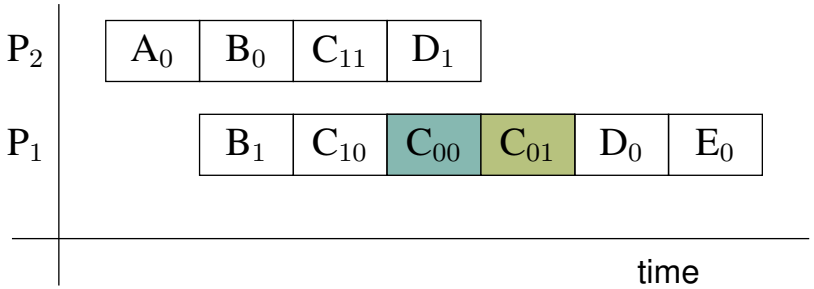
Task Symmetry : Theorem



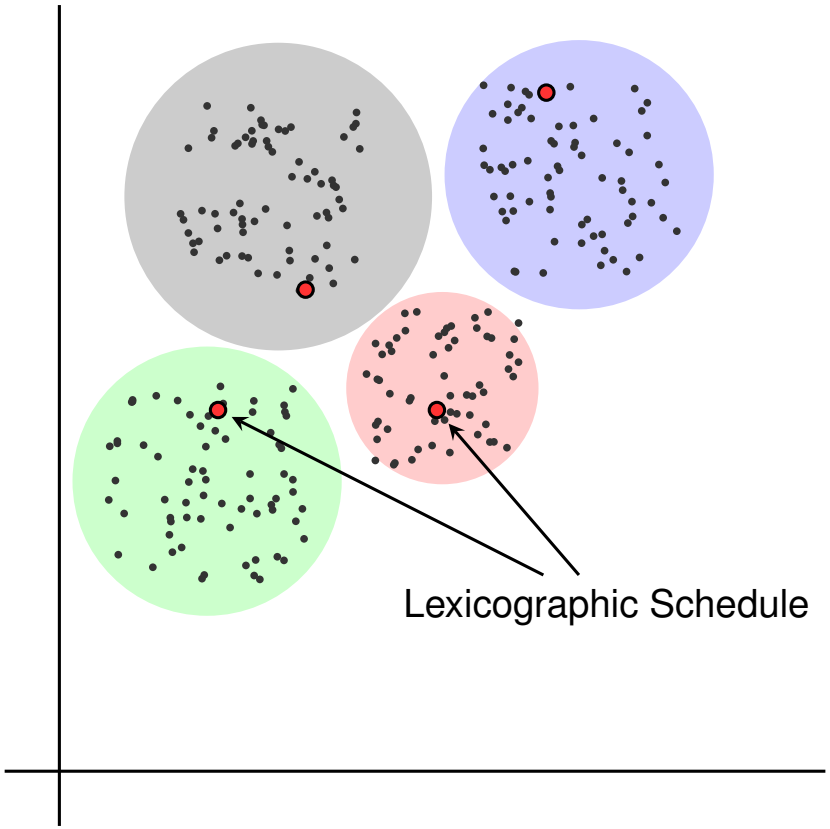
a schedule



a **permuted** schedule

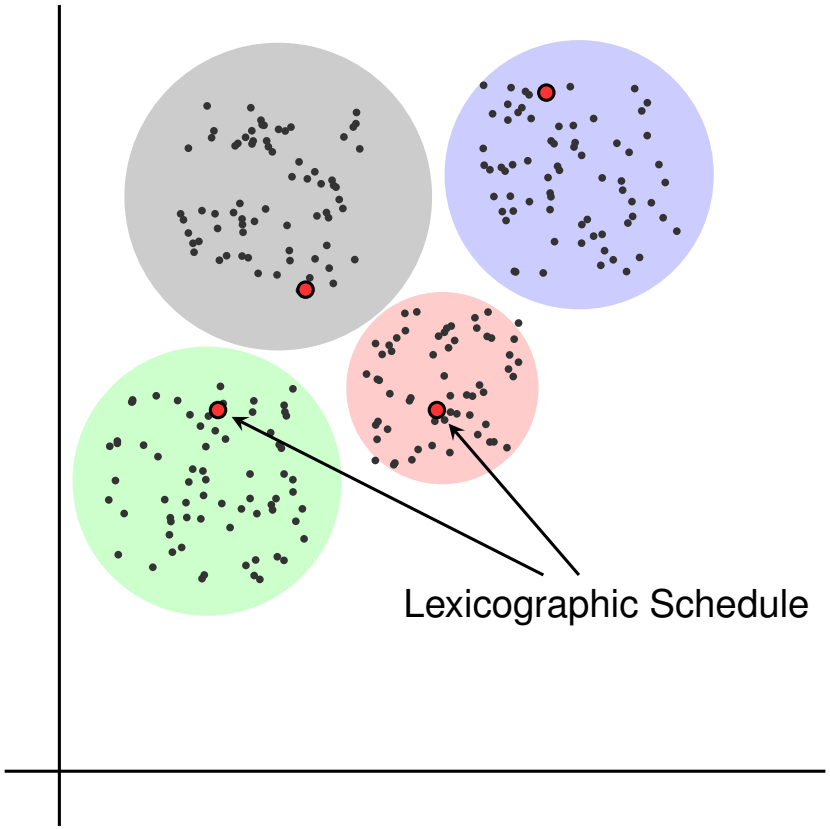


Task Symmetry : Theorem



- Theorem : Every group has a **lexicographic schedule**

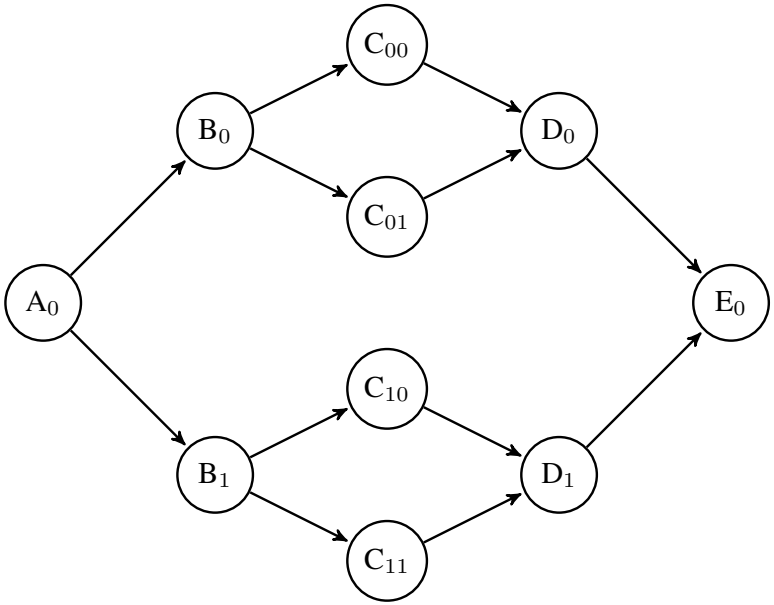
Task Symmetry : Theorem



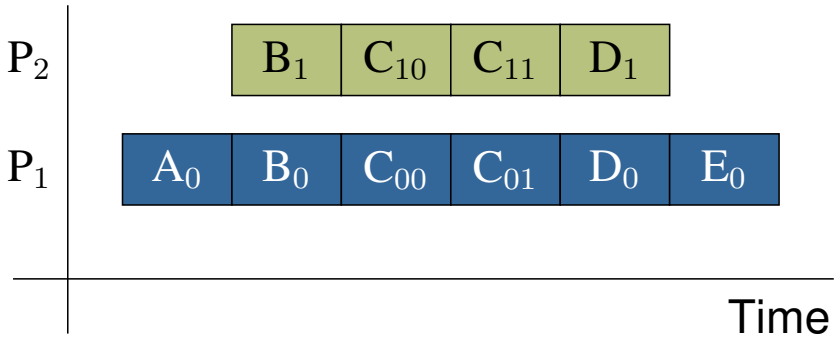
- Theorem : Every group has a **lexicographic schedule**
- Corollary : No feasible schedule is lost

Processor Symmetry

Processor Symmetry

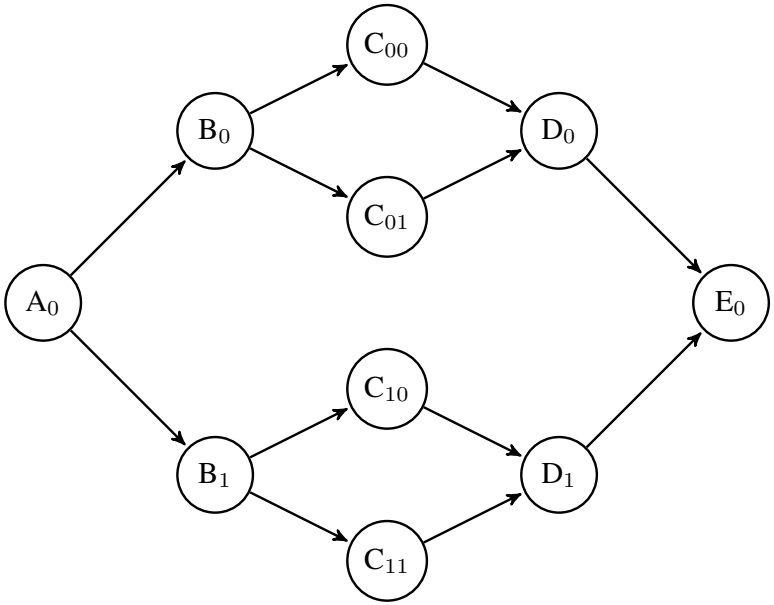


task graph

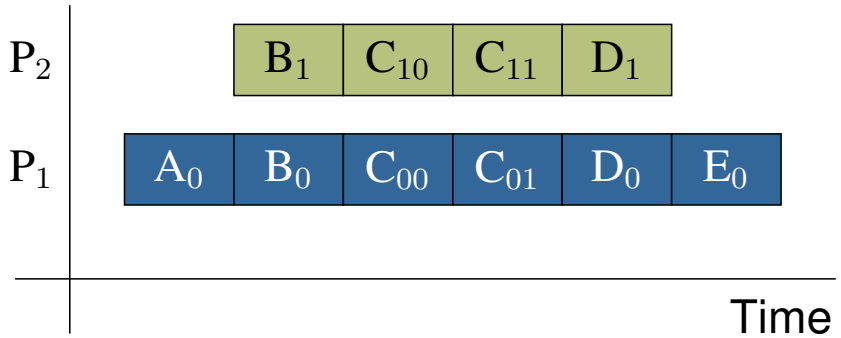


schedule

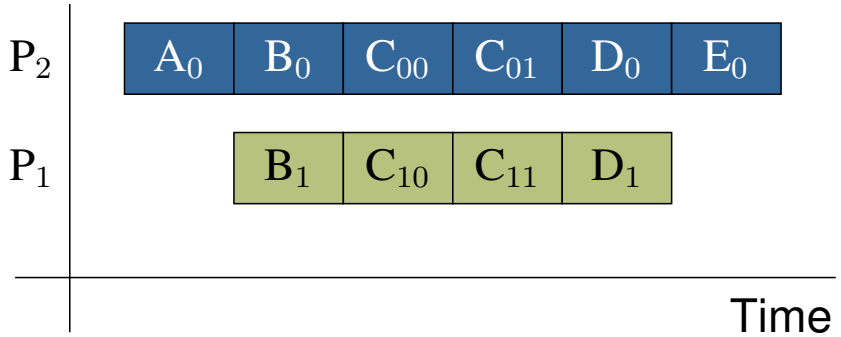
Processor Symmetry



task graph

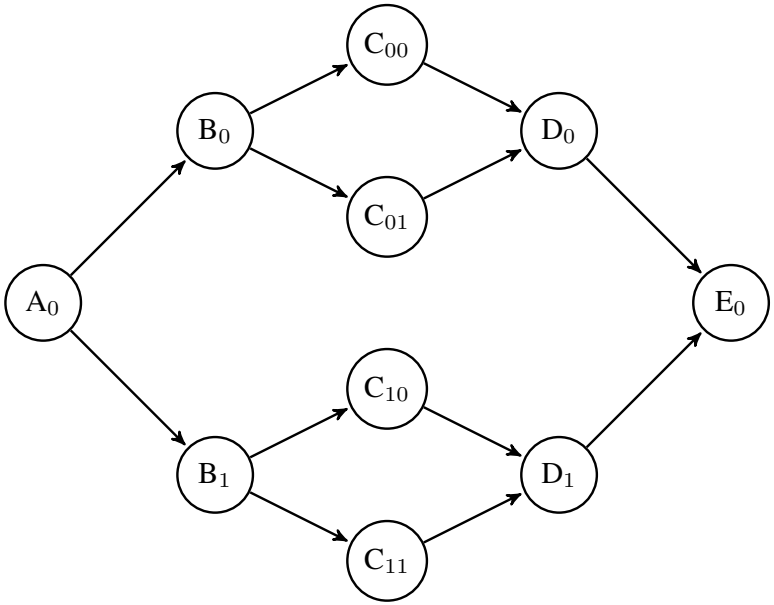


schedule

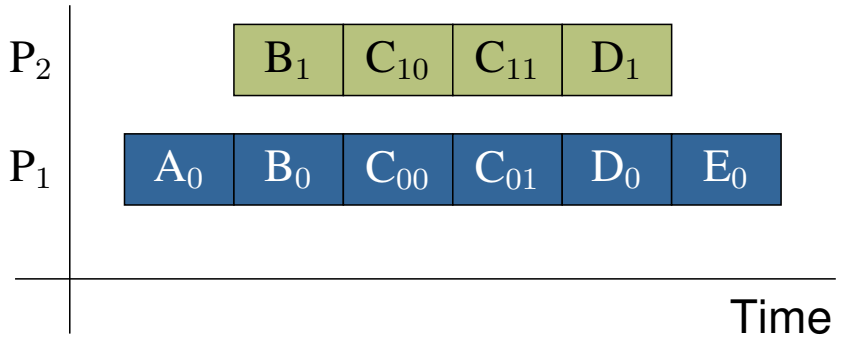


swapped P_1 and P_2

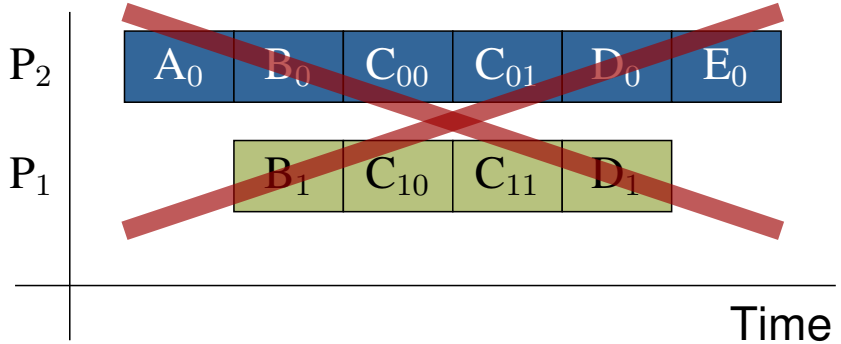
Processor Symmetry



task graph



schedule

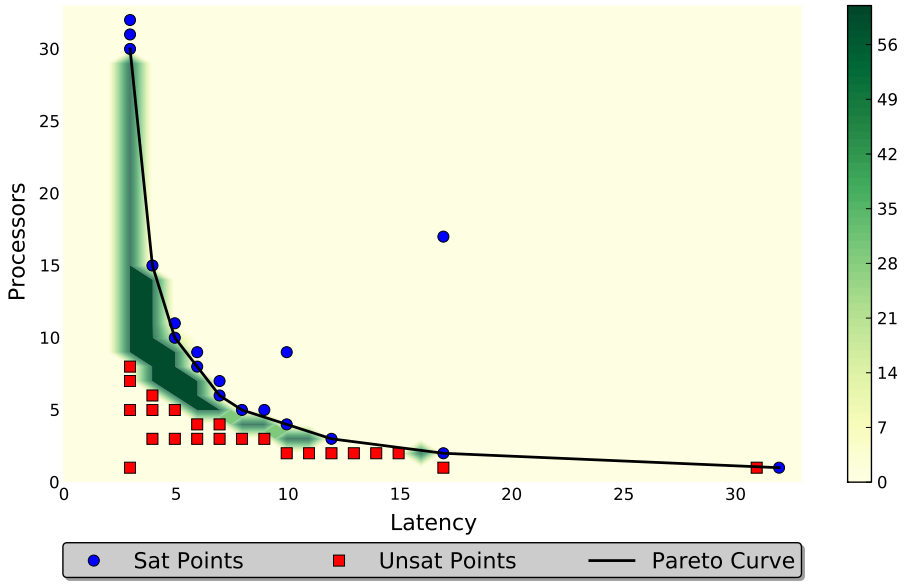


swapped P_1 and P_2

Pareto Exploration

Exploration : Processors vs Latency $\alpha = 30$

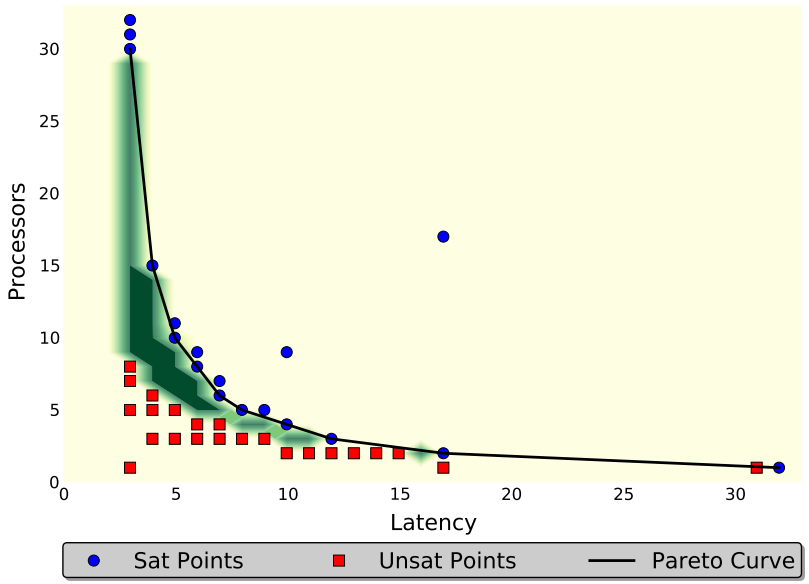
Pareto Exploration



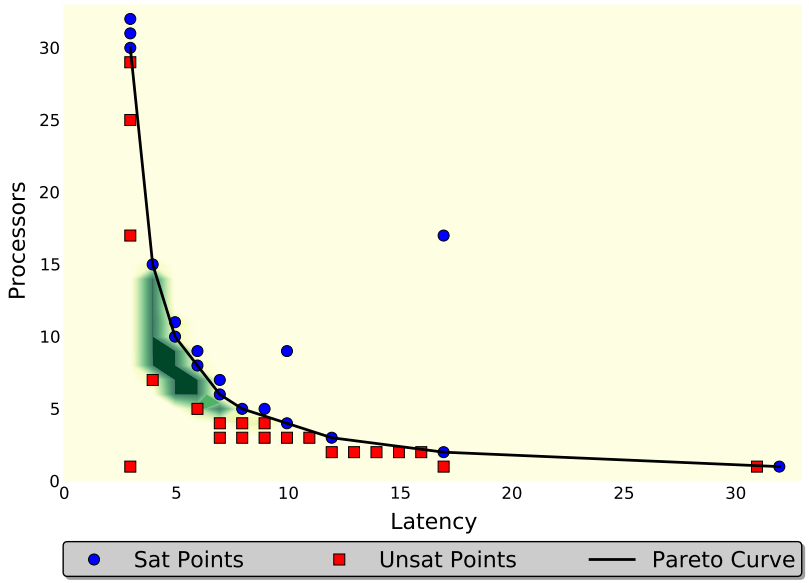
without symmetry breaking

Exploration : Processors vs Latency $\alpha = 30$

Pareto Exploration



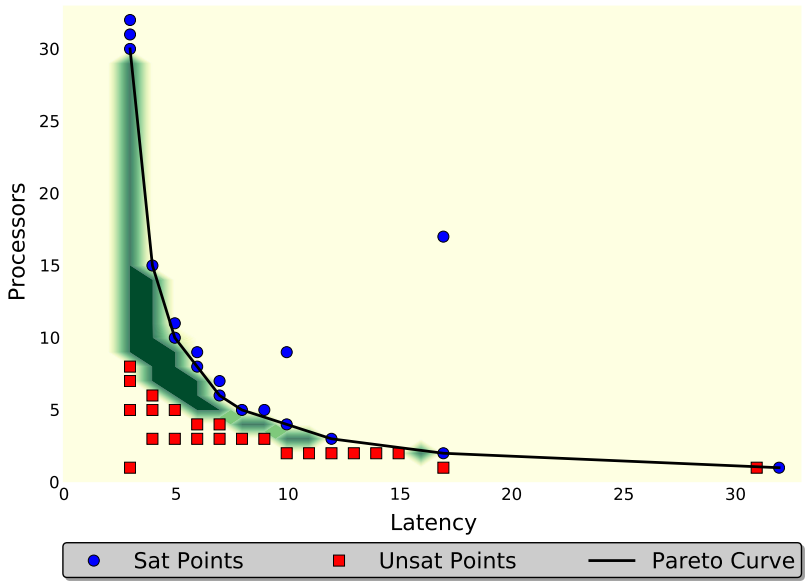
without symmetry breaking



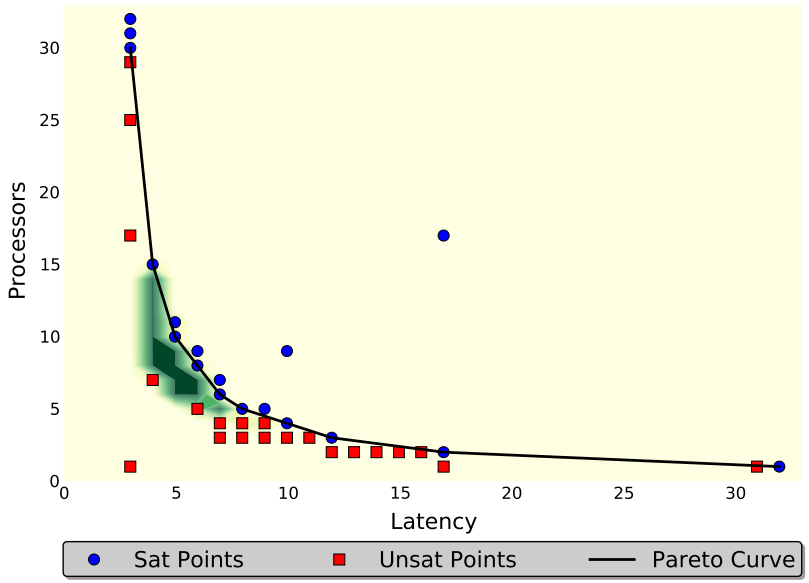
with symmetry breaking

Exploration : Processors vs Latency $\alpha = 30$

Pareto Exploration



without symmetry breaking



with symmetry breaking

Exploration : Processors vs Latency $\alpha = 30$

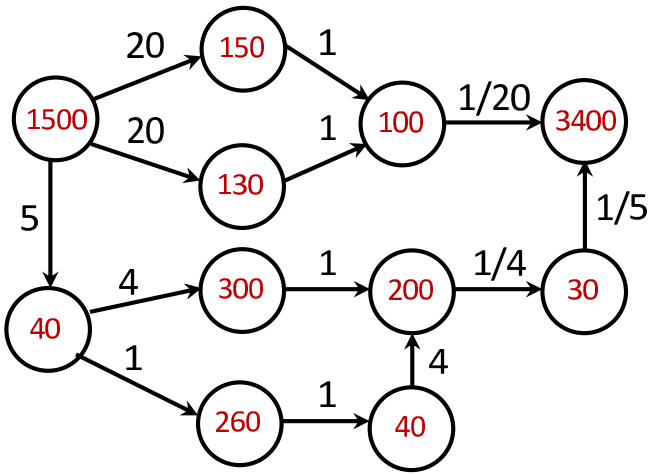
Solver Performance

- Timeouts **reduce** !
- The gap between SAT and UNSAT points **is smaller**.

Video Decoder

3D cost space $(\mathbf{C}_L, \mathbf{C}_P, \mathbf{C}_B)$ exploration, C_B - total buffer size

MPEG video decoder:

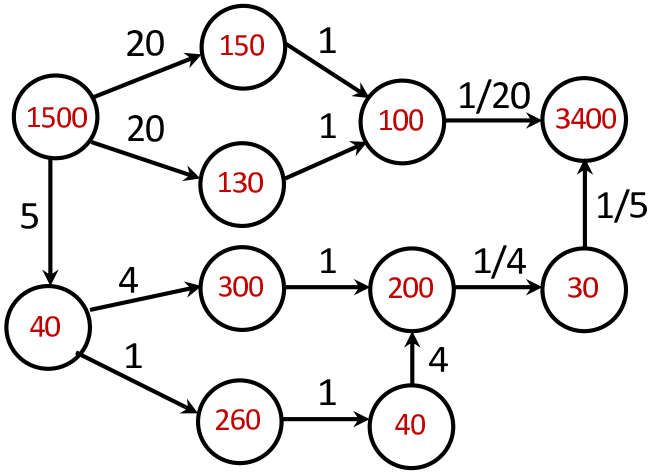


122 Tasks

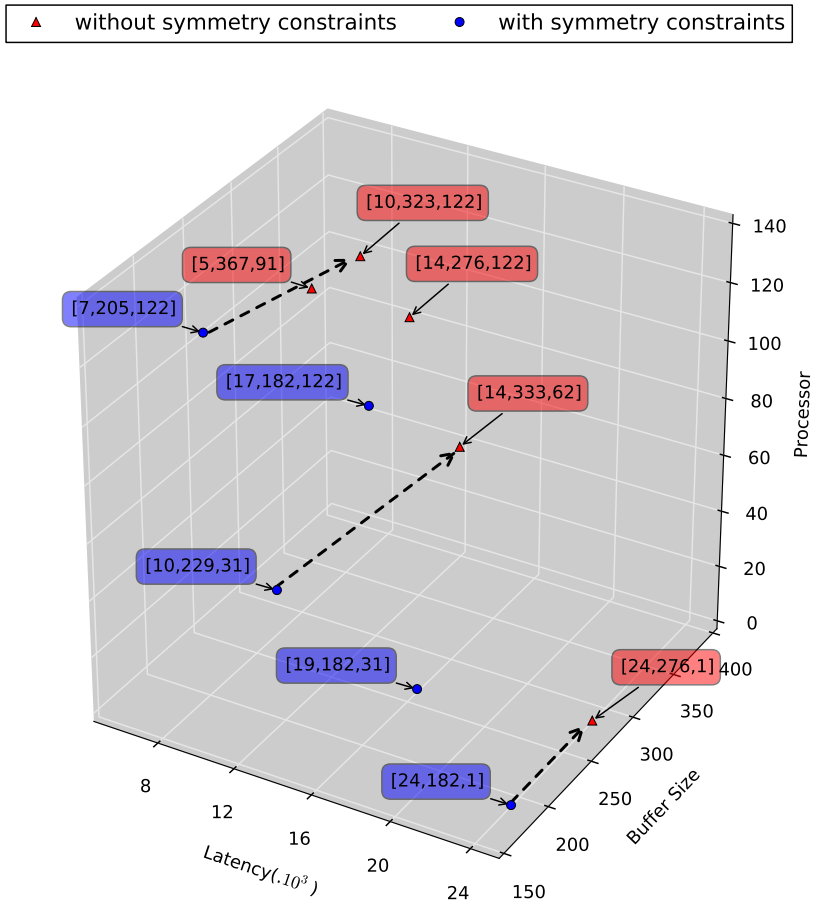
Video Decoder

3D cost space (C_L, C_P, C_B) exploration, C_B - total buffer size

MPEG video decoder:



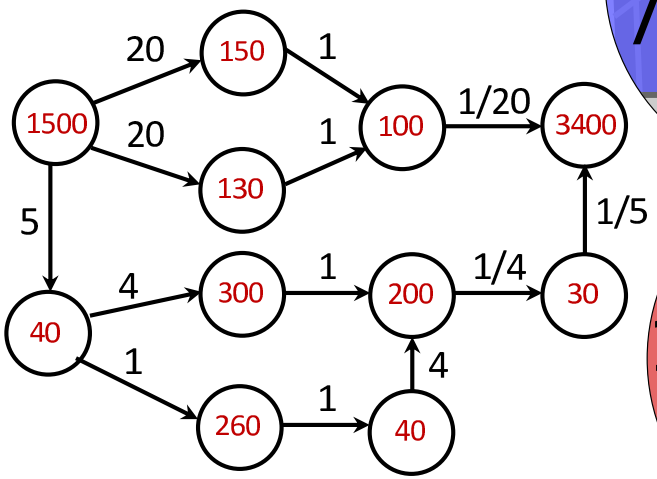
122 Tasks

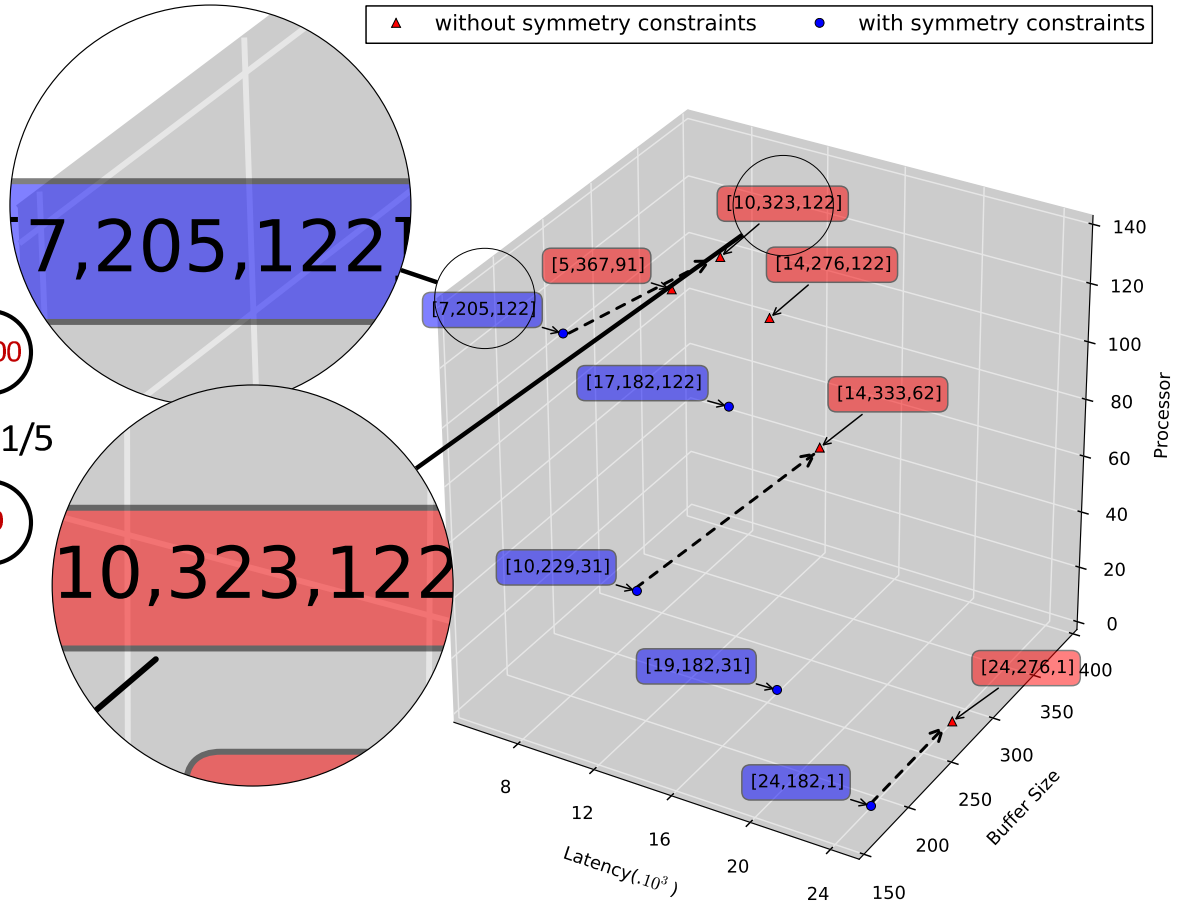


Video Decoder

3D cost space (C_L, C_P, C_B) exploration, C_B - total buffer size

MPEG video decoder:

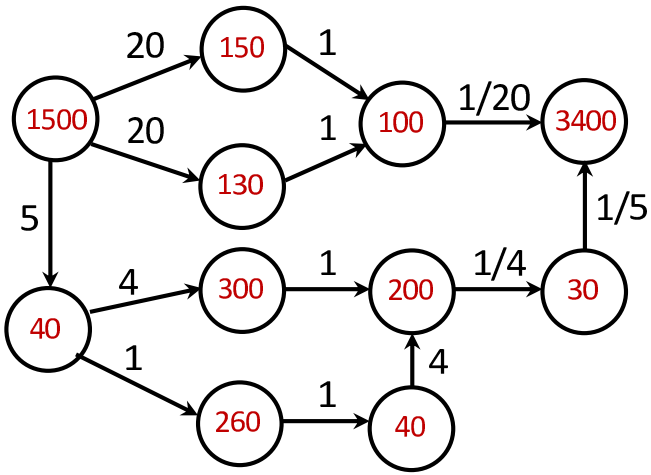




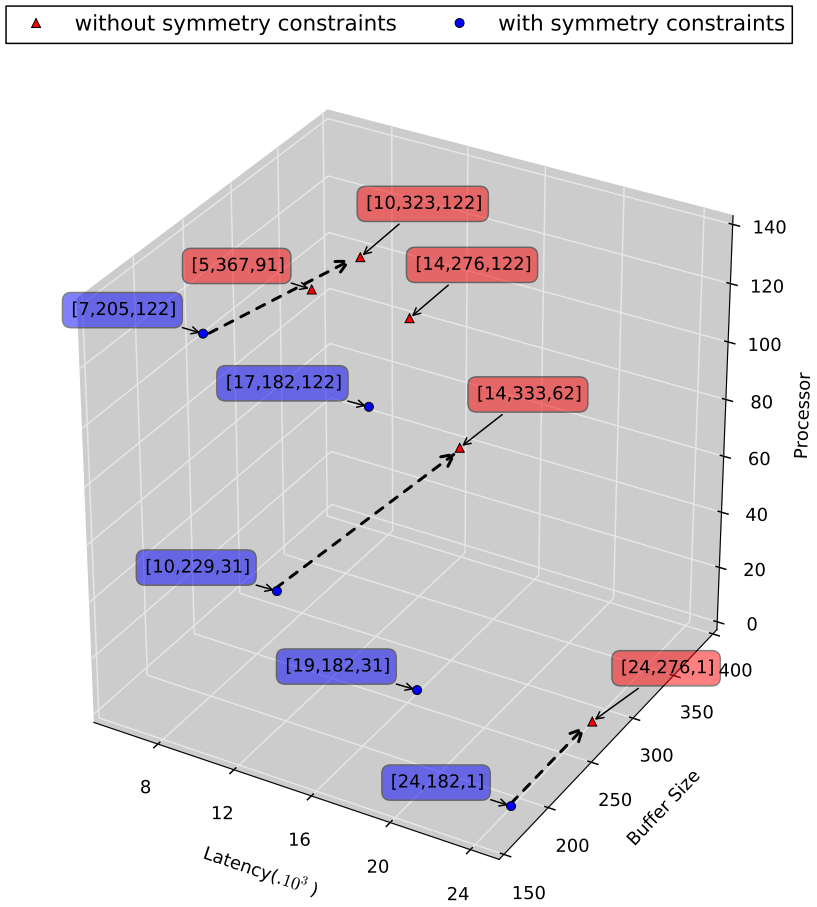
Video Decoder

3D cost space $(\mathbf{C}_L, \mathbf{C}_P, \mathbf{C}_B)$ exploration, $\mathbf{C}_B$ - total buffer size

MPEG video decoder:



122 Tasks

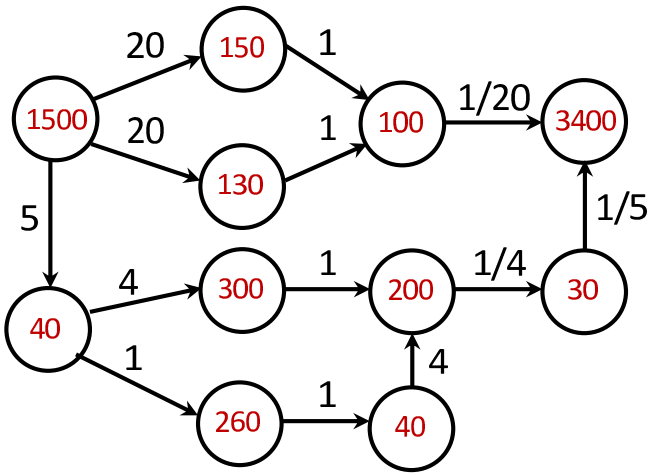


Better Pareto points

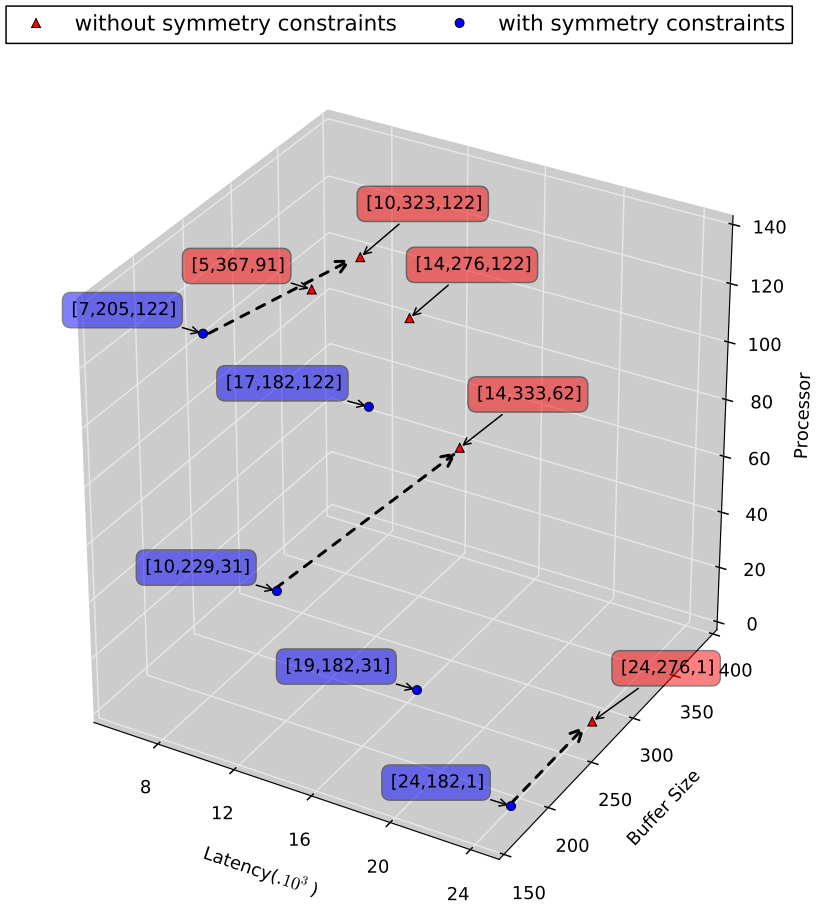
Video Decoder

3D cost space $(\mathbf{C}_L, \mathbf{C}_P, \mathbf{C}_B)$ exploration, $\mathbf{C}_B$ - total buffer size

MPEG video decoder:



122 Tasks



Better Pareto points in same **TIME-Budget** !

Distributed memory scheduling

Distributed memory scheduling

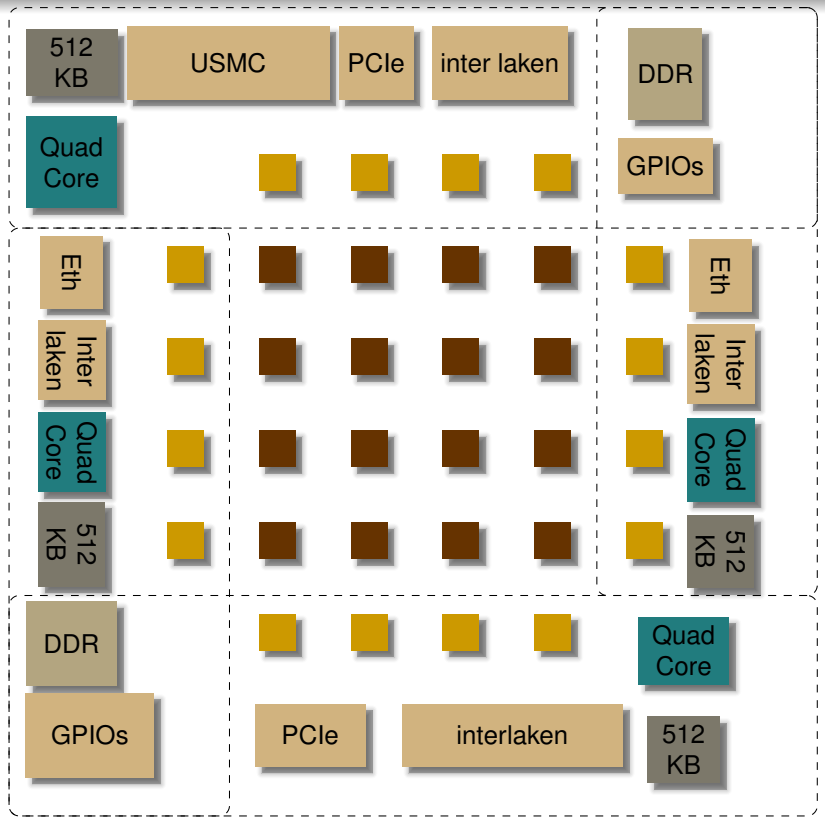
- So far we ignored the **communication** costs

Distributed memory scheduling

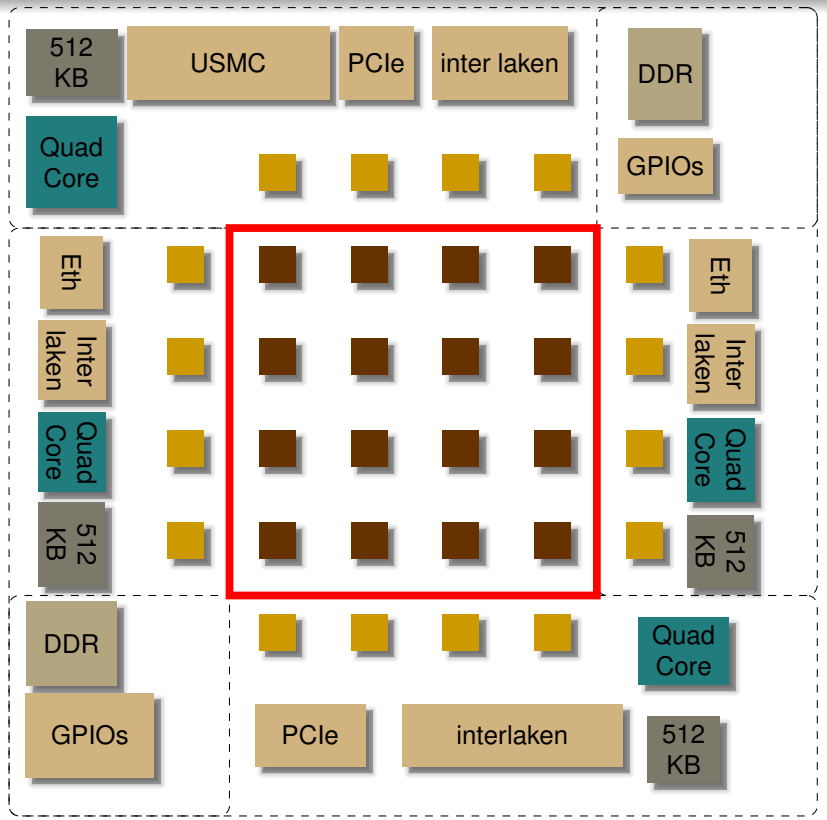
- So far we ignored the **communication** costs
- For **distributed memory**, communication needs to be modeled

Overview

Kalray MPPA-256

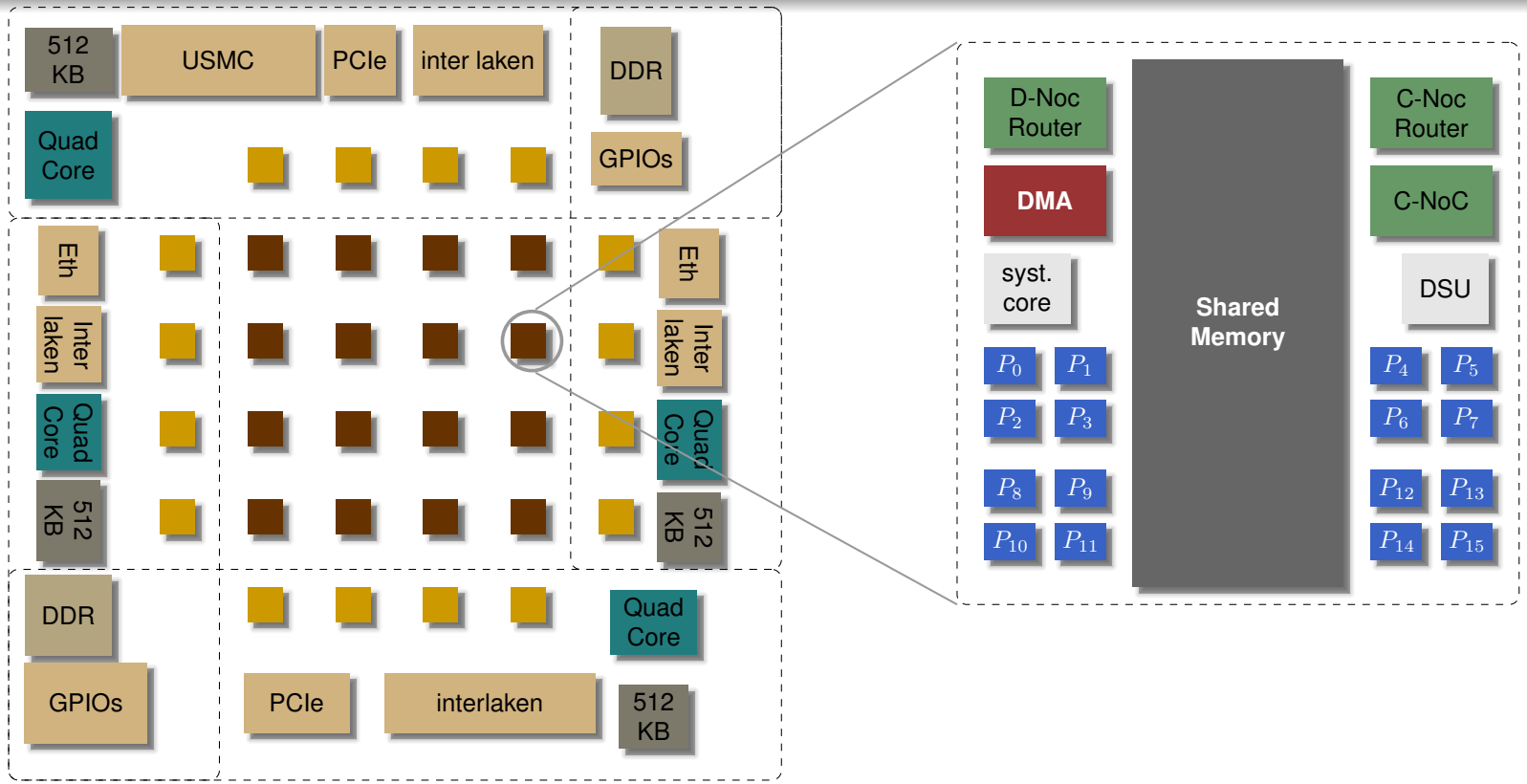


Kalray MPPA-256



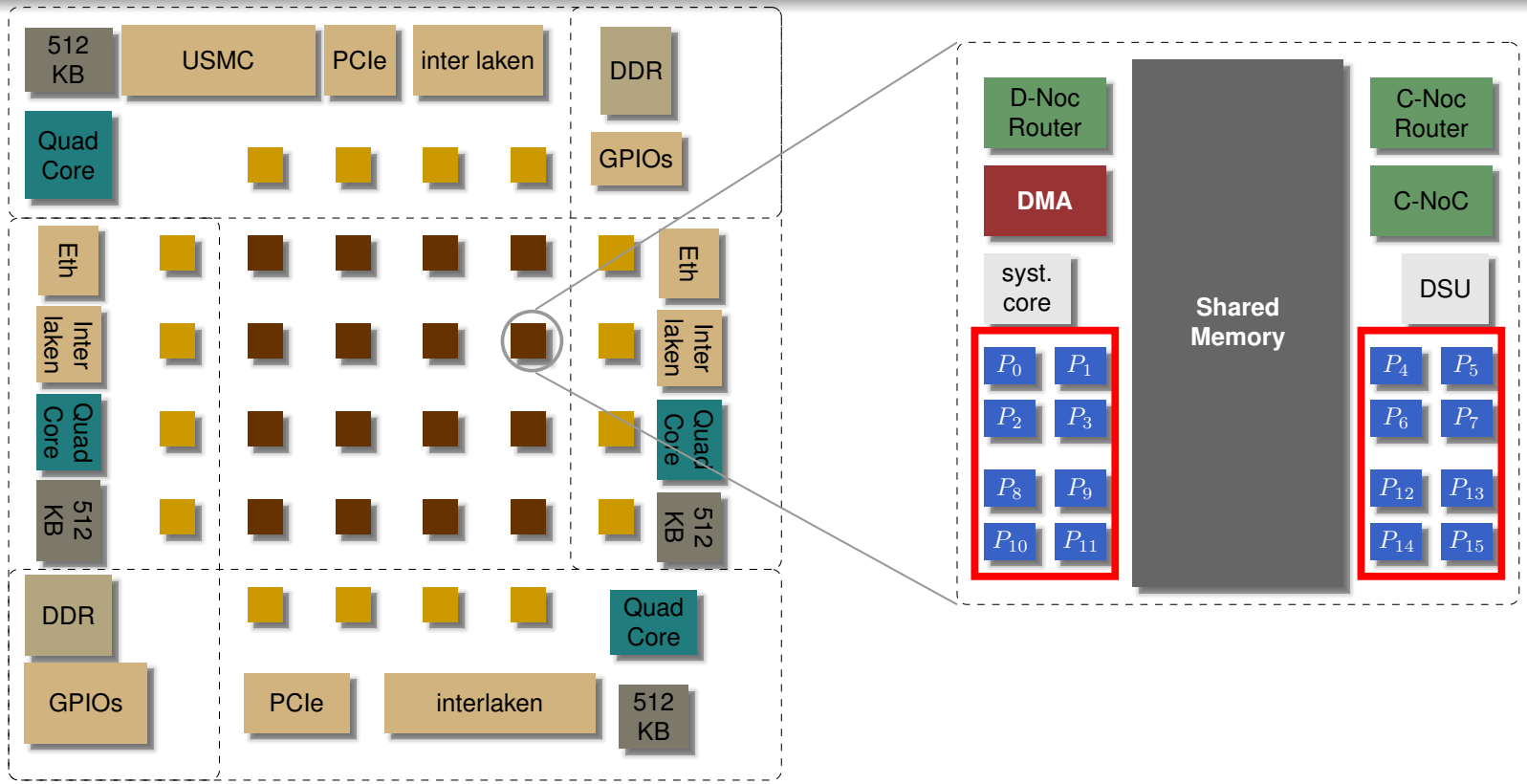
- 16 **compute clusters**

Kalray MPPA-256



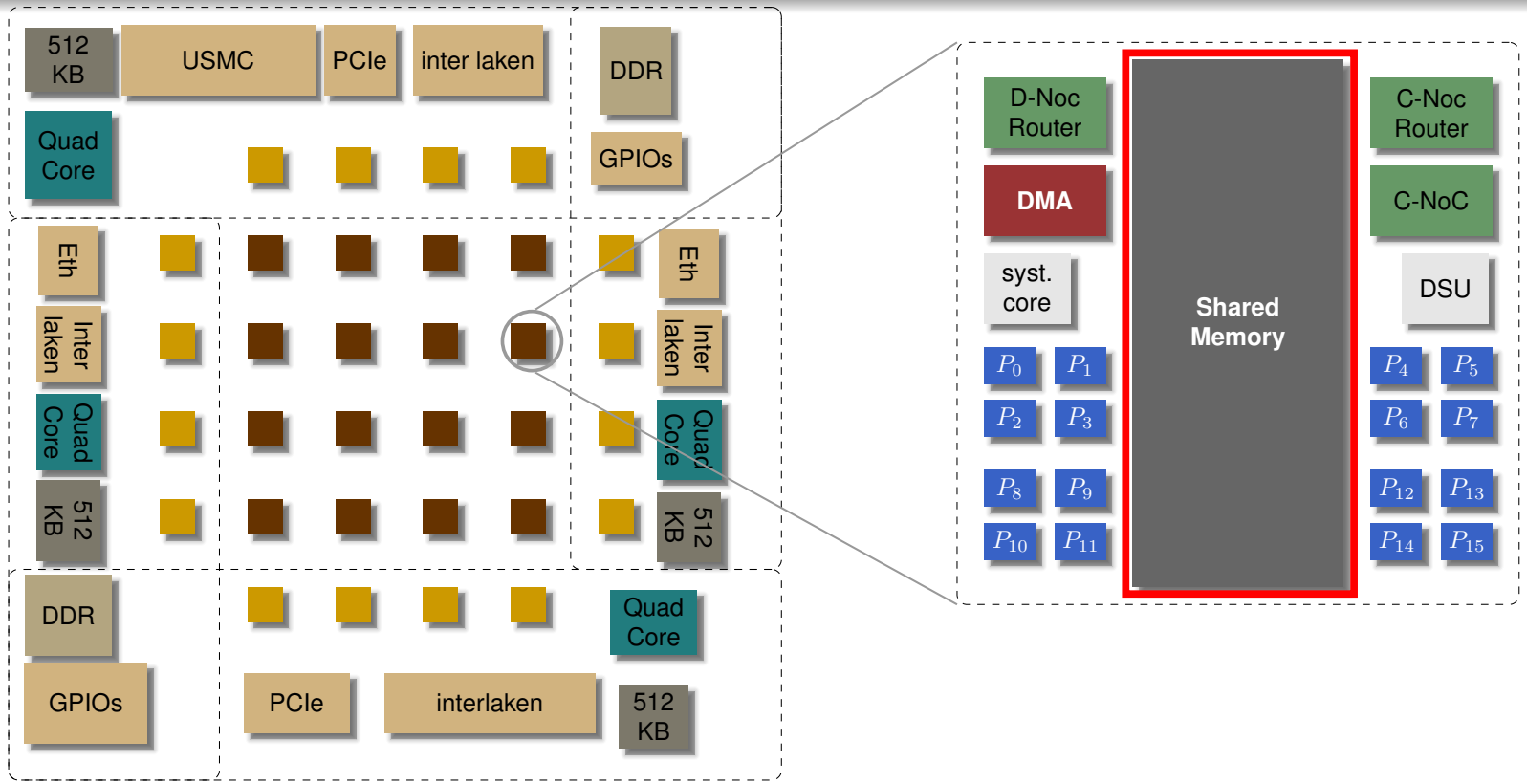
- 16 **compute clusters**

Kalray MPPA-256



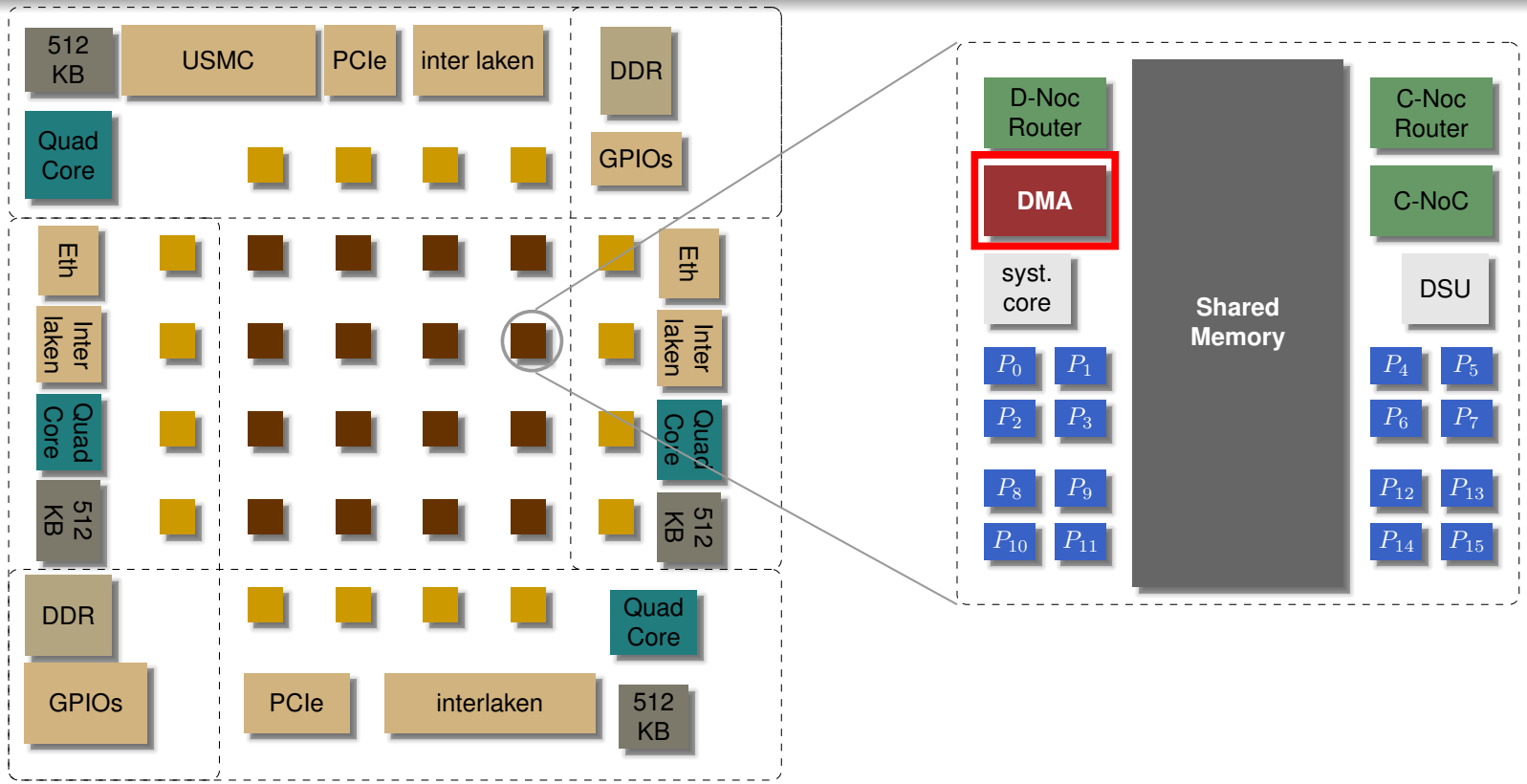
- 16 **compute clusters**
  - **16 processors**

Kalray MPPA-256



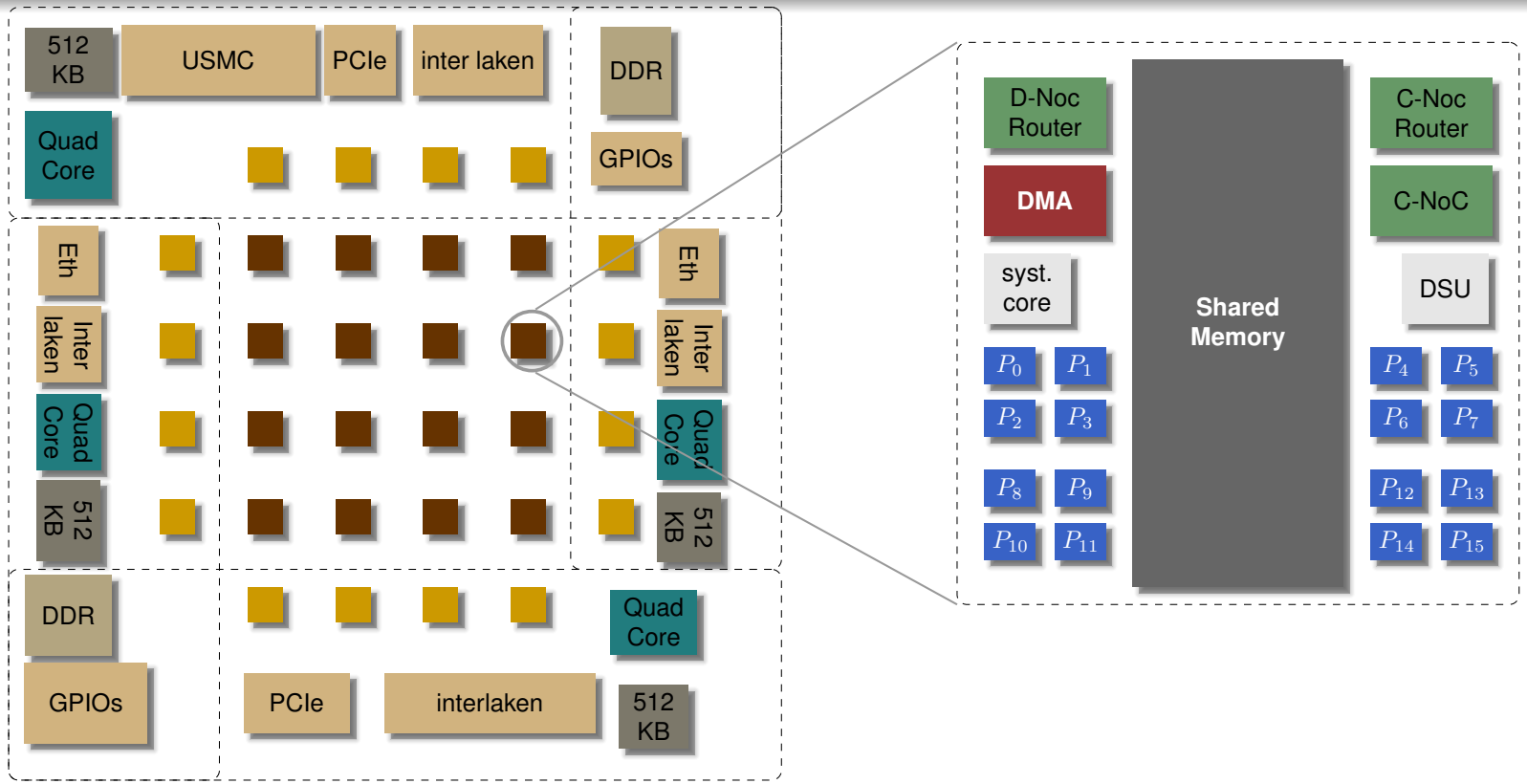
- 16 **compute clusters**
  - **16 processors**
  - 2 MB **Shared Memory**

Kalray MPPA-256



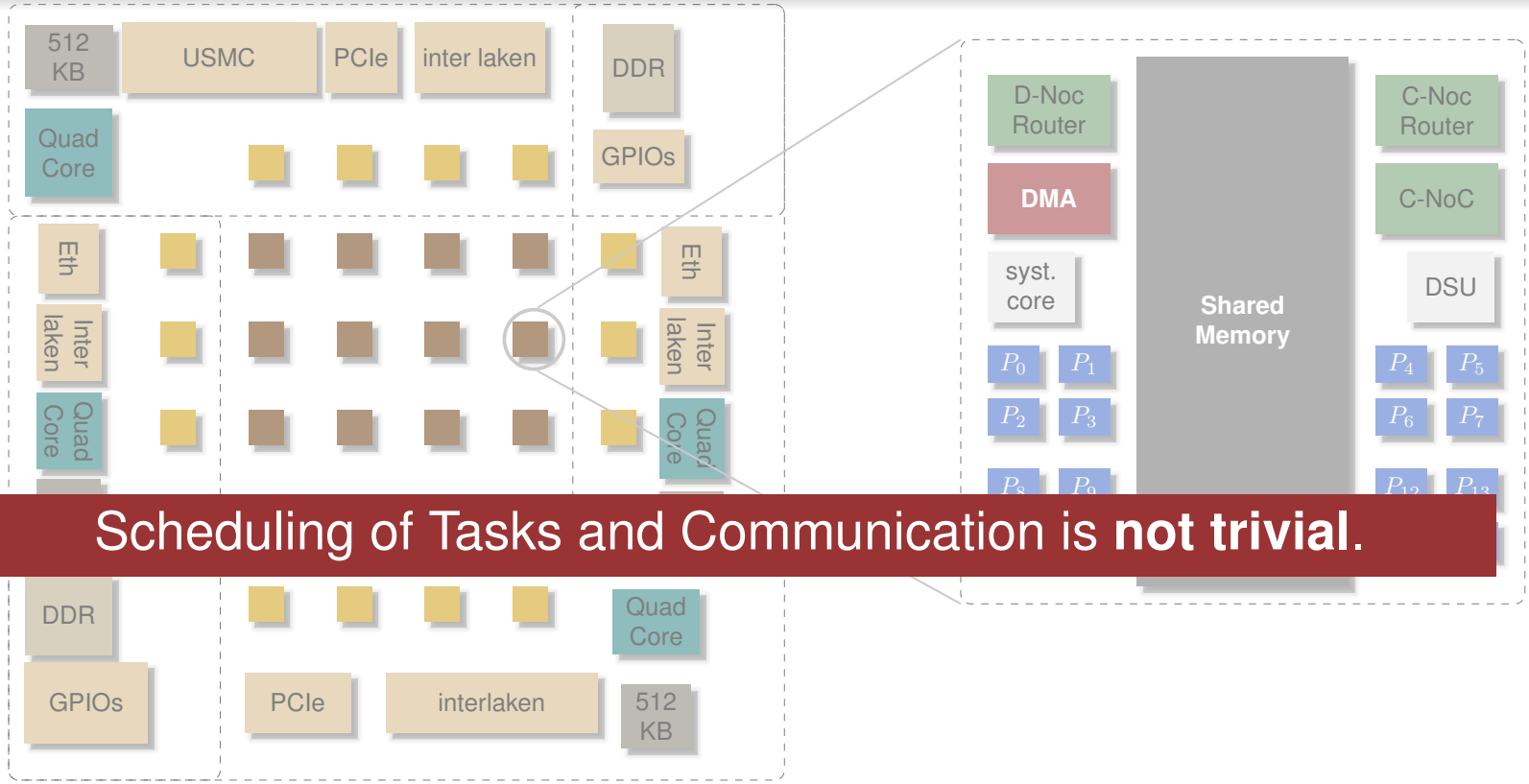
- 16 **compute clusters**
  - **16 processors**
  - 2 MB **Shared Memory**
  - **DMA**

Kalray MPPA-256



- 16 **compute clusters**
  - **16 processors**
  - 2 MB **Shared Memory**
  - **DMA**
- **Toroidal 2D** network

Kalray MPPA-256



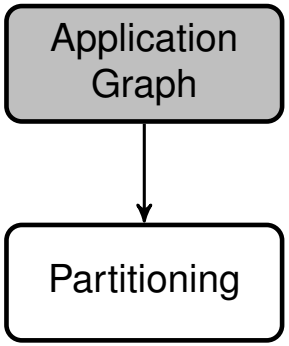
Scheduling of Tasks and Communication is **not trivial**.

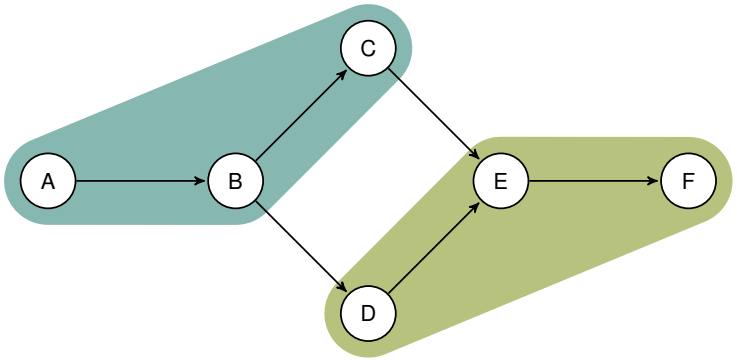
- 16 **compute clusters**
  - **16 processors**
  - 2 MB **Shared Memory**
  - **DMA**
- **Toroidal 2D** network

Design Flow

Application Graph

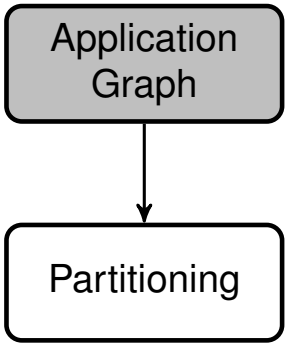
Design Flow

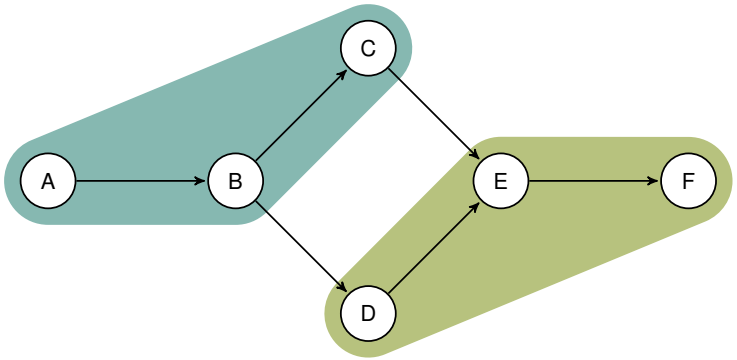




Group the Actors

Design Flow



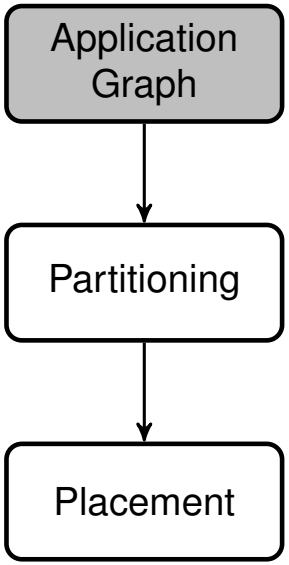


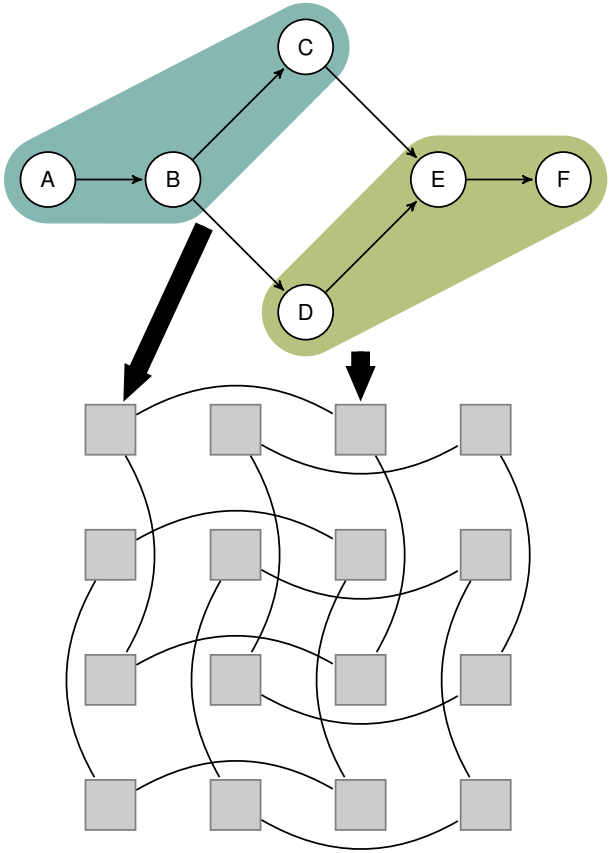
Group the Actors

Goals

- Load balance the groups
- Minimize data exchange

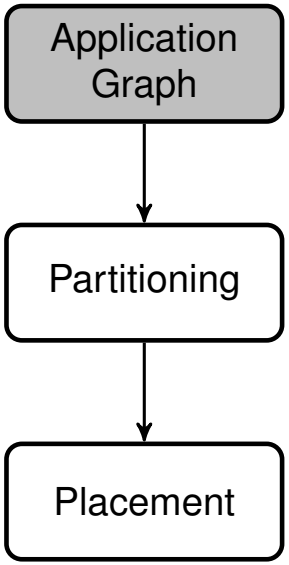
Design Flow

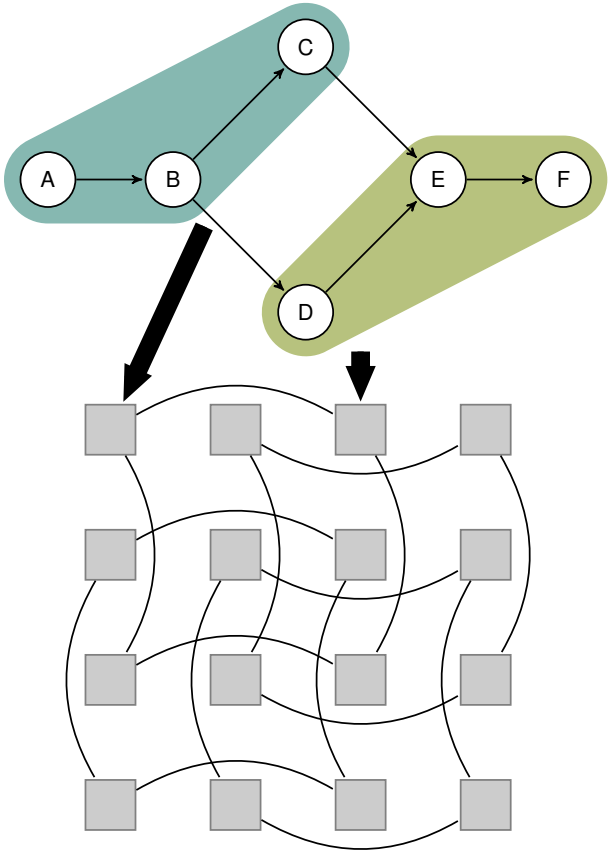




Place the Groups

Design Flow



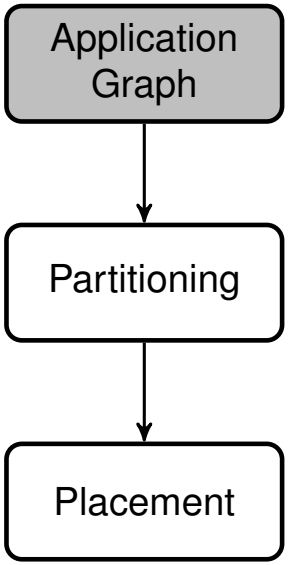


Place the Groups

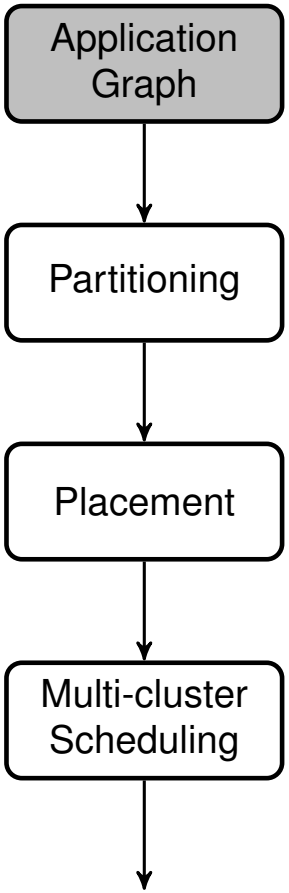
Goals

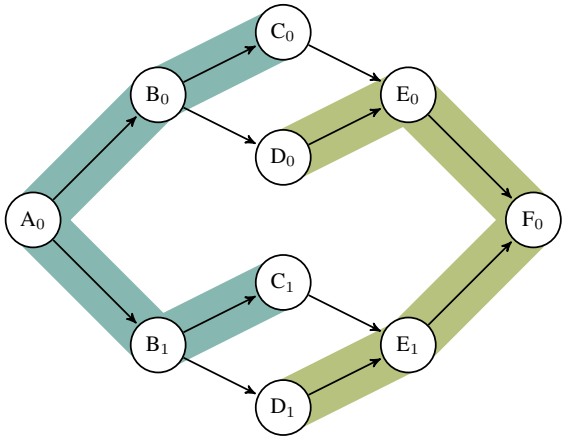
- Minimize distance between communicating groups

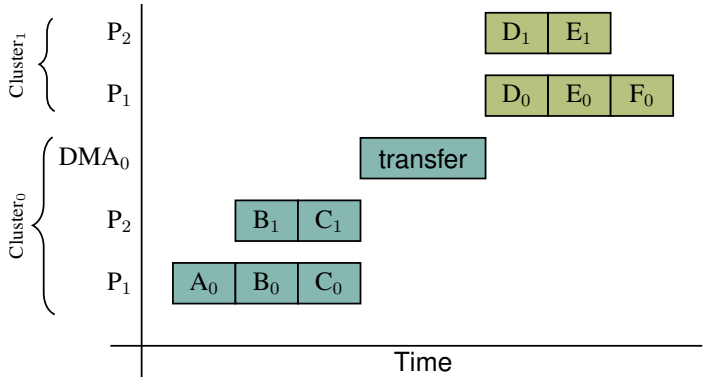
Design Flow



Design Flow



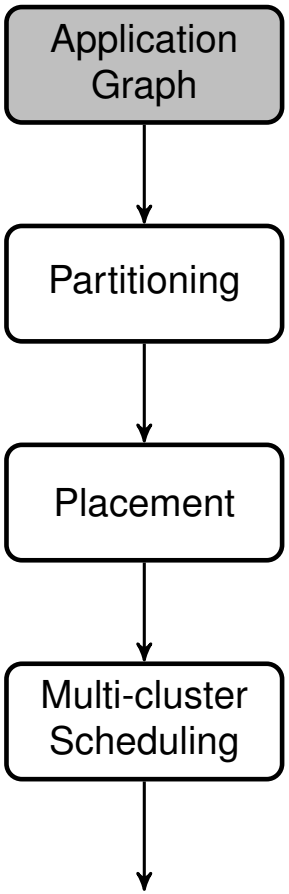


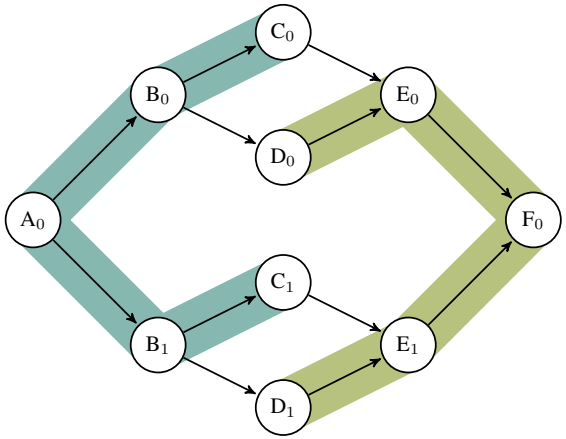


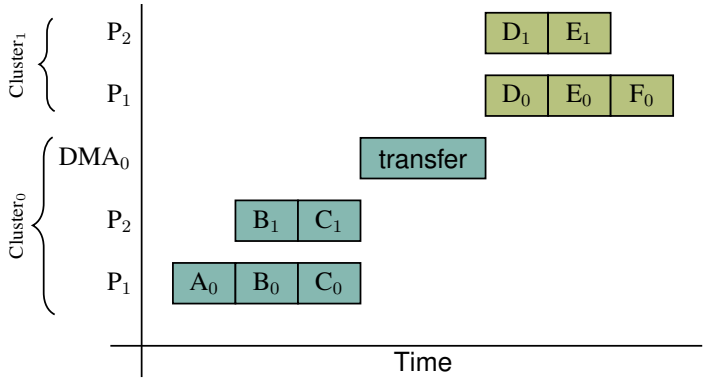
Schedule

- Tasks
- Transfer

Design Flow







Schedule

- Tasks
- Transfer

Goals

- Minimize Latency
- Minimize Buffer size

Output of Design Flow

Output of Design Flow

- Tasks and Transfers

Output of Design Flow

- Tasks and Transfers
  - **Cluster** Mapping

Output of Design Flow

- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping

Output of Design Flow

- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping
  - **Start time**

Output of Design Flow

- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping
  - **Start time**
- Edges

Output of Design Flow

- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping
  - **Start time**
- Edges
  - **Communication buffer** size

Output of Design Flow

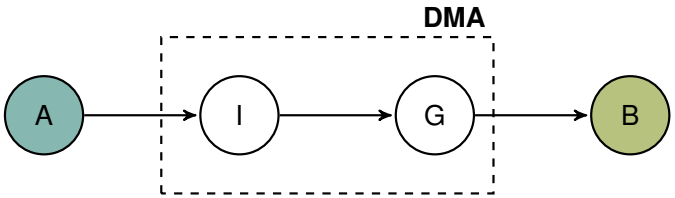
- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping
  - **Start time**
- Edges
  - **Communication buffer** size
- Application

Output of Design Flow

- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping
  - **Start time**
- Edges
  - **Communication buffer** size
- Application
  - **Latency**

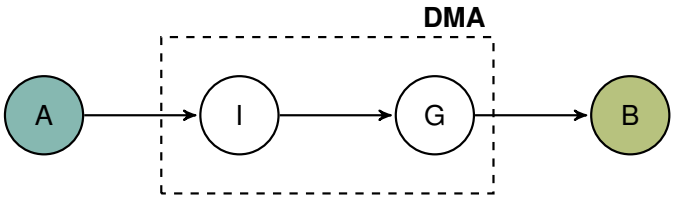
DMA Model

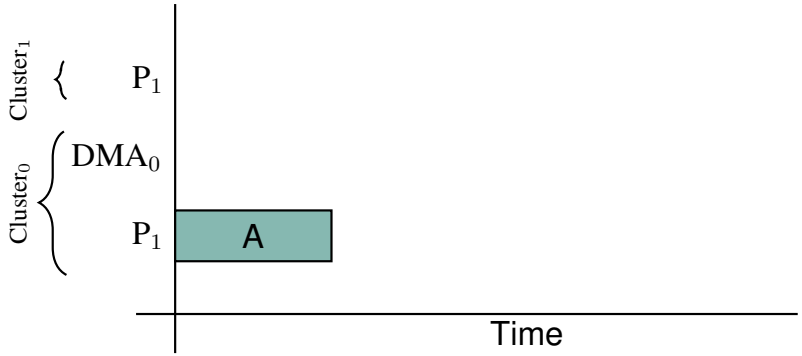
Tasks communicating via DMA:



DMA Model

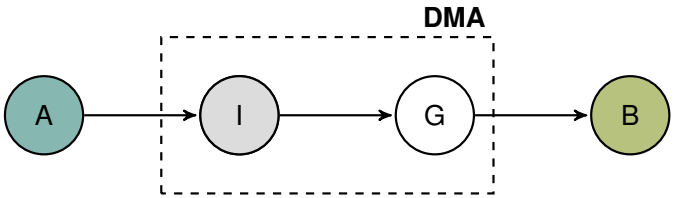
Tasks communicating via DMA:

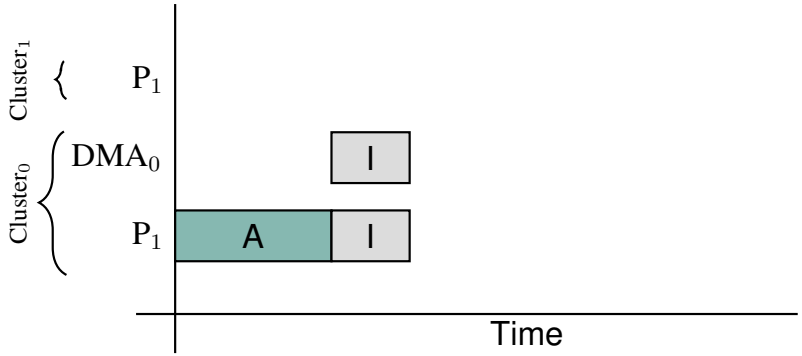




DMA Model

Tasks communicating via DMA:

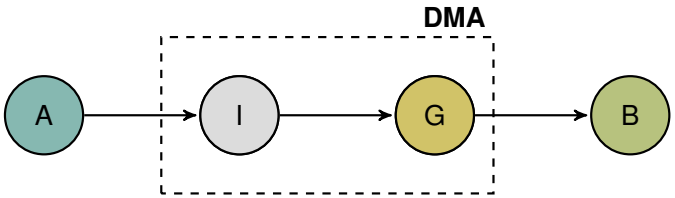


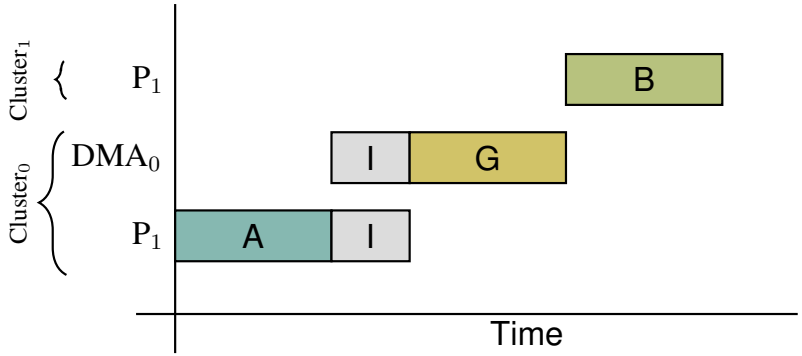


Task	Description	Resources used	Task duration
I	Initialization	Processor and DMA	Constant

DMA Model

Tasks communicating via DMA:

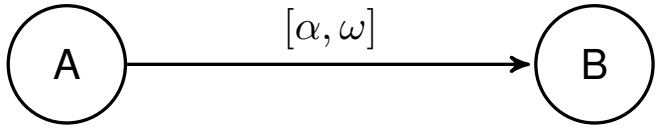




Task	Description	Resources used	Task duration
I	Initialization	Processor and DMA	Constant
G	Network Transfer	Only DMA	Transfer size dependent

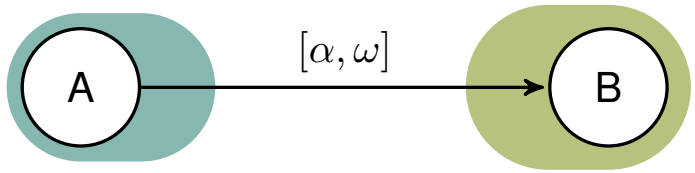
Model Transformation

An example application graph:



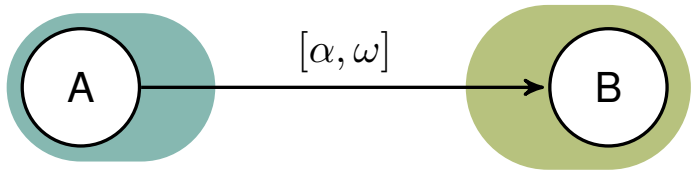
Model Transformation

An example application graph:

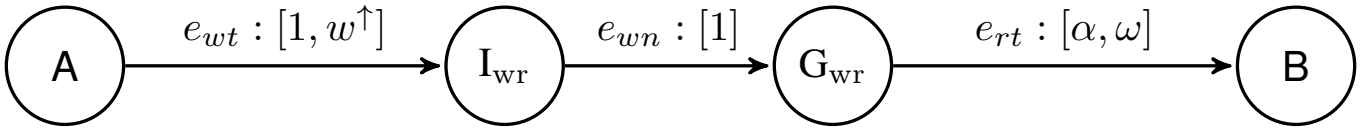


Model Transformation

An example application graph:

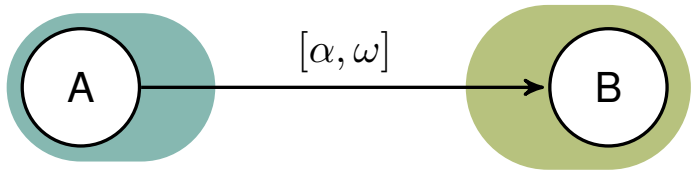


Partition-Aware graph:

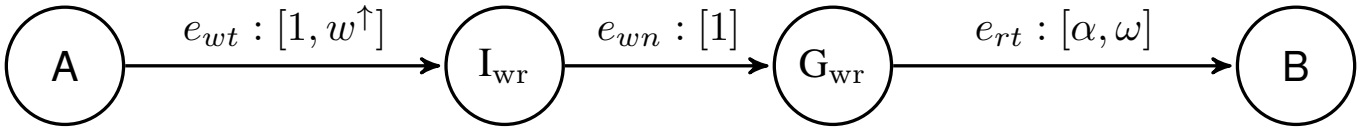


Model Transformation

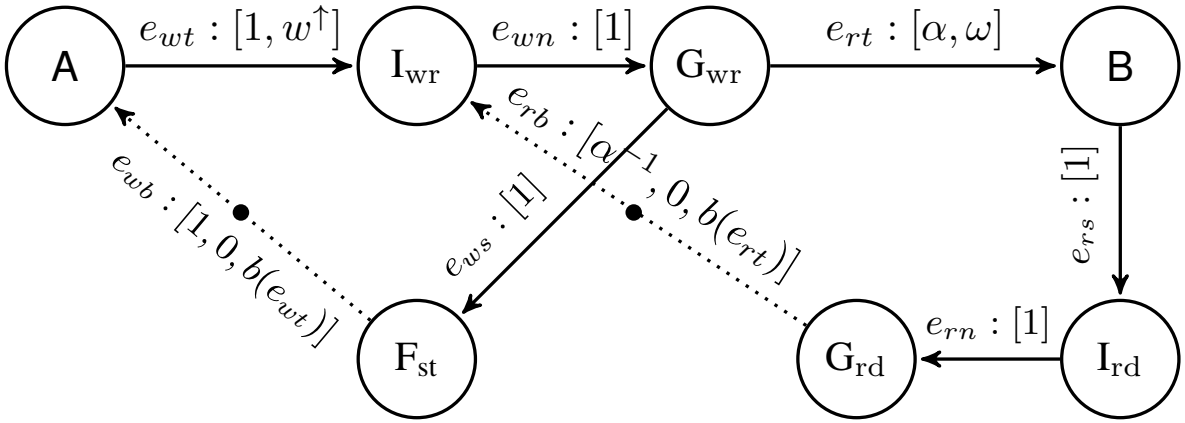
An example application graph:



Partition-Aware graph:

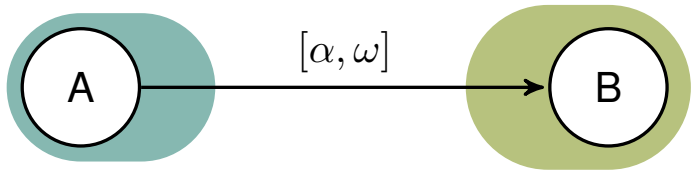


Buffer-Aware graph:

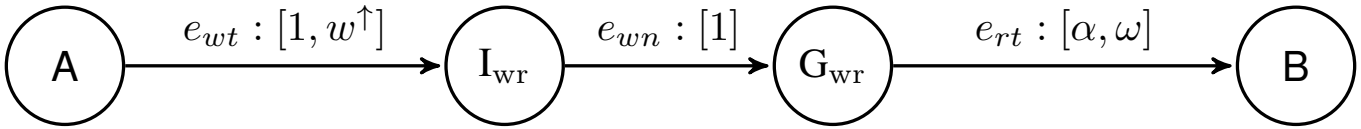


Model Transformation

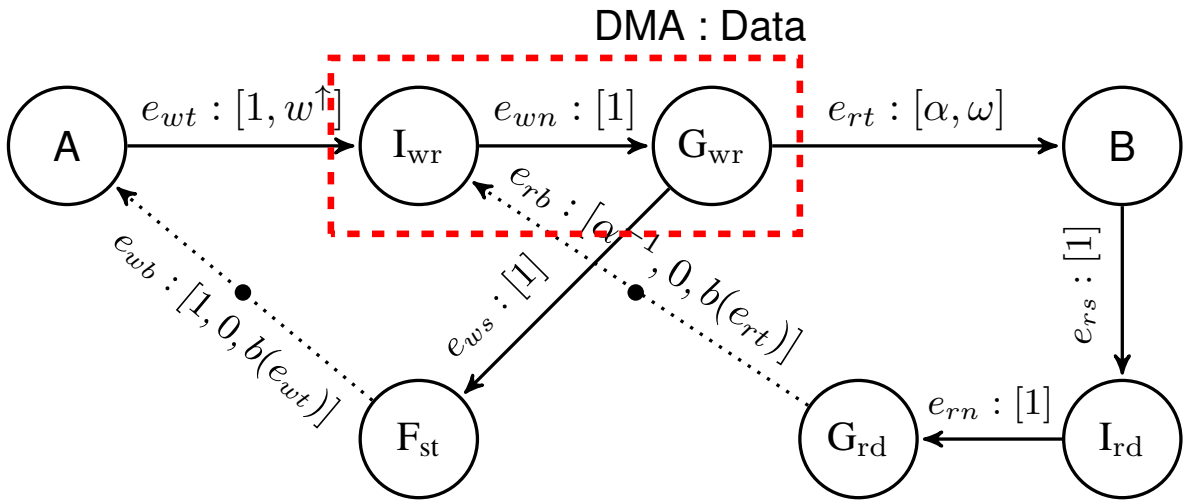
An example application graph:



Partition-Aware graph:

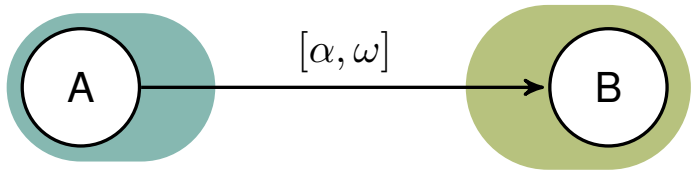


Buffer-Aware graph:

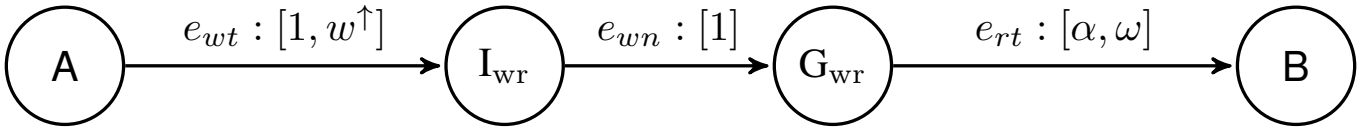


Model Transformation

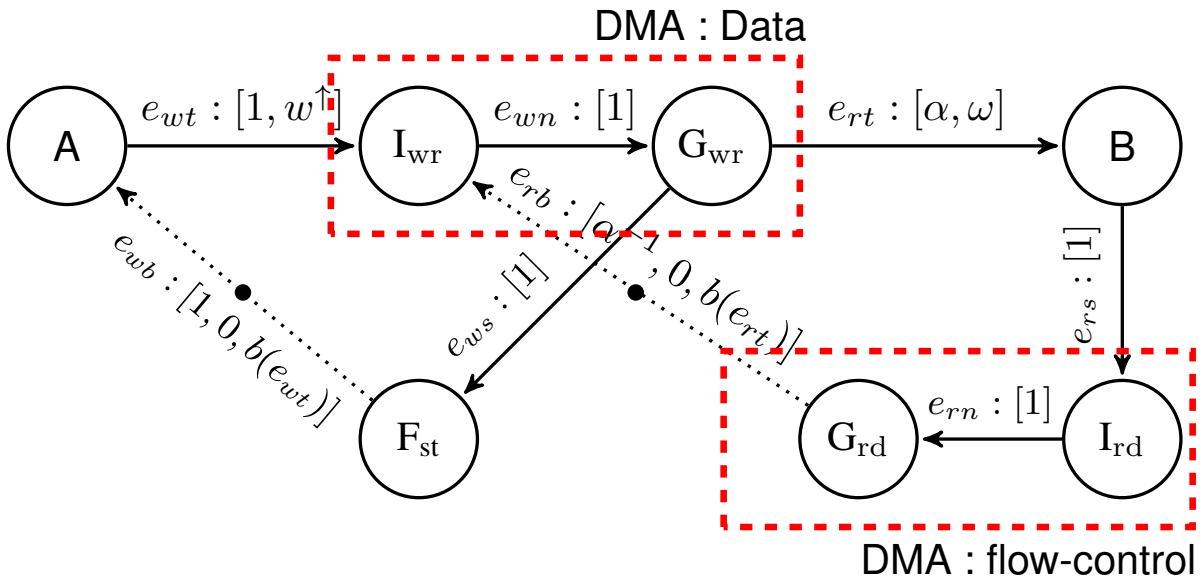
An example application graph:



Partition-Aware graph:

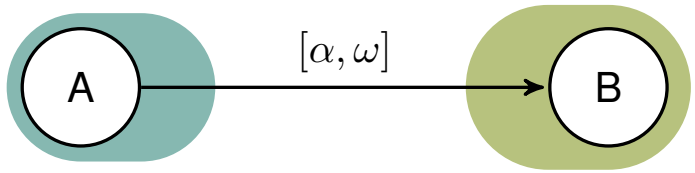


Buffer-Aware graph:

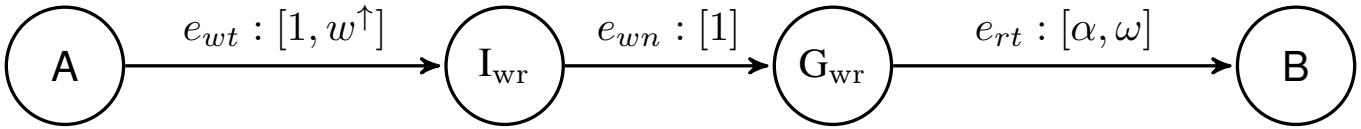


Model Transformation

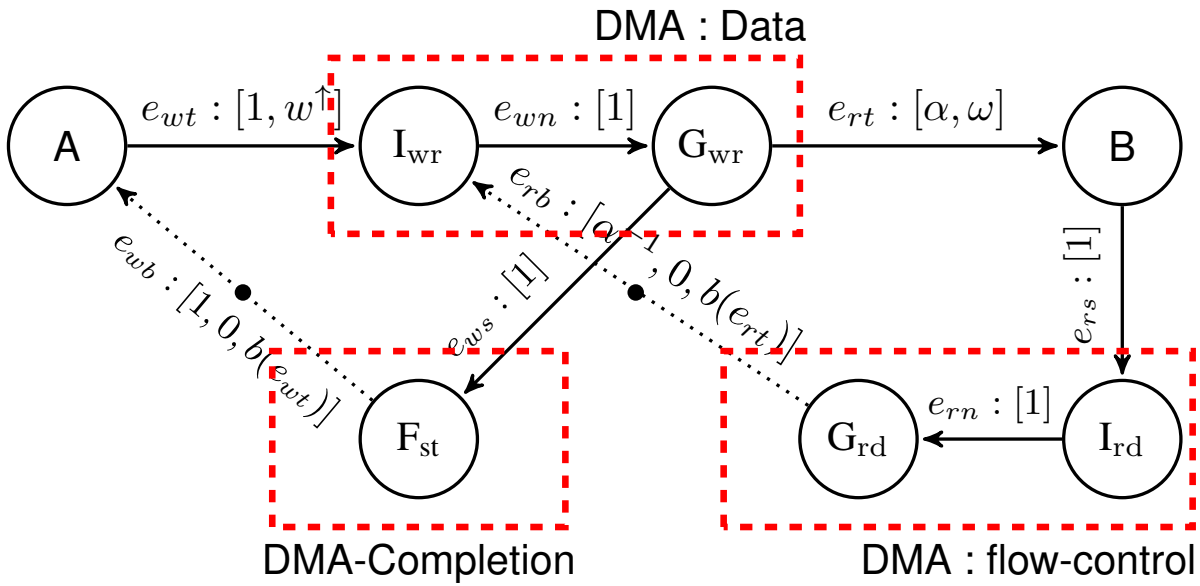
An example application graph:



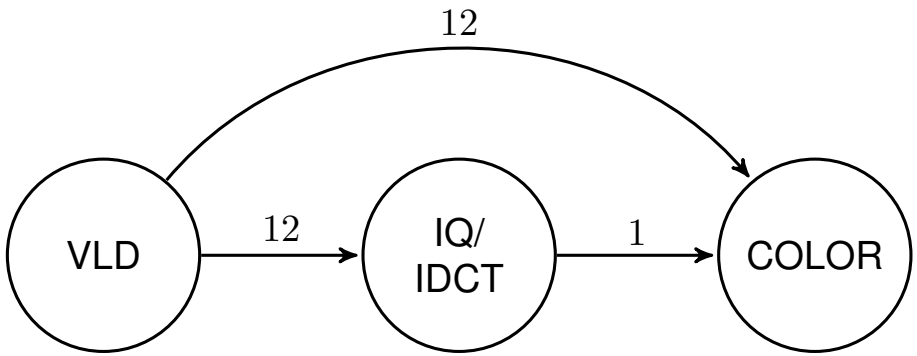
Partition-Aware graph:



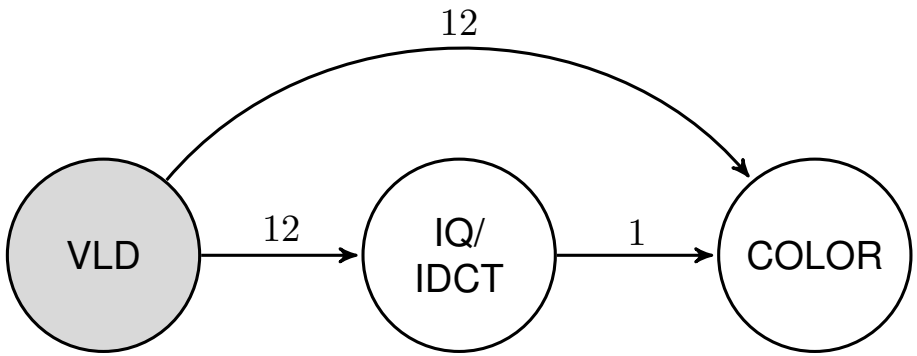
Buffer-Aware graph:



JPEG Decoder Example

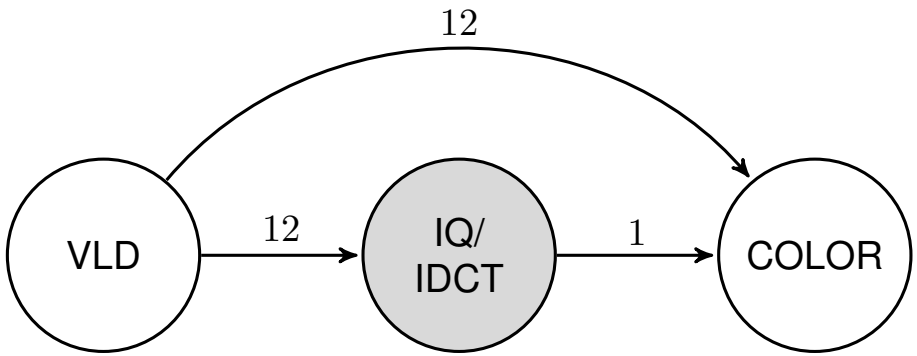


JPEG Decoder Example



VLD : Variable Length Decoder

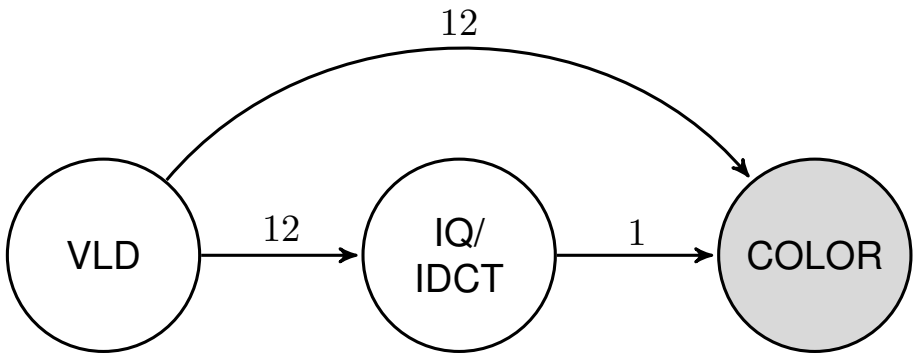
JPEG Decoder Example



VLD : Variable Length Decoder

IQ / IDCT : Inverse Quantization / Inverse Discrete Cosine Transform

JPEG Decoder Example

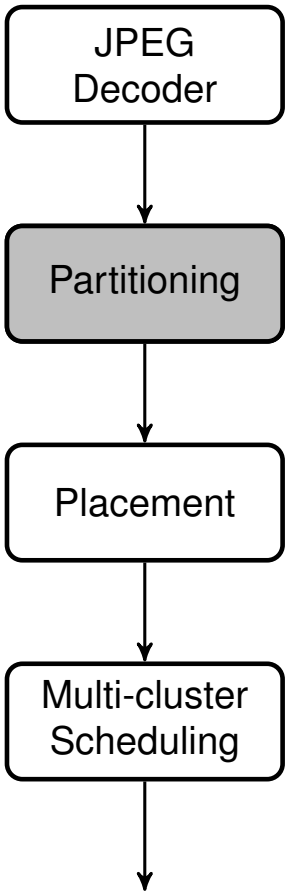


VLD : Variable Length Decoder

IQ / IDCT : Inverse Quantization / Inverse Discrete Cosine Transform

Color : Color Conversion

JPEG Decoder Example



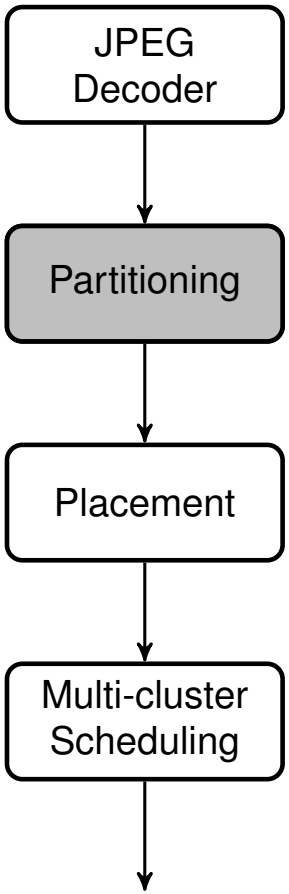
Partitioning Solutions:

$\mathbf{C}_\tau$: Max. workload per group

$\mathbf{C}_\eta$: Total communication cost

$\mathbf{C}_z$: No. of Groups

JPEG Decoder Example



Partitioning Solutions:

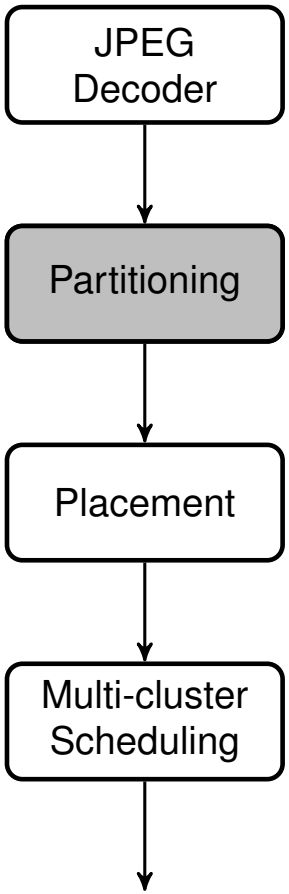
C_τ : Max. workload per group

C_η : Total communication cost

C_z : No. of Groups

Solution	Allocated group			Exploration Cost		
	vld	iq	color	C_τ	C_η	C_z
P_{s0}	0	1	2	424012	12384	3
P_{s1}	0	0	1	758116	2736	2
P_{s2}	0	0	0	934288	0	1
P_{s3}	0	1	1	510276	9648	2

JPEG Decoder Example



Partitioning Solutions:

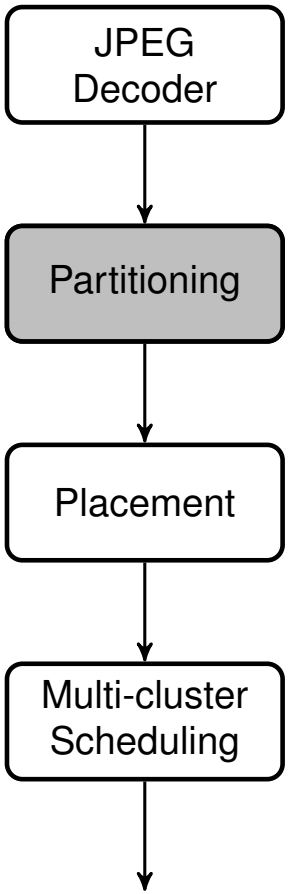
C_τ : Max. workload per group

C_η : Total communication cost

C_z : No. of Groups

Solution	Allocated group			Exploration Cost		
	vld	iq	color	C_τ	C_η	C_z
P_{s0}	0	1	2	424012	12384	3
P_{s1}	0	0	1	758116	2736	2
P_{s2}	0	0	0	934288	0	1
P_{s3}	0	1	1	510276	9648	2

JPEG Decoder Example



Partitioning Solutions:

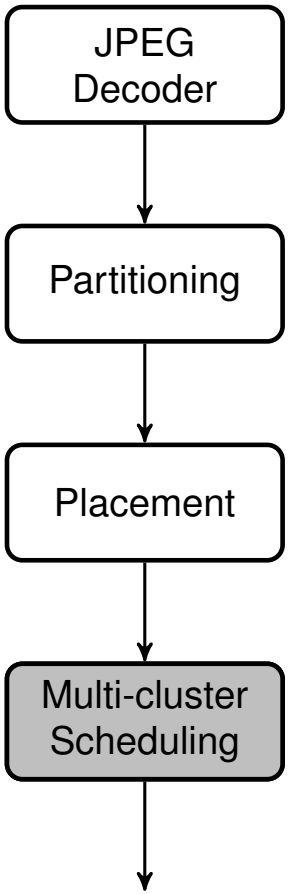
C_τ : Max. workload per group

C_η : Total communication cost

C_z : No. of Groups

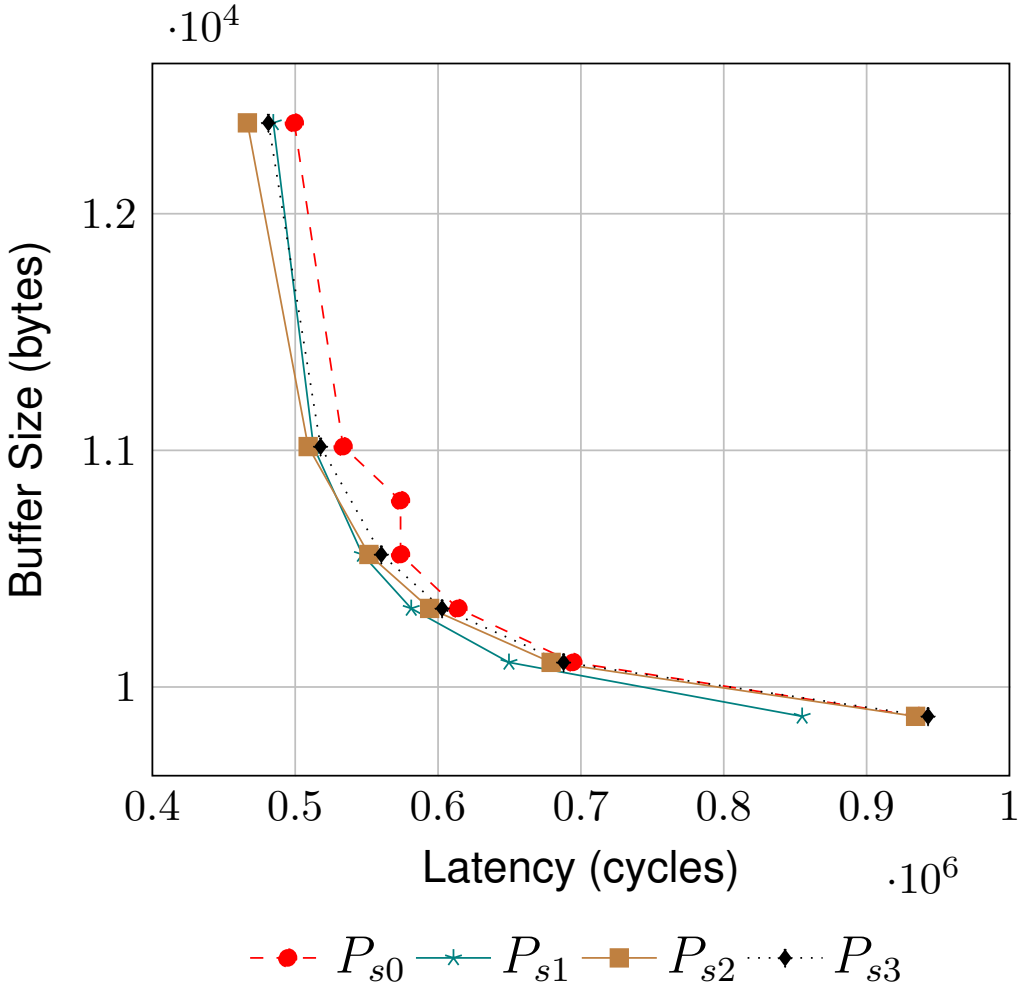
Solution	Allocated group			Exploration Cost		
	vld	iq	color	C_τ	C_η	C_z
P_{s0}	0	1	2	424012	12384	3
P_{s1}	0	0	1	758116	2736	2
P_{s2}	0	0	0	934288	0	1
P_{s3}	0	1	1	510276	9648	2

JPEG Decoder Example



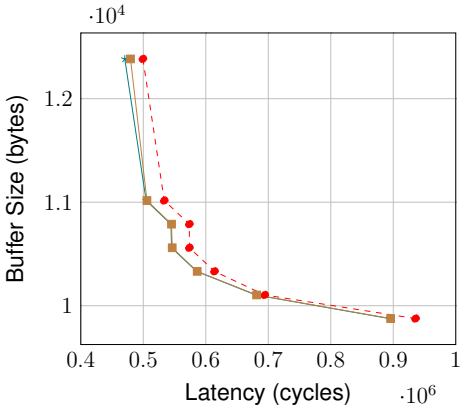
Solution
P_{s0}
P_{s1}
P_{s2}
P_{s3}

Scheduling Solutions:

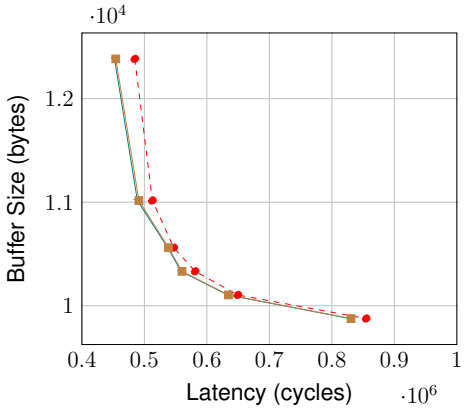


JPEG Decoder Example

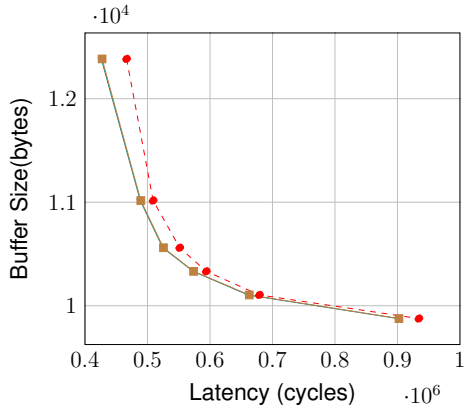
JPEG decoder latency on **Kalray** platform



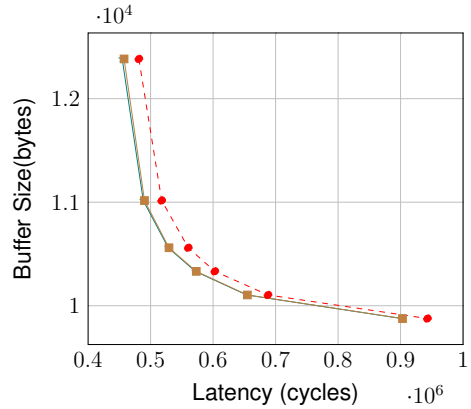
P_{s0}



P_{s1}

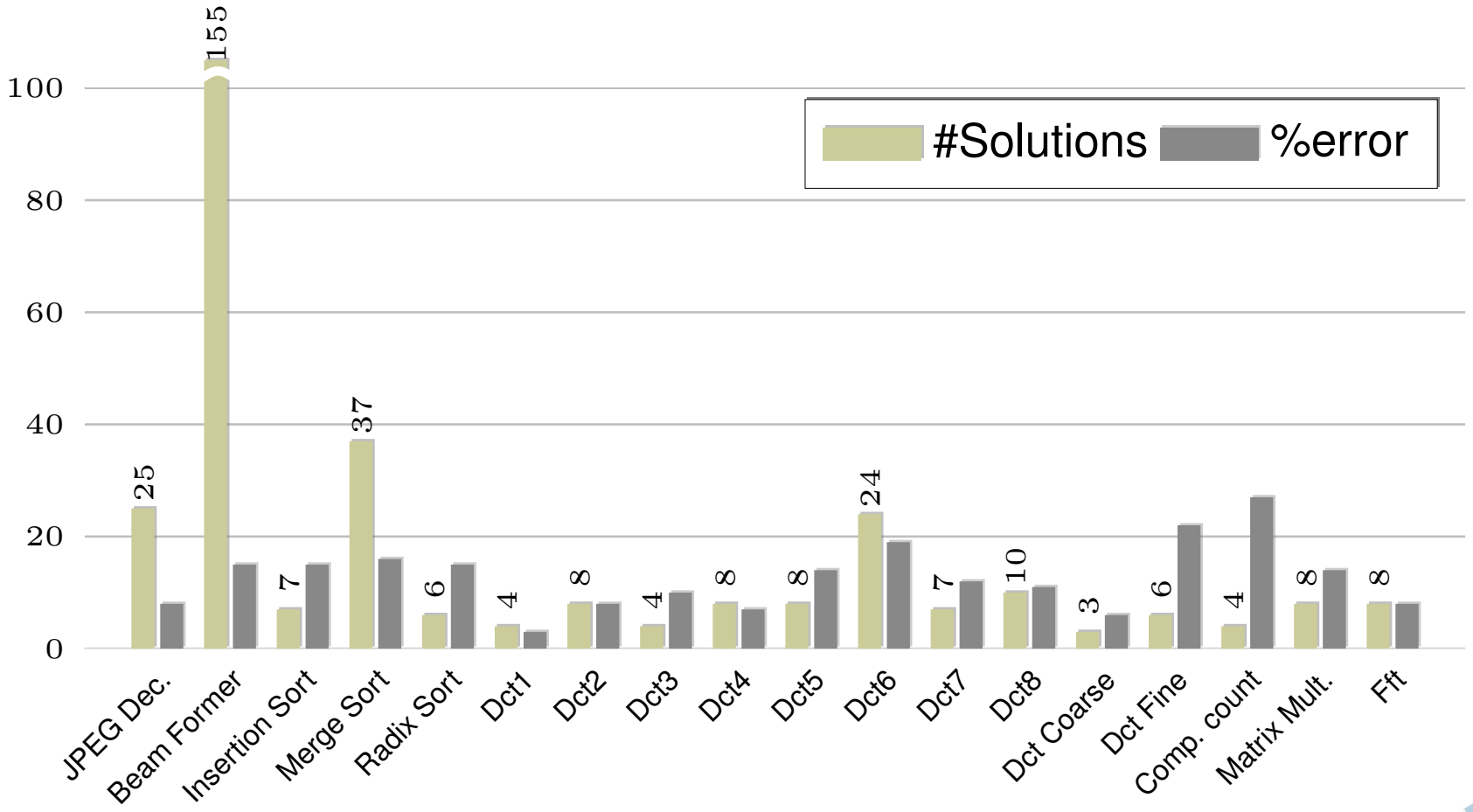


P_{s2}

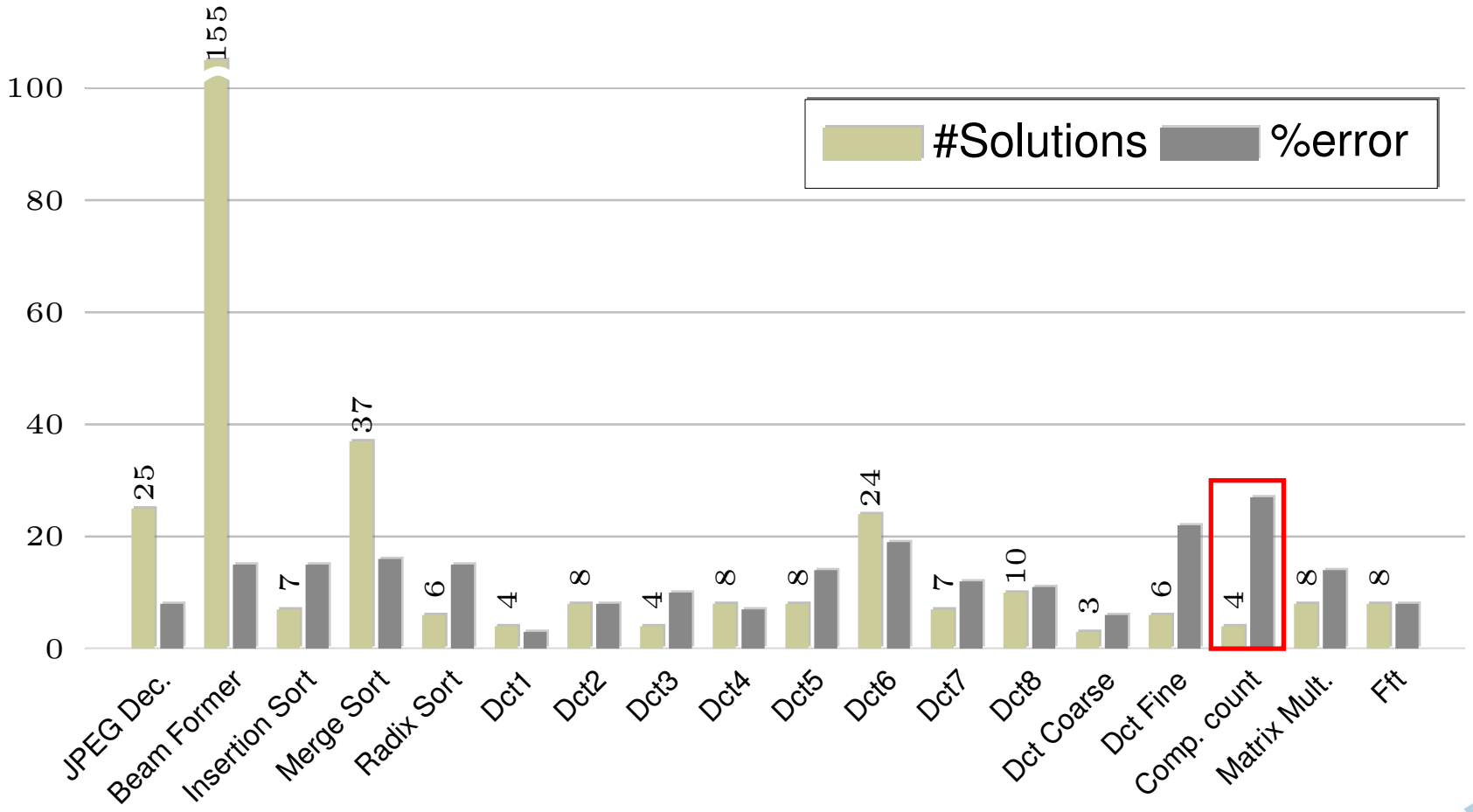


P_{s3}

StreamIt Benchmarks



StreamIt Benchmarks



Overview

Our Framework

Runtime

Our Framework

Runtime

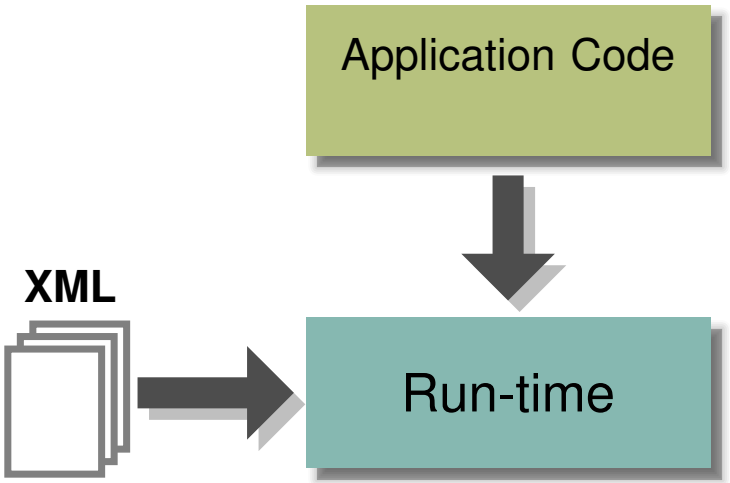
Run-time

Our Framework

Runtime

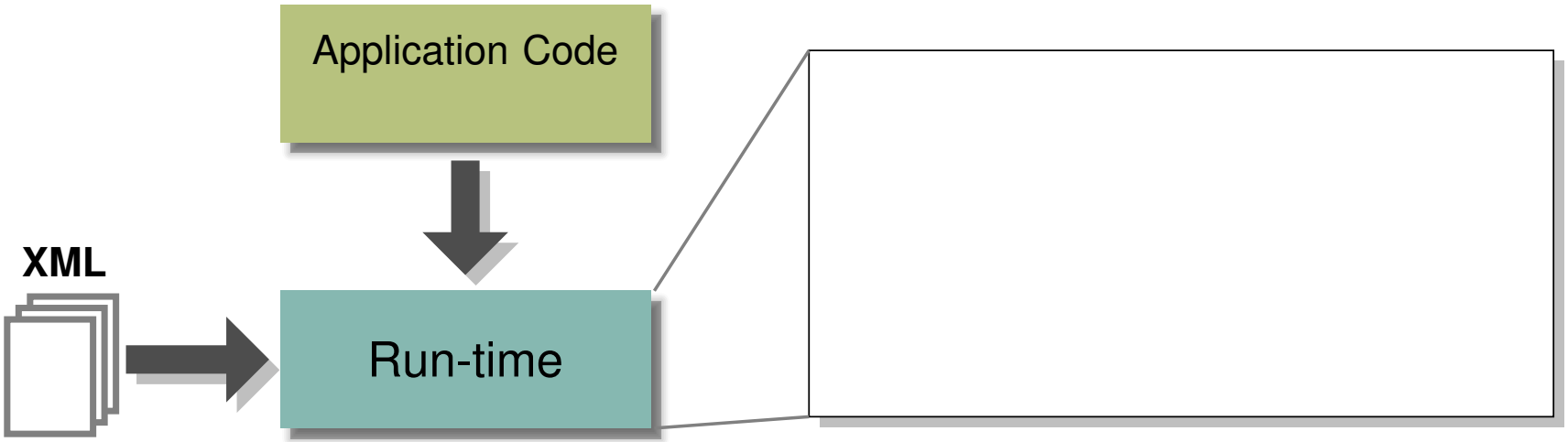
Our Framework

Runtime



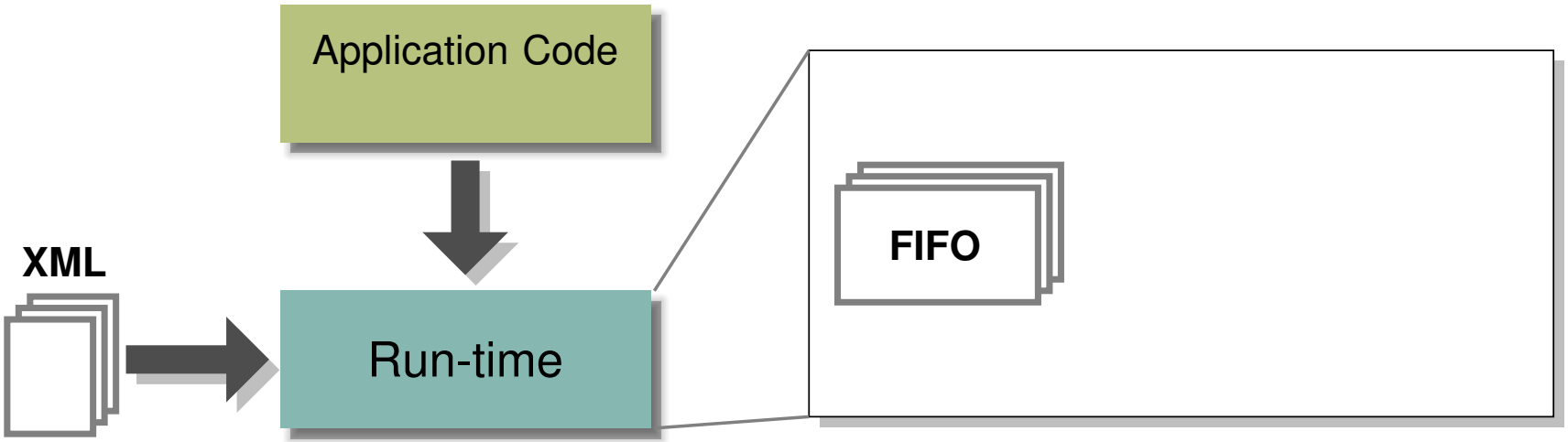
Our Framework

Runtime



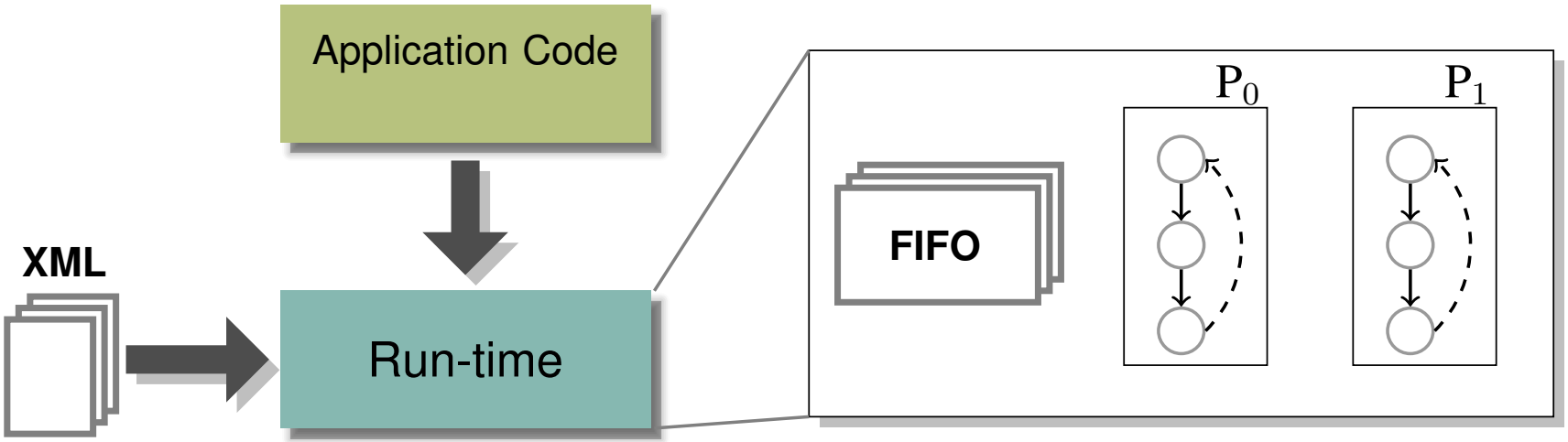
Our Framework

Runtime



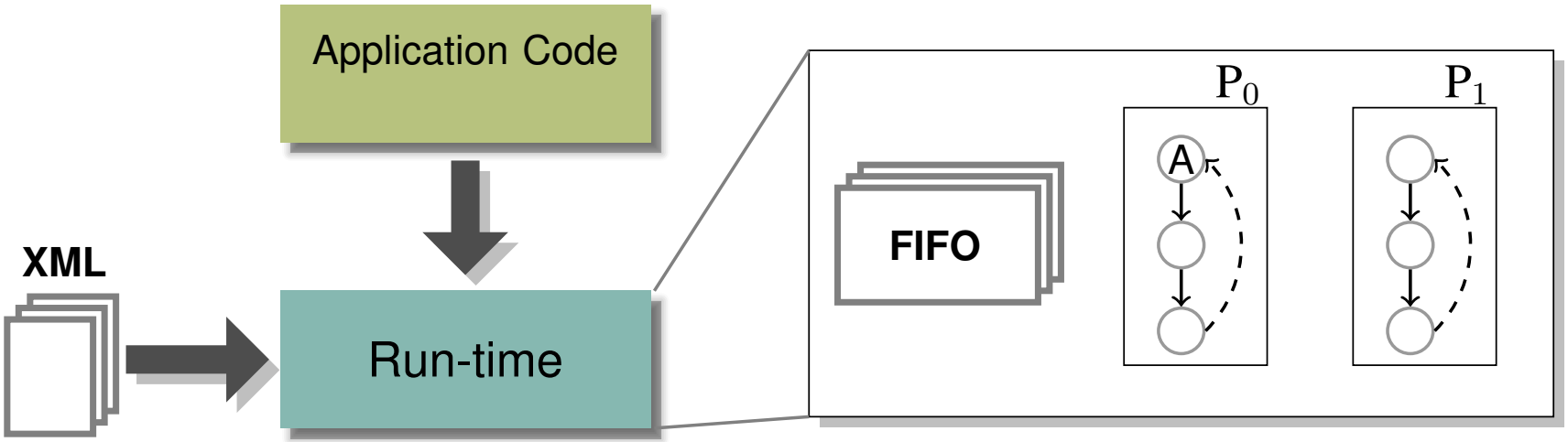
Our Framework

Runtime



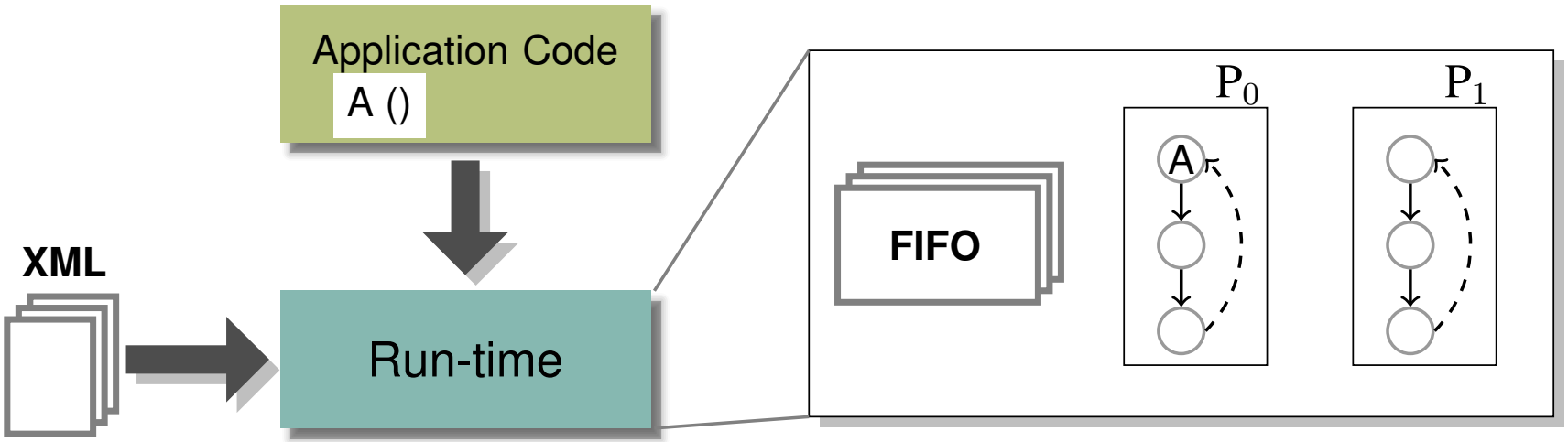
Our Framework

Runtime



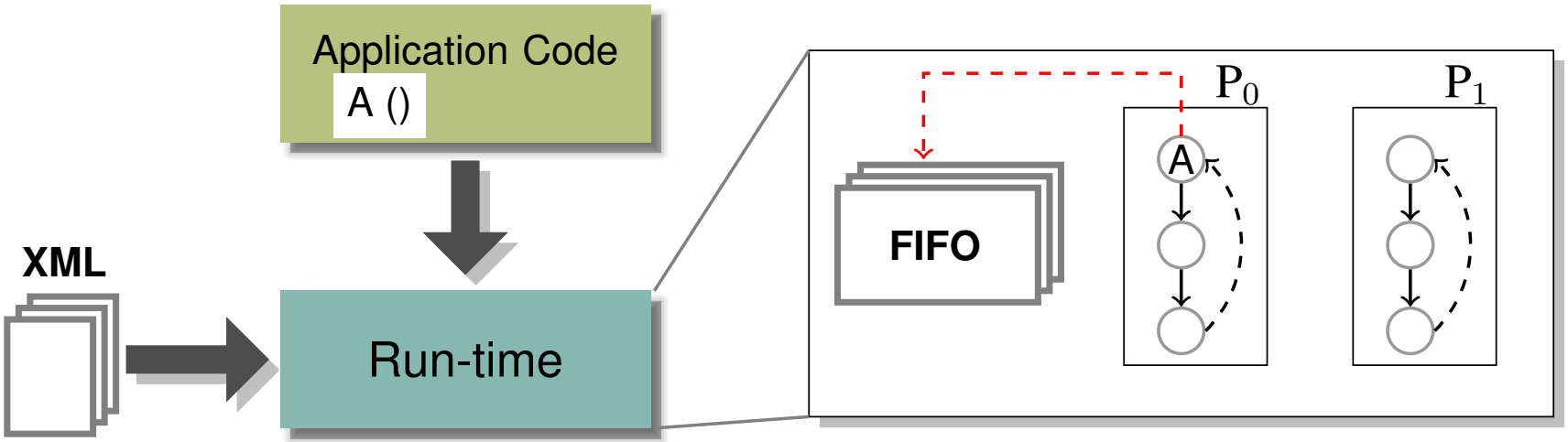
Our Framework

Runtime



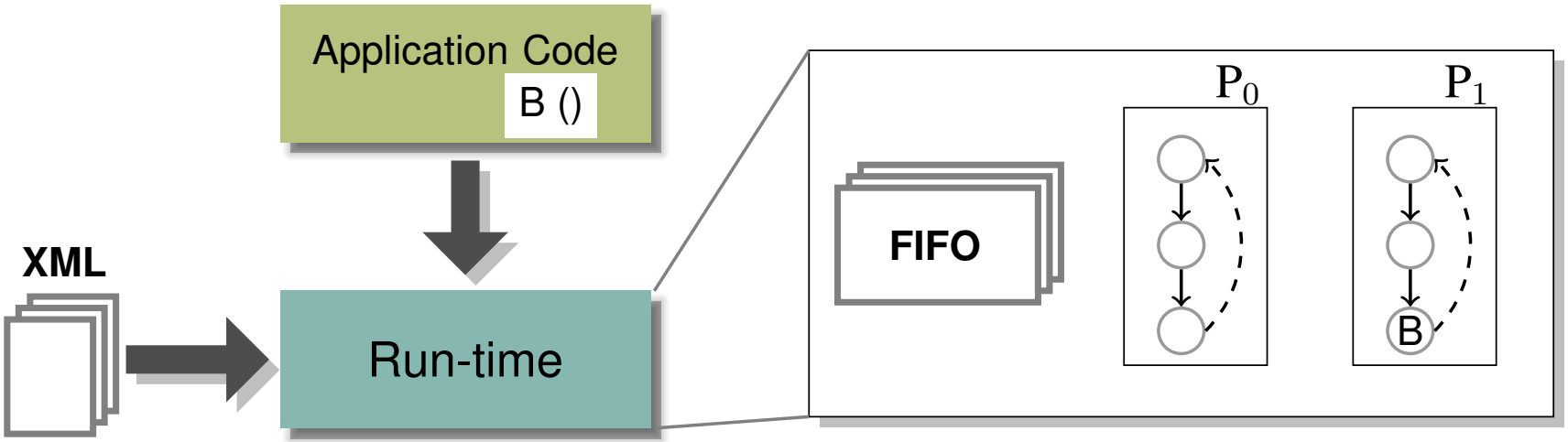
Our Framework

Runtime



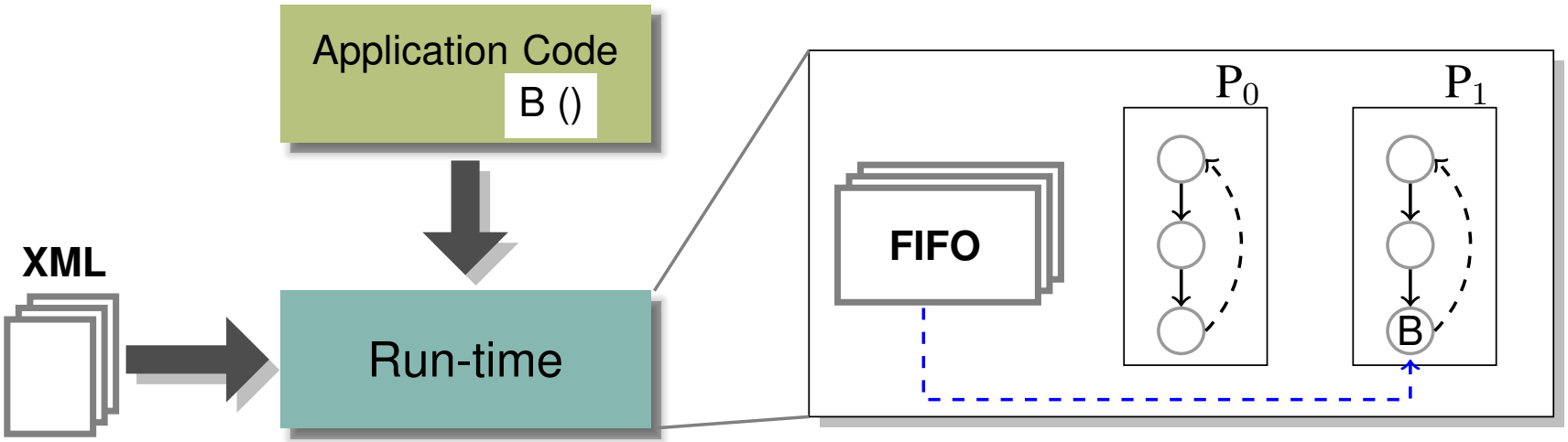
Our Framework

Runtime



Our Framework

Runtime



Our Framework

StreamExplorer

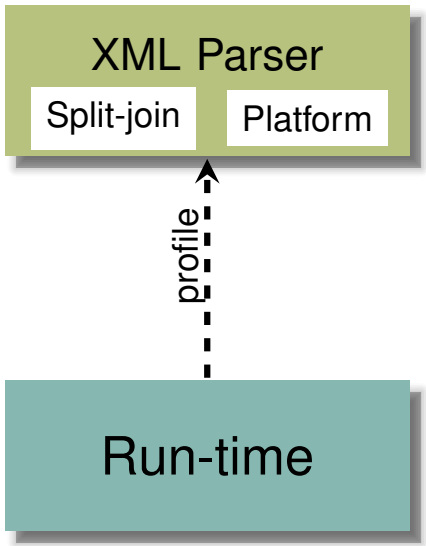
Our Framework

StreamExplorer

Run-time

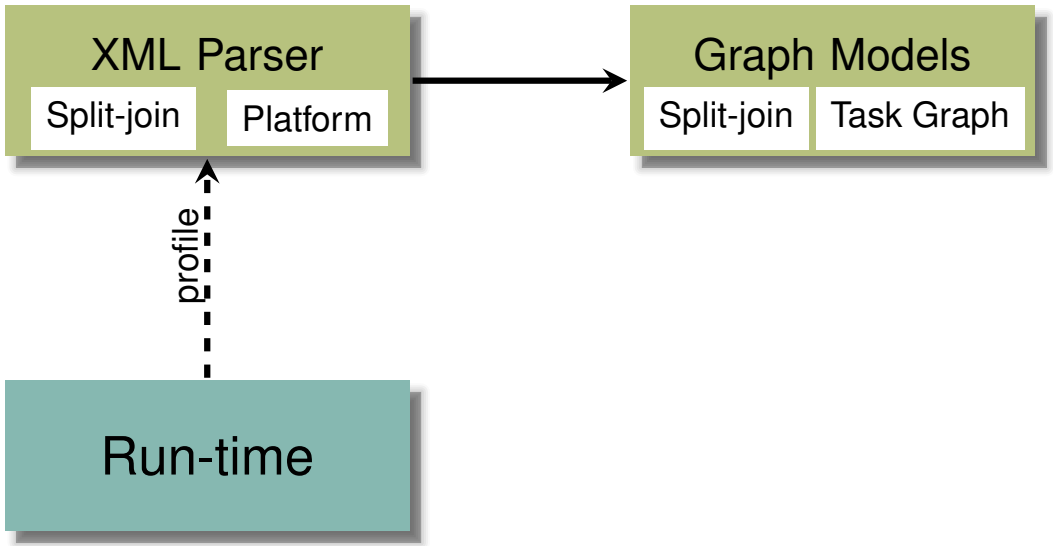
Our Framework

StreamExplorer



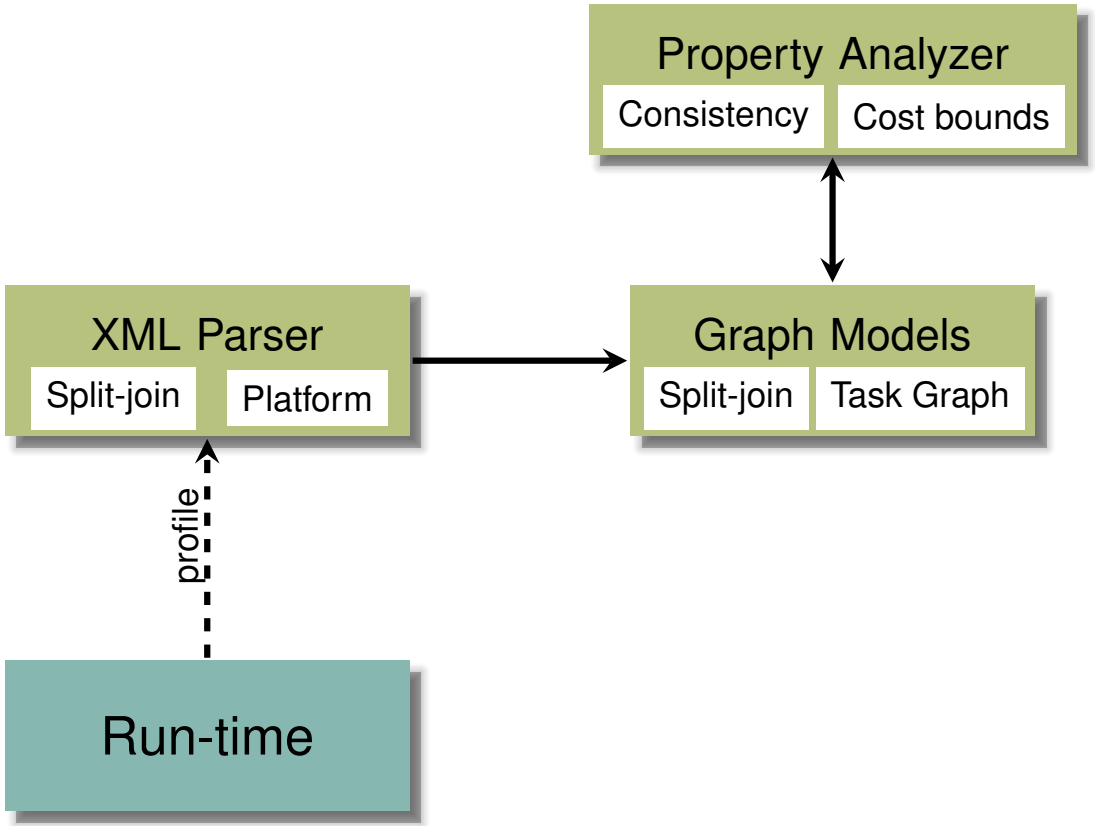
Our Framework

StreamExplorer



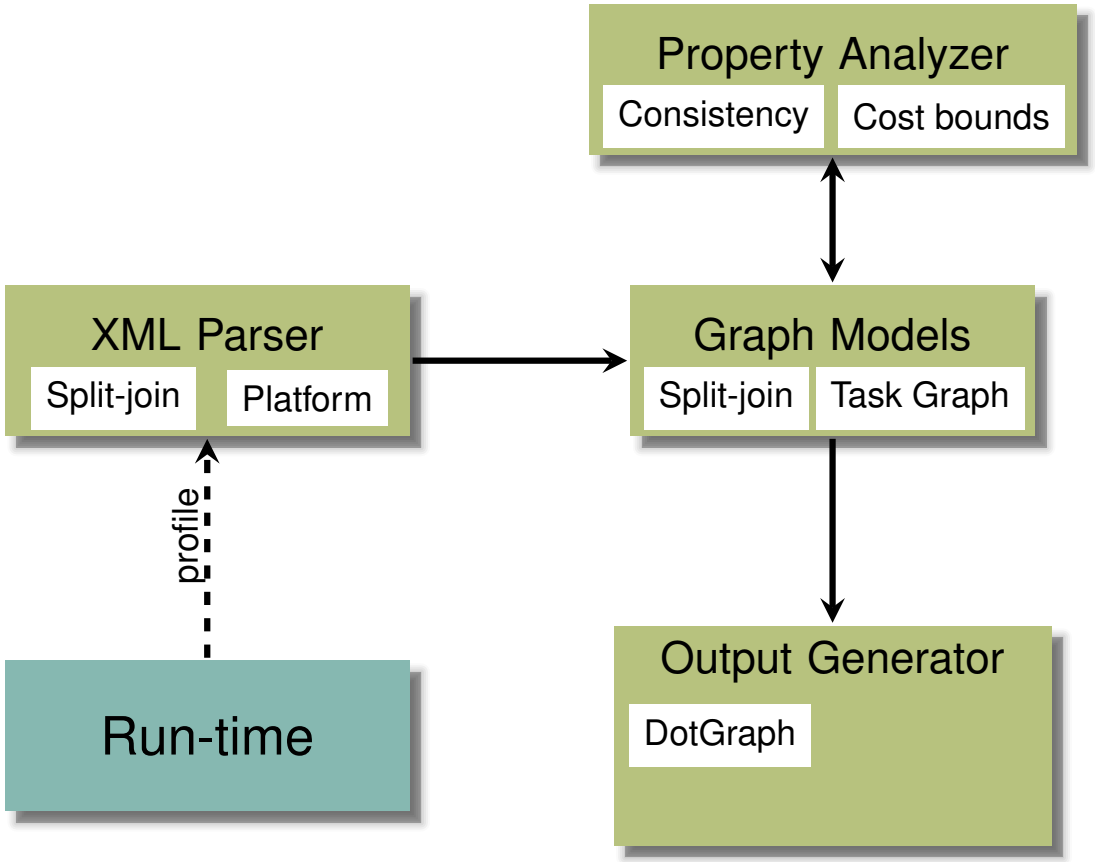
Our Framework

StreamExplorer



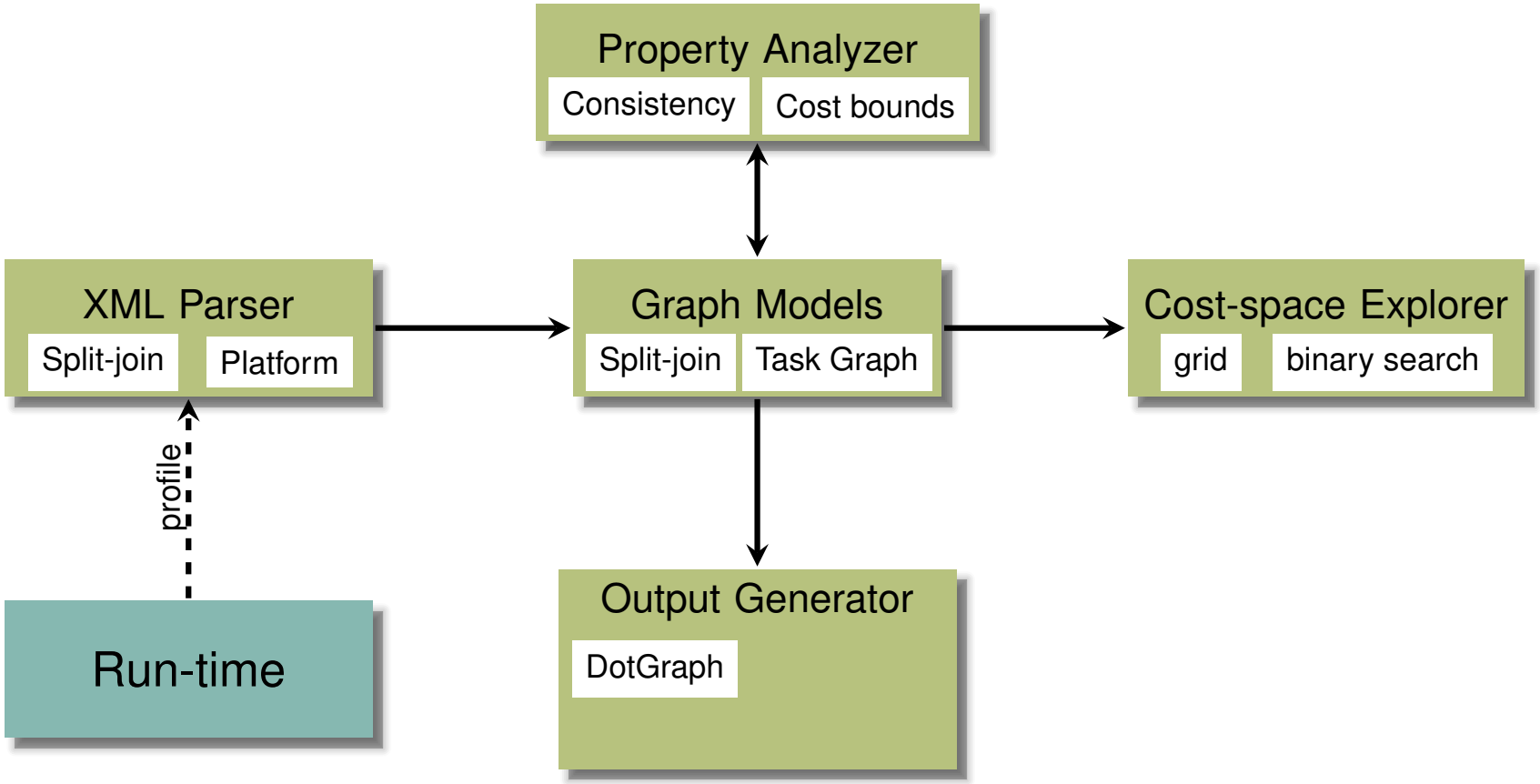
Our Framework

StreamExplorer



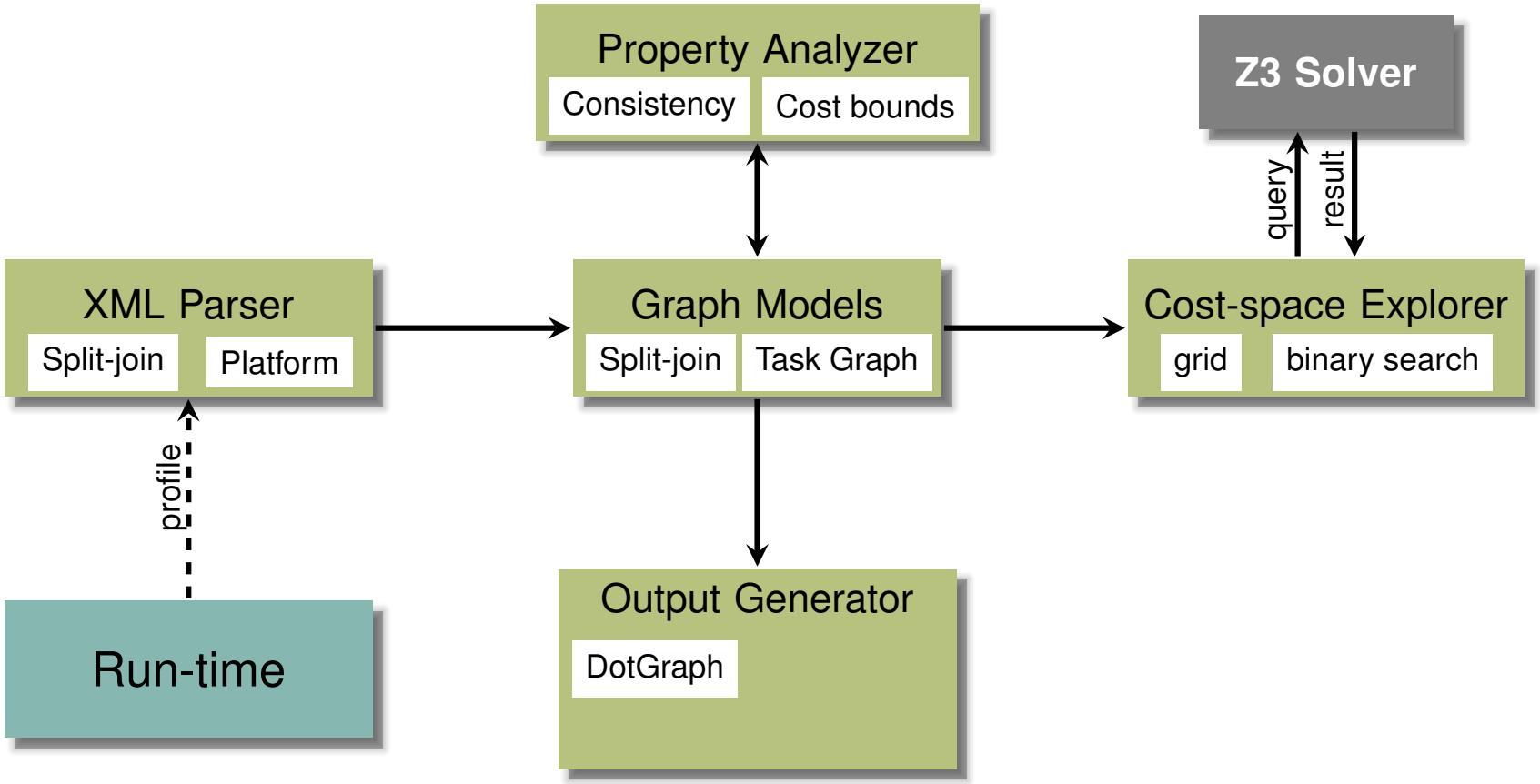
Our Framework

StreamExplorer



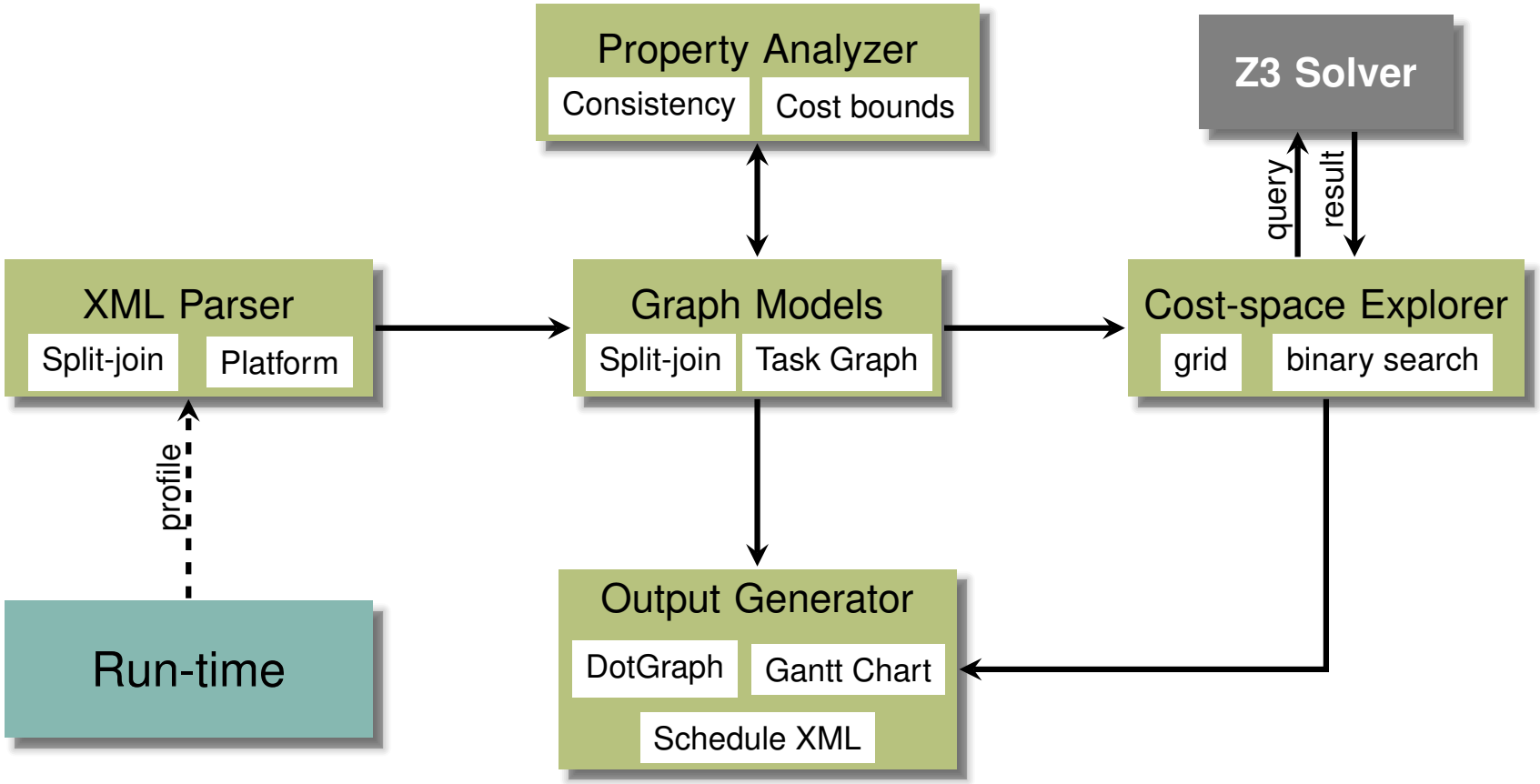
Our Framework

StreamExplorer



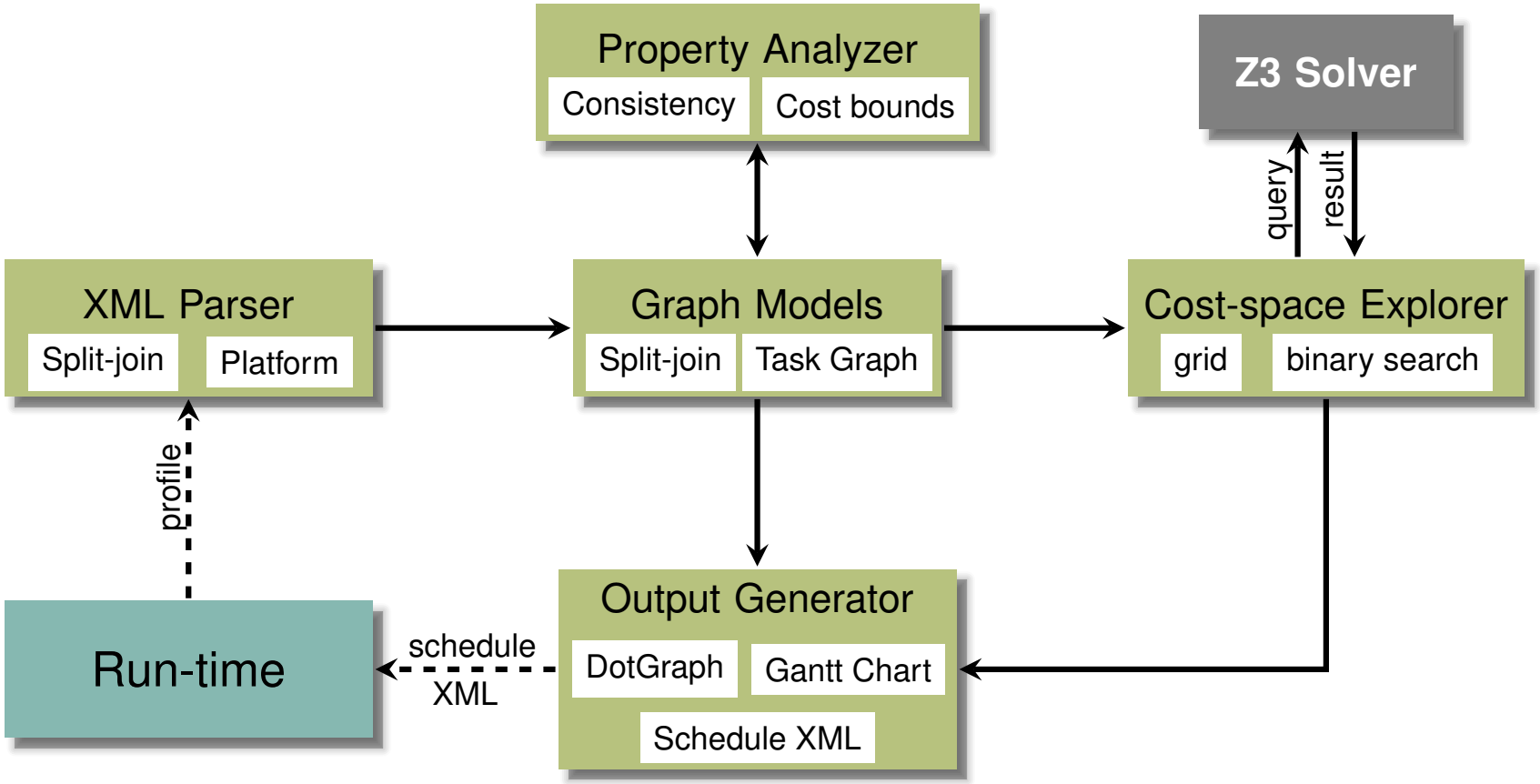
Our Framework

StreamExplorer



Our Framework

StreamExplorer



Our Framework

- **StreamExplorer**

Our Framework

- **StreamExplorer**
  - Written in Java

Our Framework

- **StreamExplorer**
  - Written in Java
  - 32k+ lines of Code.

Our Framework

- **StreamExplorer**
  - Written in Java
  - 32k+ lines of Code.
- **Runtime**
  - Written in C++

Our Framework

- **StreamExplorer**
  - Written in Java
  - 32k+ lines of Code.
- **Runtime**
  - Written in C++
  - 14k+ lines of Code.

Overview

Conclusions and Future Work

Conclusions:

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**
- **Automated design flow** for distributed memory

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**
- **Automated design flow** for distributed memory

Future Work:

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**
- **Automated design flow** for distributed memory

Future Work:

- Spread actor over **multiple clusters**

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**
- **Automated design flow** for distributed memory

Future Work:

- Spread actor over **multiple clusters**
- **Network route** selection and scheduling

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**
- **Automated design flow** for distributed memory

Future Work:

- Spread actor over **multiple clusters**
- **Network route** selection and scheduling
- **Pipelined** scheduling

Conclusions and Future Work

Conclusions:

- **Symmetry elimination** finds better solutions
- Combined Optimization with **Communication modeling**
- **Automated design flow** for distributed memory

Future Work:

- Spread actor over **multiple clusters**
- **Network route** selection and scheduling
- **Pipelined** scheduling
- Scheduling under **uncertainty**

Contributions

- P. Tendulkar, P. Poplavko, and O. Maler. "Symmetry Breaking for Multi-criteria Mapping and Scheduling on Multicores". In: *FORMATS*. 2013
- P. Tendulkar, P. Poplavko, I. Galanommatis, and O. Maler. "Many-Core Scheduling of Data Parallel Applications using SMT Solvers". In: *DSD*. 2014
- P. Tendulkar, P. Poplavko, and O. Maler. *Strictly Periodic Scheduling of Acyclic Synchronous Dataflow Graphs using SMT Solvers*. Tech. rep. Verimag Research Report, 2014

- P. Tendulkar and S. Stuijk. "A Case Study into Predictable and Composable MPSoC Reconfiguration". In: *IPDPS RAW Workshop*. 2013
- S. Saidi, P. Tendulkar, T. Lepley, and O. Maler. "Optimizing Explicit Data Transfers for Data Parallel Applications on the Cell Architecture". In: *ACM TACO* (2012)
- S. Saidi, P. Tendulkar, T. Lepley, and O. Maler. "Optimizing two-dimensional DMA transfers for scratchpad Based MPSoCs platforms". In: *Microprocessors and Microsystems* (2013)

Thank You

Questions?

Overview

Split-Join Graphs

Hypothesis supported by StreamIt:[1]

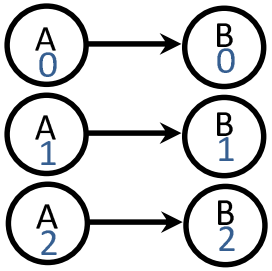
- Total 763 actors analyzed in various applications
  - 94% are stateless
  - **6%** are stateful
    - 45% have states due to algorithm
    - 55% have **avoidable** states
- Odd rates exists but are **rare**
  CD-DAT benchmark used as an example

  Converts CD audio (44.1 kHz) to digital audio tape (48 kHz)

[1] W. Thies and S. Amarasinghe. "An Empirical Characterization of Stream Programs and Its Implications for Language and Compiler Design". In: PACT. 2010

Overview

Proof Sketch

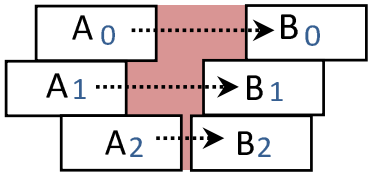


modify a feasible schedule such that:

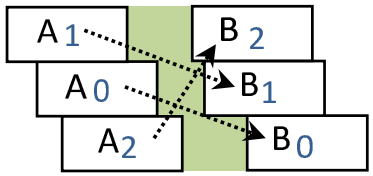
$s(v_0) \leq s(v_1) \leq s(v_2) \leq \ldots$

prove that precedence constraints are satisfied

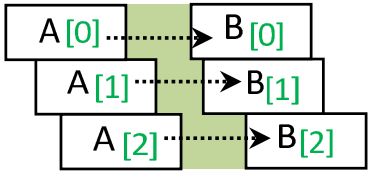
here: for neutral channels ($\alpha = 1$), unfolded to (v_h, v'_h)



lexicographic order

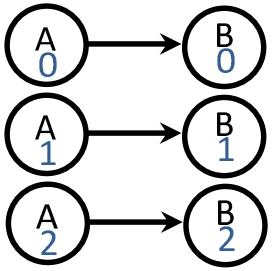


start-time compatible



new hier. index;
new precedence relation

Proof Sketch

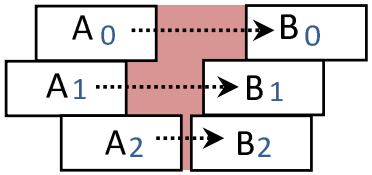


modify a feasible schedule such that:

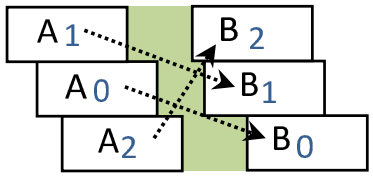
$s(v_0) \leq s(v_1) \leq s(v_2) \leq \ldots$

prove that precedence constraints are satisfied

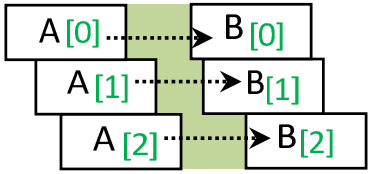
here: for neutral channels ($\alpha = 1$), unfolded to (v_h, v'_h)



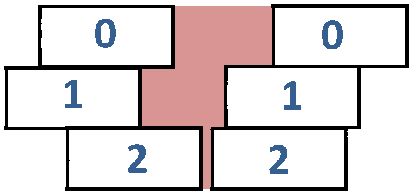
lexicographic order

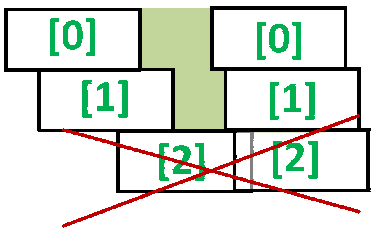


start-time compatible

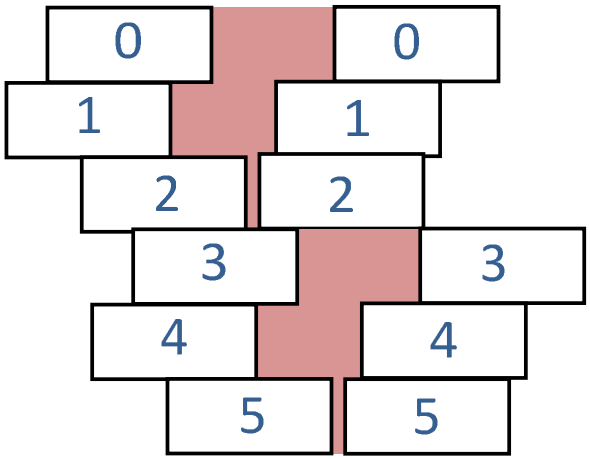


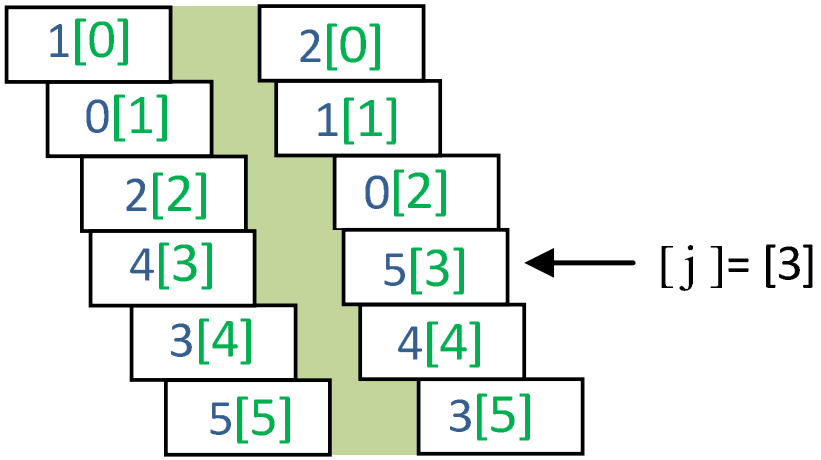
new hier. index;
new precedence relation





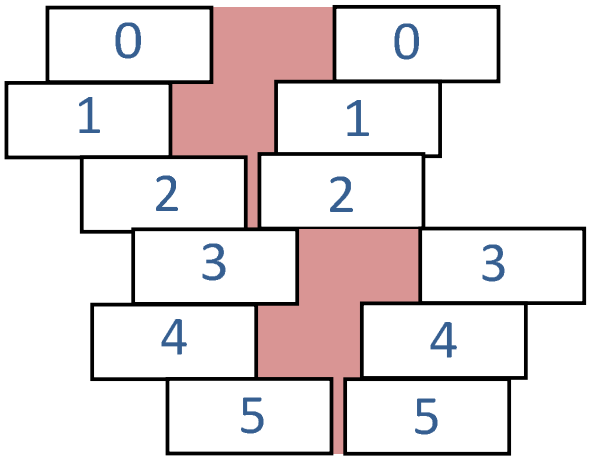
Proof Sketch

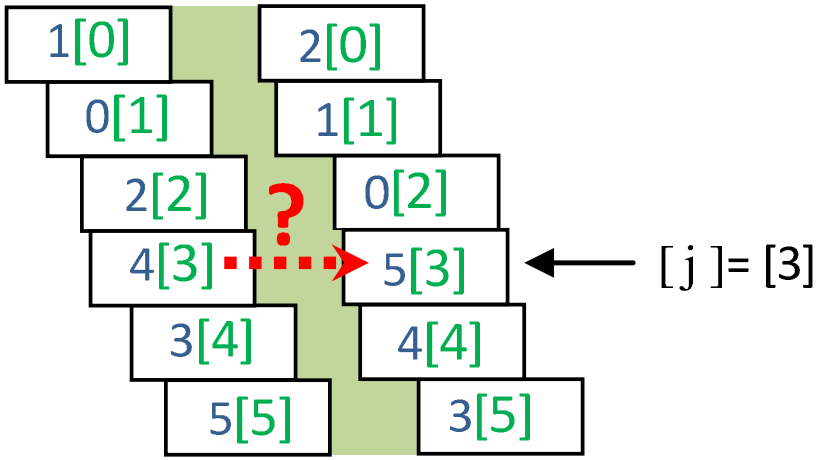




take successor $[\ j\ ]$

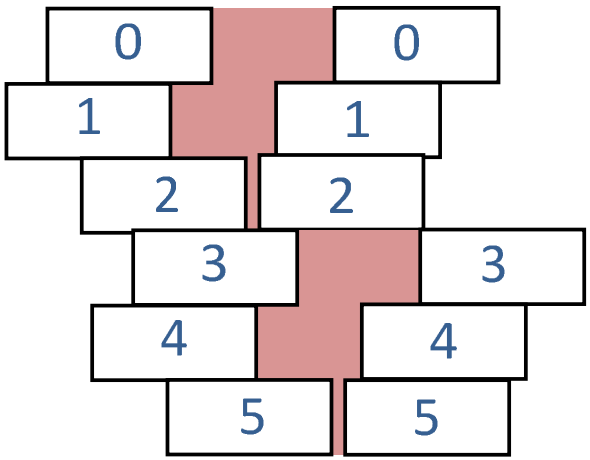
Proof Sketch

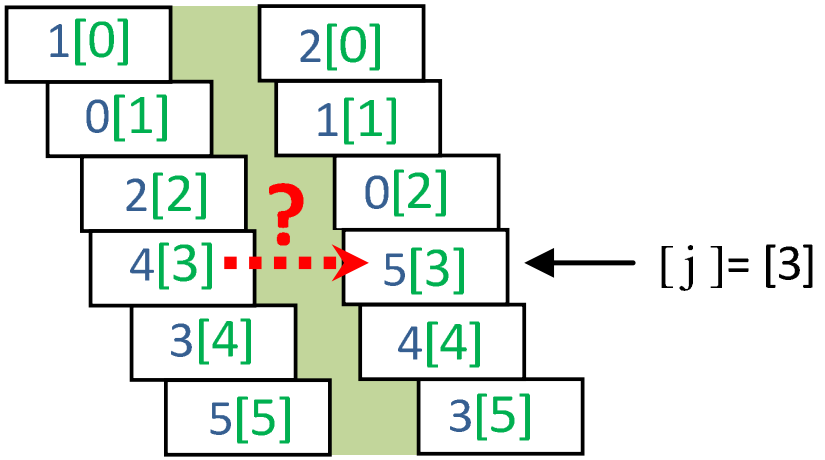




take successor $[\ j\ ]$

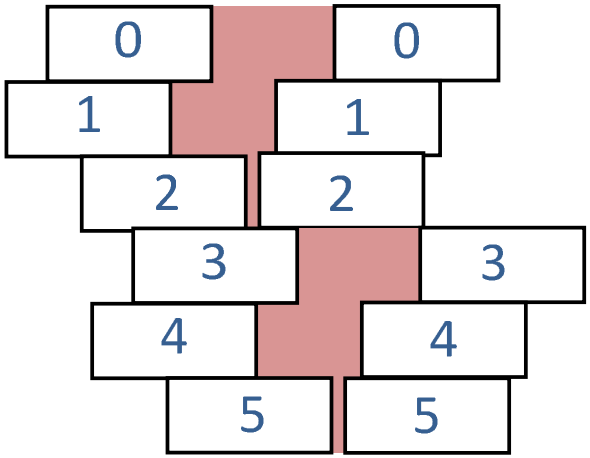
Proof Sketch

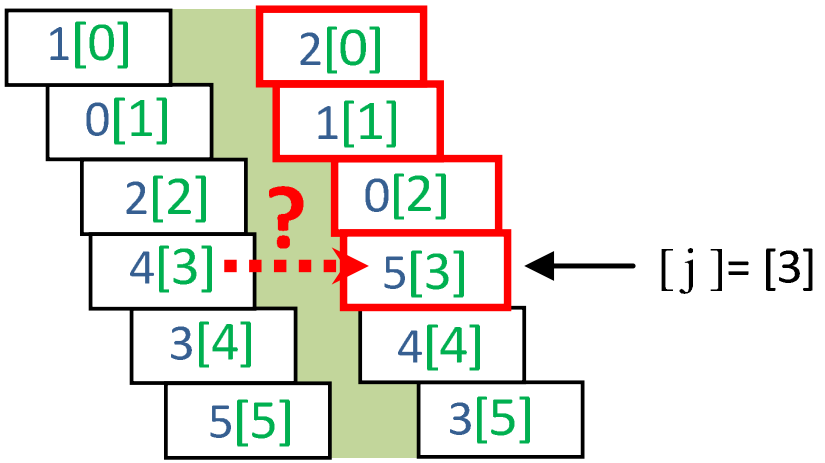




take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors

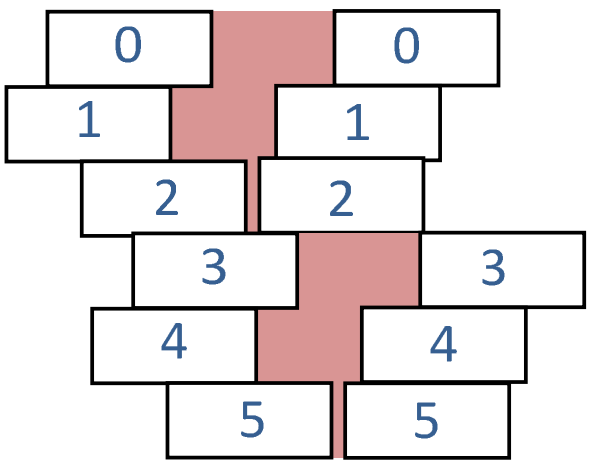
Proof Sketch

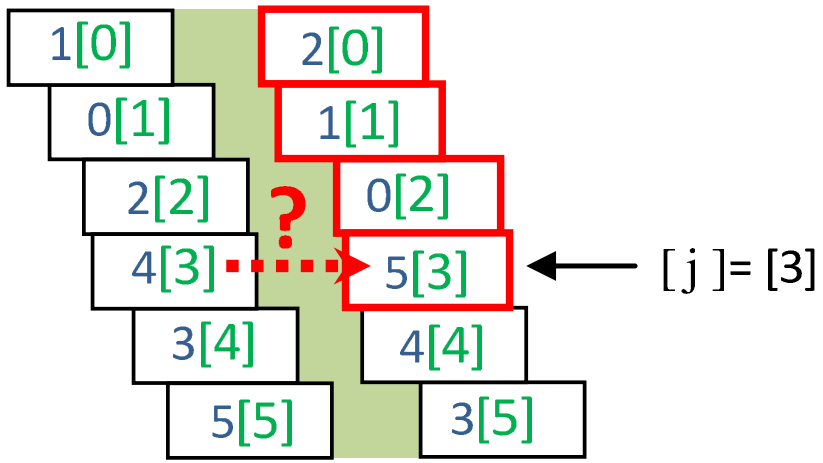




take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors

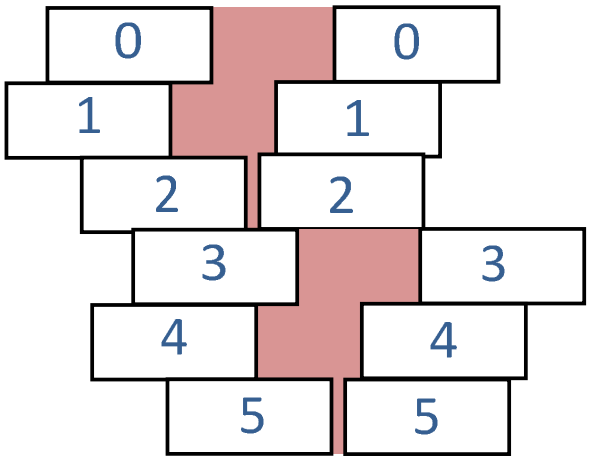
Proof Sketch

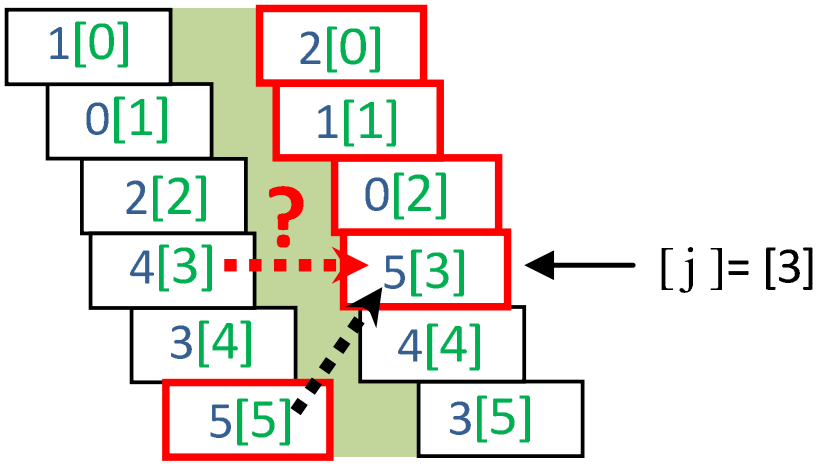




take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors their original predecessors finish before successor $[\ j\ ]$:

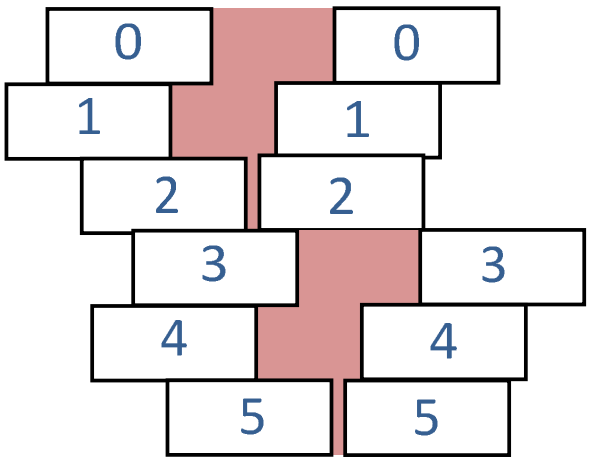
Proof Sketch

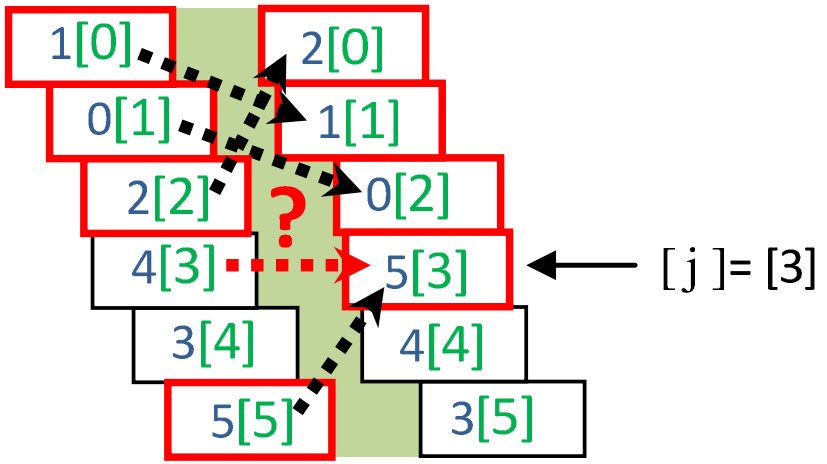




take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors their original predecessors finish before successor $[\ j\ ]$:

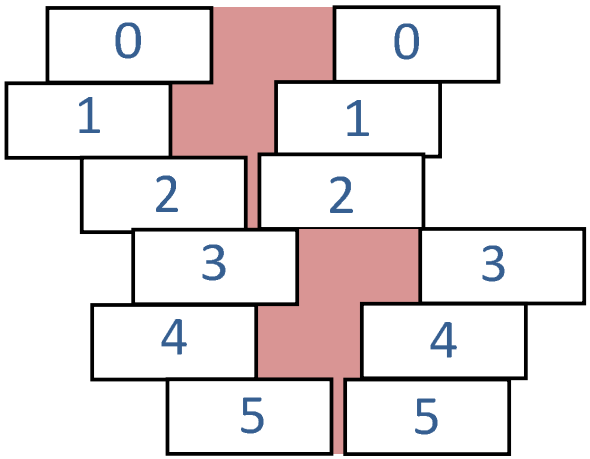
Proof Sketch

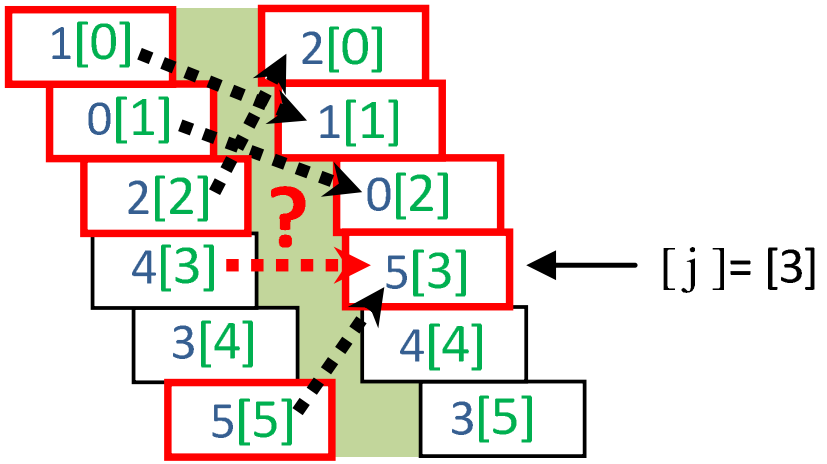




take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors their original predecessors finish before successor $[\ j\ ]$:

Proof Sketch



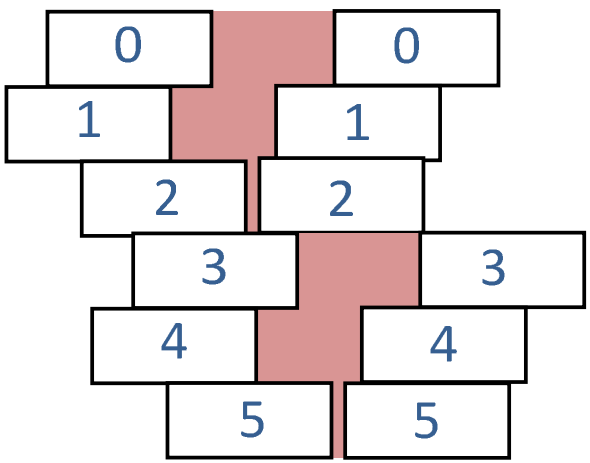


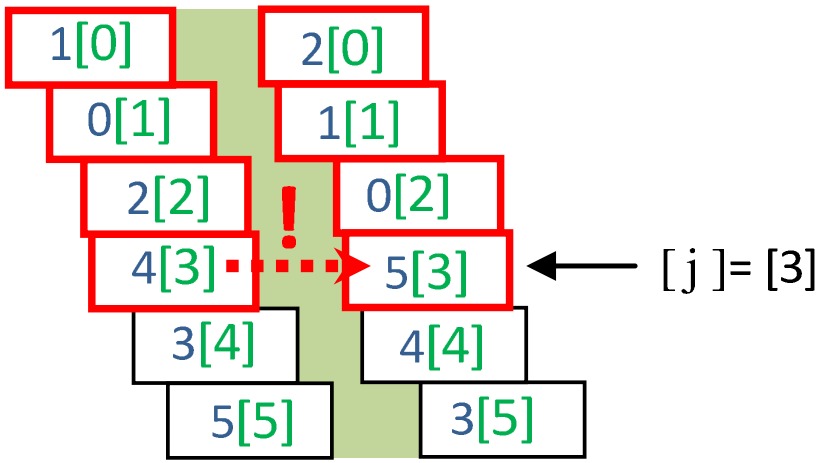
take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors

their original predecessors finish before successor $[\ j\ ]$:

$j+1$ predecessors finish before, hence the earliest $j+1$ ones as well

Proof Sketch





take successor $[\ j\ ]$ by definition there exist $j+1$ same or earlier successors

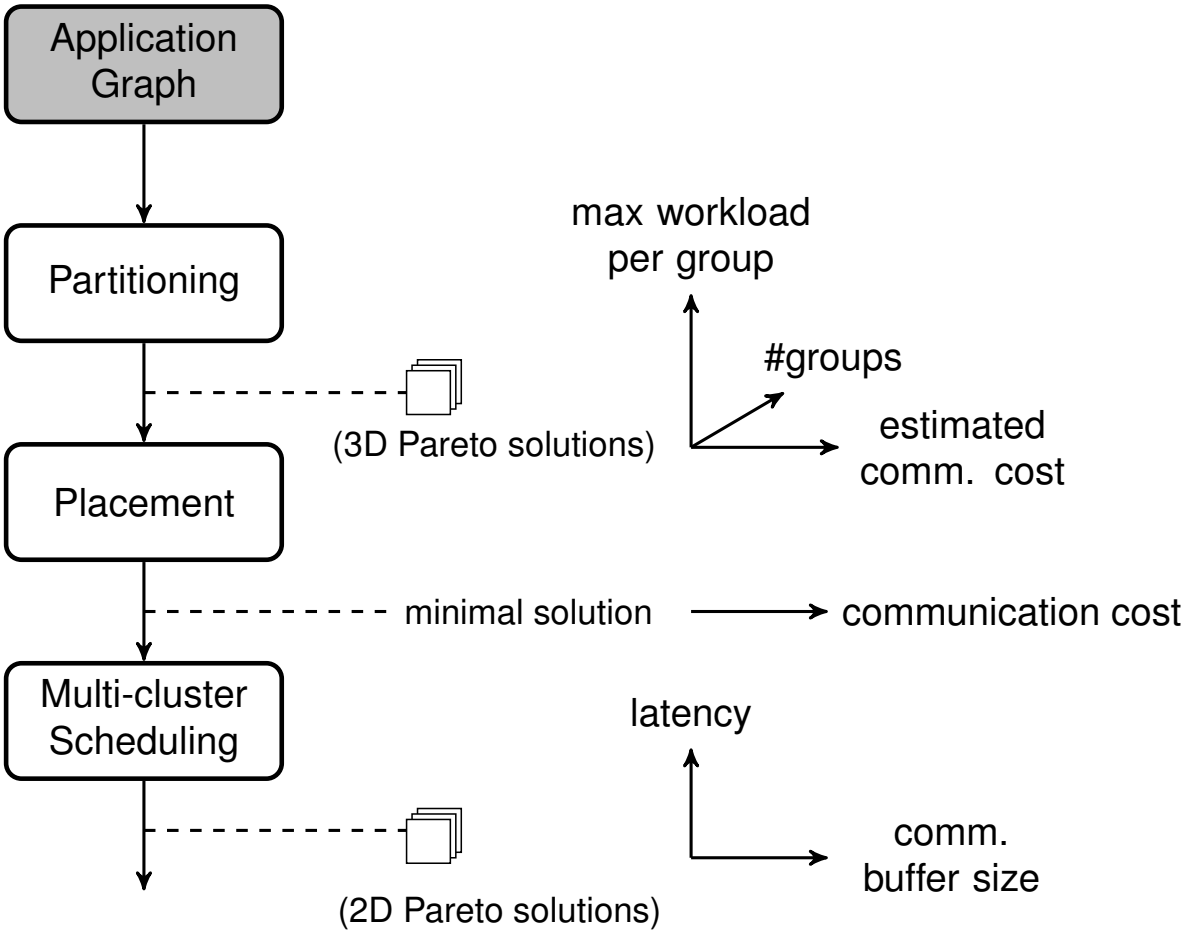
their original predecessors finish before successor $[\ j\ ]$:

$j+1$ predecessors finish before, hence the earliest $j+1$ ones as well

predecessor $[\ j\ ]$ finishes before successor $[\ j\ ]$

Overview

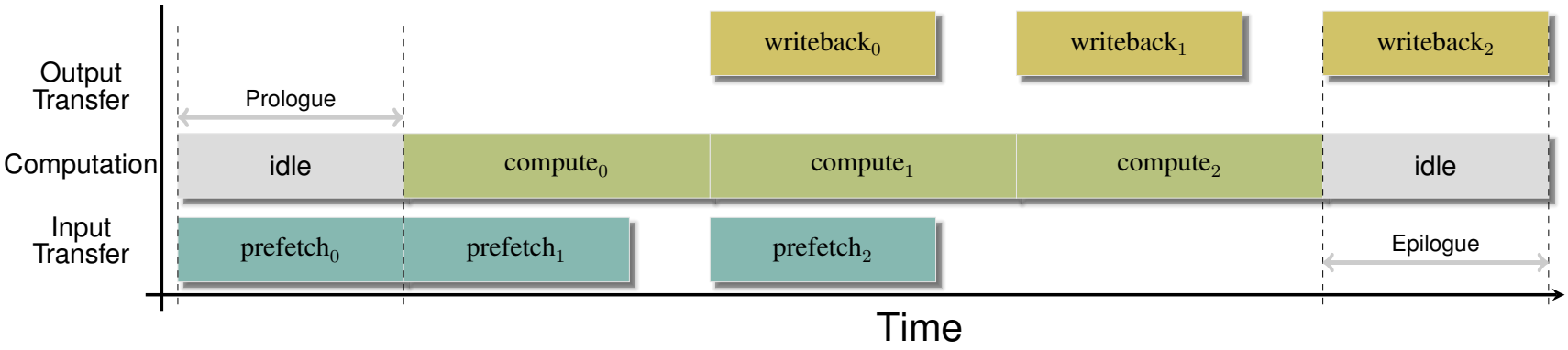
Design Flow


Application Graph
Partitioning
Placement
Multi-cluster Scheduling
max workload per group
#groups
estimated comm. cost
(3D Pareto solutions)
minimal solution
communication cost
latency
comm. buffer size
(2D Pareto solutions)

Overview

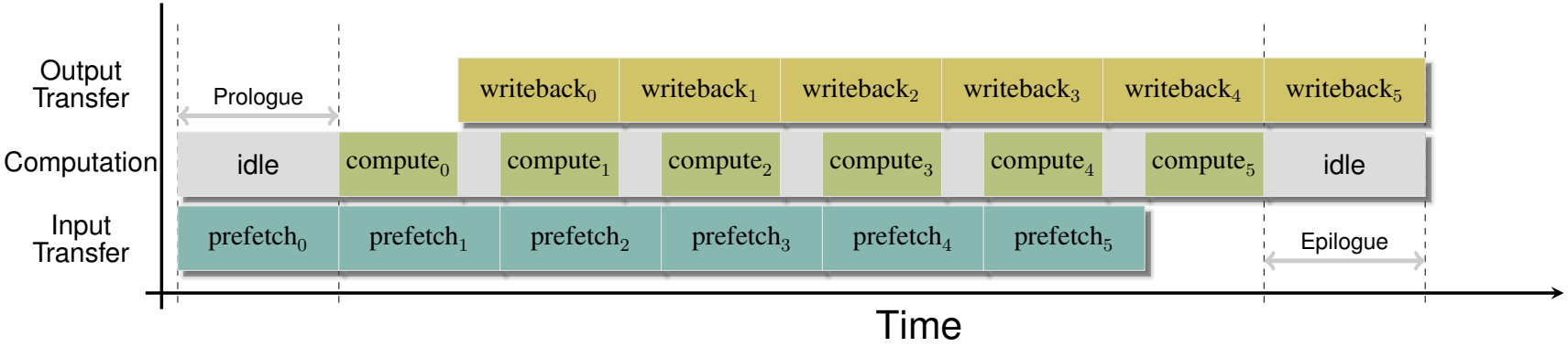
Buffering Algorithm

For architectures with DMA and limited local memory

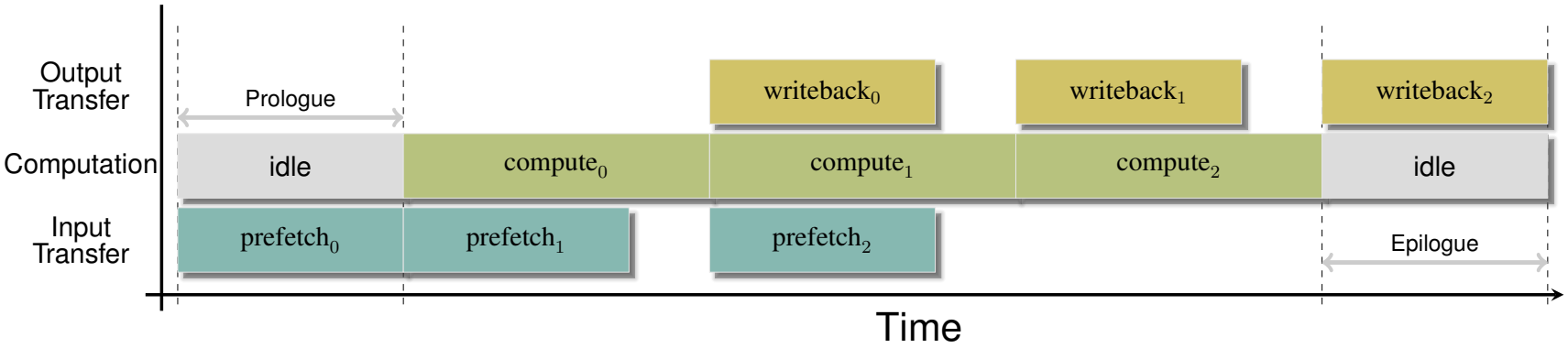


Data transfer granularity

Transfer Granularity less than optimal

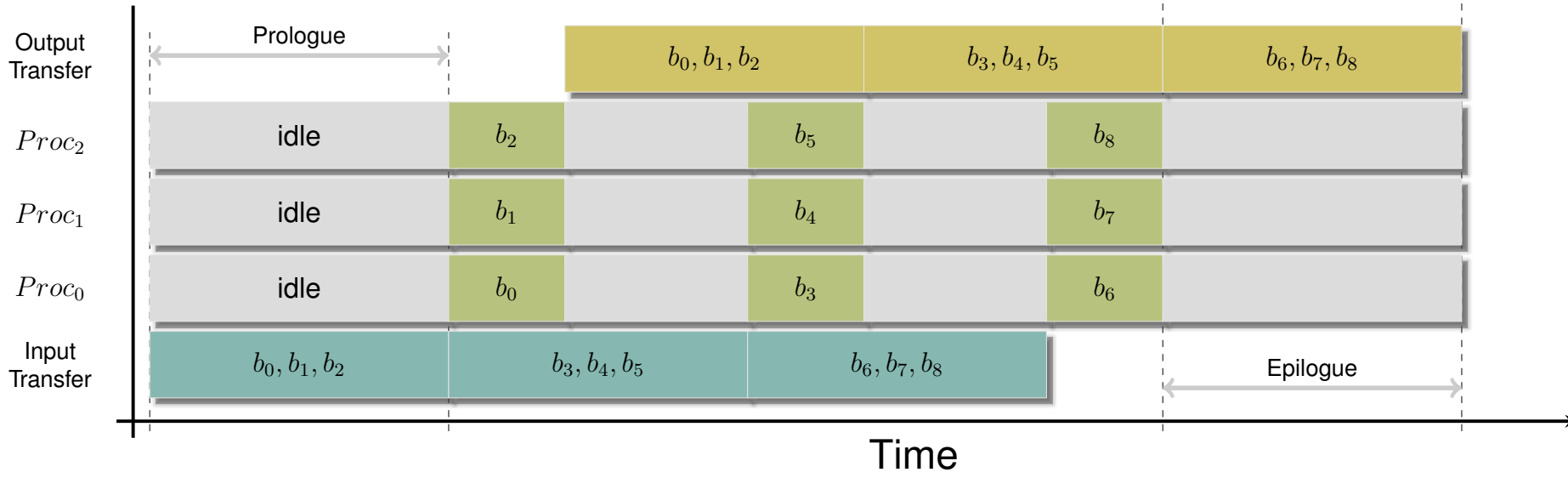


Transfer Granularity greater than optimal

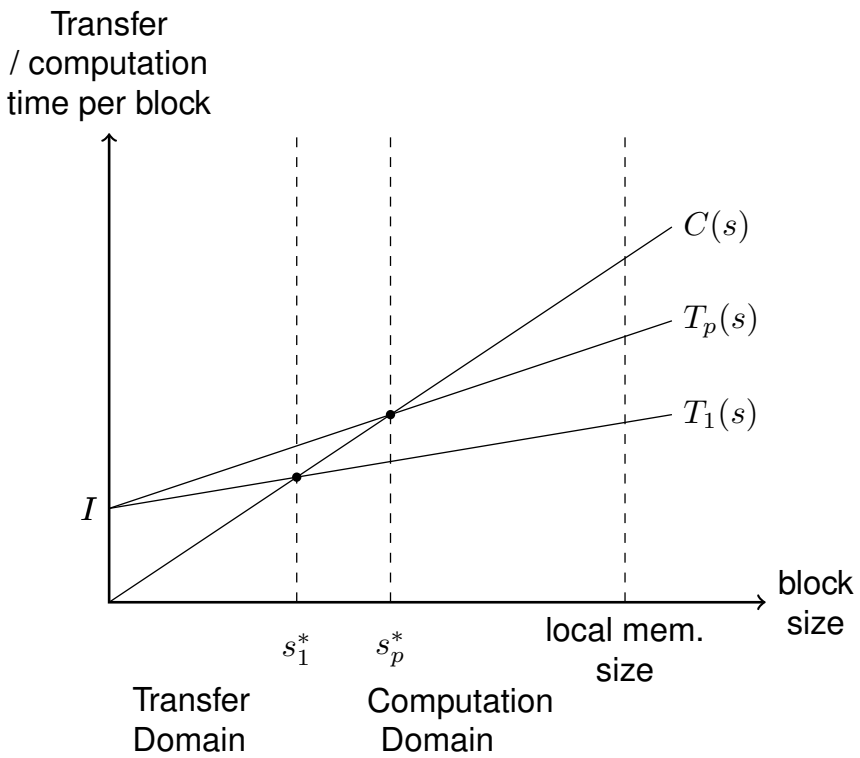


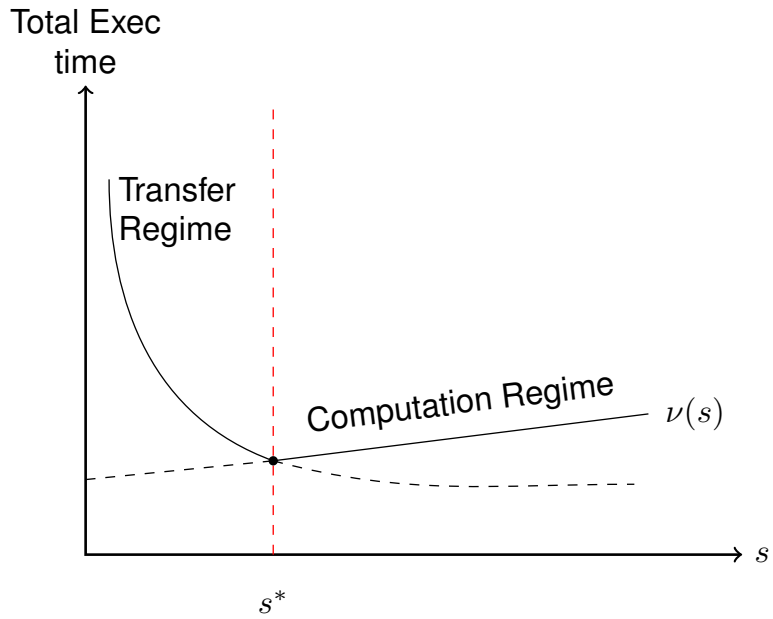
Data transfer granularity

Additional complexity with multiple processors



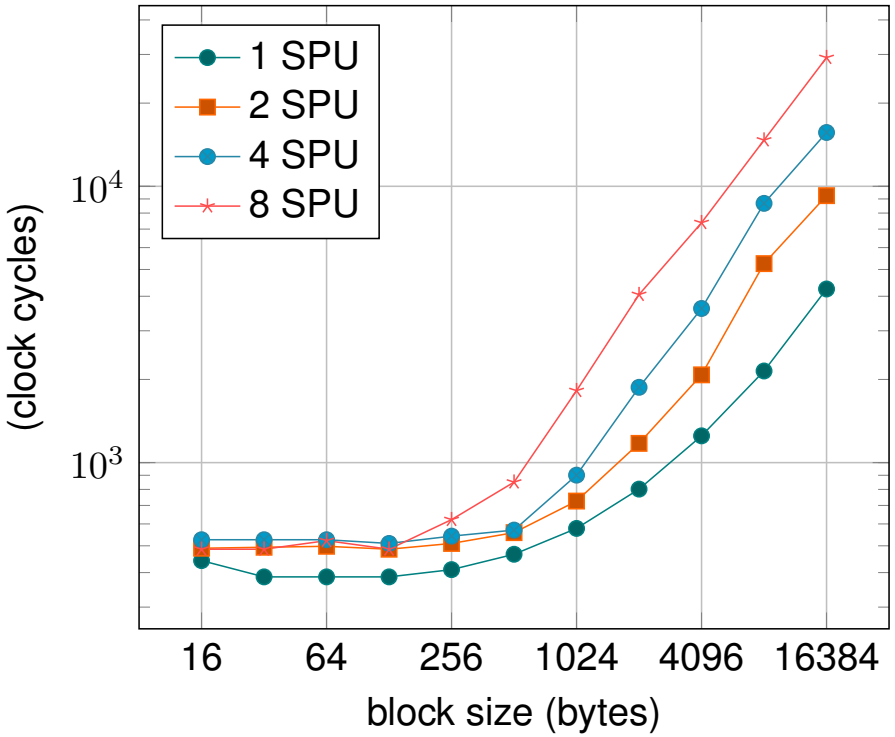
DMA transfer granularity optimization



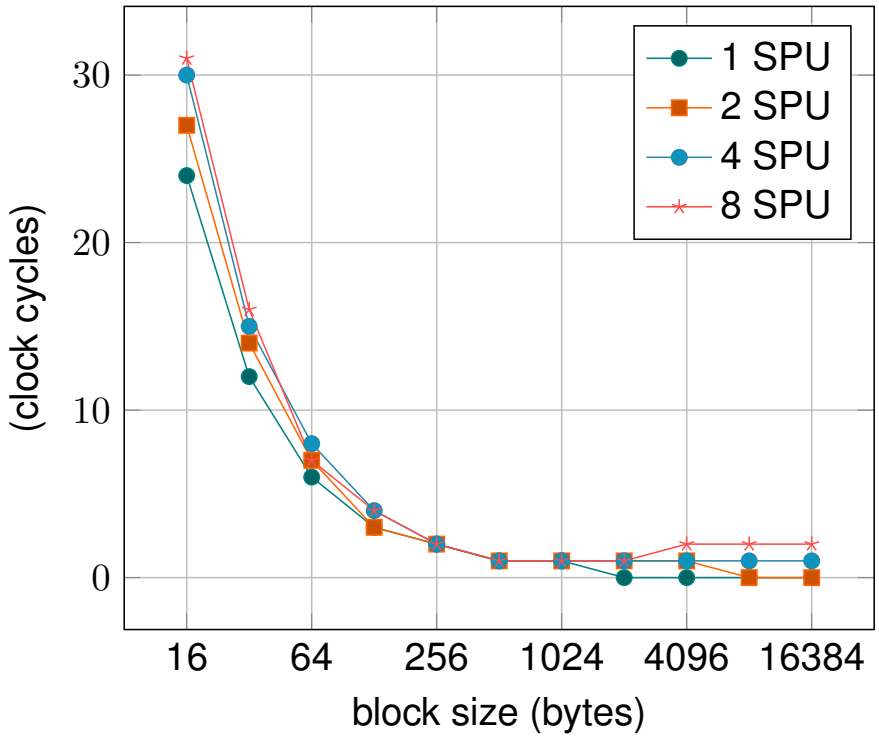


Experimental Evaluation

Characterization of DMA of IBM Cell B.E.:



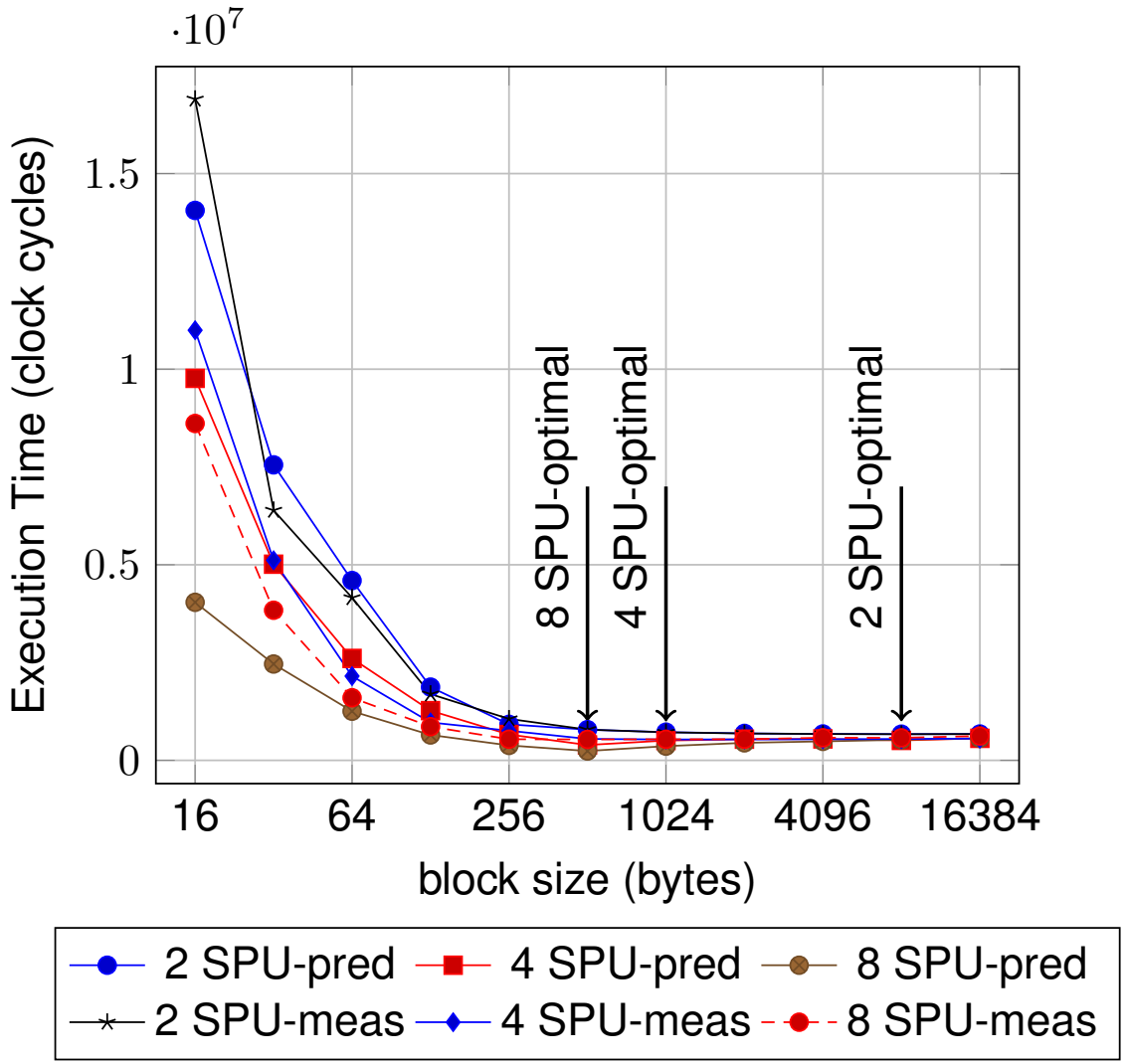
Time to read/write block



Cost per byte

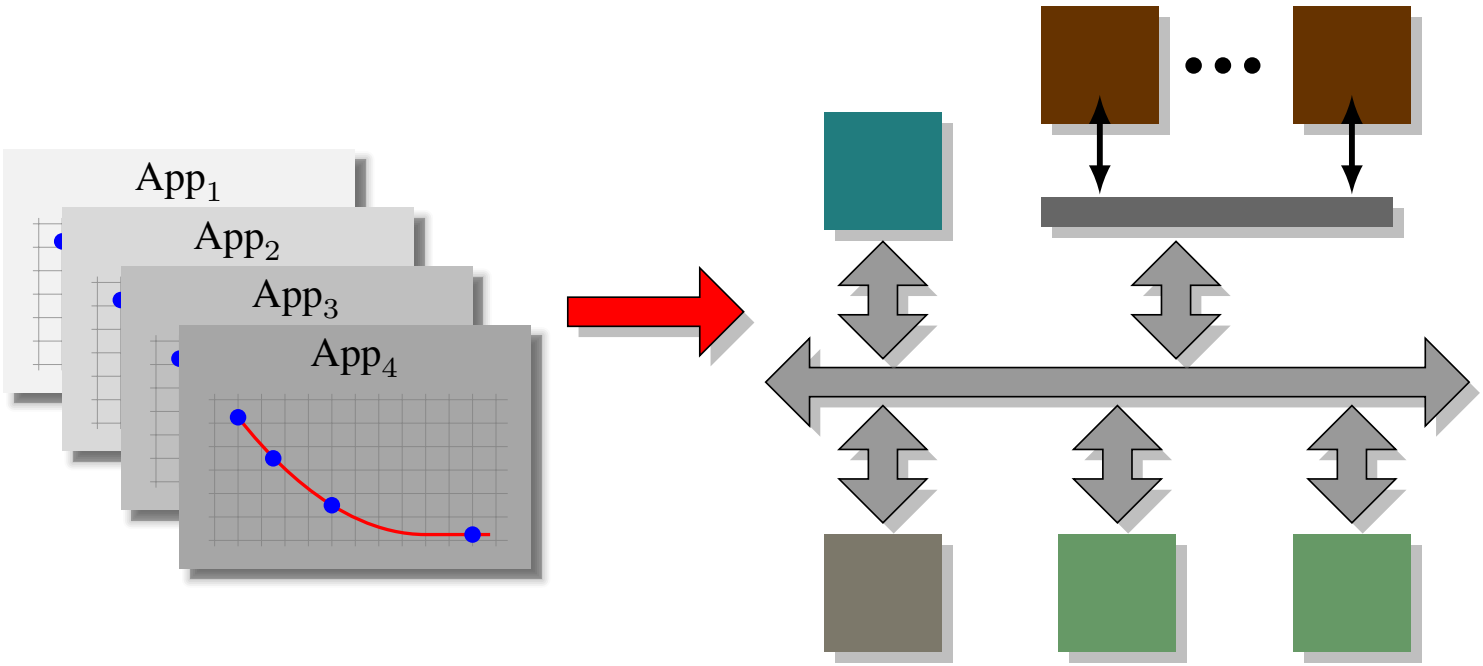
Experimental Evaluation

Synthetic Application Benchmark:



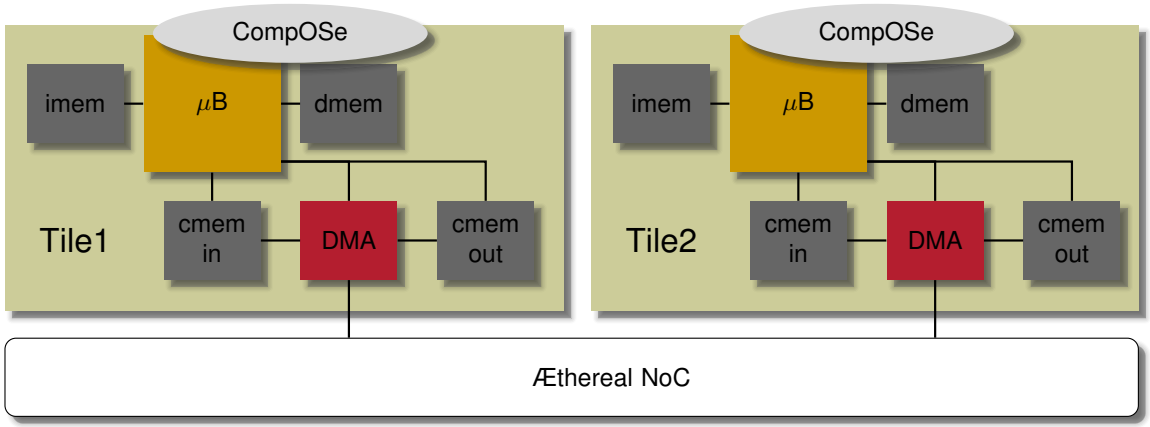
Overview

The context



- Multiple configurations for each application
- Applications start / stop dynamically
- **How to**:
  - select a configuration for each application?
  - re-configure the applications?

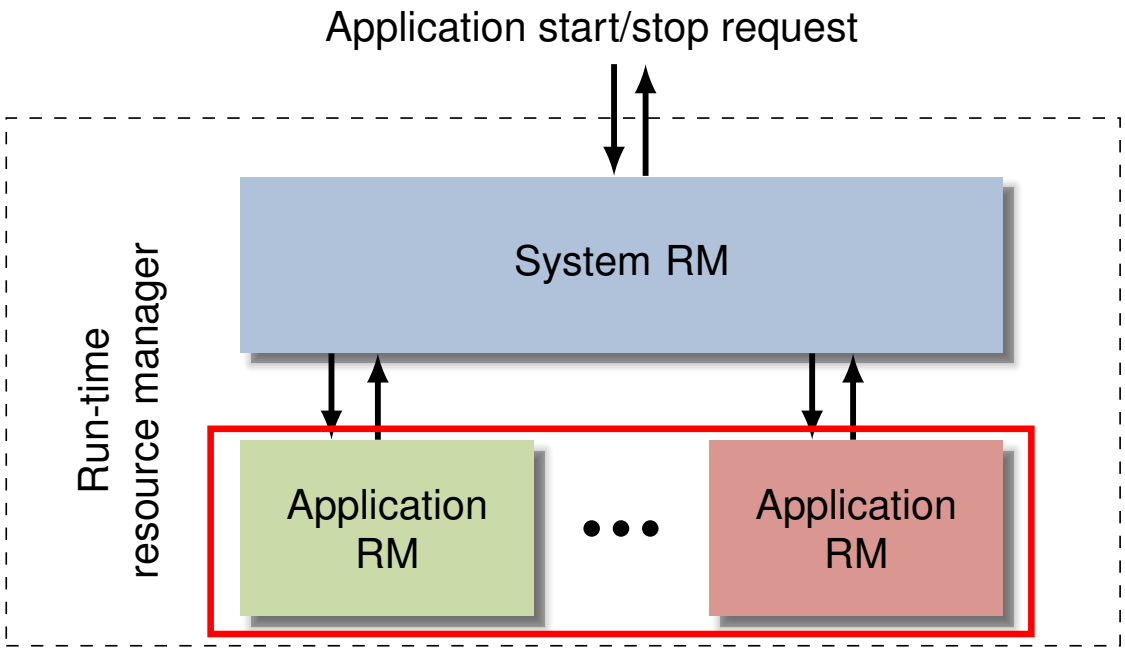
CompSoC platform



Features:

- CompOSe real-time operating system
- Predictable Æthereal network-on-chip
- TDM application scheduling for composability
- **composable**: The changes in an application don't affect other running applications

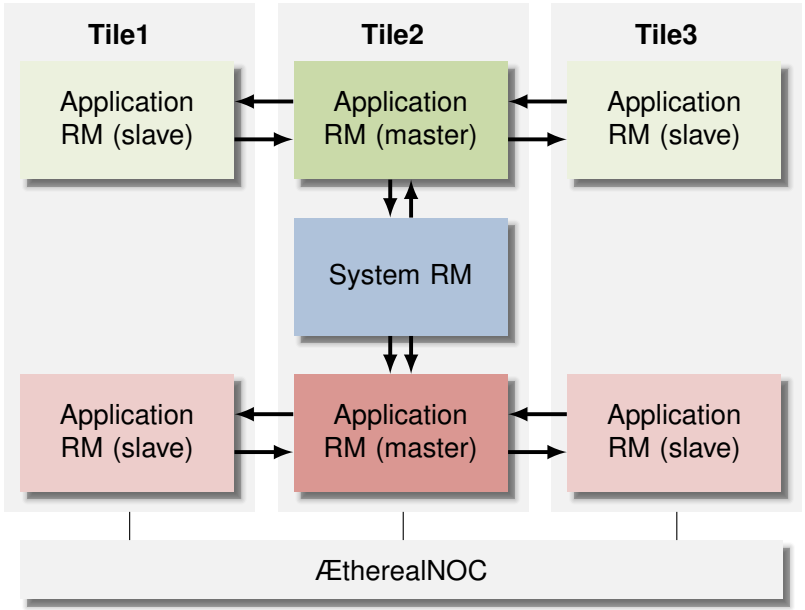
Resource Manager conceptual view



Resource manager Design:

- System RM : takes re-configuration decisions
- Application RM : implements re-configuration decisions

Resource Manager on the platform



Resource manager Implementation:

- System RM: is a separate application
- Application RM:
  - organized in master-slave(s) configuration
  - is a part of user application

Experiment with JPEG Decoder

Steps for Re-configuration:

1. Instruct application RM to reconfigure
2. Request removal of application from TDM
3. Remove application from TDM and ack.
4. Resize TDM allocation
5. Add new FIFO(s)
6. Remove old FIFO(s)
7. Add application to TDM
8. Inform system RM about completion