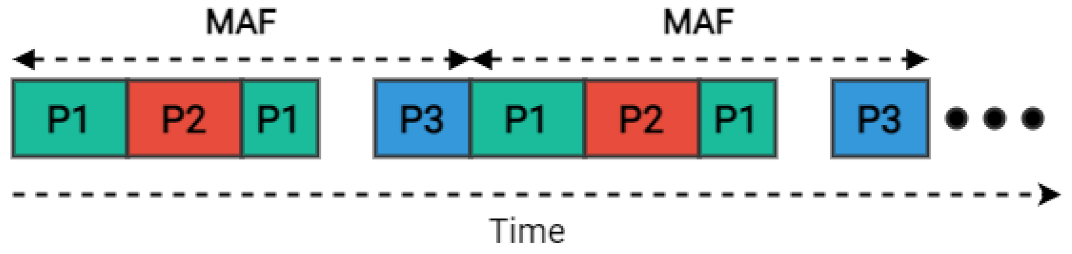
Efficient Scheduling, Mapping and Memory Bandwidth Allocation for Safety-Critical Systems

A. Torres AD[1], J-B. Lefoul[1], A. Ben-Salem[1], S. Harnois[2], F. Gohring de Magalhaes[1], and G. Nicolescu[1]

Polytechnique Montréal[1], MANNARINO Systems and Software[2]

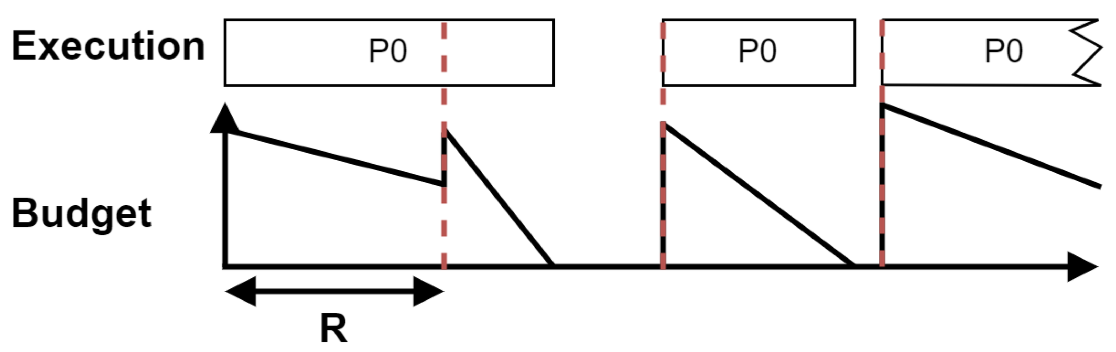
Context

- Safety critical -> **real-time, determinism and safety**
  - Fixed scheduling
  - Fixed resource allocation and management



- Multi-core architectures share components
  - Contention on memory bus
  - Generation of **interferences**
- Bus contention leads to unpredictable execution time
- **Co-scheduling** and **mapping** can increase or decrease interferences

Bandwidth control is used to limit bus contention. **Budget**: number of bus transaction for a regulation period R. When budget is **exceeded**, the **partition is paused** until next regulation period.



Objectives

- **Optimize** bandwidth allocation to maximize performances.
- **Help** the system designers to generate a mapping and scheduling scheme.
- **Extend** current state-of-the-art with scalable methods that are applicable in real life.

Contributions

- Propose a novel method capable of **mapping, scheduling and allocating** bus budgets while **scaling** with the size of the system.
- Introduce **constraints** on partitions required by **real-life** applications (dependencies, mapping and release time).
- **Overcome** previous work **limitations** (partitions with different periods, cores with different scheduling).

System Model

N cores, M partitions P_k

MAF Q_i ($i \in [0; N\text{-}1]$) divided in $\frac{Q_i}{R}$ quanta q of length R

$P_k = \langle T_k, A_k, N_k, (K_k; X_k) \rangle$. T_k: Period. A_k: Release time.

N_k : Set of cores on which P_k can mapped.

$(K_k; X_k)$: Couple of bus budget and allocated time.

- **Dependency**: P_j depends on $P_i \implies A_j \geq A_i + X_i$.
- **Release time**: Constrained release time $\implies A_k = cst$.
- **Mapping**: Constrained core set $\implies N_k = \{1, 3 \dots\}$.

Solution: array of budgets, k^{th} element $\implies$ budget of P_k.

30	45	75	35	50	Current solution

Solution Search

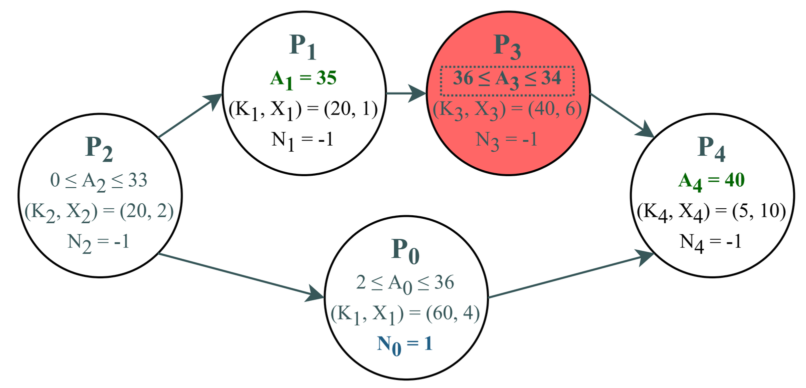
Evolutionary approach to find $(K_k; X_k)$ for all partitions:

1) Generate set of neighbor solutions
2) Precheck each neighbor
3) For each neighbor, find a scheduling / mapping scheme using best fit

Objective function based search -> not only provide a solution but also optimize it

Scaling Methods

Precheck for a given set of $(K_k; X_k)$:



$$max(A_k) = \min_{\forall j \text{ if } k \in Dep(j)} \{max(A_j)\} - X_k \quad (1)$$

$$min(A_k) = \max_{\forall j \in Dep(k)} \{min(A_j) + X_j\} \quad (2)$$

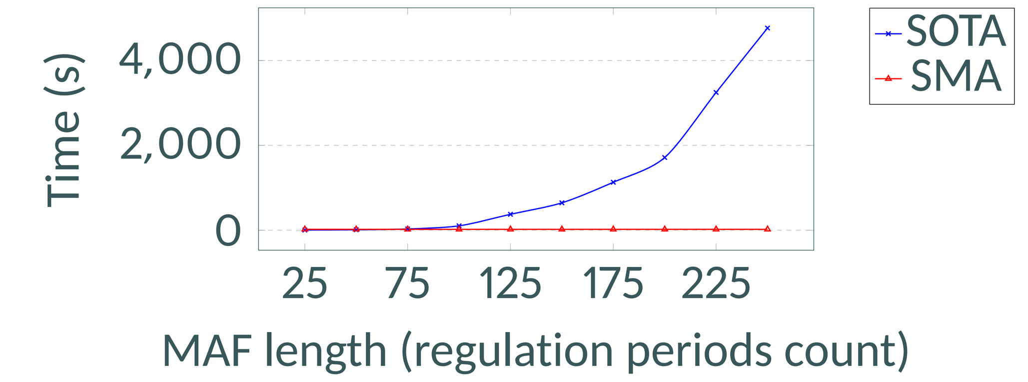
$$\textbf{Valid if } 0 \leq min(A_k) \leq max(A_k) \leq T_k - X_k \quad (3)$$

Speed up best fit: use time slot instead of quanta

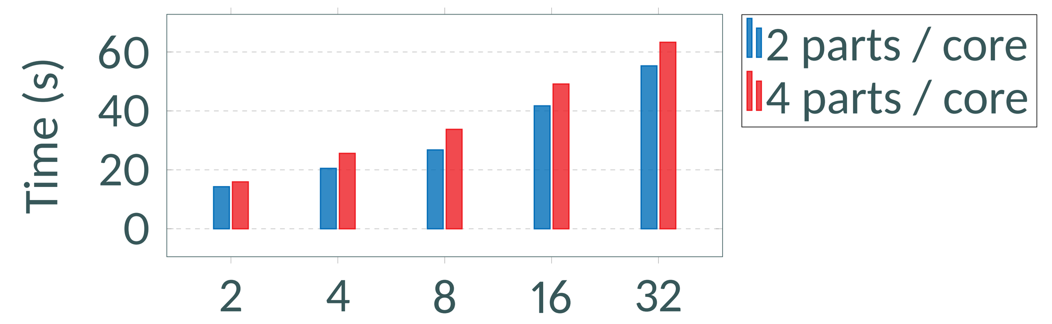
MAF Core 0	Slot 0 (free)	Slot 1 (P_0)	Slot 2 (free)	Slot 3 (P_0)

Time slots gather multiple quanta, reducing the number of comparisons for the best fit algorithm.

Problem Scaling



Comparison of the time elapsed to find a solution depending on the MAF length.



SMA solving time for different system configurations.

Conclusion

In this work we **propose** a novel method to:

- Provide **scheduling, mapping and bandwidth allocation** for partitioned systems.
- Allow the use of **real-life constraints**.

We improve the **scalability** compared to existing work:

- Our method is up to **42.26 times faster** and **scales** well with the size of the system.
- SMA **schedules between 69% and 88%** of the data sets.
- We provide **partial solutions** when the system is not schedulable.

Our approach was successfully adapted to a real-life case study (ROSACE).

Results

We extended the ILP model (SOTA) to take into account constraints and compare it with our approach (SMA). We generated 100 sets with the following parameters:

Name	Description	Values
n	Number of partitions in the system	2, 4, 8, 16, 32, 64
c	Number of cores in the system	2, 4, 8, 16
μ	Average system's memory intensity	10%, 20%, 30%, 50%
Q	MAF length (in number of quantum)	[25; 300] by step of 25
Inc	Evolutionary step size	5

Execution times of SMA and SOTA (in seconds).

	2 Cores (c = 2)				4 Cores (c = 4)			
μ(%)	SOTA		SMA		SOTA		SMA	
	4P	8P	4P	8P	8P	16P	8P	16P
10	1.3	51.2	**2.5**	**5.7**	1645.4	1611.0	**61.7**	**155.1**
20	17.5	179.3	**2.6**	**6.53**	1727.5	1721.9	**70.4**	**357.3**
30	**20.9**	1182.3	33.8	**68.3**	2047.8	2367.9	**78.1**	**452.0**
50	137.5	2409.5	**40.5**	**314.1**	4962.7	5409.5	**85.7**	**528.5**

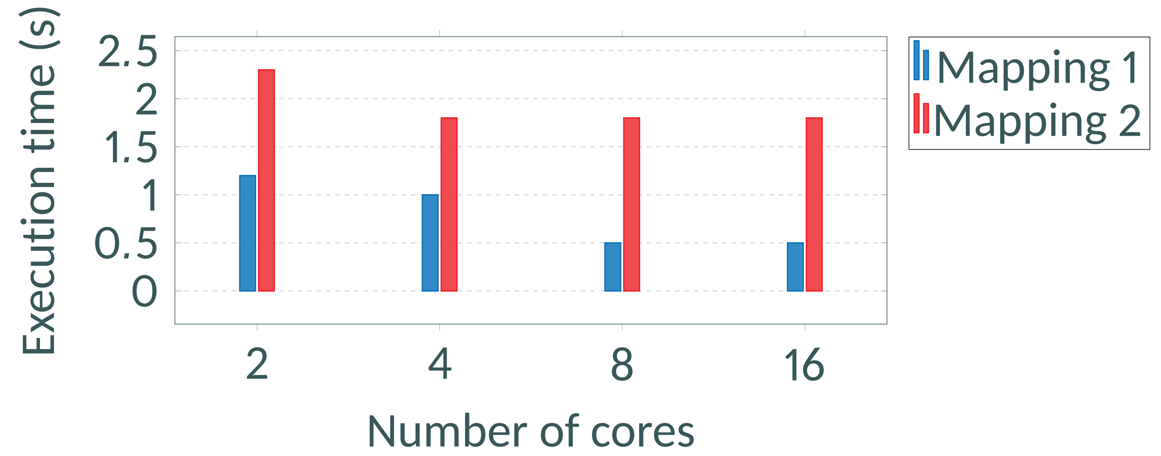
SMA is **9.64 times faster** for 2-cores systems and **42.26 times faster** for 4-cores systems.

Schedulability results for SOTA and SMA (c = 4).

	SOTA		SMA		Partial solutions	
μ (%)	8P	16P	8P	16P	8P	16P
10	91%	91.3%	91%	91.1%	7	14
20	86.3%	86.8%	84.3%	84.5%	7	13
30	79%	78.8%	75.3%	75.1%	6	13
50	73.1%	73.2%	70%	69.7%	6	11

SMA **schedules between 69% and 88%** of the partitions. We provide the reasons (deadline miss, budget overflow, etc.) of the non-schedulability.

Solving the real-life ROSACE system on all cores (Mapping 1) or two cores (Mapping 2).



Acknowledgements and Contact

Contact: alexy.torres-aurora-dugo@polymtl.ca

The authors would like to thank NSERQ and CRIAQ for supporting this research.

POLYTECHNIQUE MONTRÉAL UNIVERSITÉ D'INGÉNIERIE MANNARINO NSERC CRSNG CRIAQ