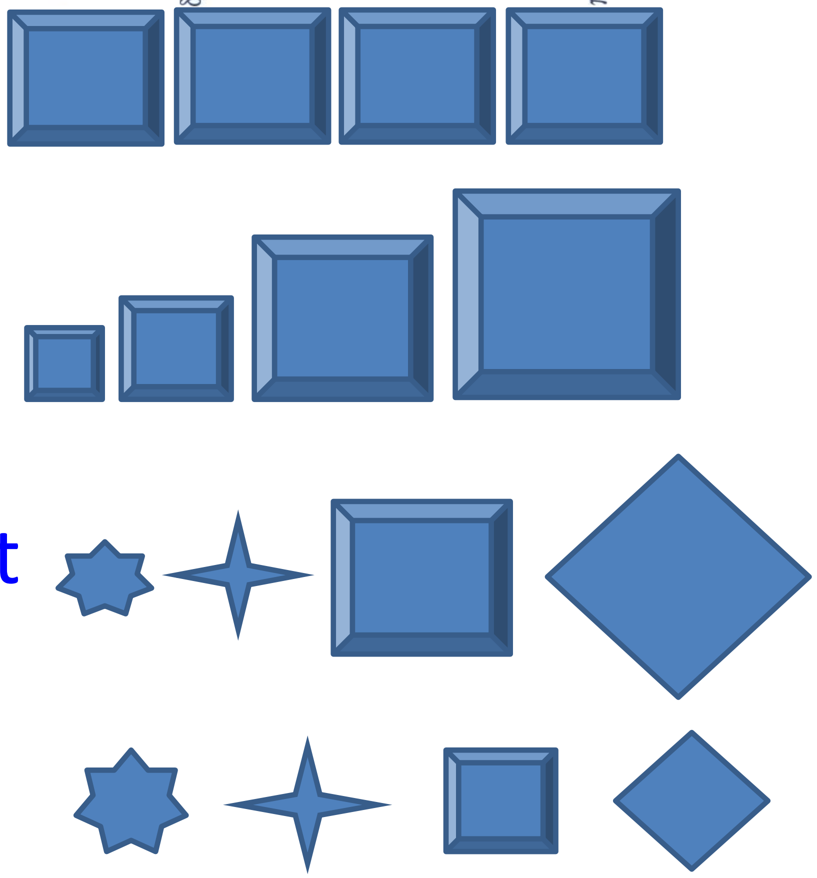
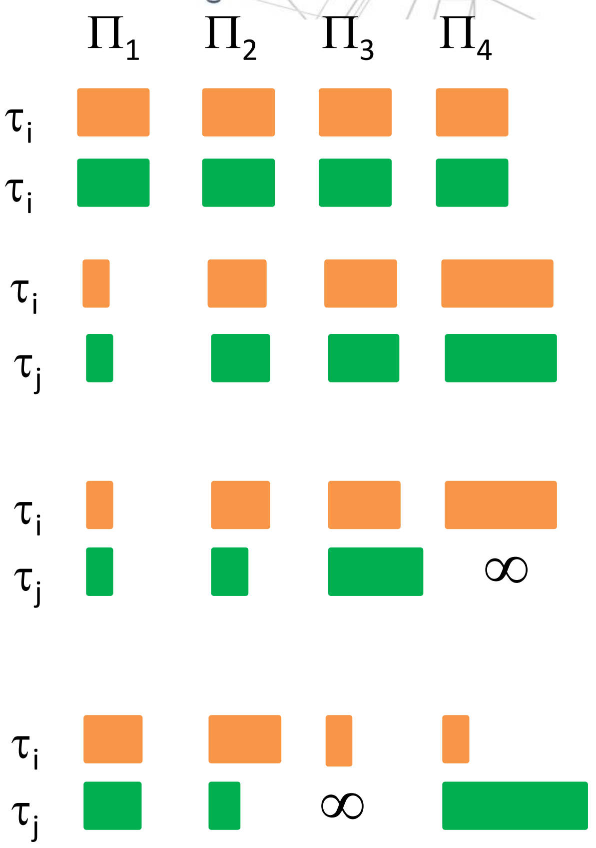
Global Scheduling on Heterogenous MPSoCs

Emmanuel Grolleau
grolleau@ensma.fr
Based on a work done with :
Antoine Bertout, Joël Goossens, Xavier Poczekajlo, Roy Jamil

Classification of mp platforms

- Identical
- Uniform
- Consistent
- Unrelated





Important results

- Seminal paper in operations research scheduling Lawler & Labetoulle 1978
- Seminal paper in r-t scheduling of S. Baruah 2004
  - Input: (strictly) periodic independent synchronous implicit deadline tasks, defined by $u_i = C_i/T_i$, C_i defined on a fictional reference core, and for each core Π_j, a rate r_{ij}

utilization	$\Pi_{fictional}$
τ_1	u_1
τ_2	u_2
τ_3	u_3

rates	Π_1	Π_2
τ_1	$r_{11}=1$	$r_{12}=2$
τ_2	$r_{21}=0.5$	$r_{22}=1$
τ_3	$r_{31}=0$	$r_{32}=2$

  - Linear Program: what fraction of core to what task?
  - Theorem: the system is feasible **if and only** if the LP has a solution

Linear Programming problem

Workload assignment	Π_1	Π_2
τ_1	x_{11} × 1	x_{12} × 2
τ_2	0,5 × x_{21}	x_{22} × 1
τ_3		2 × x_{32}

x_{11} x_{21} ≤ 1

x_{12} x_{22} x_{32} ≤ 1

$$\forall \text{core } j, \sum_{\forall \text{task } i} x_{ij} \leq 1$$

x_{11}× 1 x_{12}× 2 = u_1

x_{11} x_{12} ≤ 1

x_{21} × 0,5

x_{22} = u_2

x_{21} x_{22} ≤ 1

x_{32}× 2 = u_3

x_{32} ≤ 1

$$\forall \text{task } i, \sum_{\forall \text{core } j} x_{ij} r_{ij} = u_i$$

$$\forall \text{task } i, \sum_{\forall \text{core } j} x_{ij} \leq 1$$

From workload assignment to feasible schedule

Workload assignment	Π_1	Π_2	$Latency_1$	$Latency_2$	$Latency_3$	
τ_1	x_{11}	x_{12}	$l_1 = 1-(x_{11}+x_{12})$			$\Sigma=1$
τ_2	x_{21}	x_{22}		$l_2 = 1-(x_{21}+x_{22})$		$\Sigma=1$
τ_3		x_{32}			$l_3 = 1-(x_{31}+x_{32})$	$\Sigma=1$
$Idle_1$	$1-(x_{11}+x_{21}) = i_1$		x_{11}	x_{21}		$\Sigma=1$
$Idle_2$			x_{12}	x_{22}	x_{32}	$\Sigma=1$
	$\Sigma=1$	$\Sigma=1$	$\Sigma=1$	$\Sigma=1$	$\Sigma=1$	

- This is a Doubly Stochastic (DS) matrix!

Birkhoff-von Neumann (BvN) theorem (1946)

- A DS matrix
  - Square, non negative values, sum of each row and column is 1
- A DS matrix can be expressed as a convex combination of permutation matrices

$$\begin{pmatrix} 0.3 & 0.4 & 0.3 \\ 0.5 & 0.5 & 0 \\ 0.2 & 0.1 & 0.7 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \times 0.1 + \begin{pmatrix} 0.3 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0 \\ 0.2 & 0 & 0.7 \end{pmatrix}$$

- Note that $1/0.9 \times \begin{pmatrix} 0.3 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0 \\ 0.2 & 0 & 0.7 \end{pmatrix}$ is DS

Convex combination of permutation matrices

$$\begin{pmatrix} 0.3 & 0.4 & 0.3 \\ 0.5 & 0.5 & 0 \\ 0.2 & 0.1 & 0.7 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \times 0.1 + \begin{pmatrix} 0.3 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0 \\ 0.2 & 0 & 0.7 \end{pmatrix}$$

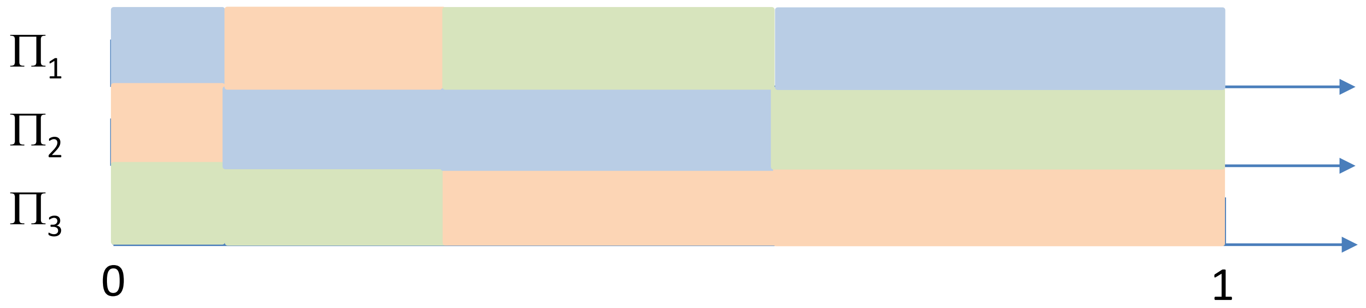
$$\begin{pmatrix} 0.3 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0 \\ 0.2 & 0 & 0.7 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \times 0.2 + \begin{pmatrix} 0.3 & 0.4 & 0 \\ 0.4 & 0.3 & 0 \\ 0 & 0 & 0.7 \end{pmatrix}$$

$$\begin{pmatrix} 0.3 & 0.4 & 0 \\ 0.4 & 0.3 & 0 \\ 0 & 0 & 0.7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \times 0.3 + \begin{pmatrix} 0 & 0.4 & 0 \\ 0.4 & 0 & 0 \\ 0 & 0 & 0.4 \end{pmatrix}$$

Convex combination of permutation matrices

$$\begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.5 & 0.5 & 0 \\ 0.2 & 0.1 & 0.7 \end{bmatrix} = 0.1 \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} + 0.2 \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} + 0.3 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + 0.4 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

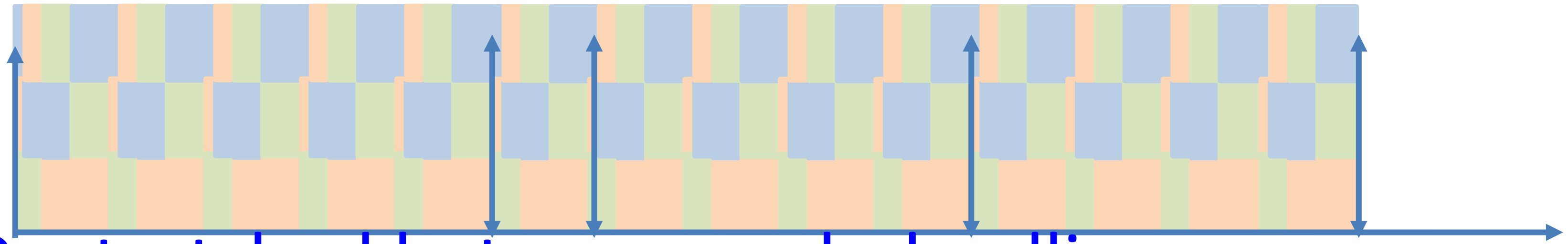
- As a real-time scheduling guy, I can interpret it as a schedule…



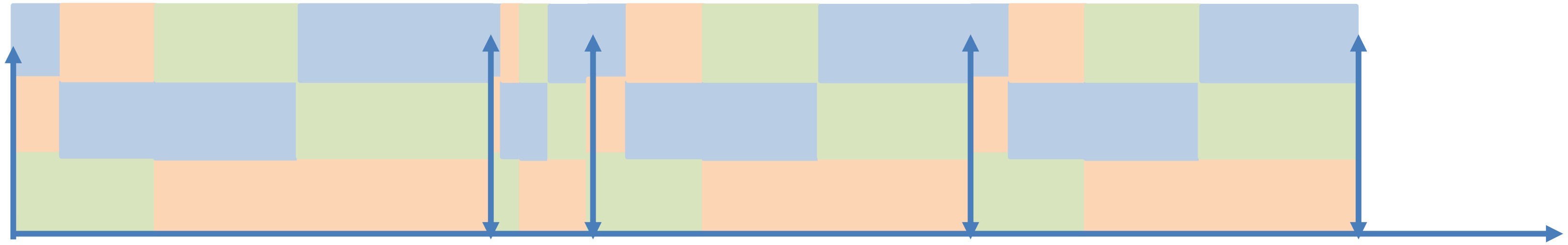
- Each permutation matrix generates a scheduling point
  - Preemption and/or migration

From template schedule to schedule

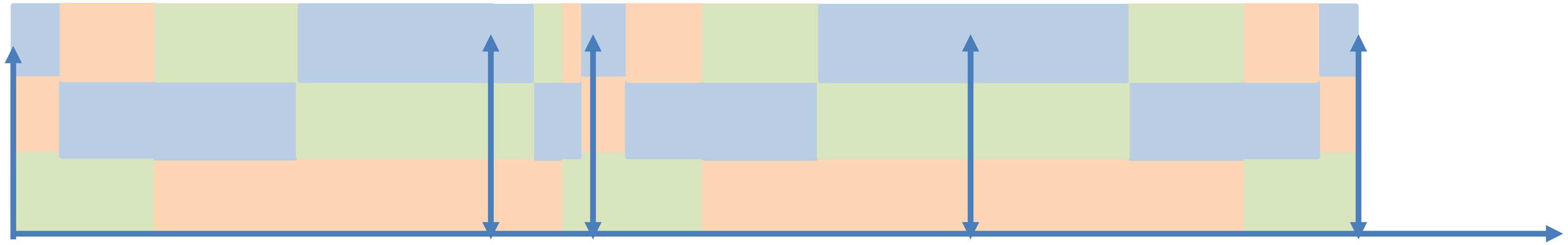
- A template schedule can be repeated on each time unit



- Or stretched between each deadline



- Mirrored every other time



Discussion about the template schedule

- Can always be obtained from a DS matrix (BvN decomposition theorem)
- Obtaining a valid workload assignment matrix is a necessary and sufficient schedulability condition
  - Under the hypothesis of no preemption cost, no migration cost
- The produced off-line schedule supposes an « almost fluid » scheduler, able to preempt/migrate tasks several times per time unit
- Number of permutation matrices = number of scheduling points per template schedule

BvN decomposition strategies

- Minimizing the number of permutation matrices is NP-hard in the strong sense (from 3-Partition in [Dufossé 2015])
- Heuristic to reduce migrations/preemptions
  - In the previous example, Birkhoff method (minimum non null value)
  - Maximize locally the duration of each assignment?

$$\begin{pmatrix} 0.3 & 0.4 & 0.3 \\ 0.5 & 0.5 & 0 \\ 0.2 & 0.1 & 0.7 \end{pmatrix} = 0.4 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} + 0.2 \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} + 0.1 \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

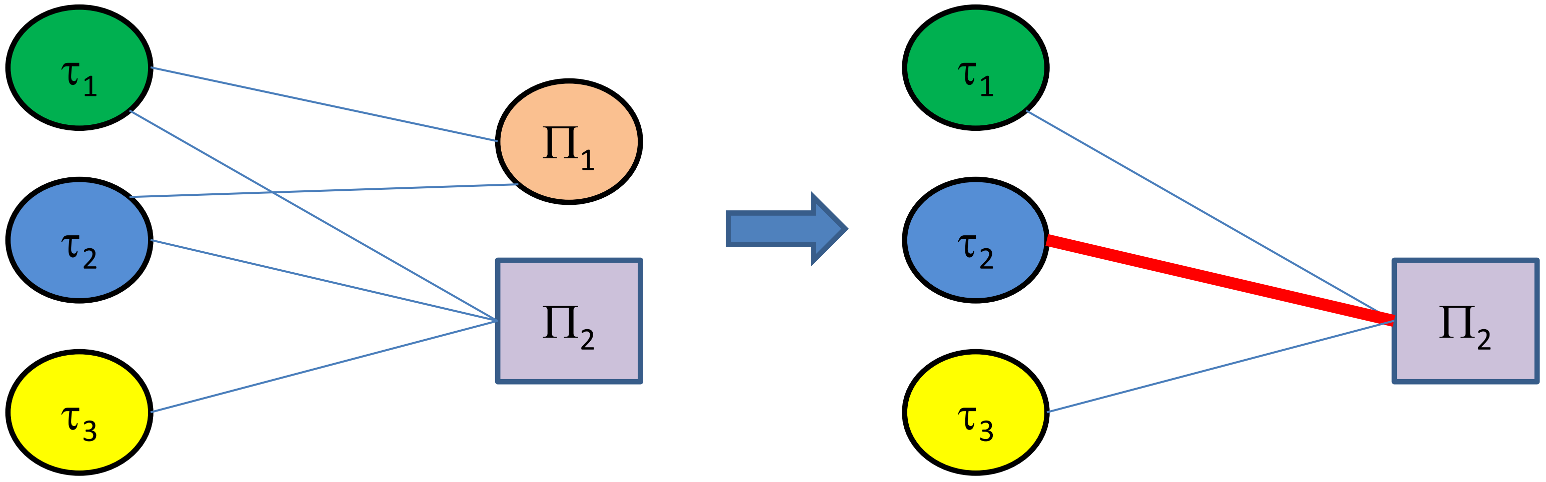
  - ✓ Solving a Linear Bottleneck Assignment Problem at each step
  - ✓ Polynomial time using Hungarian method

S. Baruah's strategy

Workload assignment	Π_1	Π_2	$Latency_1$	$Latency_2$	$Latency_3$	
τ_1	x_{11}	x_{12}	$l_1 = 1-(x_{11}+x_{12})$			$\Sigma=1$
τ_2	x_{21}	x_{22}		$l_2 = 1-(x_{21}+x_{22})$		$\Sigma=1$
τ_3		x_{32}			$l_3 = 1-(x_{31}+x_{32})$	$\Sigma=1$
$Idle_1$	$1-(x_{11}+x_{21}) = i_1$		x_{11}	x_{21}		$\Sigma=1$
$Idle_2$			x_{12}	x_{22}	x_{32}	$\Sigma=1$
	$\Sigma=1$	$\Sigma=1$	$\Sigma=1$	$\Sigma=1$	$\Sigma=1$	

An "as conservative" template schedule as possible

- Square node = urgent task or full processor, assign absolutely
- Circle node = non urgent, non full, assign if necessary

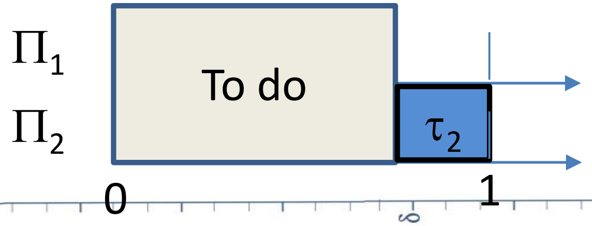


Arbitrary marriage

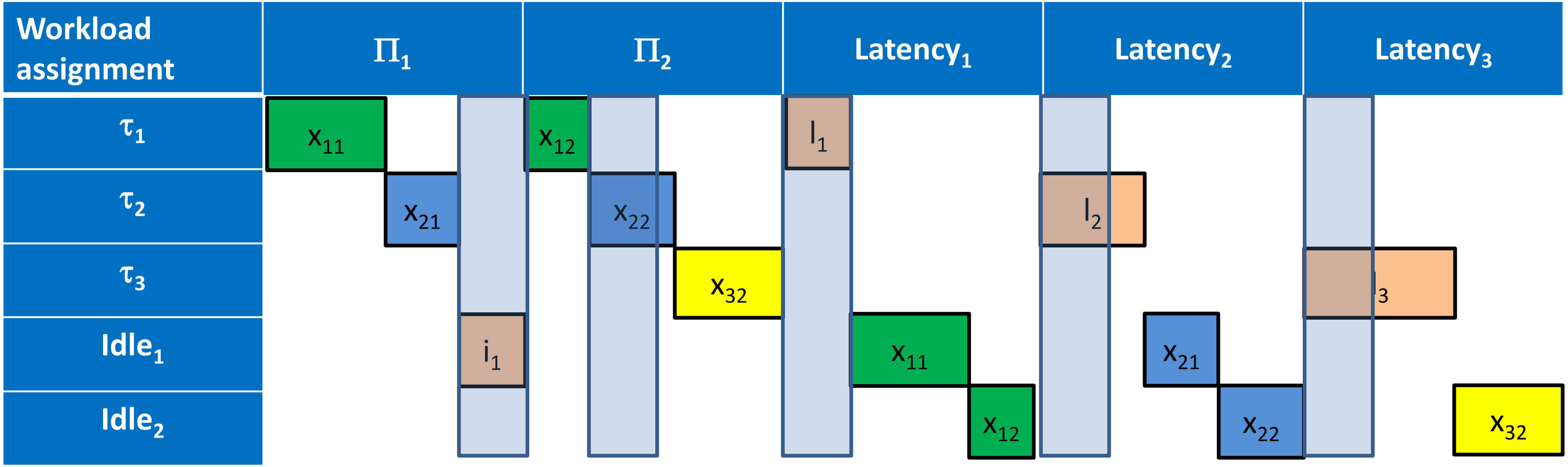
Reverse construction of template schedule

Workload assignment	Π_1	Π_2	$Latency_1$	$Latency_2$	$Latency_3$
τ_1	x_{11}	x_{12}	l_1		
τ_2	x_{21}	x_{22}		l_2	
τ_3		x_{32}			l_3
$Idle_1$	i_1		x_{11}	x_{21}	
$Idle_2$			x_{12}	x_{22}	x_{32}

Smallest non null value

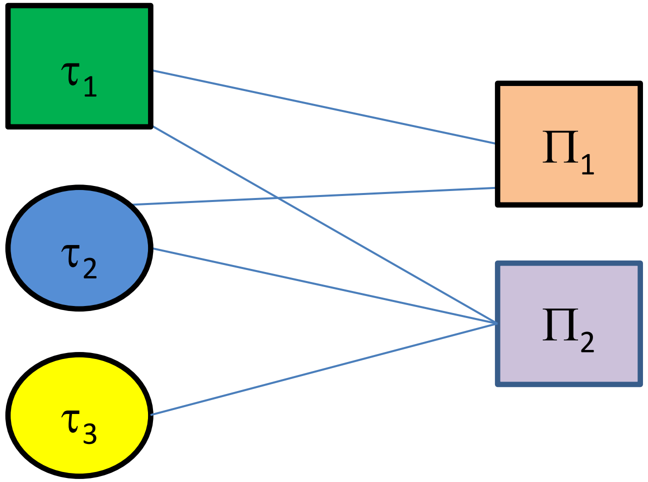


Reverse construction of template schedule$_{/contd}$



- Π_1 is now full, like Π_2, and they will remain full until the… beginning of the template schedule
- τ_1 is now urgent, and will remain urgent until the beginning

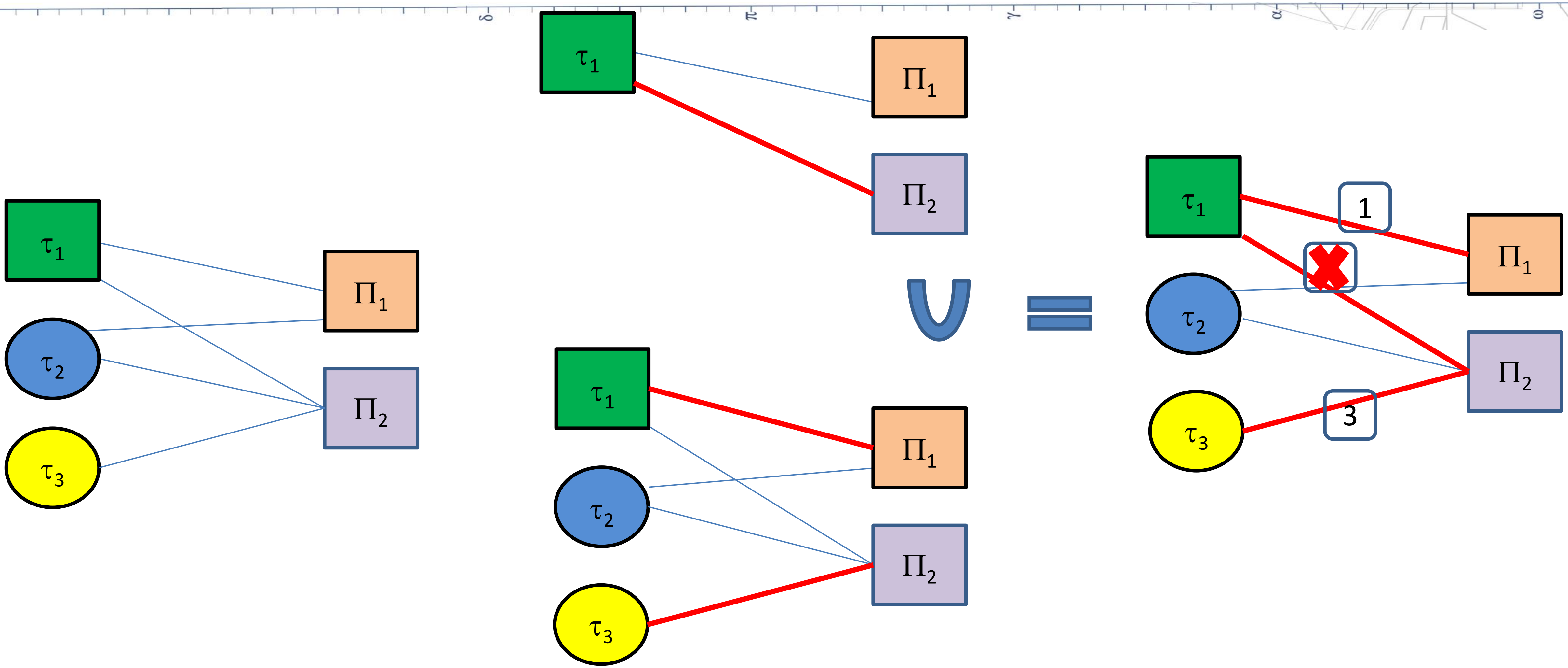
Marriage problem



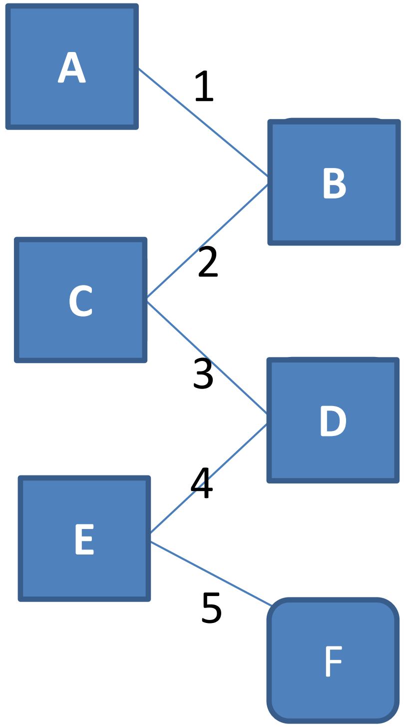
- Find a marriage such that each square node is married
- Circle nodes are "spare nodes"

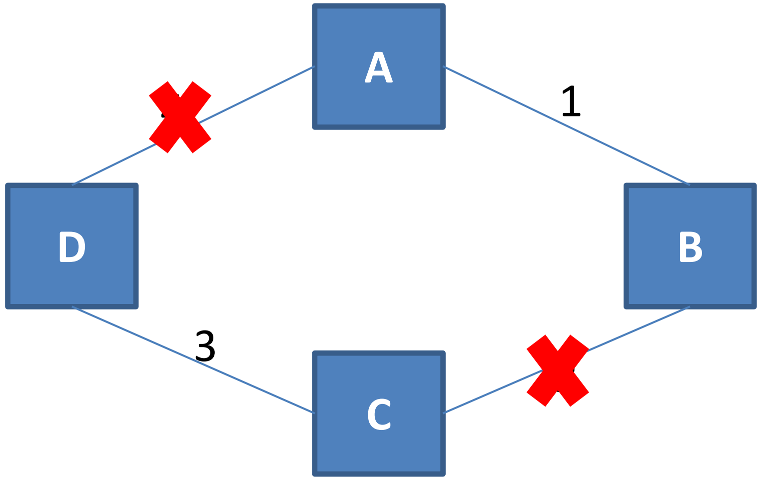
- Could we invent a new "marriage in the nobility" problem?

Baruah's method



Cleaning method

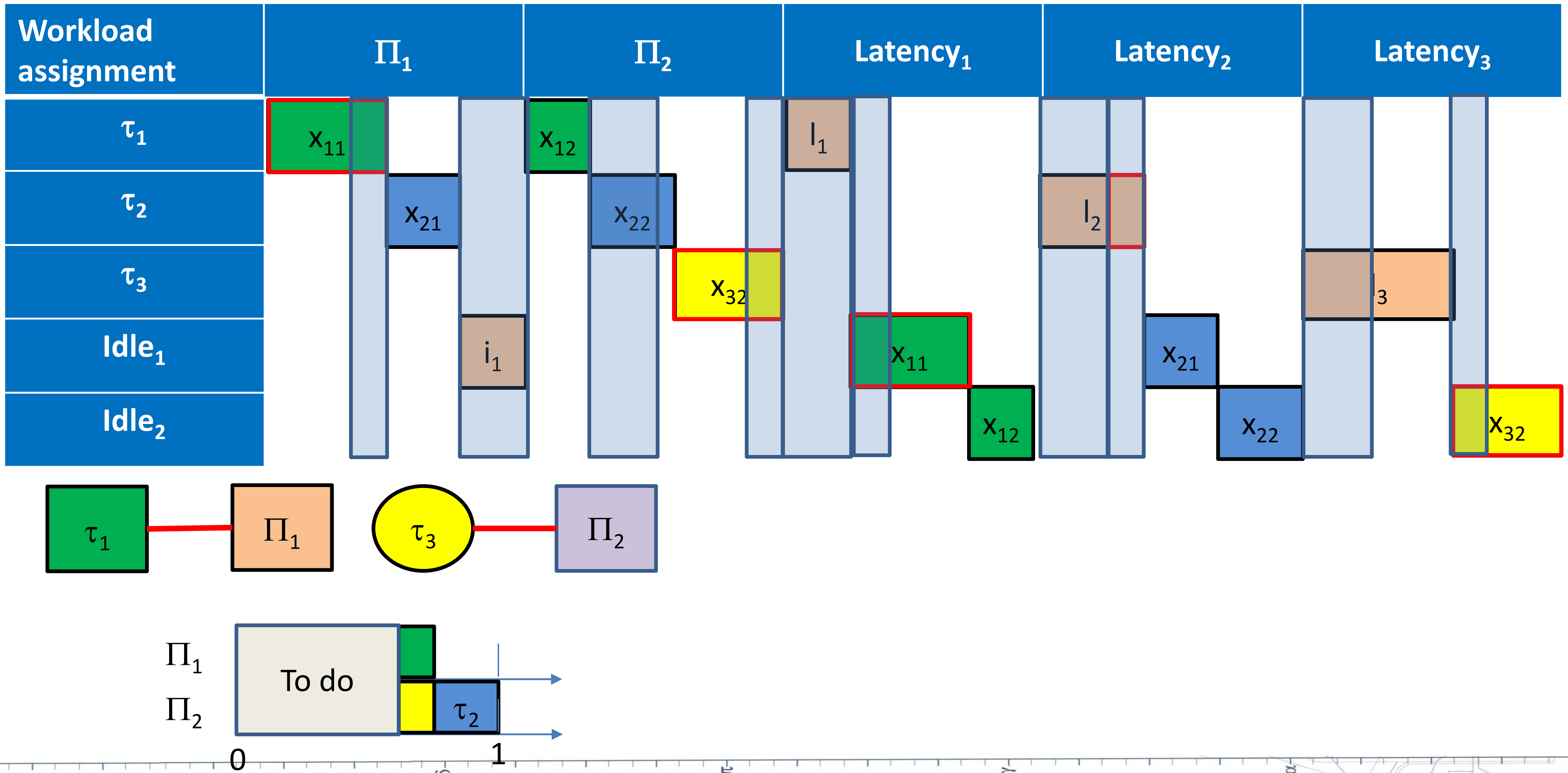




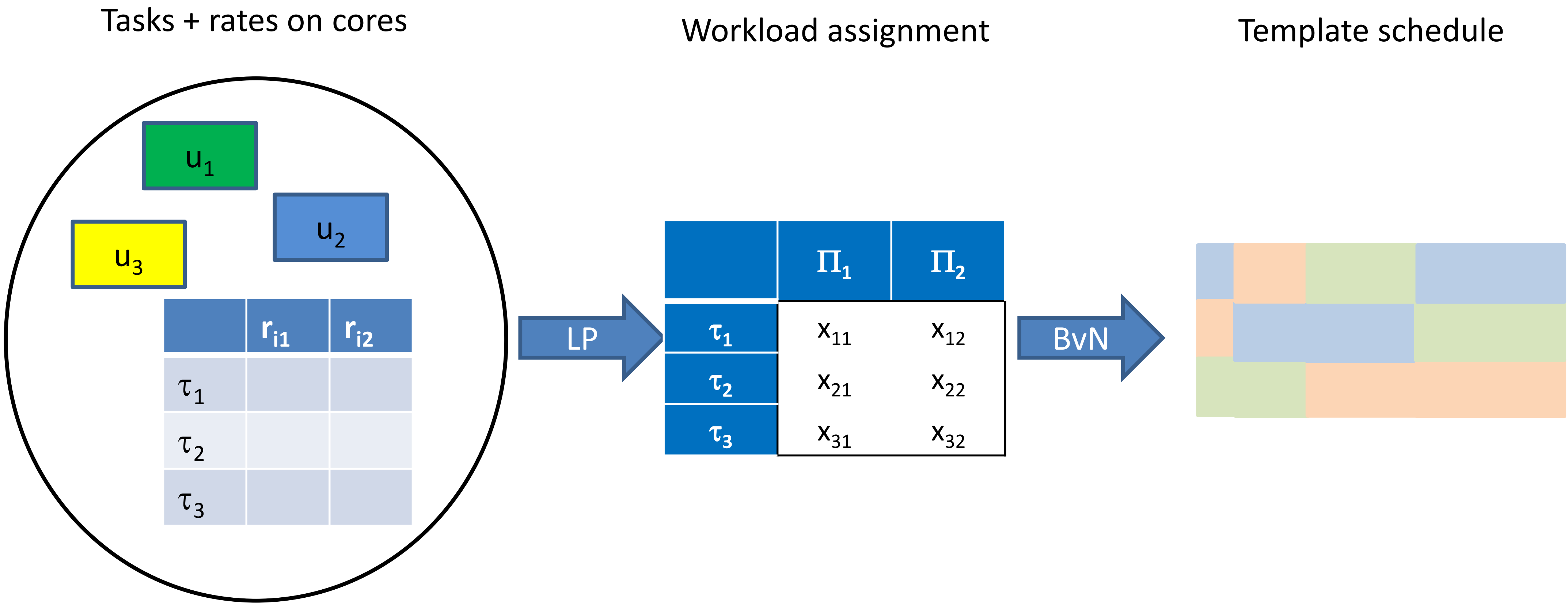
- Number any path starting from an important node
- Cut the edges with an even number

- Only important nodes can be the endpoints of two edges
- An non cyclic even length path has a non important node as one of its extermetiies
- A cycle can only have an even length path

Template schedule construction



Summary



Tasks + rates on cores
Workload assignment
Template schedule
u_1
u_2
u_3
r_{i1} r_{i2}
τ_1
τ_2
τ_3
LP
Π_1 Π_2
τ_1 x_{11} x_{12}
τ_2 x_{21} x_{22}
τ_3 x_{31} x_{32}
BvN

Remarks on the LP

$$\forall \text{task } i, \sum_{\forall \text{core } j} x_{ij} r_{ij} = u_i$$

$$\forall \text{task } i, \sum_{\forall \text{core } j} x_{ij} \le 1$$

$$\forall \text{core } j, \sum_{\forall \text{task } i} x_{ij} \le 1$$

$$\forall \text{task } i, \sum_{\forall \text{core } j} x_{ij} r_{ij} = u_i$$

$$\forall \text{task } i, \sum_{\forall \text{core } j} x_{ij} \le \boldsymbol{L}$$

$$\forall \text{core } j, \sum_{\forall \text{task } i} x_{ij} \le \boldsymbol{L}$$

LP Feas Obj: min L

- The objective function can be any

LP Load Obj: min $\sum_j \sum_i x_{ij}$

- Room left for e.g. energy or heat dissipation optimization
- Constraints can be added

$$\left.\begin{array}{r} \forall \text{task } i, \forall \text{core } j, b_{ij} \epsilon \{0,1\} \\ x_{ij} \le b_{ij} \\ b_{ij} < 1 + x_{ij} \end{array}\right\} b_{ij} \text{ is 1 iff } \tau_i \text{ uses } \Pi_j\text{, 0 else}$$

ILP Mig Obj: min $\sum_j \sum_i b_{ij}$

LP Feas vs. LP Load

	C_i	T_i
τ_1	5	10
τ_2	5	10

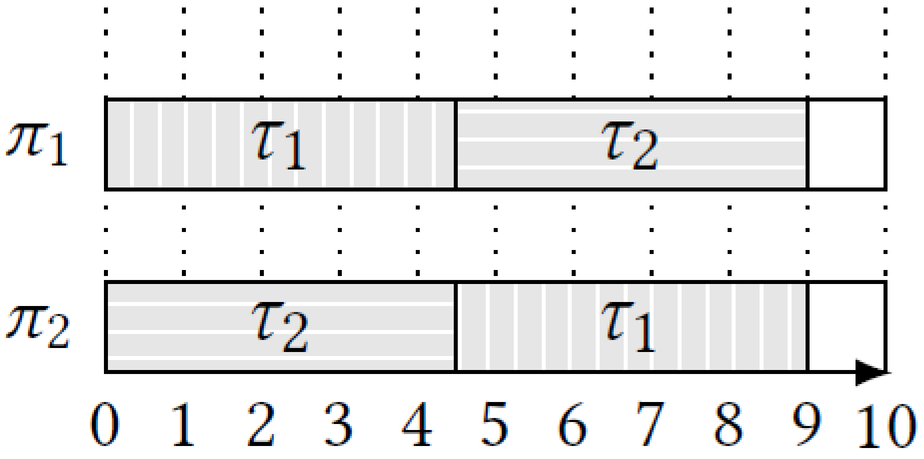
Rates	π_1	π_2
τ_1	10	1
τ_2	10	1

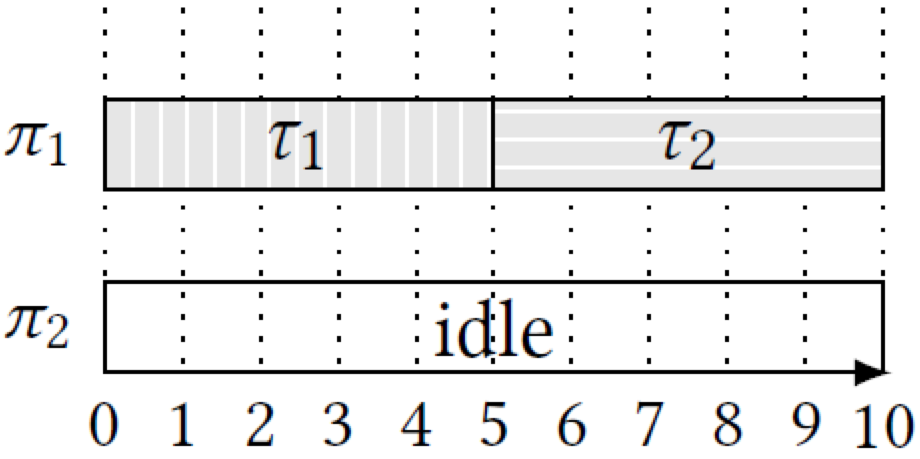
LP Feas

$$\begin{bmatrix} 5/11 & 5/11 \\ 5/11 & 5/11 \end{bmatrix}$$

LP Load

$$\begin{bmatrix} 0{,}5 & 0{,}5 \\ 0 & 0 \end{bmatrix}$$





BvN, LBAP or Conservative decomposition?

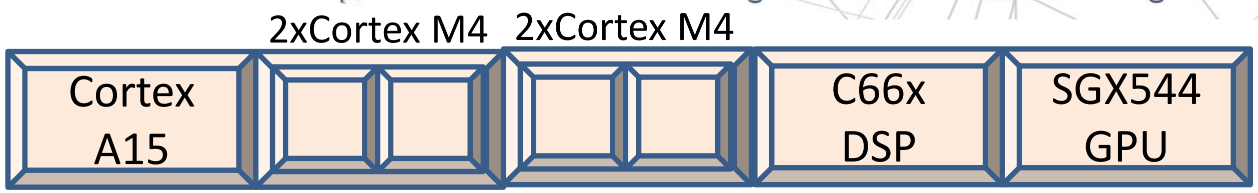
- Metrics: number of migrations & preemptions
- And (experimentally) the winner is…

- The conservative decomposition

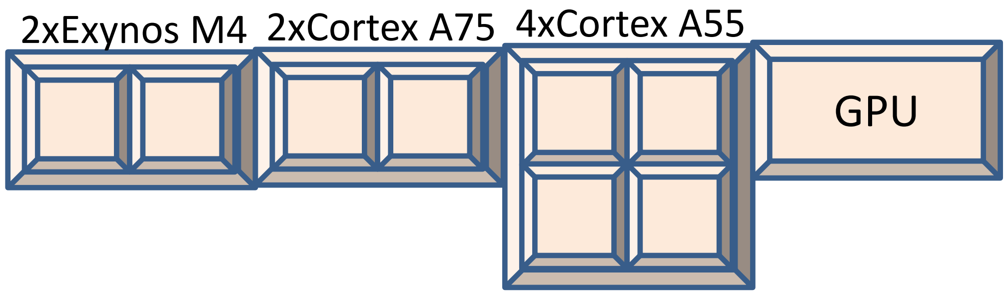
- Traveler salesman problem

Heterogeneous MPSoCs : a growing trend

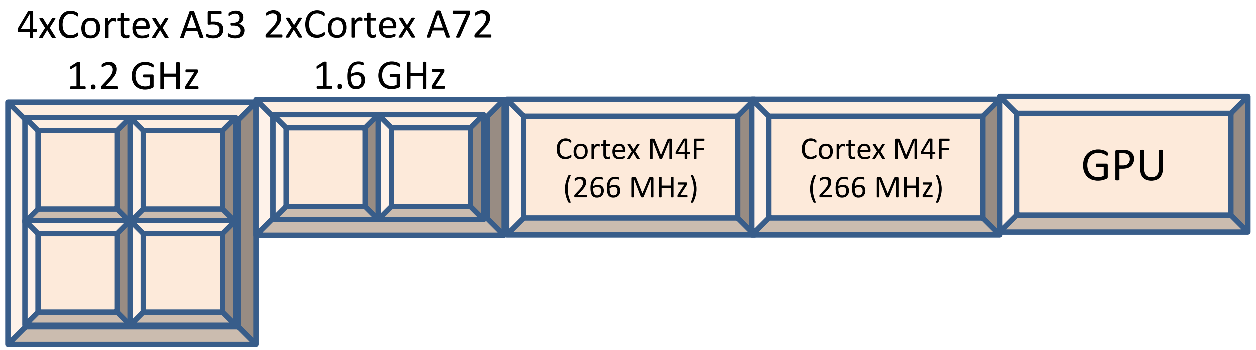
- ❑TI Sitara AM57x
  - ➢Embedded computing, robotics, avionics, medical imaging, etc.



- ❑Samsung Exynos 9 9820
  - ➢Smartphone

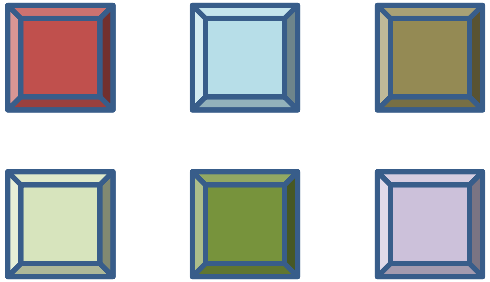


- ❑NXP i.MX 8 QuadMax
  - ➢Automotive, etc.

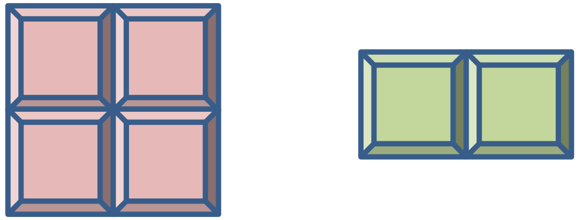


Clustered platform

- Rather than having a heterogeneous platform



- Consider a set of clusters of identical cores



Heterogeneous MPSoCs

$$\forall \text{task } i, \sum_{\forall \text{cluster } j} x_{ij} r_{ij} = u_i$$

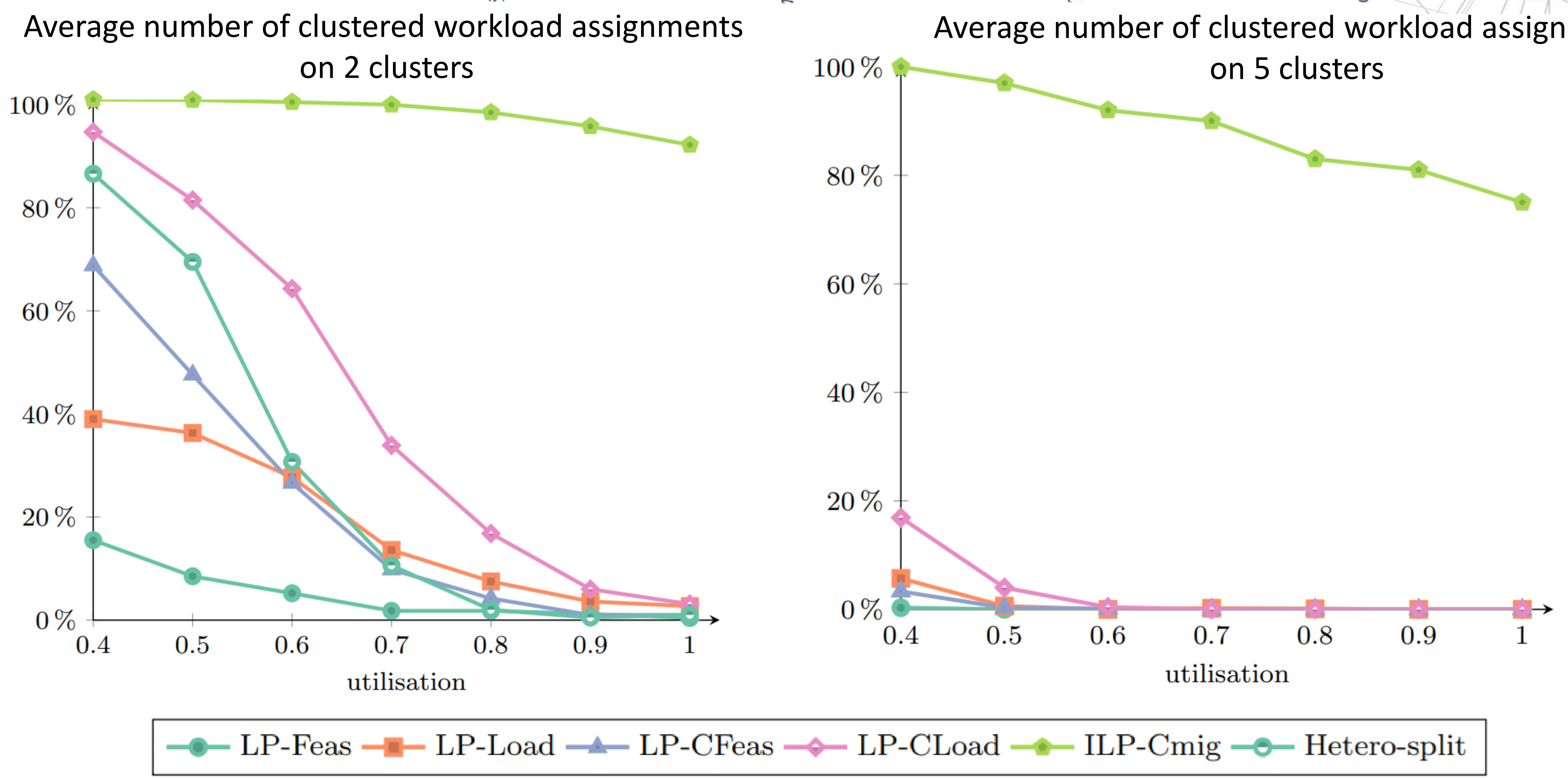
$$\forall \text{task } i, \sum_{\forall \text{cluster } j} x_{ij} \leq 1$$

$$\forall \text{cluster } j, \sum_{\forall \text{task } i} x_{ij} \leq m_j$$

Workload assignment	$\dot{\Pi}_1$	$\dot{\Pi}_2$	
τ_1	x_{11} × 1	x_{12} × 2	$\Sigma \leq 1$
τ_2	0,5 × x_{21}	1 × x_{22}	$\Sigma \leq 1$
τ_3	2 ×	x_{32}	$\Sigma \leq 1$
	$\Sigma \leq m_1$	$\Sigma \leq m_2$	

- A system is feasible iff LP has a solution => always possible to build a DS matrix
- Less variables (rate r_{ij} per cluster)
  - ILP for inter-cluster migrations minimization smaller
- Inter-cluster ≠ Intra-cluster migration
  - Experimentally 10 to 70µs vs. 1 to 2 µs on i.mx8 and STM32MP1

Comparison flat vs. clustered



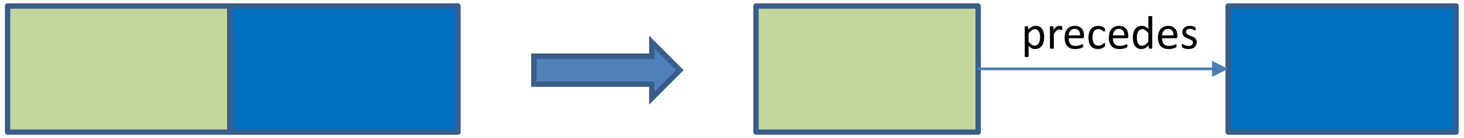
Performance gain

- Average execution time (in seconds)

	2 clusters	5 clusters
LP-Feas	0.013	0.464
LP-Load	0.012	0.562
LP-CFeas	0.002	0.027
LP-CLoad	0.002	0.029
Hetero-split	0.007	N/A
ILP-Mig	0.061	N/A
ILP-CMig	0.023	0.156

How (un)realistic is the model?

- Zero cost for preemption & migration => already NP-hard for uniprocessor
- « If any portion of a thread is executed for 5% of the time on a core of rate 2, it executes for 10% »
  - What if the first half of a thread uses intensively integers, and the second half uses intensively floats?



- Limited to implicit deadlines, strictly periodic tasks
  - Sporadic tasks with explicit deadlines?
- Offline schedule hard to implement
  - Dynamic schedule à la U-EDF?

Conclusion

- Are used and will be more in the future
- Energy saving possibilities
  - DVFS, DPM
- Global scheduling
  - Can use up to 100% of the platform
  - Can be seen as saving more energy in the future
- Lots to do…