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

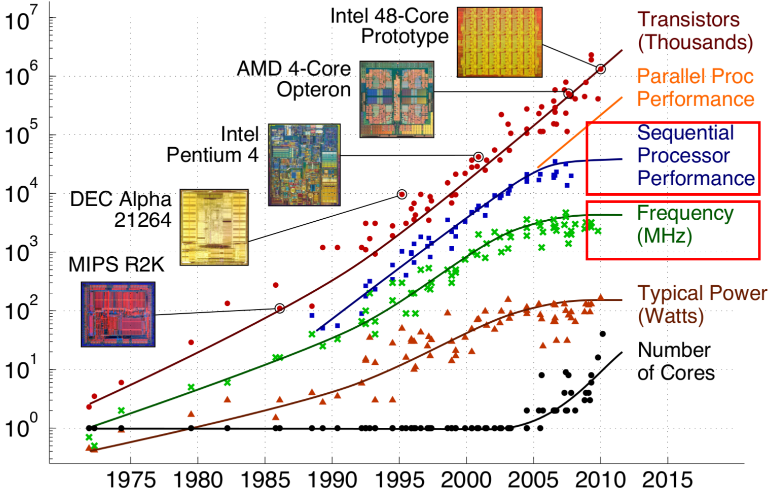
Phones

Cars

Smart-TV

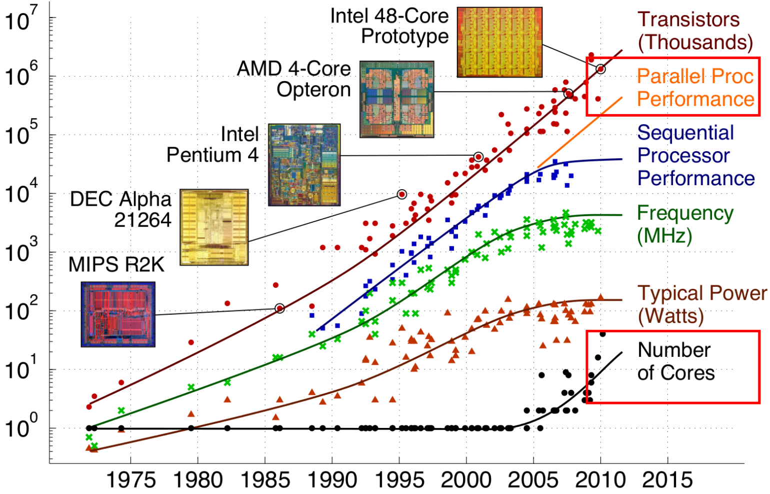
Cameras

Multi-core Processors Everywhere



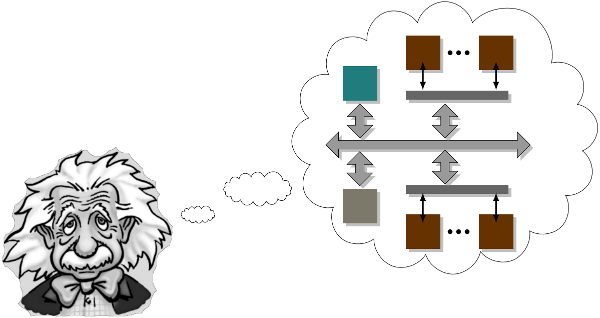
source : http://www.csl.cornell.edu/courses/ece5745/handouts.html

Multi-core Processors Everywhere



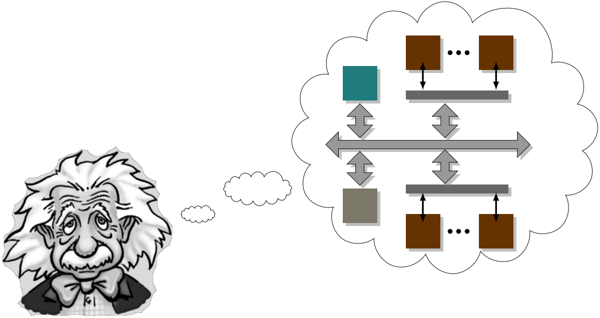
source : http://www.csl.cornell.edu/courses/ece5745/handouts.html

Multi-core systems



How To:

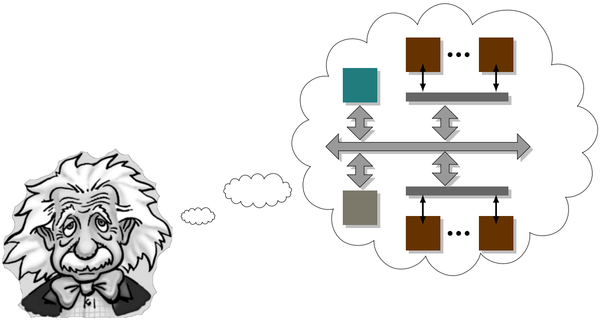
Multi-core systems



How To:

- **Deploy** the application to the platform

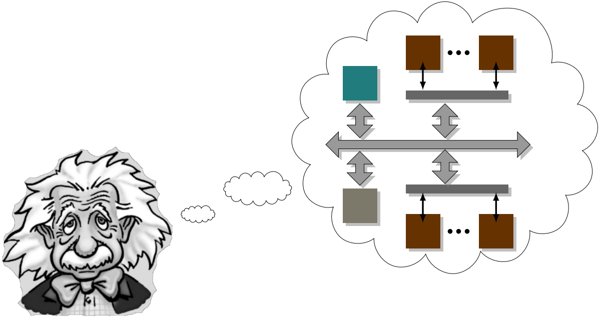
Multi-core systems



How To:

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

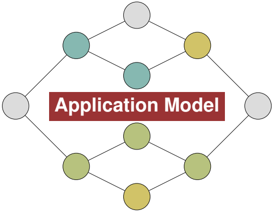
Multi-core systems



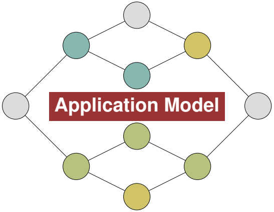
How To:

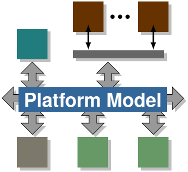
- **Deploy** the application to the platform
- Decide number of **processors** to use?
- **Allocate** tasks to processors and **schedule** them

Our Deployment Framework

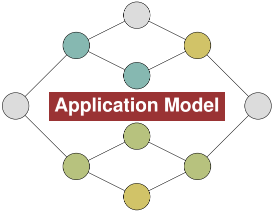


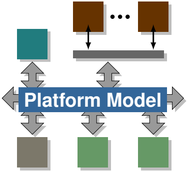
Our Deployment Framework

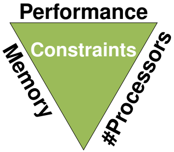




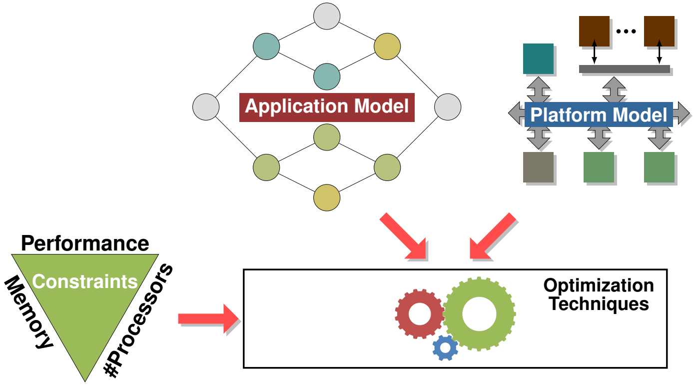
Our Deployment Framework



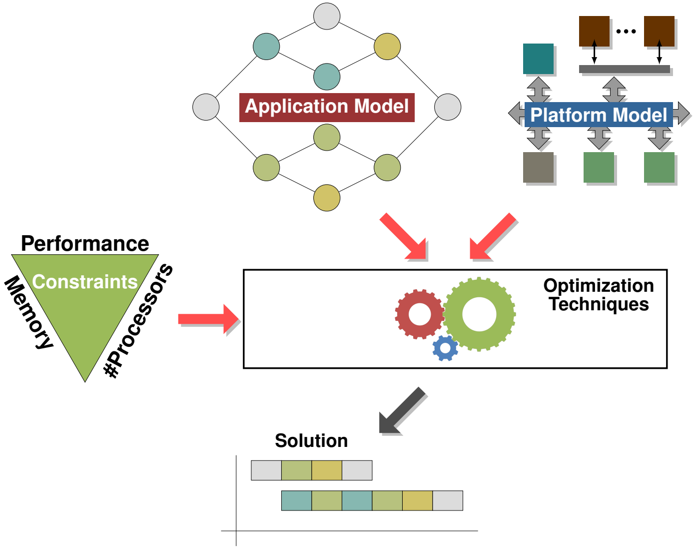




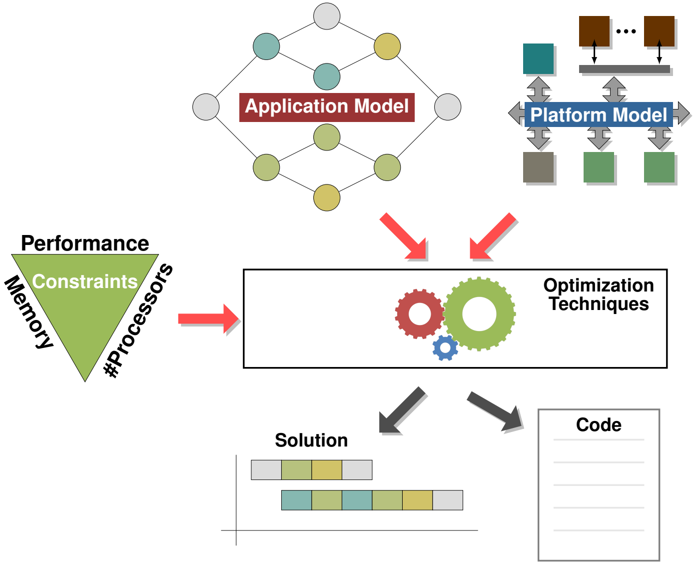
Our Deployment Framework



Our Deployment Framework

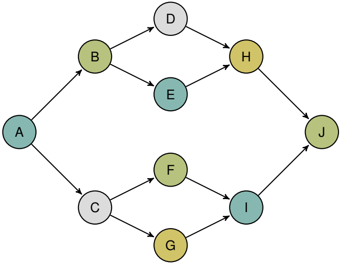


Our Deployment Framework



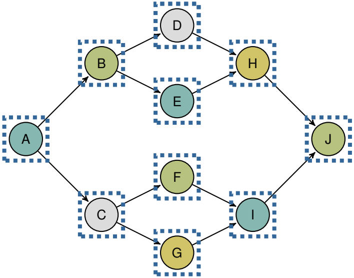
Application Model

Task Graph



Application Model

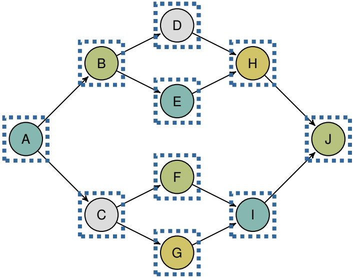
Task Graph



- **Tasks** : Software procedure

Application Model

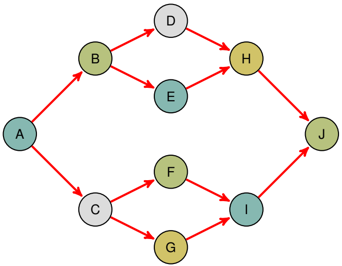
Task Graph



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

Application Model

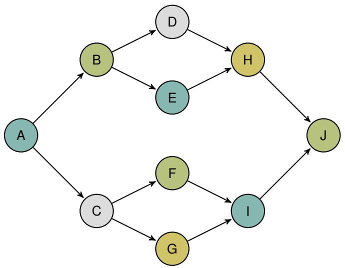
Task Graph



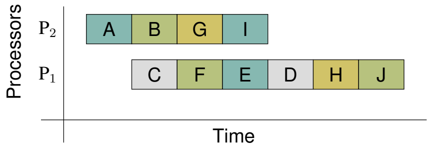
- **Tasks** : Software procedure
- **Edges** : Precedence relations

Deployment Problem

Task Graph



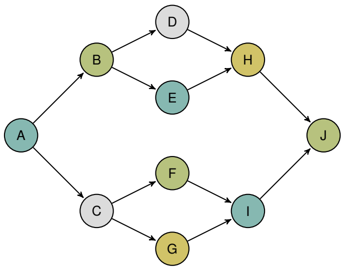
Deployment Solution



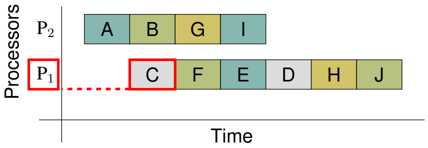
- **Tasks** : Software procedure
- **Edges** : Precedence relations

Deployment Problem

Task Graph



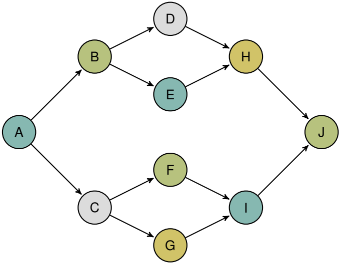
Deployment Solution



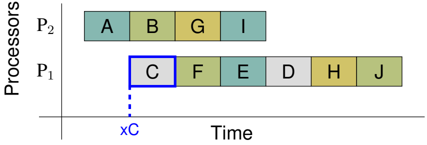
- **Tasks** : Software procedure
- **Edges** : Precedence relations
- **Mapping** : Task ⇒ Processor

Deployment Problem

Task Graph



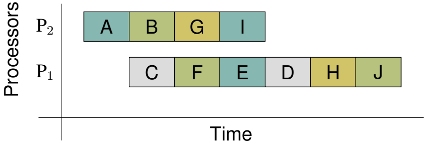
Deployment Solution



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

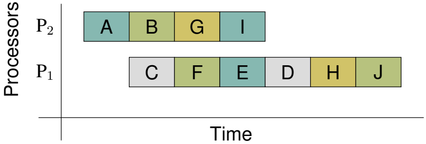
Deployment Problem

Solution1:

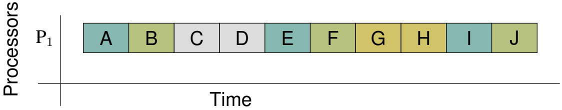


Deployment Problem

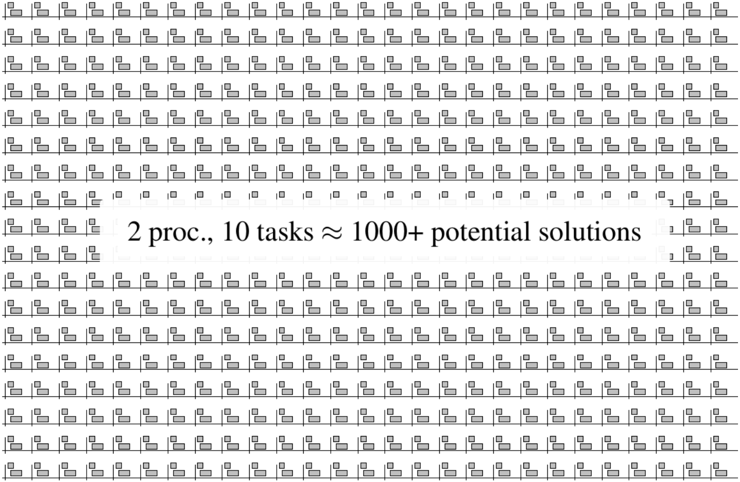
Solution1:



Solution2:



Solution space is large



Deployment problem

How to:

- find optimal solutions in exponential design space.

Deployment problem

How to:

- find optimal solutions in exponential design space.
- model **complex hardware** which has Processors, Network, DMA

Deployment problem

How to:

- find optimal solutions in exponential design space.
- model **complex hardware** which has Processors, Network, DMA
- evaluate **multiple criteria**
  - Latency
  - Memory used
  - Processors used
  - ...

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

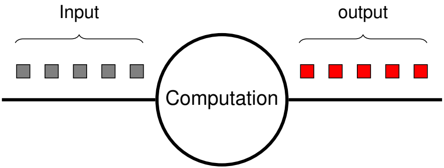
represents **Streaming Applications**

Model of Computation

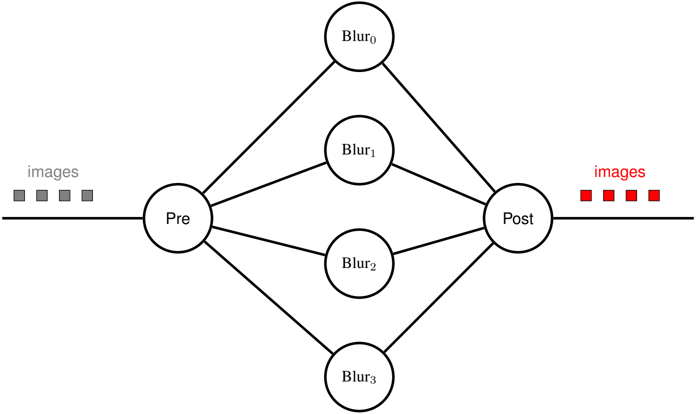
Synchronous Dataflow graphs (SDF)

by Edward Lee and David Messerschmitt in 1987

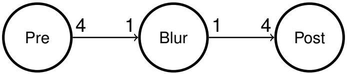
represents **Streaming Applications**



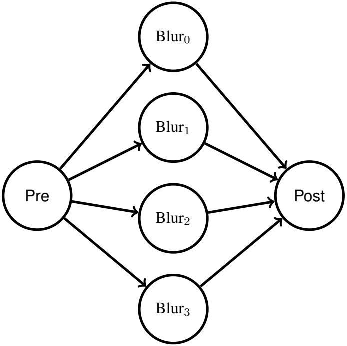
Synchronous DataFlow



Synchronous DataFlow

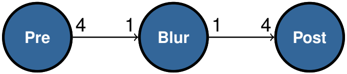


SDF Graph



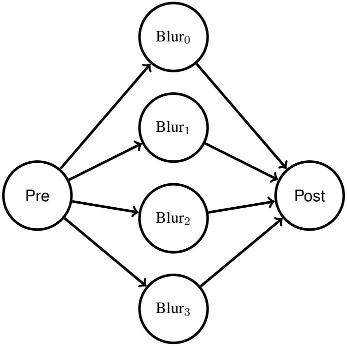
Task Graph

Synchronous DataFlow



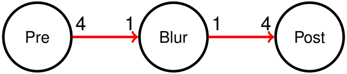
SDF Graph

- **Actors**



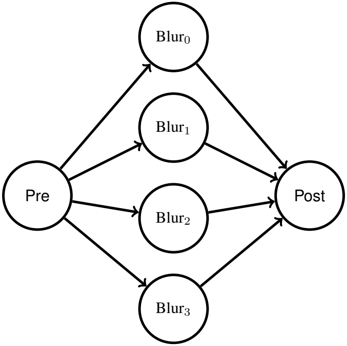
Task Graph

Synchronous DataFlow



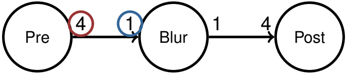
SDF Graph

- Actors
- Edges



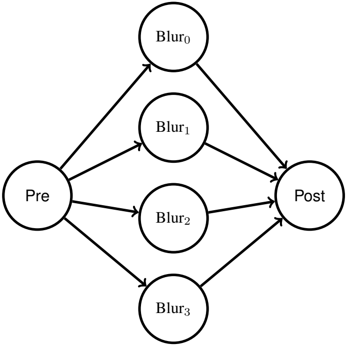
Task Graph

Synchronous DataFlow



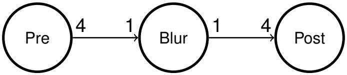
SDF Graph

- Actors
- Edges
- Rates

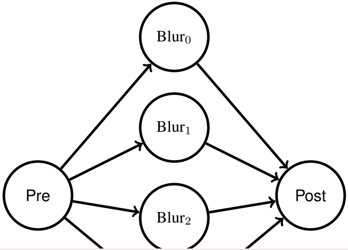


Task Graph

Synchronous DataFlow



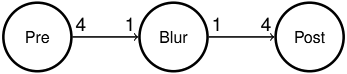
SDF Graph



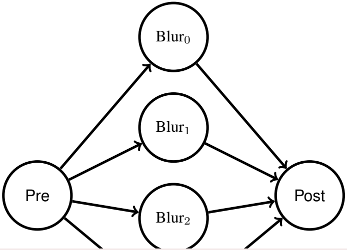
- Actor Blur is **compact representation** of data parallel tasks.

Task Graph

Synchronous DataFlow



SDF Graph



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

Task Graph

Split-Join Graphs

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

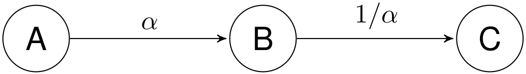
still covering perhaps 90% of use cases in the literature

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

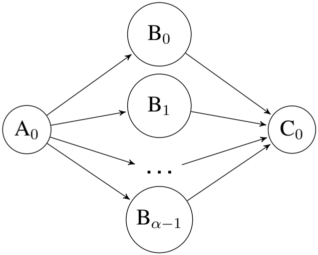
still covering perhaps 90% of use cases in the literature

a simple example:

$$A \xrightarrow{\alpha} B \xrightarrow{1/\alpha} C$$

α : spawn and split

$1/\alpha$: wait and join



Restrictions compared to general SDF

Split-join does not support:

- **Stateful** actors

Restrictions compared to general SDF

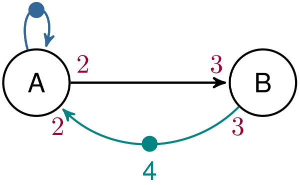
Split-join does not support:

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

Restrictions compared to general SDF

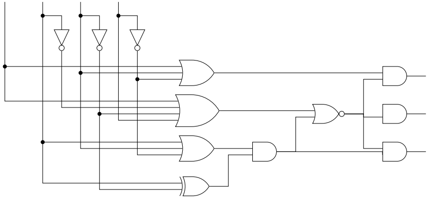
Split-join does not support:

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

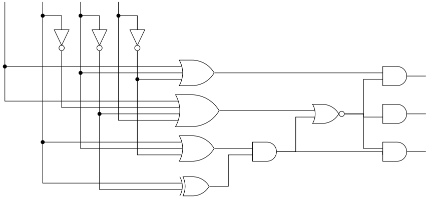


Overview

SATisfiability solver (SAT / SMT)

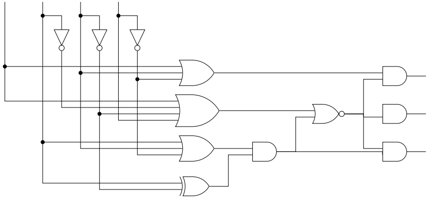


SATisfiability solver (SAT / SMT)



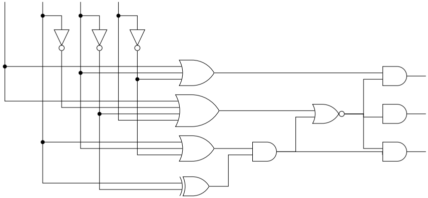
- Boolean variables
  - in_0, in_1, in_2 ...
  - out_0, out_1, out_2 ...

SATisfiability solver (SAT / SMT)

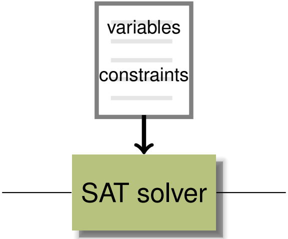


- Boolean variables
  - in_0, in_1, in_2 ...
  - out_0, out_1, out_2 ...
- Constraints
  - $\mathrm{out}_0 = \mathrm{in}_0 \vee \mathrm{in}_1 \oplus \mathrm{in}_2$ ...

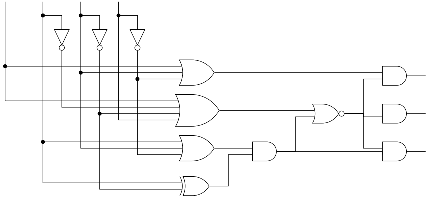
SATisfiability solver (SAT / SMT)



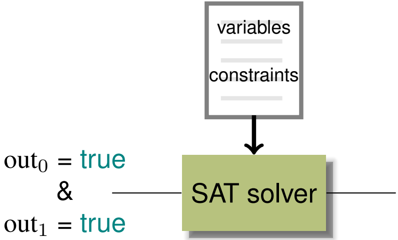
- Boolean variables
  - $\mathrm{in}_0, \mathrm{in}_1, \mathrm{in}_2$ ...
  - $\mathrm{out}_0, \mathrm{out}_1, \mathrm{out}_2$ ...
- Constraints
  - $\mathrm{out}_0 = \mathrm{in}_0 \vee \mathrm{in}_1 \oplus \mathrm{in}_2$ ...



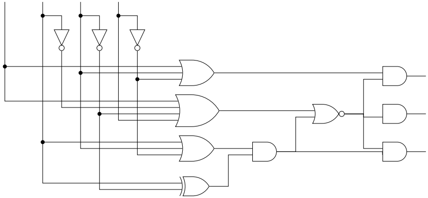
SATisfiability solver (SAT / SMT)



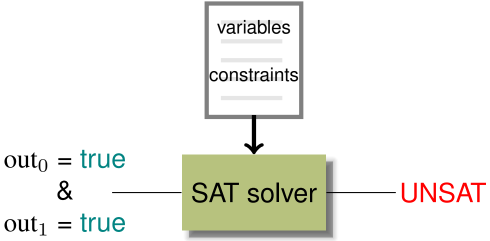
- Boolean variables
  - $\mathrm{in}_0, \mathrm{in}_1, \mathrm{in}_2$ ...
  - $\mathrm{out}_0, \mathrm{out}_1, \mathrm{out}_2$ ...
- Constraints
  - $\mathrm{out}_0 = \mathrm{in}_0 \vee \mathrm{in}_1 \oplus \mathrm{in}_2$ ...



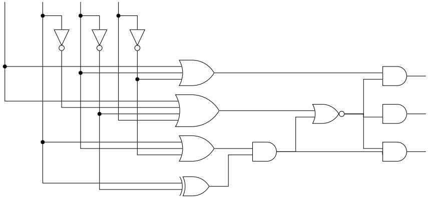
SATisfiability solver (SAT / SMT)



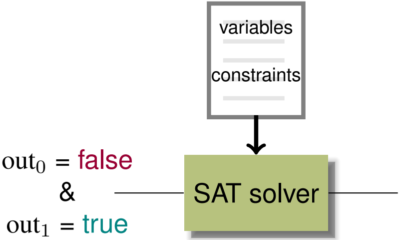
- Boolean variables
  - $\mathrm{in}_0, \mathrm{in}_1, \mathrm{in}_2$ ...
  - $\mathrm{out}_0, \mathrm{out}_1, \mathrm{out}_2$ ...
- Constraints
  - $\mathrm{out}_0 = \mathrm{in}_0 \vee \mathrm{in}_1 \oplus \mathrm{in}_2$ ...



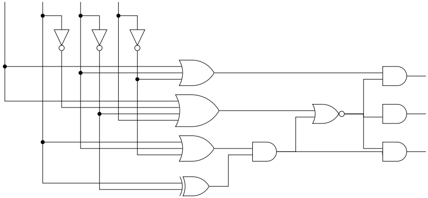
SATisfiability solver (SAT / SMT)



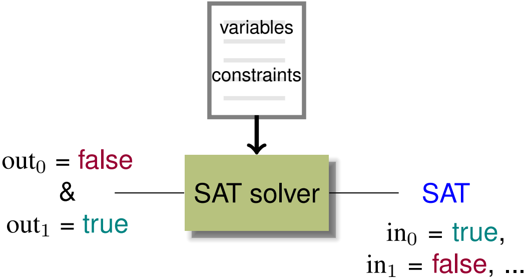
- Boolean variables
  - $\mathrm{in}_0, \mathrm{in}_1, \mathrm{in}_2$ ...
  - $\mathrm{out}_0, \mathrm{out}_1, \mathrm{out}_2$ ...
- Constraints
  - $\mathrm{out}_0 = \mathrm{in}_0 \vee \mathrm{in}_1 \oplus \mathrm{in}_2$ ...



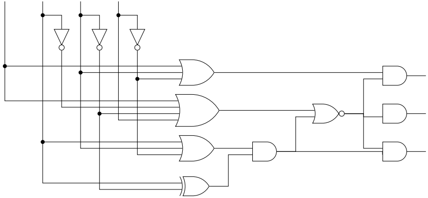
SATisfiability solver (SAT / SMT)



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

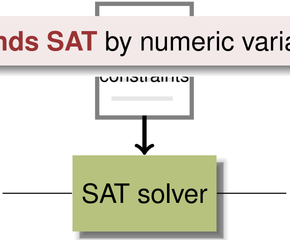


SATisfiability solver (SAT / SMT)

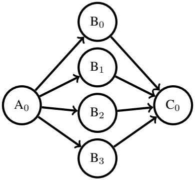


- Boolean variables
  - $\mathrm{in}_0, \mathrm{in}_1, \mathrm{in}_2 \ldots$
  - $\mathrm{out}_0, \mathrm{out}_1, \mathrm{out}_2 \ldots$
- Constraints
  - $\mathrm{out}_0 = \mathrm{in}_0 \vee \mathrm{in}_1 \oplus \mathrm{in}_2 \ldots$

SMT **extends SAT** by numeric variables and constants



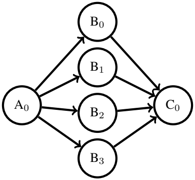
Encoding deployment with constraints



Task Graph

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

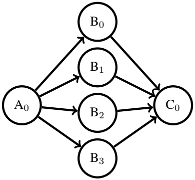
Encoding deployment with constraints



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

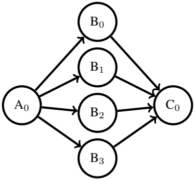
Encoding deployment with constraints



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

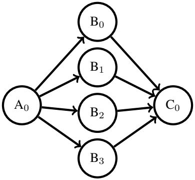
Encoding deployment with constraints



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

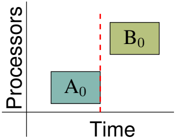
Encoding deployment with constraints



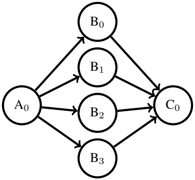
Task Graph

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
  - $xB_0 \geq (xA_0 + dA)$



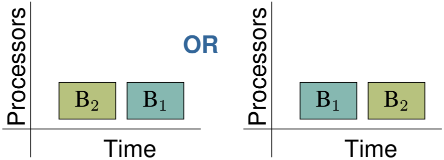
Encoding deployment with constraints



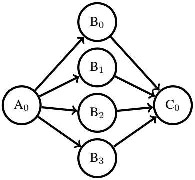
Task Graph

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

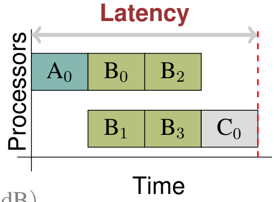


Encoding deployment with constraints



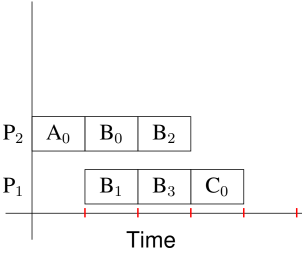
Task Graph

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

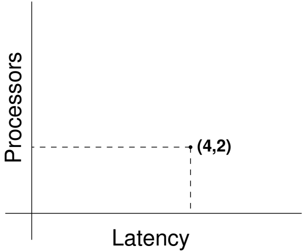


- Precedence Constraints
  - $xB_0 \geq (xA_0 + dA)$
- Mutual Exclusion Constraints
  - if $(pB_1 = pB_2)$ then
    $xB_1 \geq (xB_2 + dB) \ \vee \ xB_2 \geq (xB_1 + dB)$
- Latency Cost
  - $\text{Latency} = (xC_0 + dC)$

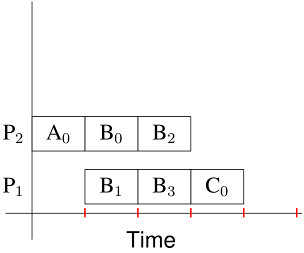
Multi-criteria Problem



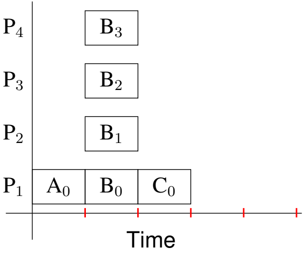
Latency = 4
#Proc = 2



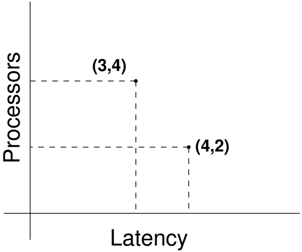
Multi-criteria Problem



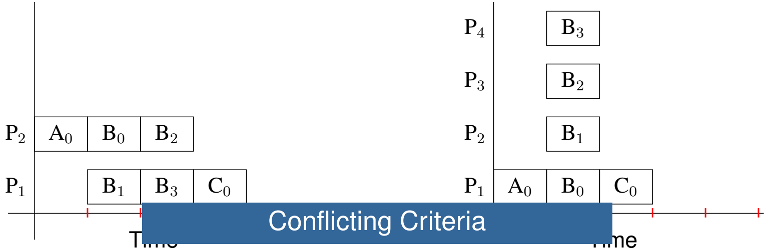
Latency = 4
#Proc = 2



Latency = 3
#Proc = 4

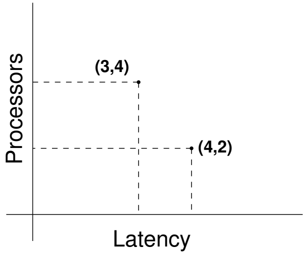


Multi-criteria Problem

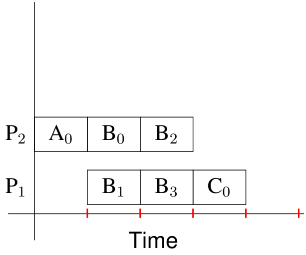


Latency = 4
#Proc = 2

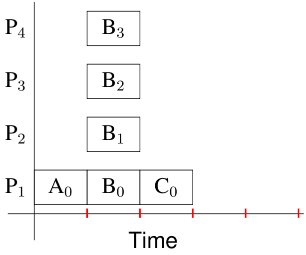
Latency = 3
#Proc = 4



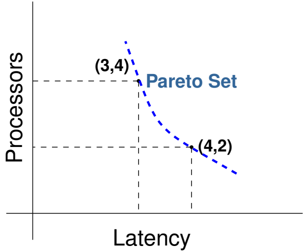
Multi-criteria Problem



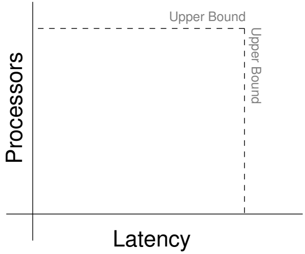
Latency = 4
#Proc = 2



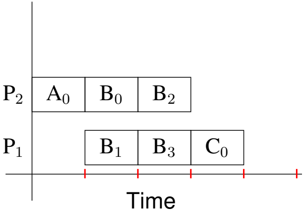
Latency = 3
#Proc = 4



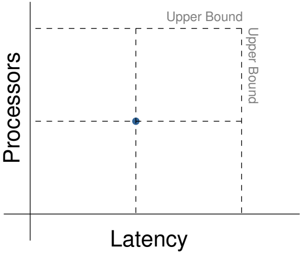
Problem Monotonicity



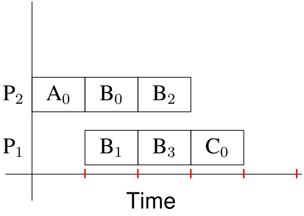
Problem Monotonicity



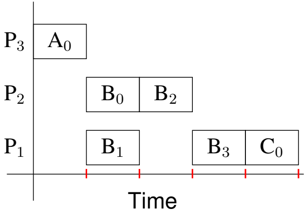
Latency = 4
#Proc = 2



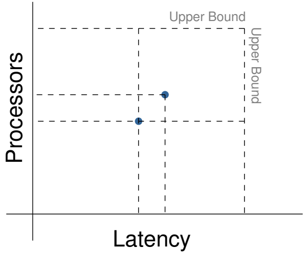
Problem Monotonicity



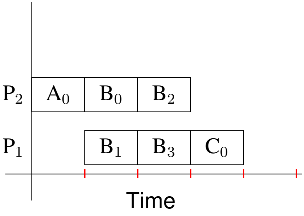
Latency = 4
#Proc = 2



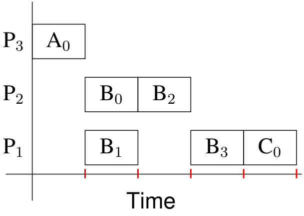
Latency = 5
#Proc = 3



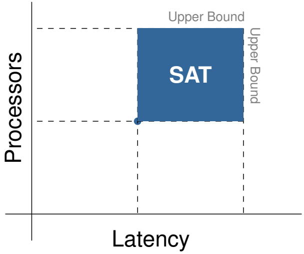
Problem Monotonicity



Latency = 4
#Proc = 2

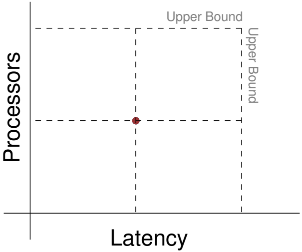


Latency = 5
#Proc = 3



Problem Monotonicity

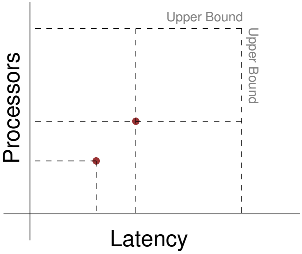
Latency = 4
#Proc = 2
Not Possible



Problem Monotonicity

Latency = 4
#Proc = 2
Not Possible

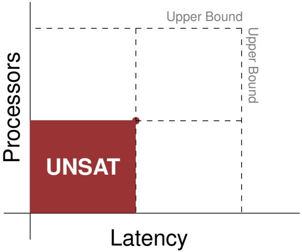
Latency = 2
#Proc = 1
Also Not Possible



Problem Monotonicity

Latency = 4
#Proc = 2
Not Possible

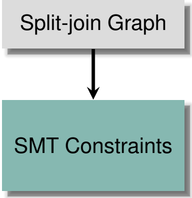
Latency = 2
#Proc = 1
Also Not Possible



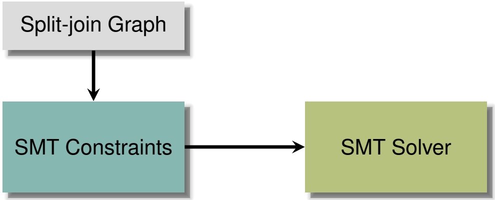
Design Space Exploration

Split-join Graph

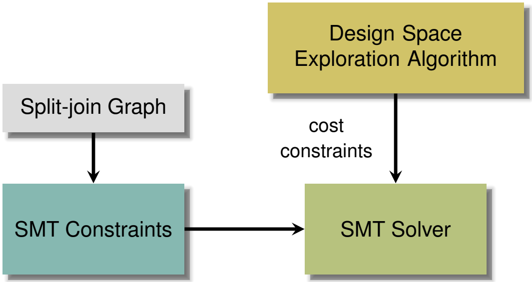
Design Space Exploration



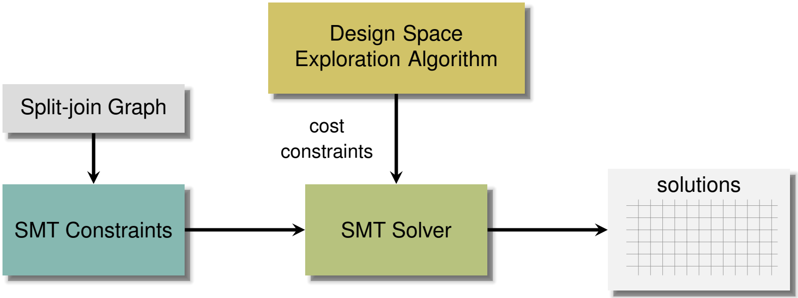
Design Space Exploration



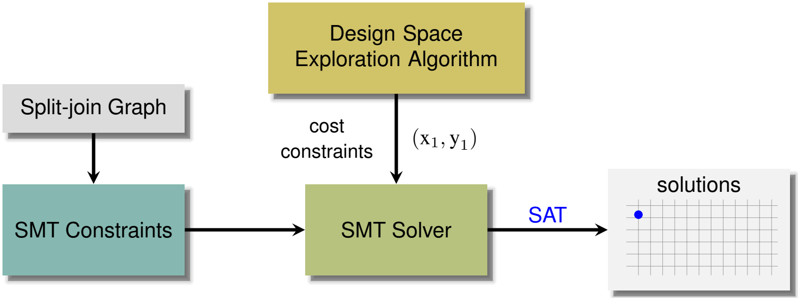
Design Space Exploration



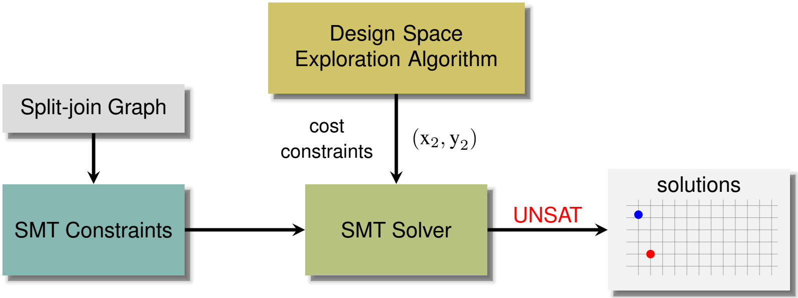
Design Space Exploration



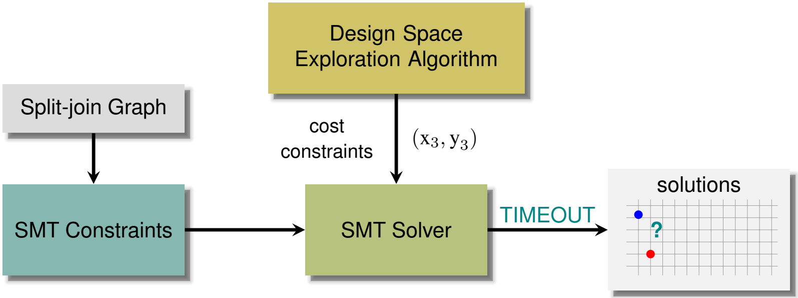
Design Space Exploration



Design Space Exploration

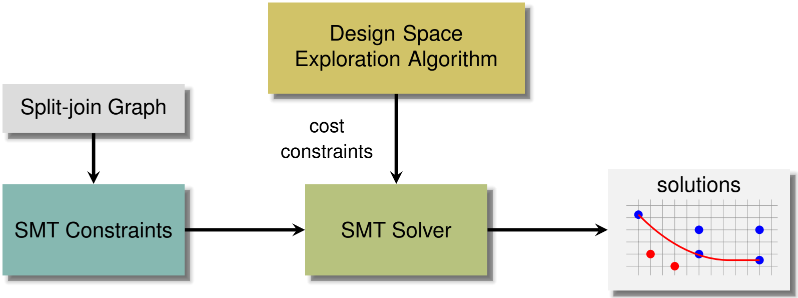


Design Space Exploration

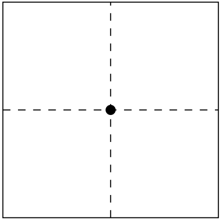


Timeout:
Cannot decide SAT / UNSAT in a given **TIME-BUDGET**.

Design Space Exploration

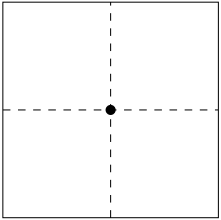


Exploration Algorithm



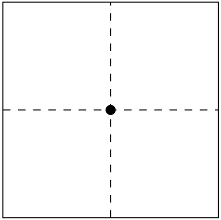
Exploration Algorithm

- **Divide cost space** using grids



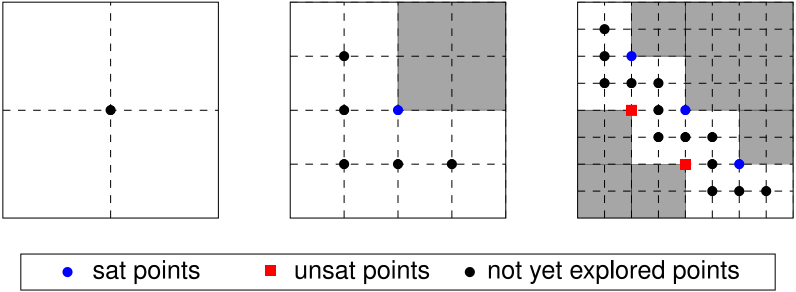
Exploration Algorithm

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



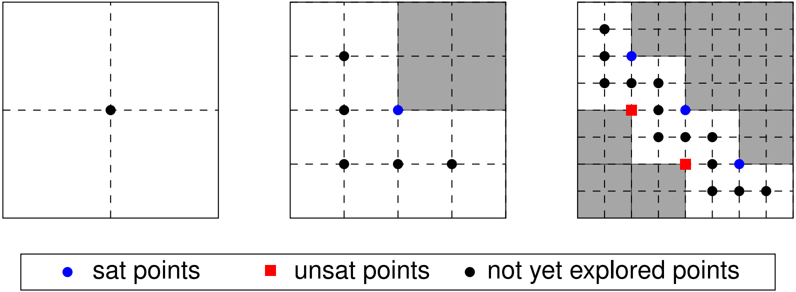
Exploration Algorithm

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



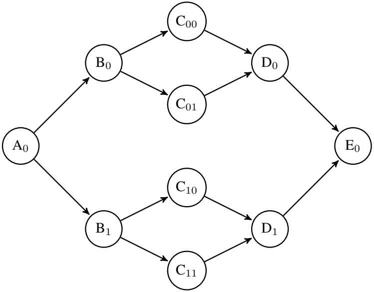
Exploration Algorithm

- **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

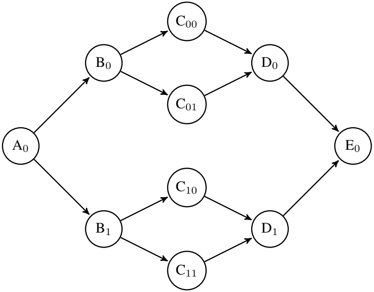
Task Symmetry



task graph

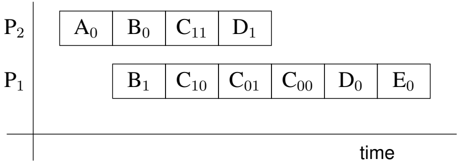
- all instances of actor C are similar (**symmetric**)

Task Symmetry



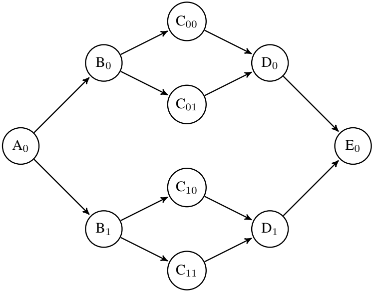
task graph

a schedule



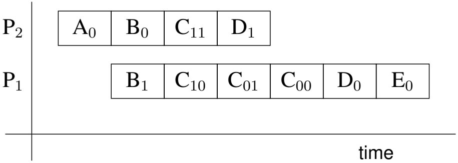
- all instances of actor C are similar (**symmetric**)

Task Symmetry

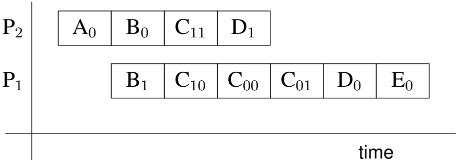


task graph

a schedule

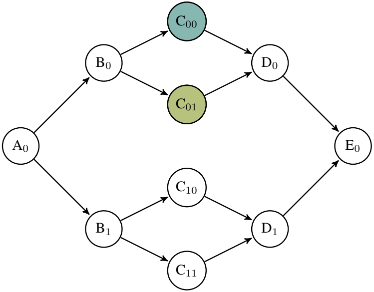


a **permuted** schedule



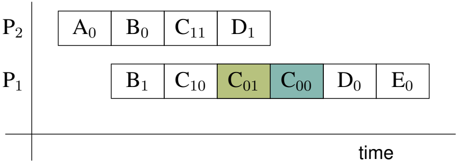
- all instances of actor C are similar (**symmetric**)

Task Symmetry

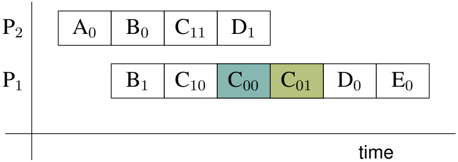


task graph

a schedule

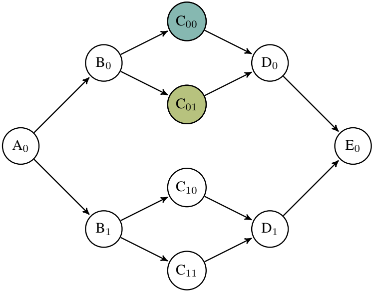


a **permuted** schedule

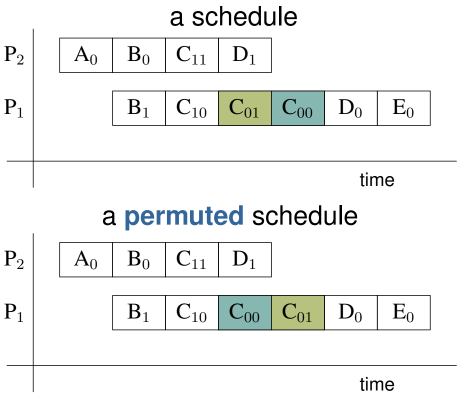


- all instances of actor C are similar (**symmetric**)

Task Symmetry

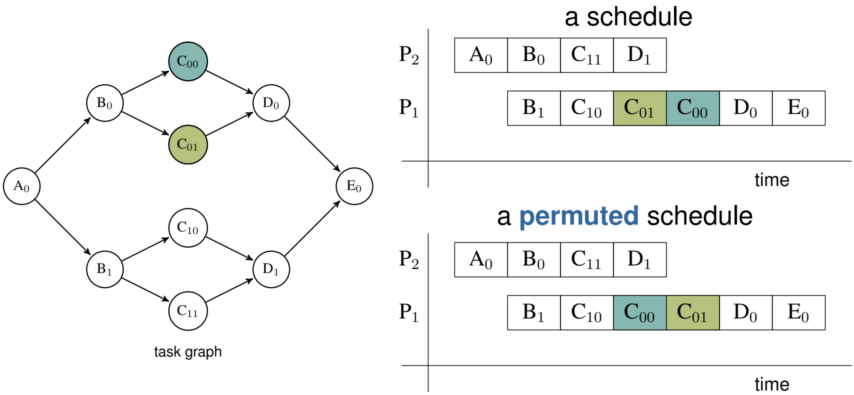


task graph



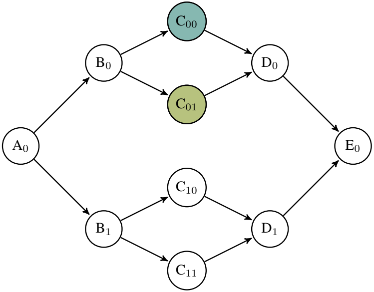
- all instances of actor C are similar (**symmetric**)
- **No change** in latency !

Task Symmetry



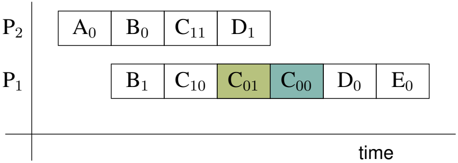
- all instances of actor C are similar (**symmetric**)
- **No change** in latency !
- **Huge number** of such symmetric solutions

Task Symmetry

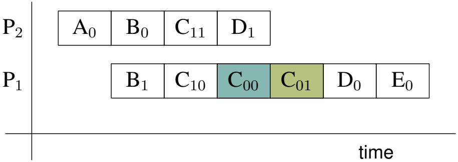


task graph

a schedule

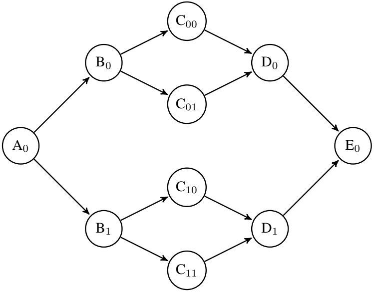


a **permuted** schedule



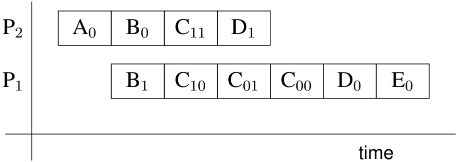
- all instances of actor C are similar (**symmetric**)
- **No change** in latency !
- **Huge number** of such symmetric solutions
- Add constraints to **eliminate** all but one

Task Symmetry

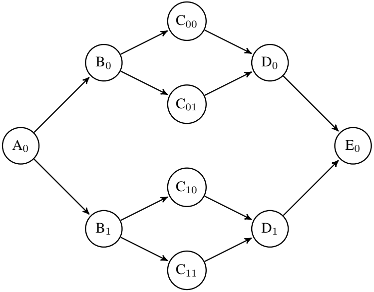


task graph

a schedule

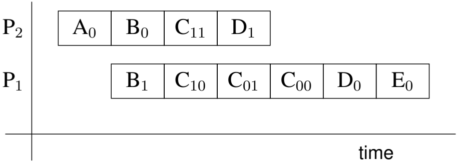


Task Symmetry



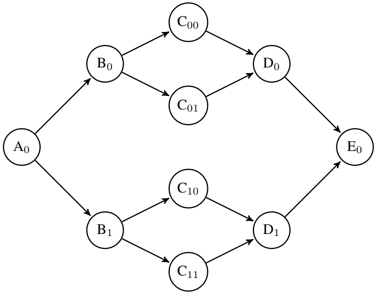
task graph

a schedule



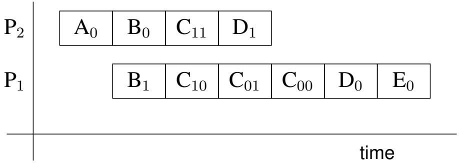
- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$

Task Symmetry



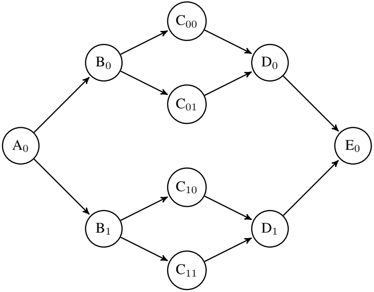
task graph

a schedule



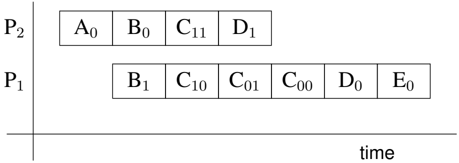
- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$
- enforce lexicographic order in schedule: $s(u) \leq s(u') $ for $u \ll u'$

Task Symmetry



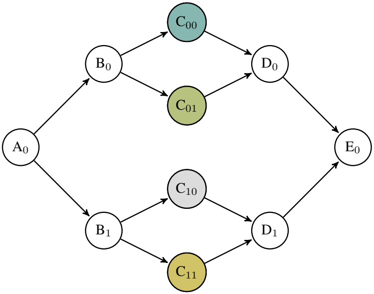
task graph

a schedule

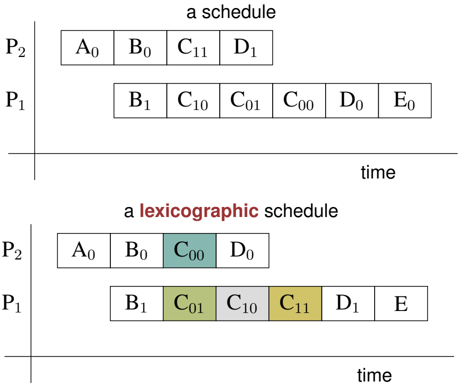


- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$
- enforce lexicographic order in schedule:
  $s(u) \leq s(u')$ for $u \ll u'$
- $s(C_{00}) \leq s(C_{01}) \leq s(C_{10}) \leq s(C_{11})$

Task Symmetry

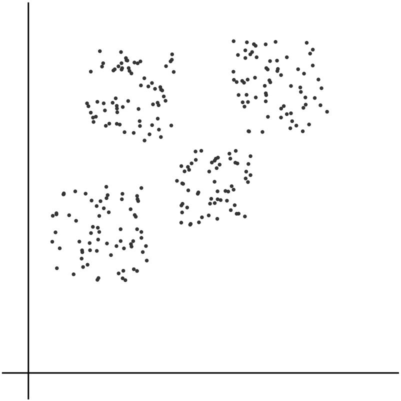


task graph

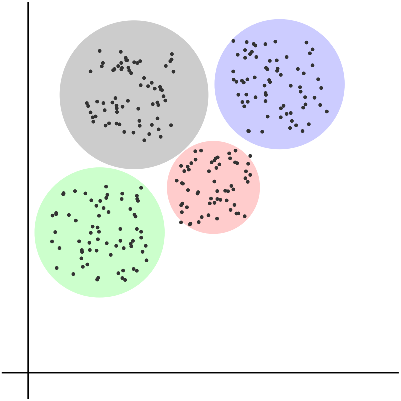


- **lexicographic** order : $C_{00} \ll C_{01} \ll C_{10} \ll C_{11}$
- enforce lexicographic order in schedule:
  $s(u) \leq s(u')$ for $u \ll u'$
- $s(C_{00}) \leq s(C_{01}) \leq s(C_{10}) \leq s(C_{11})$

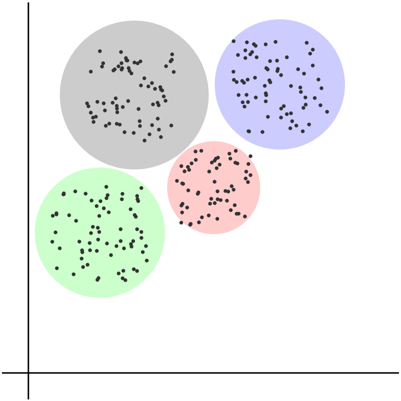
Task Symmetry : Theorem



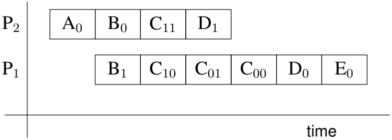
Task Symmetry : Theorem



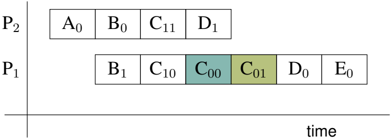
Task Symmetry : Theorem



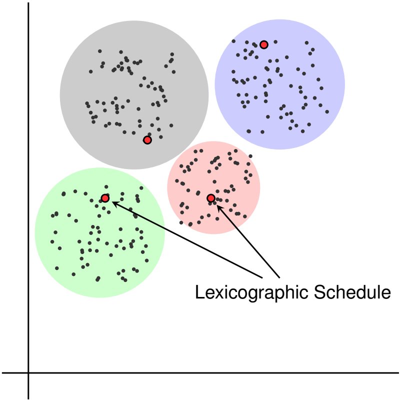
a schedule



a **permuted** schedule

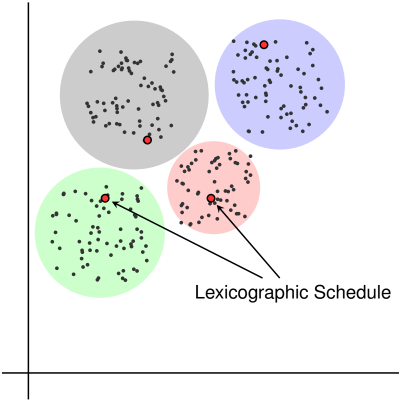


Task Symmetry : Theorem



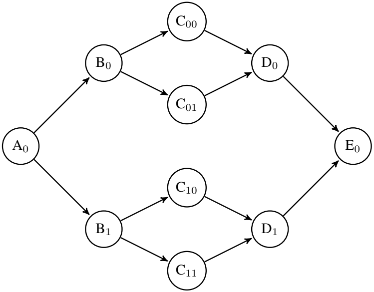
- Theorem : Every group has a **lexicographic schedule**

Task Symmetry : Theorem

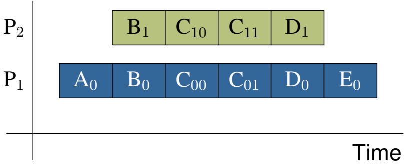


- Theorem : Every group has a **lexicographic schedule**
- Corollary : No feasible **cost** is lost

Processor Symmetry

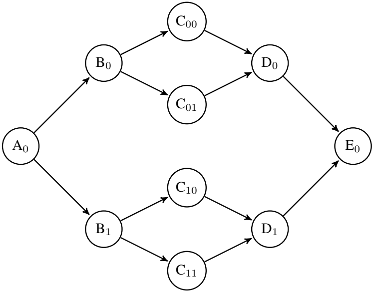


task graph

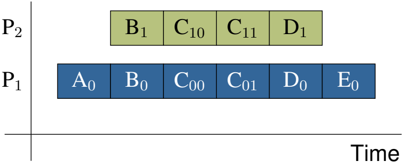


schedule

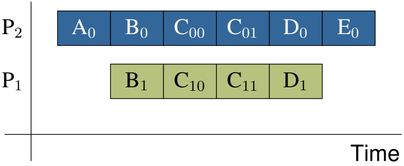
Processor Symmetry



task graph

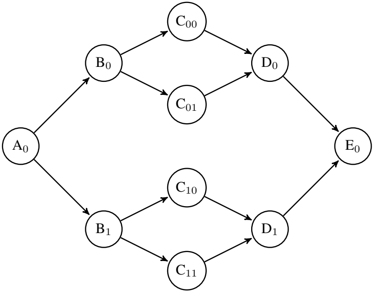


schedule

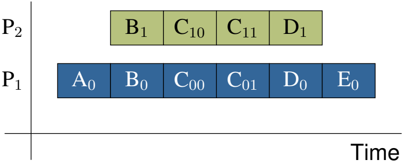


swapped P_1 and P_2

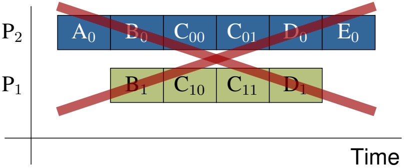
Processor Symmetry



task graph



schedule

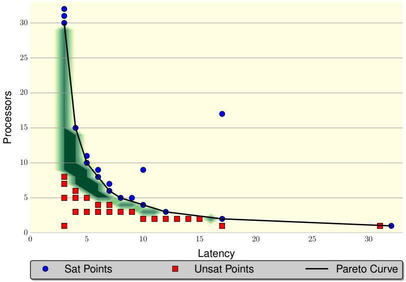


swapped P_1 and P_2

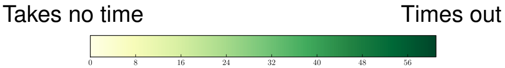
Pareto Exploration

Exploration : Processors vs Latency $\alpha = 30$

Pareto Exploration

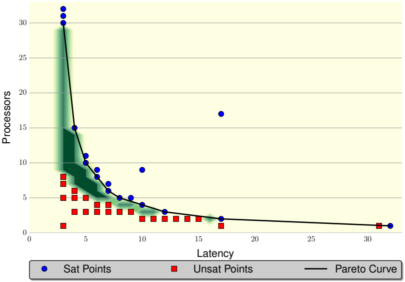


without symmetry breaking

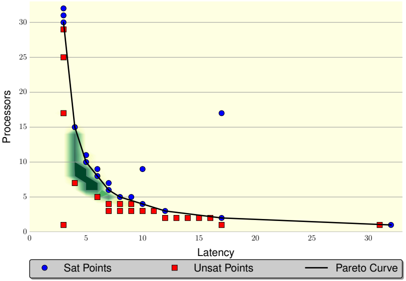


Exploration : Processors vs Latency $\alpha = 30$

Pareto Exploration



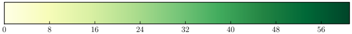
without symmetry breaking



with symmetry breaking

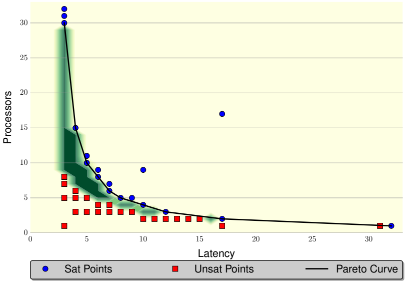
Takes no time

Times out

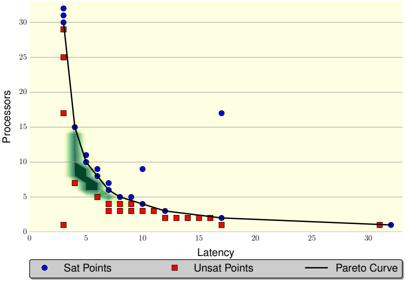


Exploration : Processors vs Latency $\alpha = 30$

Pareto Exploration



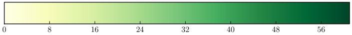
without symmetry breaking



with symmetry breaking

Takes no time

Times out



Exploration : Processors vs Latency $\alpha = 30$

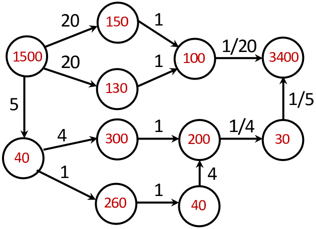
Solver Performance

- 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:

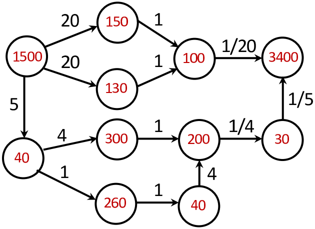


122 Tasks

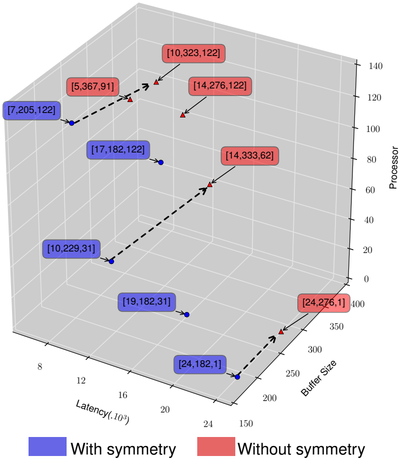
Video Decoder

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

MPEG video decoder:



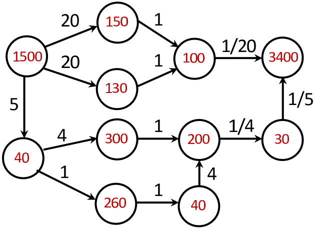
122 Tasks



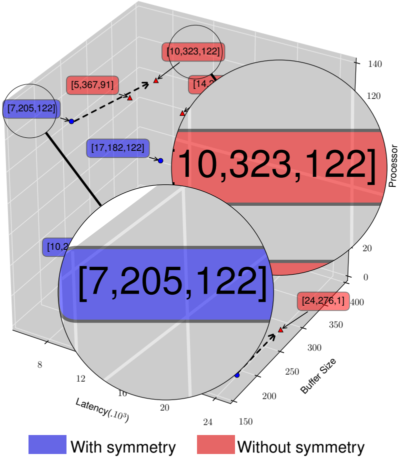
Video Decoder

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

MPEG video decoder:



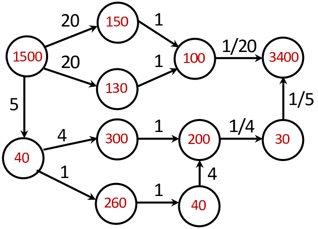
122 Tasks



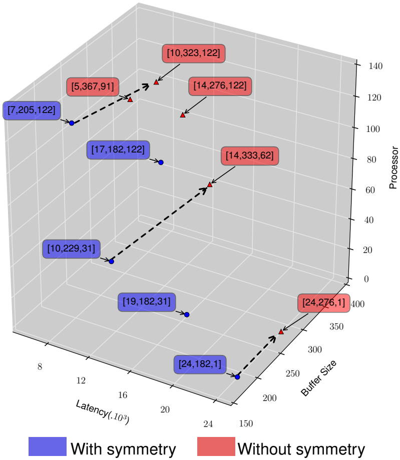
Video Decoder

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

MPEG video decoder:



122 Tasks

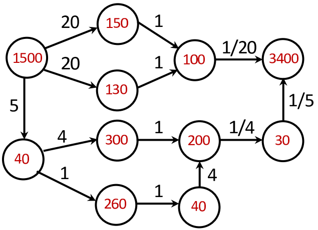


Better Pareto points

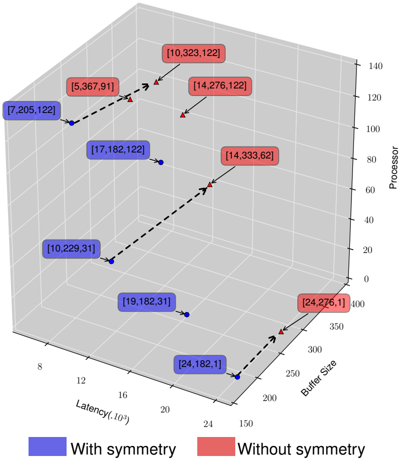
Video Decoder

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

MPEG video decoder:



122 Tasks



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

Distributed memory scheduling

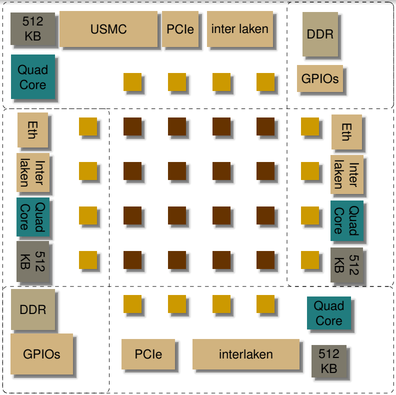
- So far we ignored the **communication** costs

Distributed memory scheduling

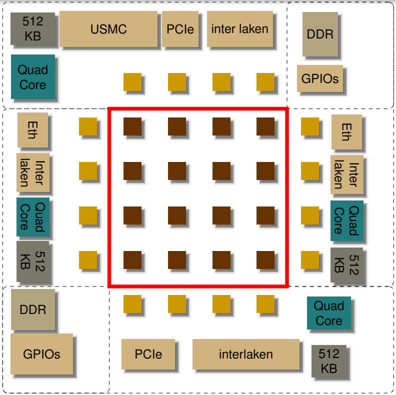
- So far we ignored the **communication** costs
- For **distributed memory**, communication needs to be modeled

Overview

Kalray MPPA-256

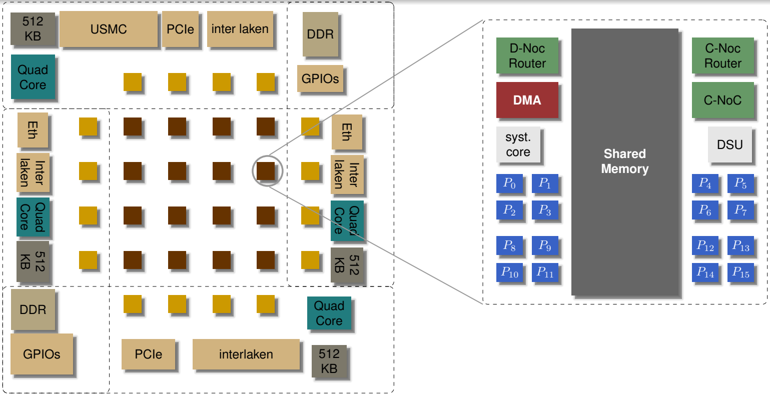


Kalray MPPA-256



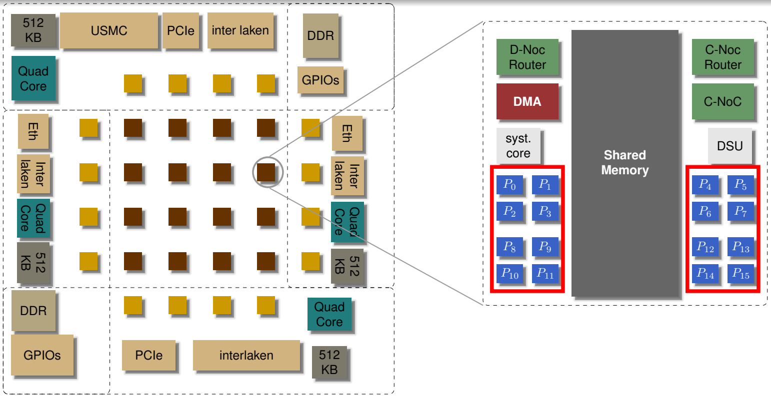
- 16 **compute clusters**

Kalray MPPA-256



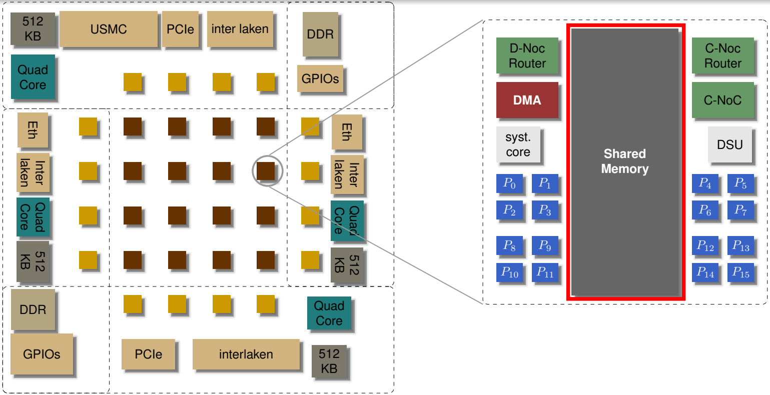
- 16 **compute clusters**

Kalray MPPA-256



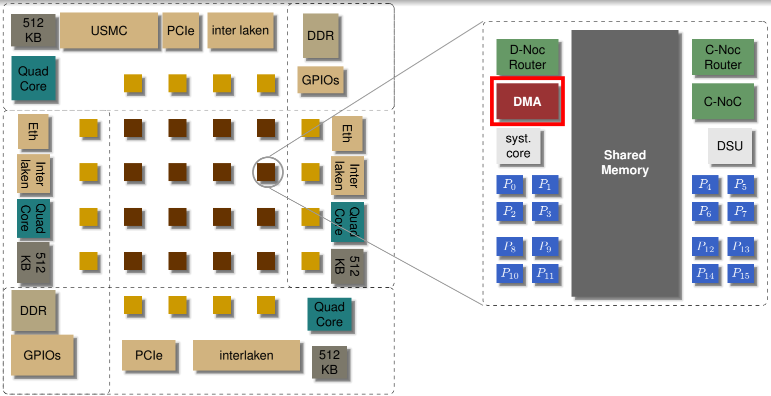
- 16 **compute clusters**
  - **16 processors**

Kalray MPPA-256



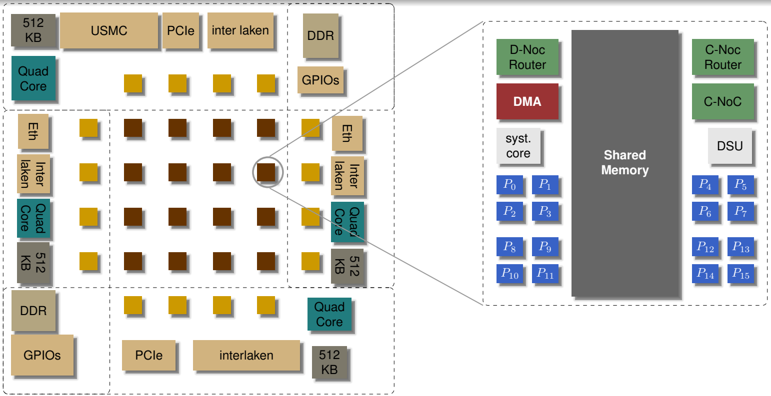
- 16 **compute clusters**
  - **16 processors**
  - 2 MB **Shared Memory**

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

The problem?

- **Which cluster** to allocate?
- **Which processor** to allocate?
- Connected tasks in **same or different** cluster?
- Communicating tasks if to be added, **which DMA**?
- And the constraints
  - Precedence
  - Mutual Exclusion
  - Costs

For 10 tasks, 256 processors,

$1.20892582 \times 10^{24}$ potential solutions!

The problem?

- **Which cluster** to allocate?
- **Which processor** to allocate?
- Connected tasks in **same or different** cluster?
- Communicating tasks if to be added, **which DMA**?
- And the constraints
  - Precedence
  - Mutual Exclusion
  - Costs

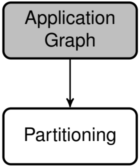
Split the problem into sub-problems.

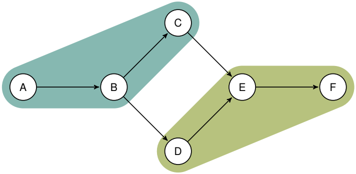
For 10 tasks, 256 processors,
$1.20892582 \times 10^{24}$ potential solutions!

Design Flow

Application Graph

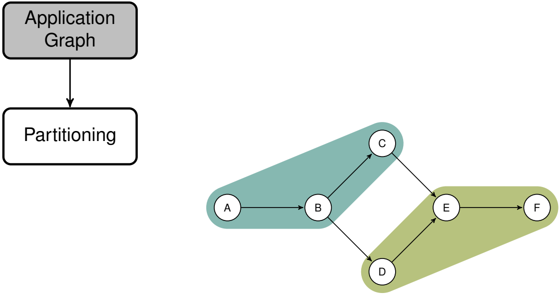
Design Flow





Group the Actors

Design Flow

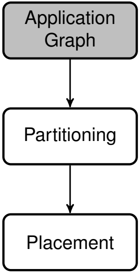


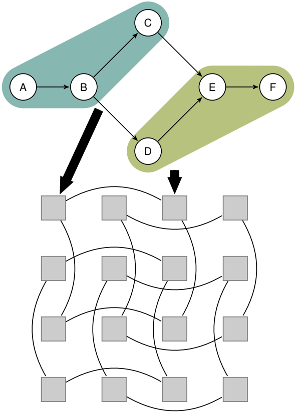
Group the Actors

Goals

- Load balance the groups
- Minimize data exchange

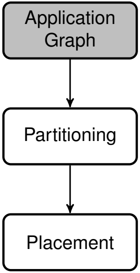
Design Flow

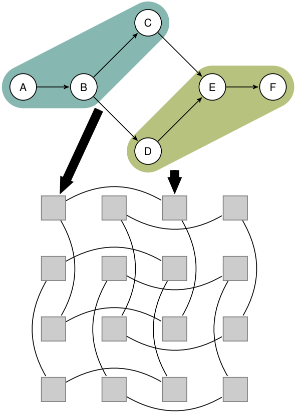




Place the Groups

Design Flow



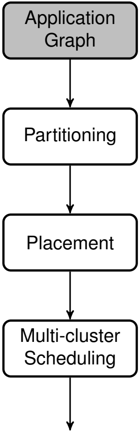


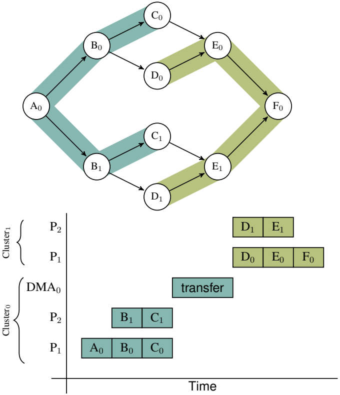
Place the Groups

Goals

- Minimize distance between communicating groups

Design Flow

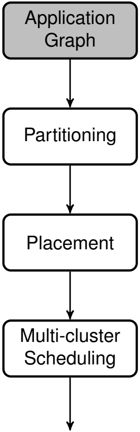


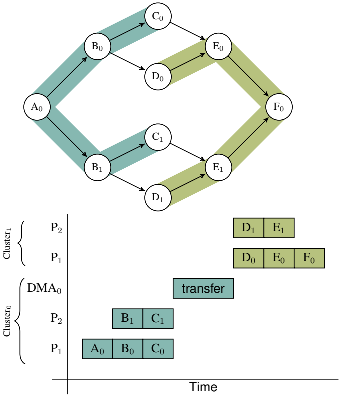


Schedule

- Tasks
- Transfer

Design Flow





Schedule

- Tasks
- Transfer

Goals

- Minimize Latency
- Minimize Buffer size

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

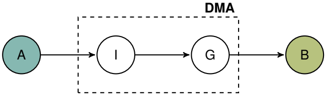
- Tasks and Transfers
  - **Cluster** Mapping
  - **Processor** and **DMA** Mapping
  - **Start time**
- Edges
  - **Communication buffer** size

Output of Design Flow

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

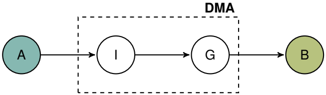
DMA Model

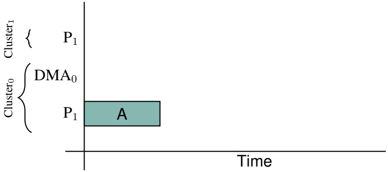
Tasks communicating via DMA:



DMA Model

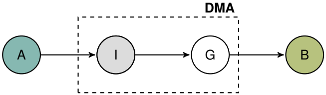
Tasks communicating via DMA:

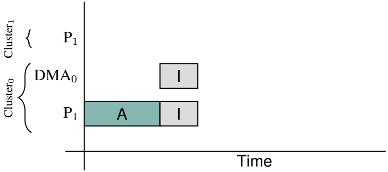




DMA Model

Tasks communicating via DMA:

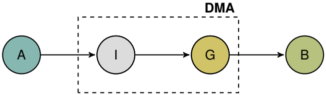


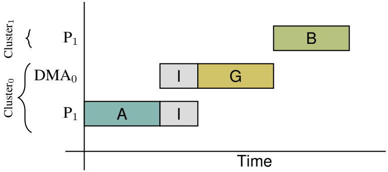


Task	Description	Resources used	Task duration
I	Initialization	Processor and DMA	Constant

DMA Model

Tasks communicating via DMA:

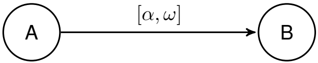




Task	Description	Resources used	Task duration
I	Initialization	Processor and DMA	Constant
G	Network Transfer	Only DMA	Transfer size dependent

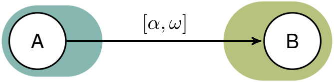
Model Transformation

An example application graph:



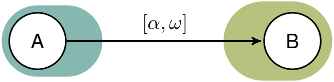
Model Transformation

An example application graph:

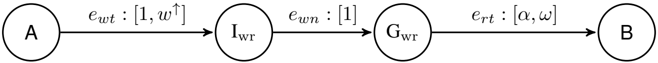


Model Transformation

An example application graph:

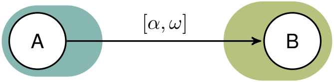


Partition-Aware graph:

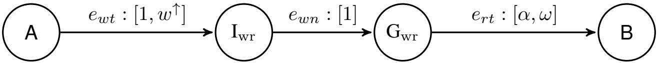


Model Transformation

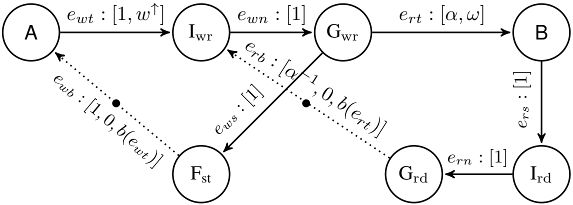
An example application graph:



Partition-Aware graph:

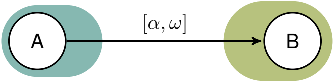


Buffer-Aware graph:

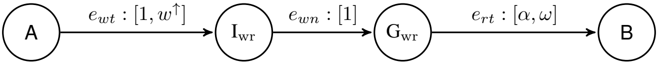


Model Transformation

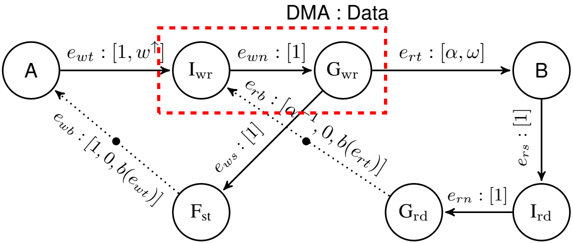
An example application graph:



Partition-Aware graph:

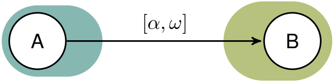


Buffer-Aware graph:

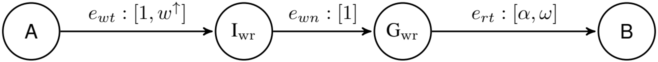


Model Transformation

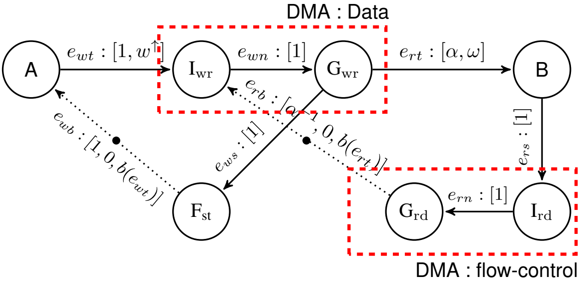
An example application graph:



Partition-Aware graph:

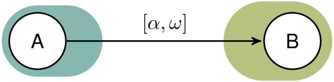


Buffer-Aware graph:

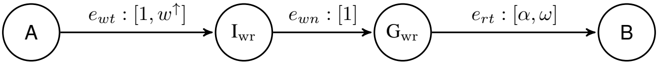


Model Transformation

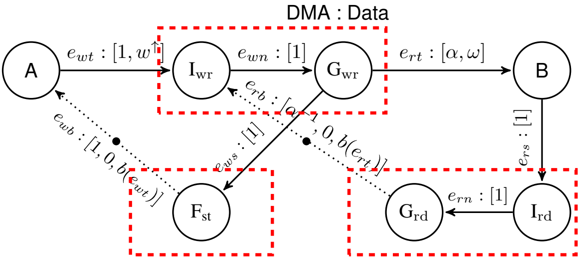
An example application graph:



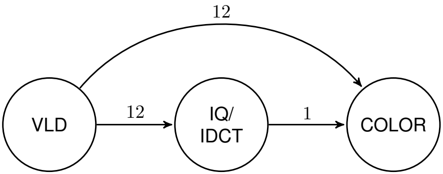
Partition-Aware graph:



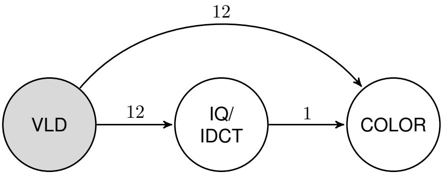
Buffer-Aware graph:



JPEG Decoder Example

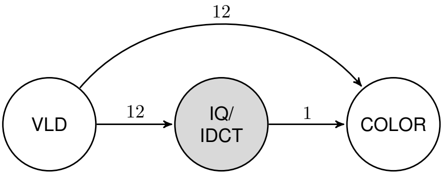


JPEG Decoder Example



VLD : Variable Length Decoder

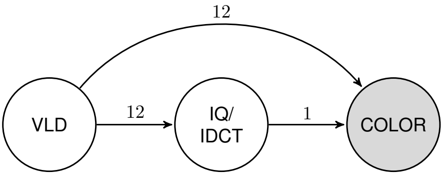
JPEG Decoder Example



VLD : Variable Length Decoder

IQ / IDCT : Inverse Quantization / Inverse Discrete Cosine Transform

JPEG Decoder Example

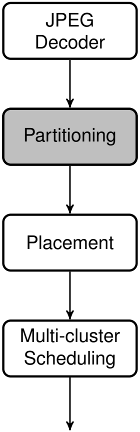


VLD : Variable Length Decoder

IQ / IDCT : Inverse Quantization / Inverse Discrete Cosine Transform

Color : Color Conversion

JPEG Decoder Example



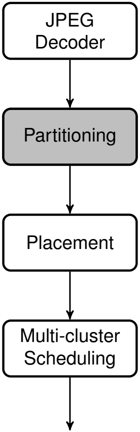
Partitioning Solutions:

$\mathrm{C}_{\mathbf{z}}$: No. of Groups

C_{η} : Total communication cost

C_{τ} : Max. workload per group

JPEG Decoder Example



Partitioning Solutions:

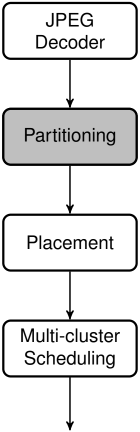
$\mathrm{C}_{\mathbf{z}}$: No. of Groups

C_{η} : Total communication cost

C_{τ} : Max. workload per group

Solution	Allocated group			Exploration Cost		
	vld	iq	color	$\mathrm{C}_{\mathbf{z}}$	C_{η}	C_{τ}
P_{s0}	0	1	2	3	12384	424012
P_{s1}	0	0	1	2	2736	758116
P_{s2}	0	0	0	1	0	934288
P_{s3}	0	1	1	2	9648	510276

JPEG Decoder Example



Partitioning Solutions:

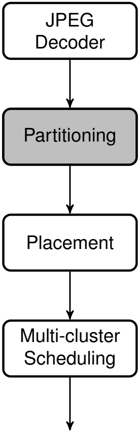
$\mathrm{C}_{\mathbf{z}}$: No. of Groups

C_{η} : Total communication cost

C_{τ} : Max. workload per group

Solution	Allocated group			Exploration Cost		
	vld	iq	color	$\mathrm{C}_{\mathbf{z}}$	C_{η}	C_{τ}
P_{s0}	0	1	2	3	12384	424012
P_{s1}	0	0	1	2	2736	758116
P_{s2}	0	0	0	1	0	934288
P_{s3}	0	1	1	2	9648	510276

JPEG Decoder Example



Partitioning Solutions:

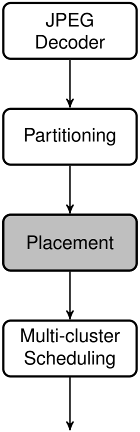
$\mathrm{C}_{\mathbf{z}}$: No. of Groups

C_{η} : Total communication cost

C_{τ} : Max. workload per group

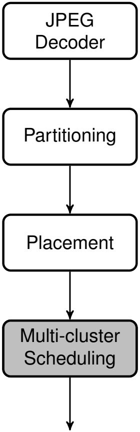
Solution	Allocated group			Exploration Cost		
	vld	iq	color	$\mathrm{C}_{\mathbf{z}}$	C_{η}	C_{τ}
P_{s0}	0	1	2	3	12384	424012
P_{s1}	0	0	1	2	2736	758116
P_{s2}	0	0	0	1	0	934288
P_{s3}	0	1	1	2	9648	510276

JPEG Decoder Example



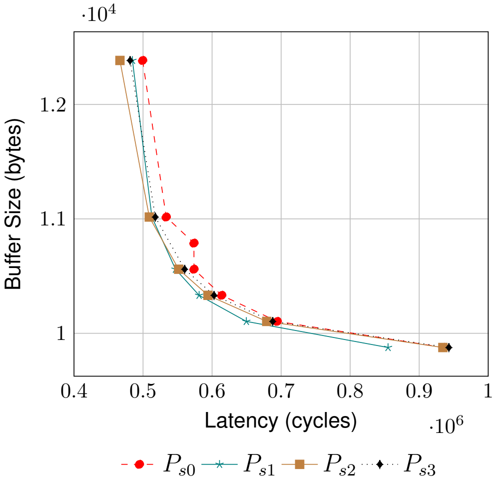
Solution	Allocated group vld	iq	color	Exploration Cost C_z	C_η	C_τ
P_{s0}	0	1	2	3	12384	424012
P_{s1}	0	0	1	2	2736	758116
P_{s2}	0	0	0	1	0	934288
P_{s3}	0	1	1	2	9648	510276

JPEG Decoder Example



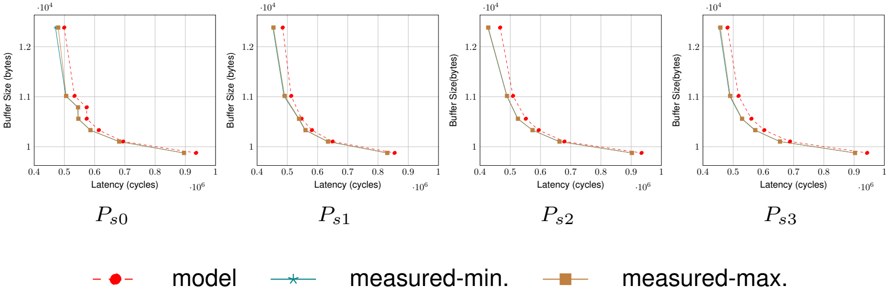
Solution
P_{s0}
P_{s1}
P_{s2}
P_{s3}

Scheduling Solutions:



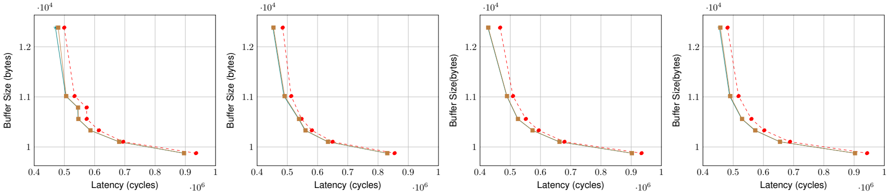
JPEG Decoder Example

JPEG decoder latency measured on **Kalray** platform



JPEG Decoder Example

JPEG decoder latency measured on **Kalray** platform

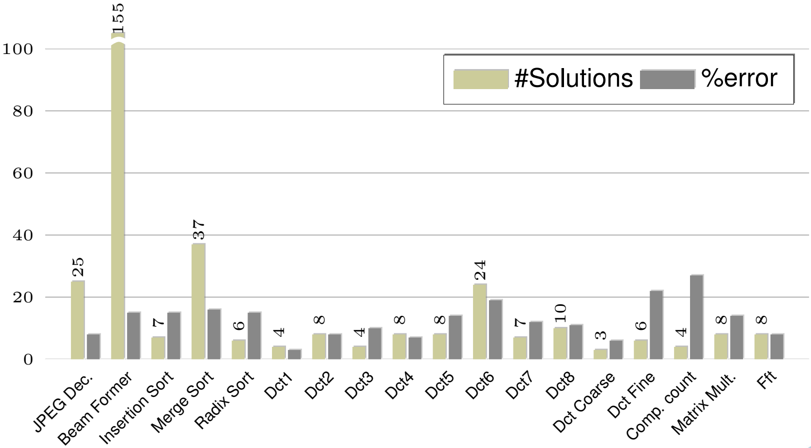


P_{s0} P_{s1} P_{s2} P_{s3}

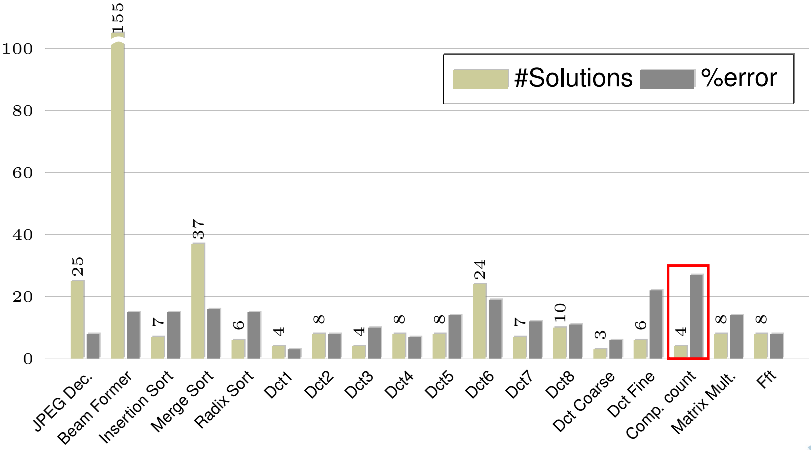
model measured-min. measured-max.

Maximum prediction error of 9%

StreamIt Benchmarks

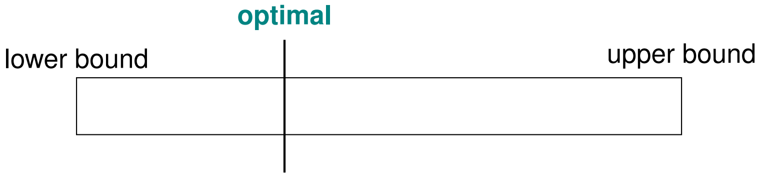


StreamIt Benchmarks

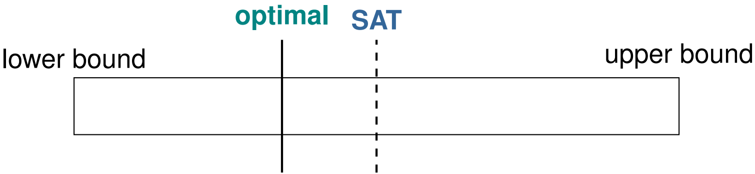


Overview

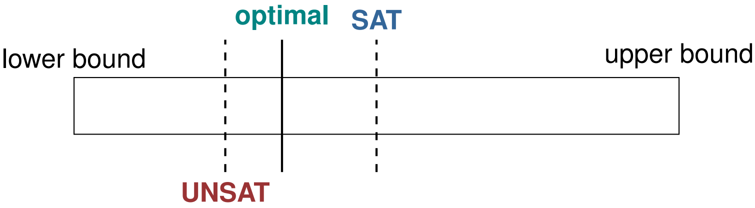
Lessons learnt from SMT solver



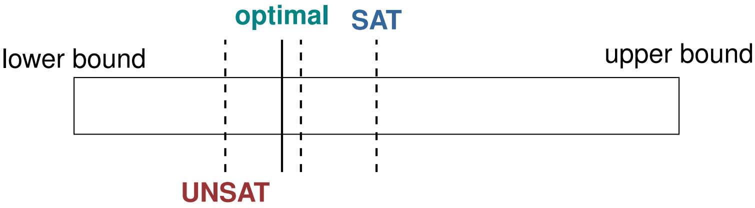
Lessons learnt from SMT solver



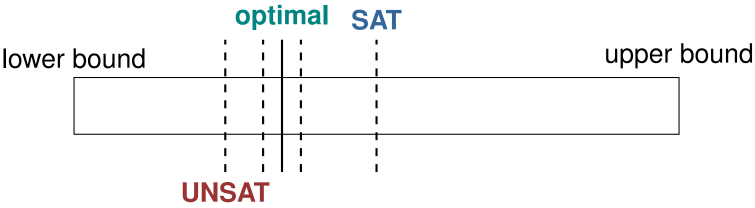
Lessons learnt from SMT solver



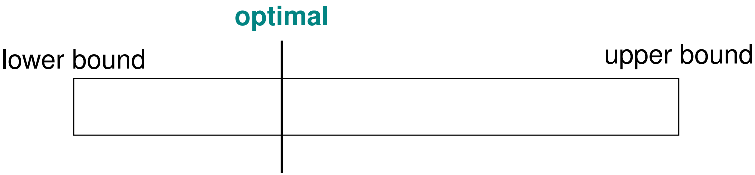
Lessons learnt from SMT solver



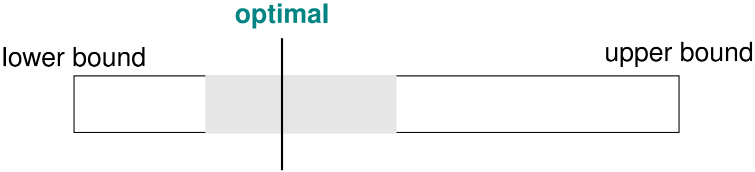
Lessons learnt from SMT solver



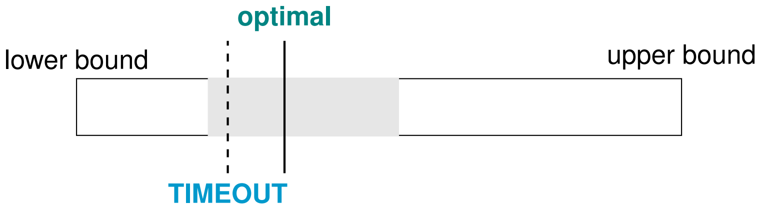
Lessons learnt from SMT solver



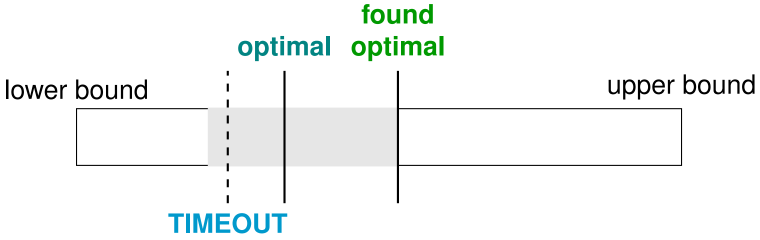
Lessons learnt from SMT solver



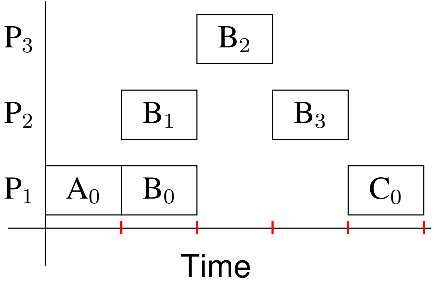
Lessons learnt from SMT solver



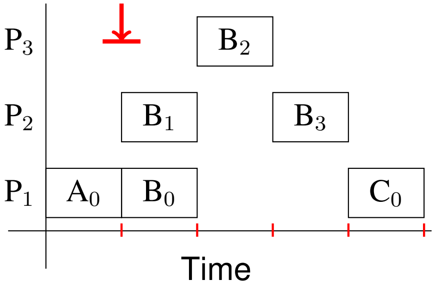
Lessons learnt from SMT solver



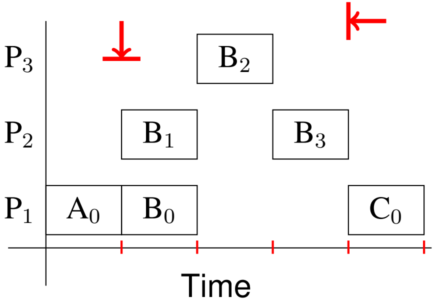
Lessons learnt from SMT solver



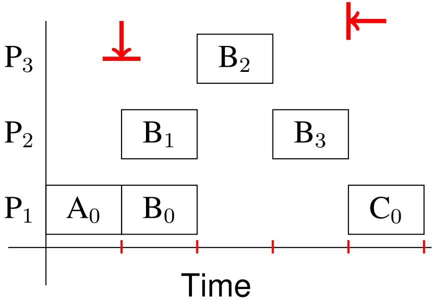
Lessons learnt from SMT solver

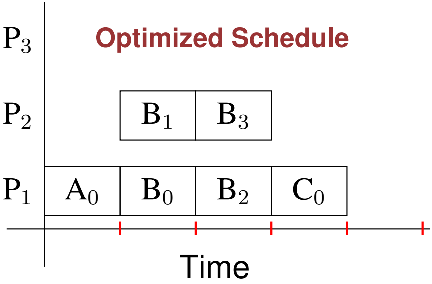


Lessons learnt from SMT solver

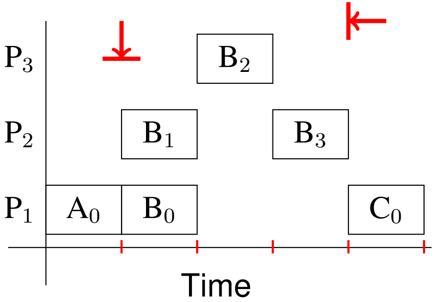


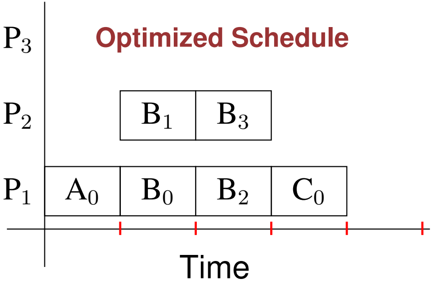
Lessons learnt from SMT solver





Lessons learnt from SMT solver

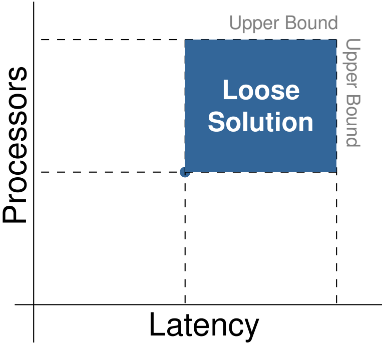




Such constraints makes the problem **harder** for SMT

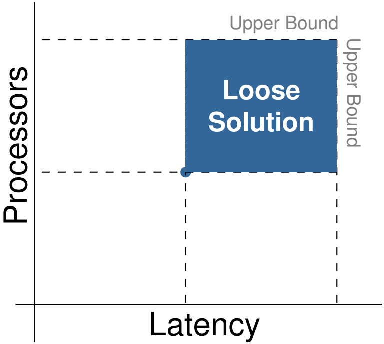
Two-step optimization

- Get a **loose** schedule from the solver



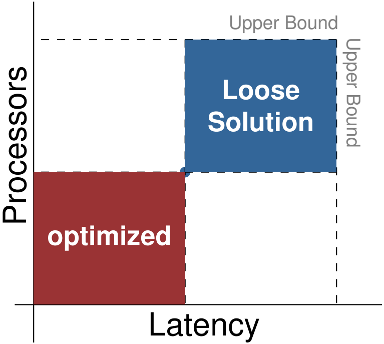
Two-step optimization

- Get a **loose** schedule from the solver
- **Optimize** it for:
  - Latency
  - Processors used



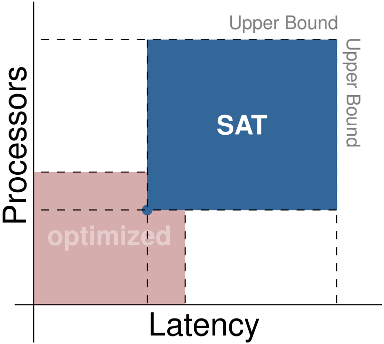
Two-step optimization

- Get a **loose** schedule from the solver
- **Optimize** it for:
  - Latency
  - Processors used



Two-step optimization

- Get a **loose** schedule from the solver
- **Optimize** it for:
  - Latency
  - Processors used



Overview

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

References

- 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

Thank You

Questions?