Memory Feasibility Analysis of Parallel Tasks Running on Scratchpad-Based Architectures

Daniel Casini,[+] **Alessandro Biondi**,[+] Geoffrey Nelissen,[*] and Giorgio Buttazzo[+]

[+] *Scuola Superiore Sant'Anna, Pisa, Italy*

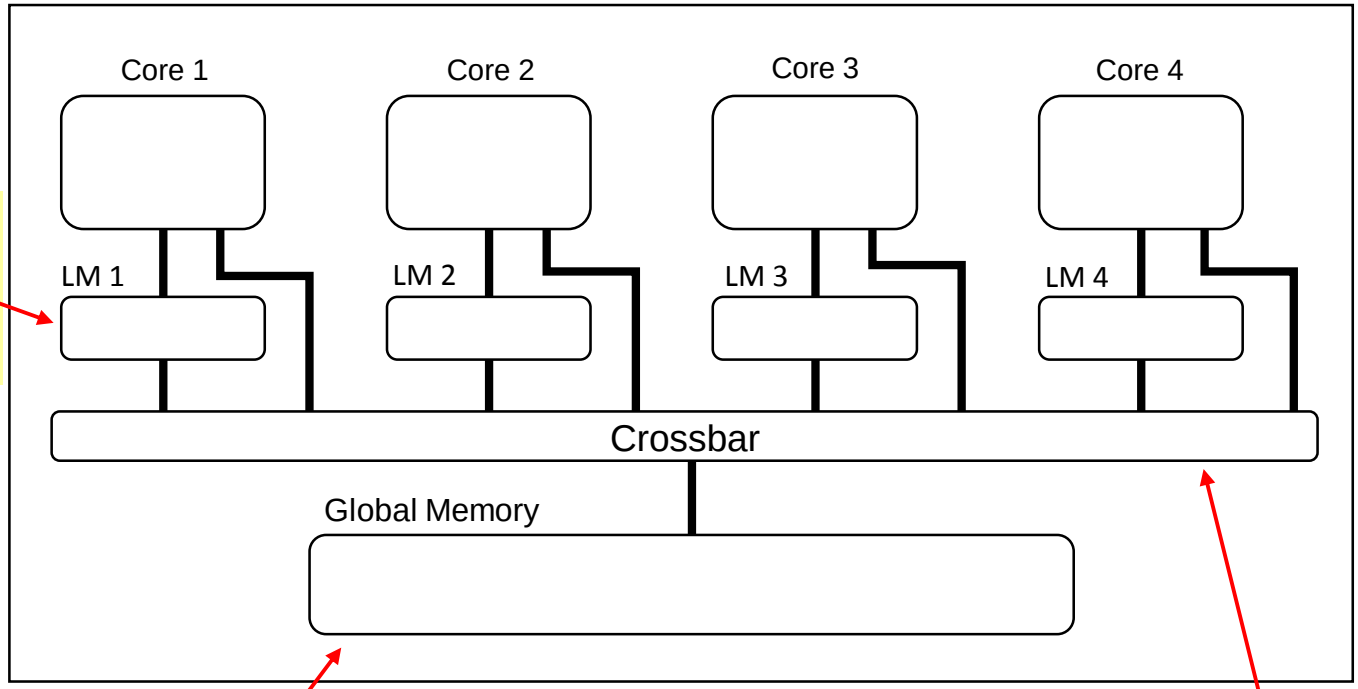
[*] *CISTER, ISEP, Polytechnic Institute of Porto, Portugal*

Sant'Anna
Scuola Universitaria Superiore Pisa

CISTER
Research Center in
Real-Time & Embedded
Computing Systems

This Talk

- **Focus of this talk**: multicore platforms with local memories (scratch-pads)

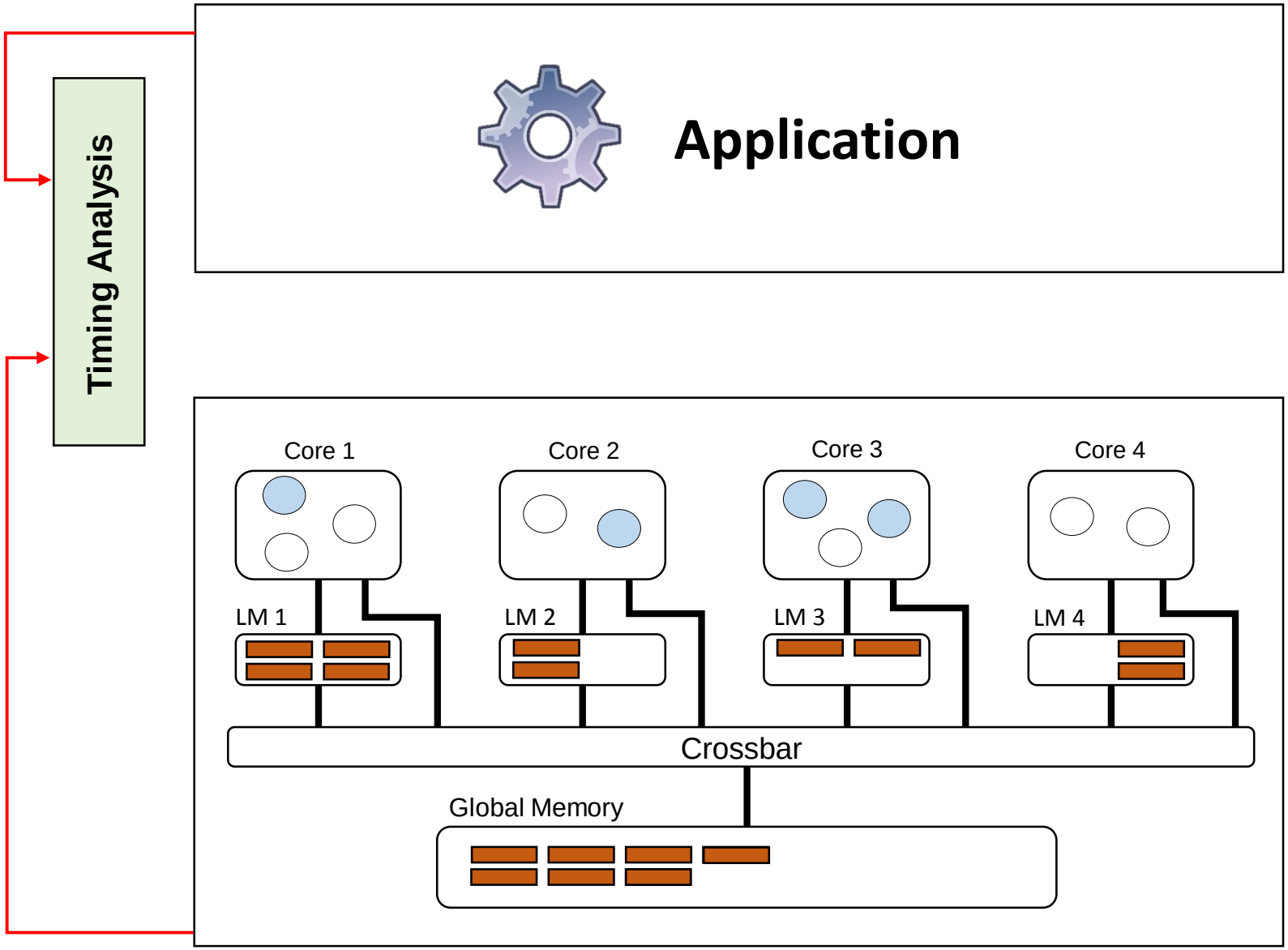


core local memory (*scratchpad*)

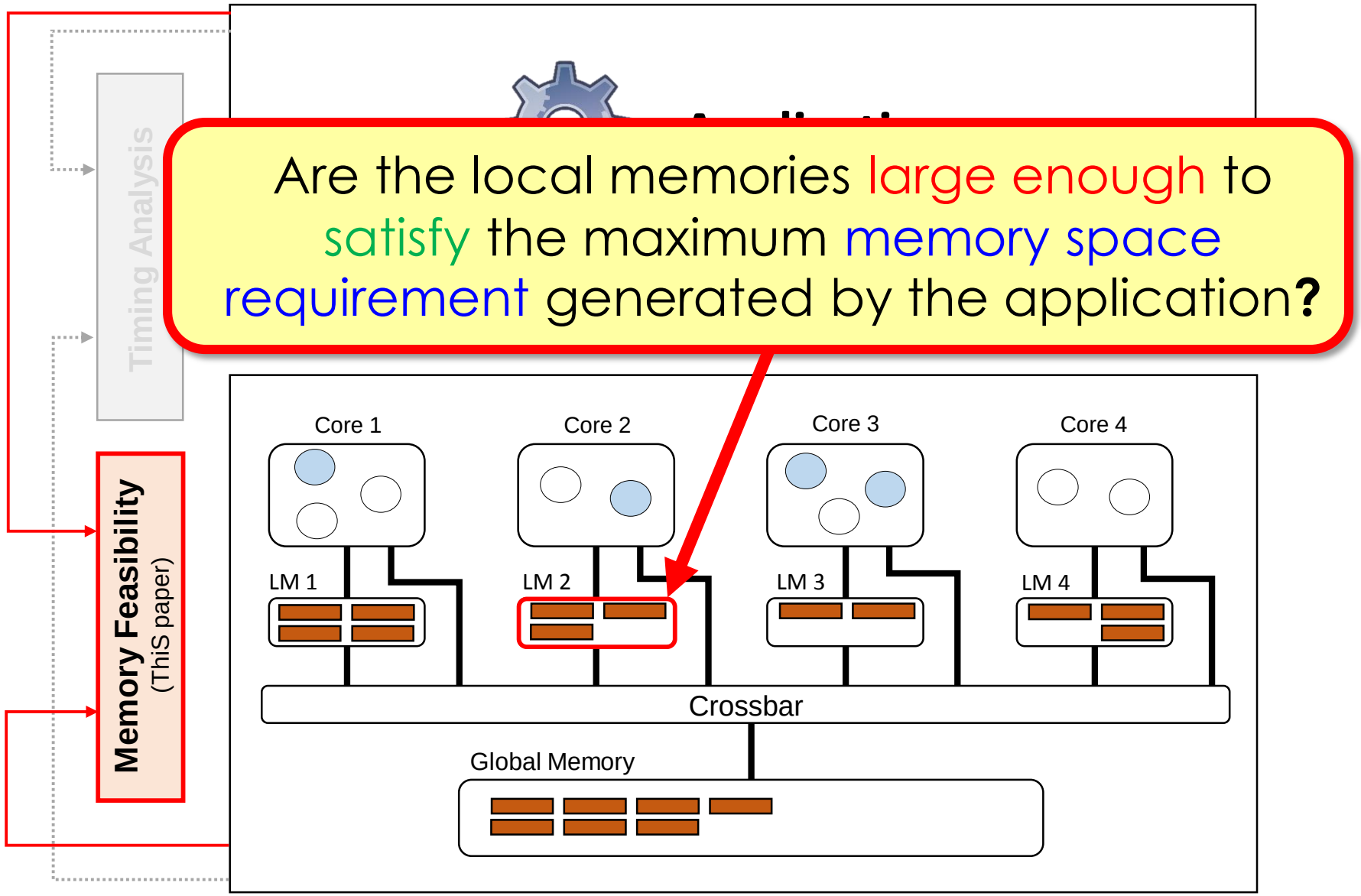
global **shared** memory

provides **point-to-point** communication between each core and memory

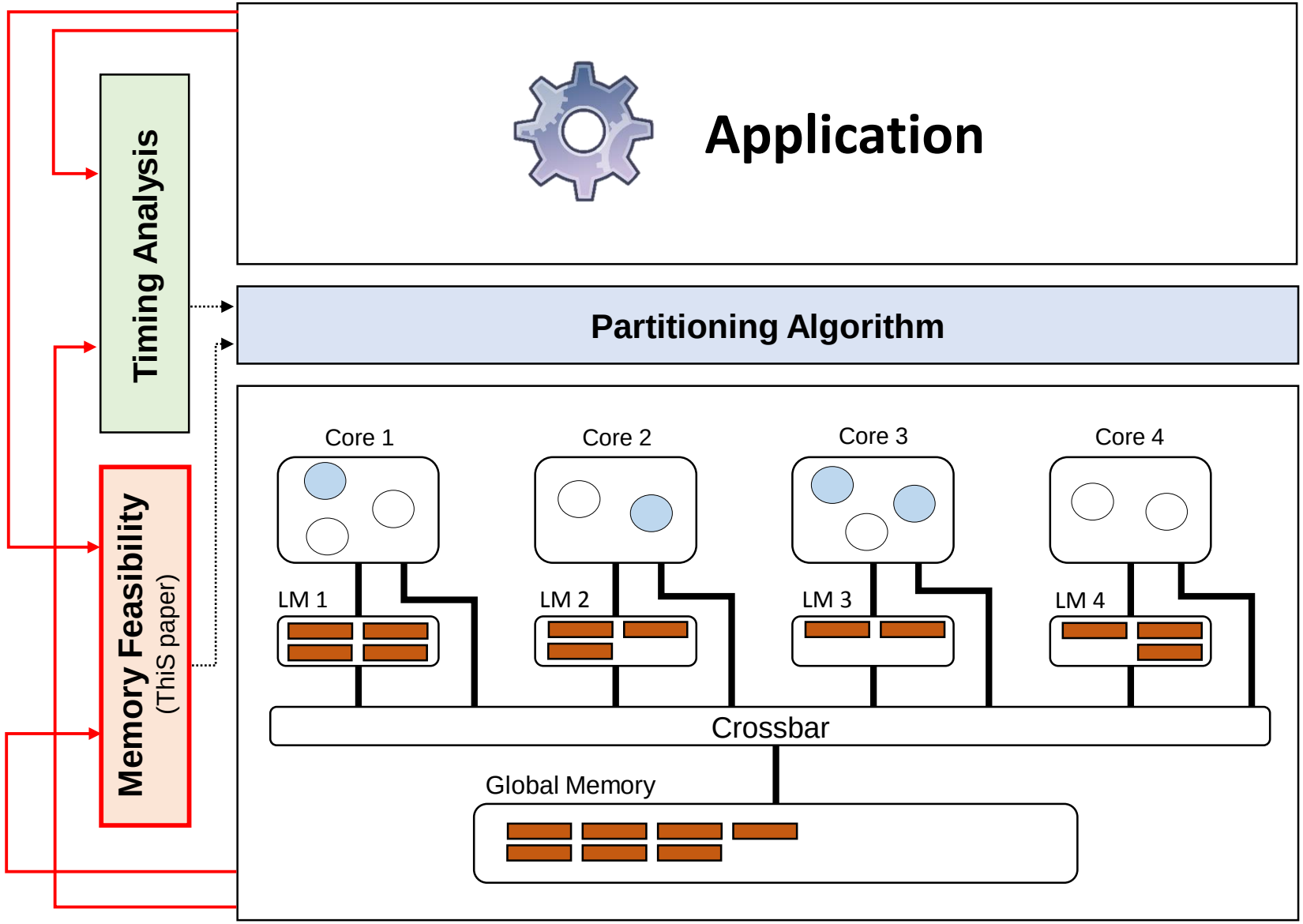
This Talk


Timing Analysis
Application
Core 1
Core 2
Core 3
Core 4
LM 1
LM 2
LM 3
LM 4
Crossbar
Global Memory

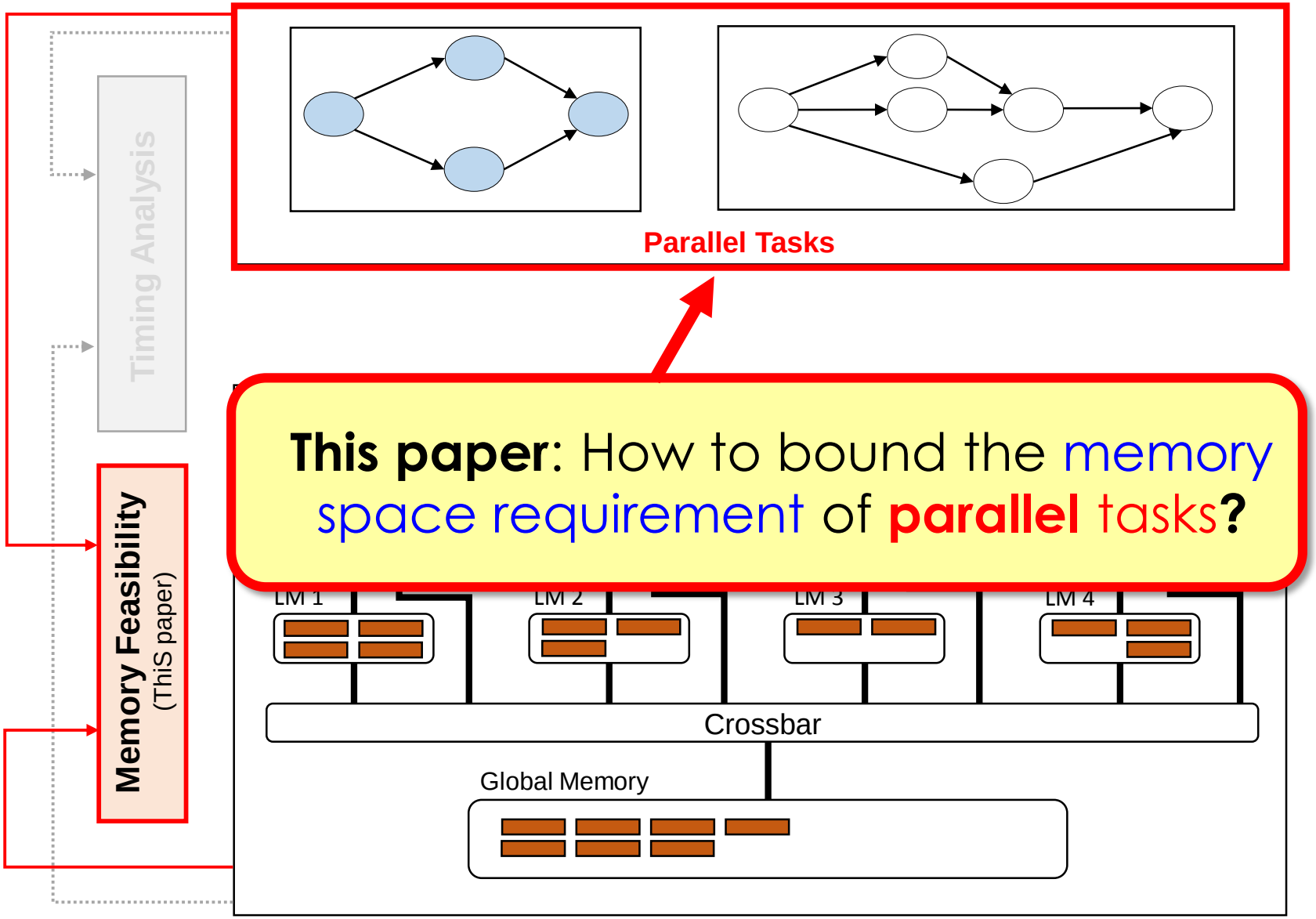
This Talk


Timing Analysis
Memory Feasibility
(ThiS paper)
Are the local memories large enough to satisfy the maximum memory space requirement generated by the application?
Core 1
Core 2
Core 3
Core 4
LM 1
LM 2
LM 3
LM 4
Crossbar
Global Memory

This Talk


Application
Timing Analysis
Partitioning Algorithm
Memory Feasibility
(ThiS paper)
Core 1
Core 2
Core 3
Core 4
LM 1
LM 2
LM 3
LM 4
Crossbar
Global Memory

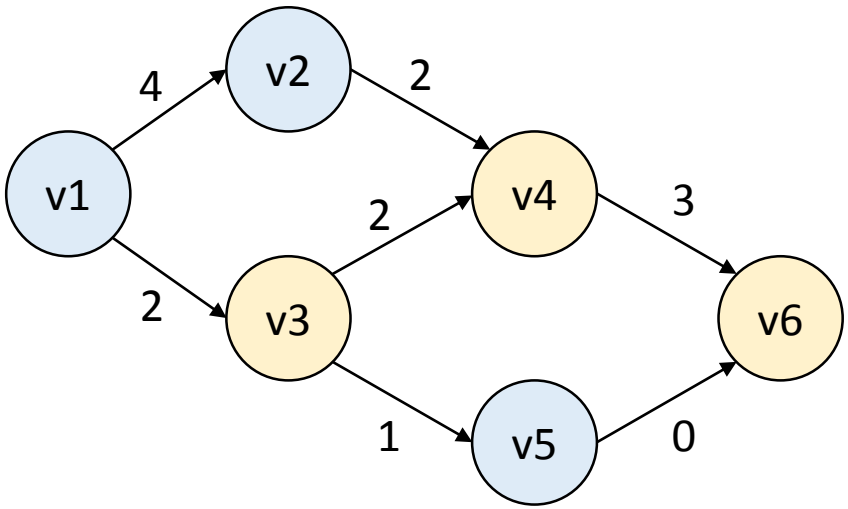
Objective of this Paper

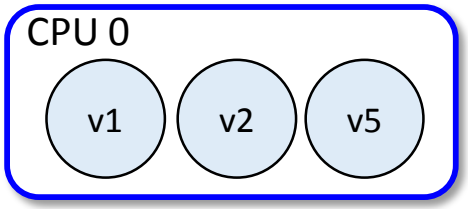


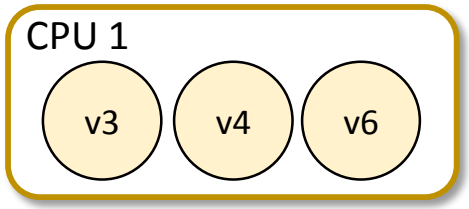
This paper: How to bound the memory space requirement of **parallel** tasks**?**

Task Model

- Directed Acyclic Graph (DAG)
- Each node is statically assigned to a processor and executed in a non-preemptive manner



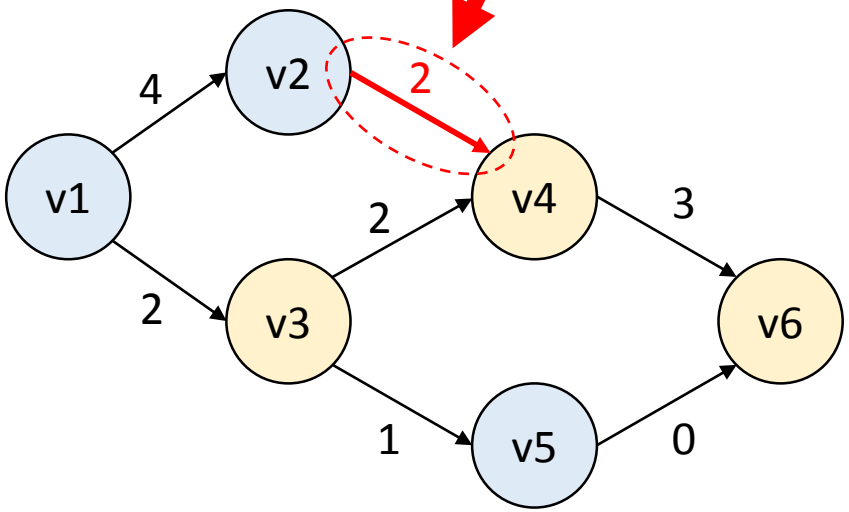


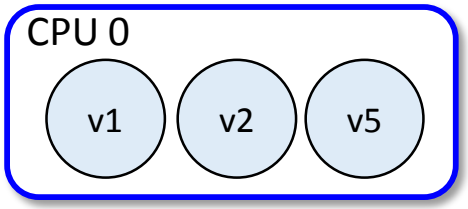


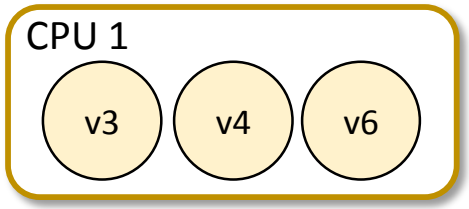
Task Model

- Dir
- Ea in a r

Edges denote precedence constraints and communications. **Weights** denote the amount of data exchanged by nodes (*bytes*, or *blocks*, or *memory pages*, …)



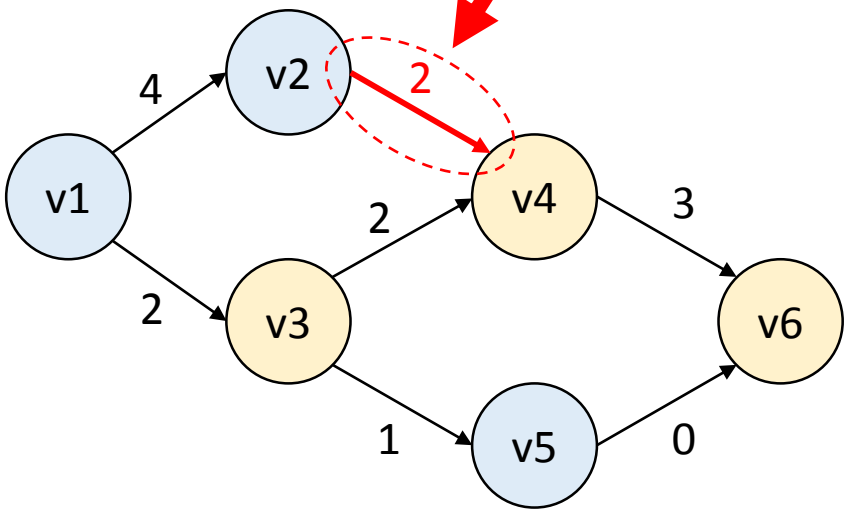


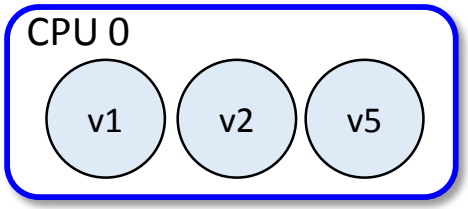


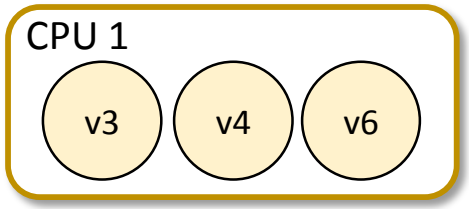
Task Model

- Directed Acycli
- Each node is st xecuted in a non-preempt

Inter-core communication realized with shared buffer in global memory

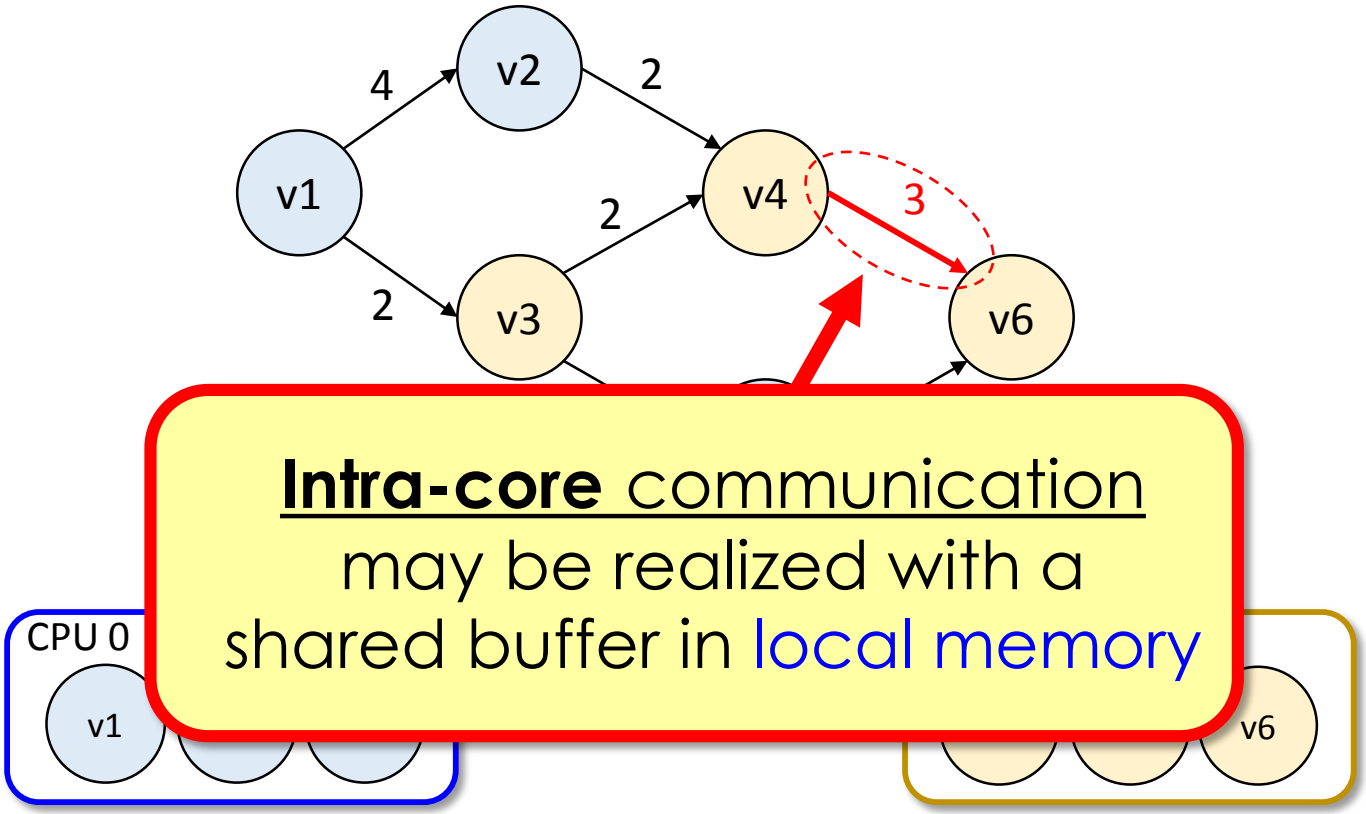






Task Model

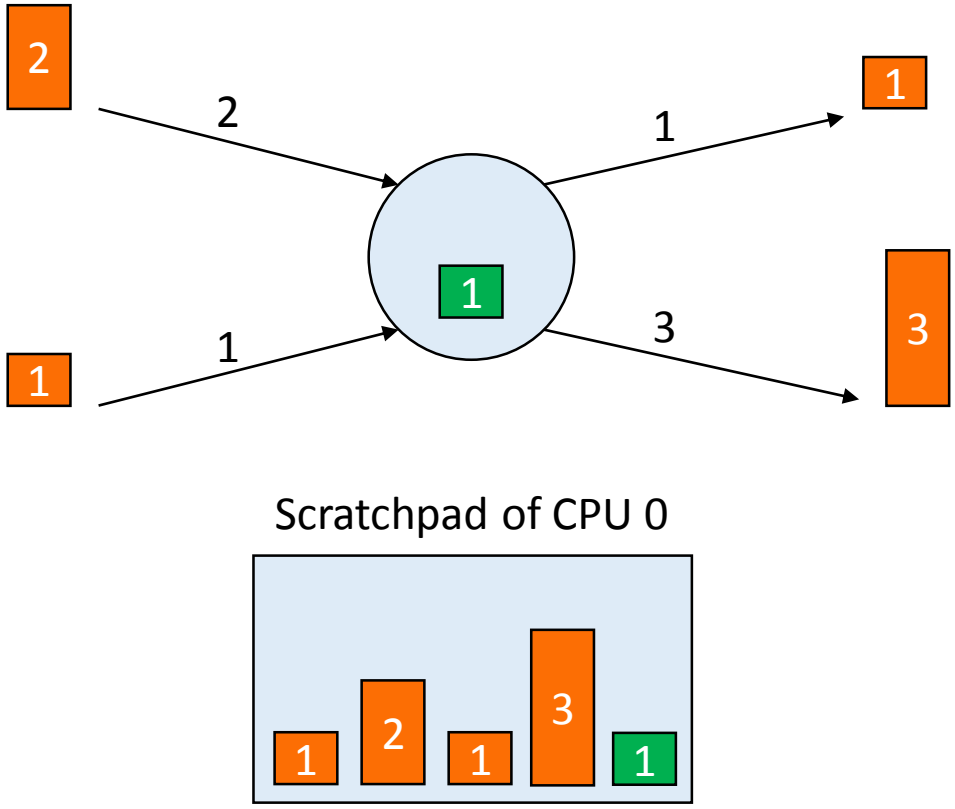
- Directed Acyclic Graph (DAG)
- Each node is statically assigned to a processor and executed in a non-preemptive manner



<u>Intra-core</u> <u>communication</u> may be realized with a shared buffer in local memory

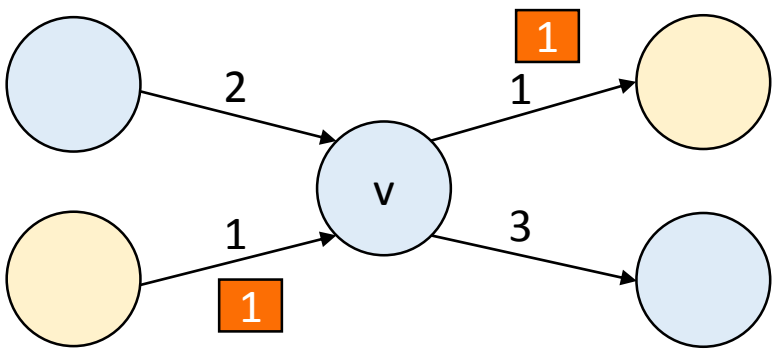
Memory Space Requirement: Nodes

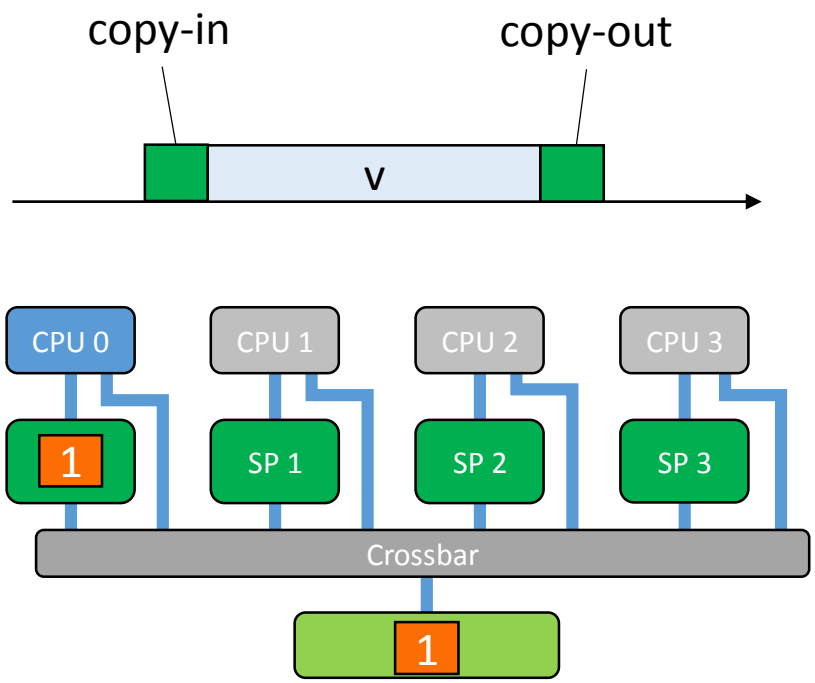
- During their whole execution, nodes must dispose of all input and output buffers allocated in the scratchpad of their core
- Nodes are also characterized by a **local** memory space requirement (e.g., to model the *stack* of functions)



Memory Space Requirement: Nodes

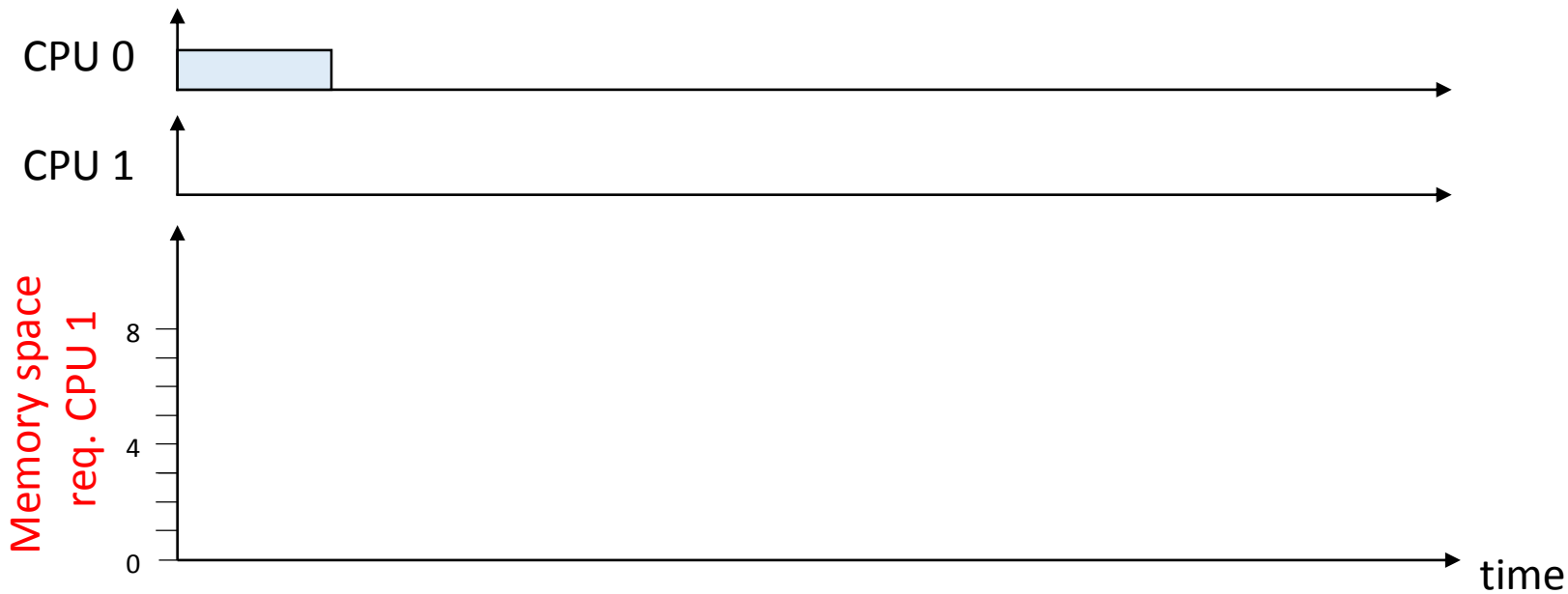
- **Intra-core communications**
  - Input buffers are left allocated by *predecessors*
  - Output buffers are allocated when the node starts executing
- **Inter-core communications**
  - Input buffers: copy-in phase from global memory to scratchpads
  - Output buffers: copy-out phase from scratchpads to global memory



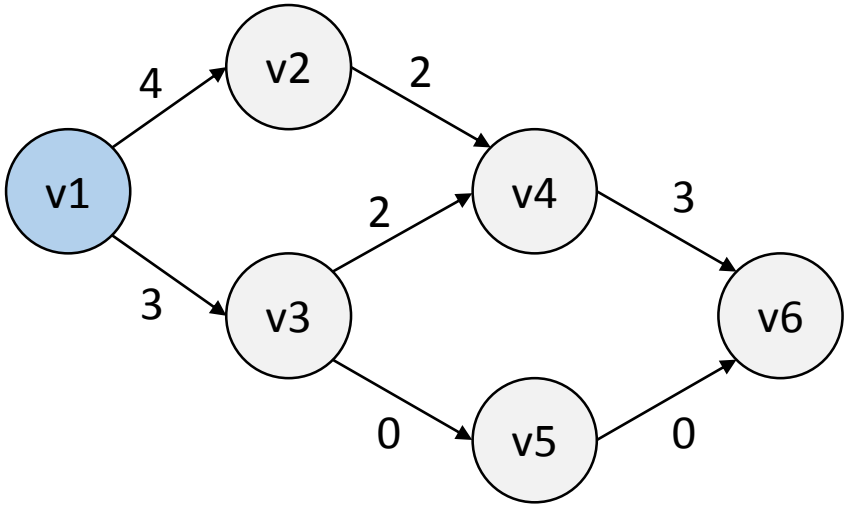


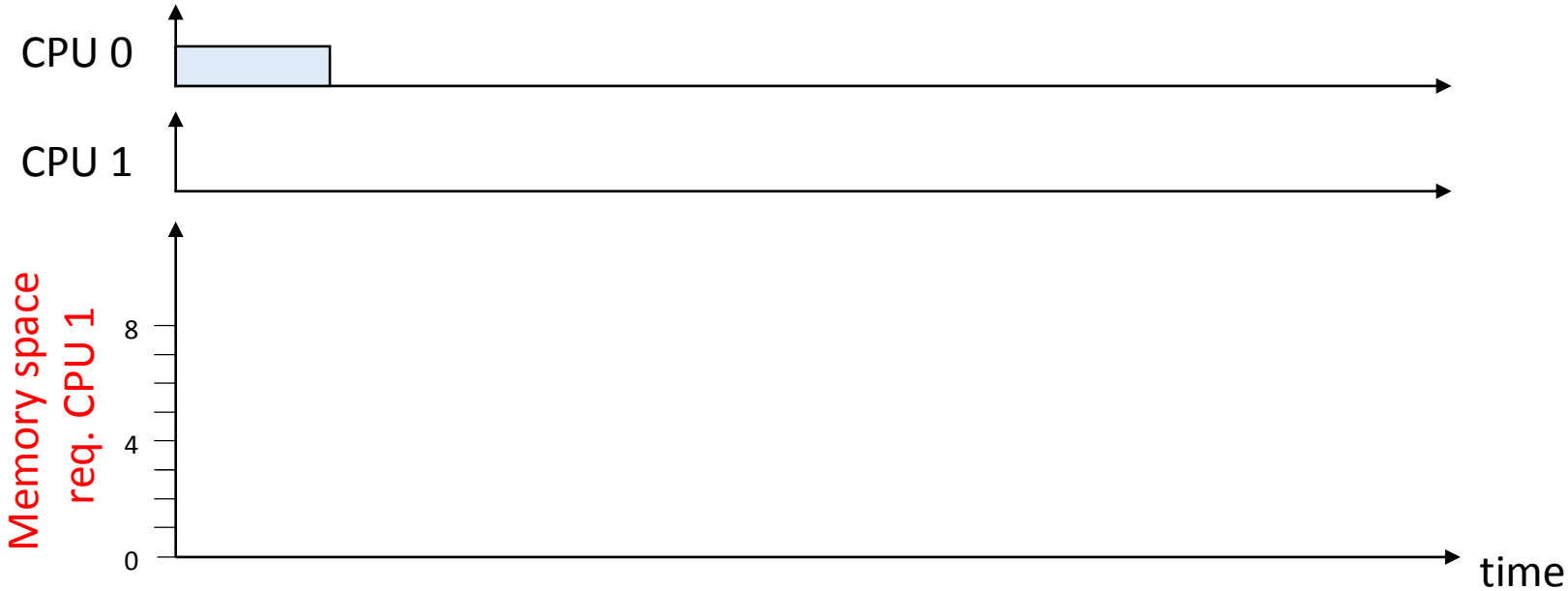
Memory Space Requirement

What's the effect of **precedence constraints** on memory space requirement?

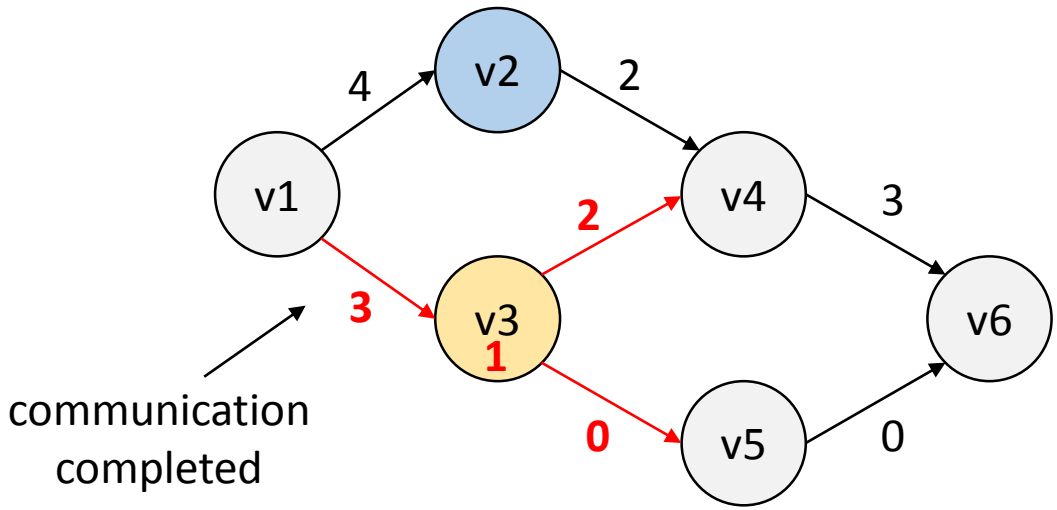


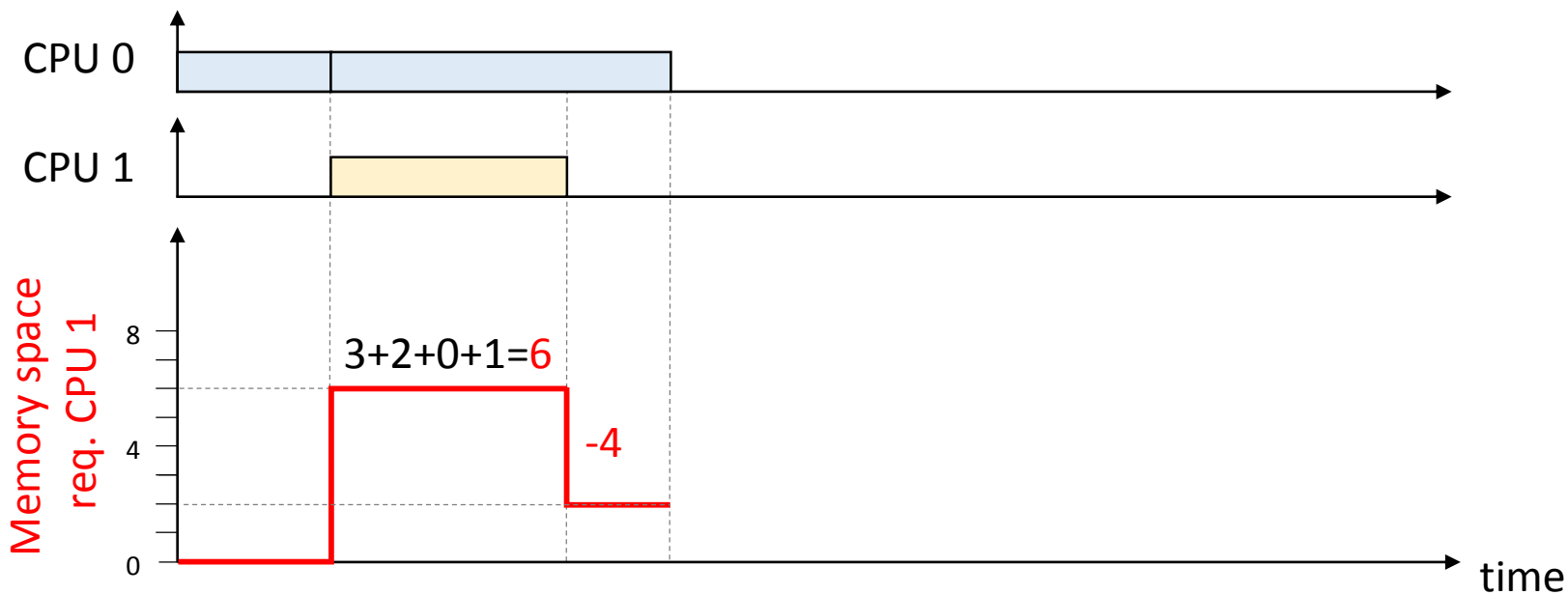
Memory Space Requirement



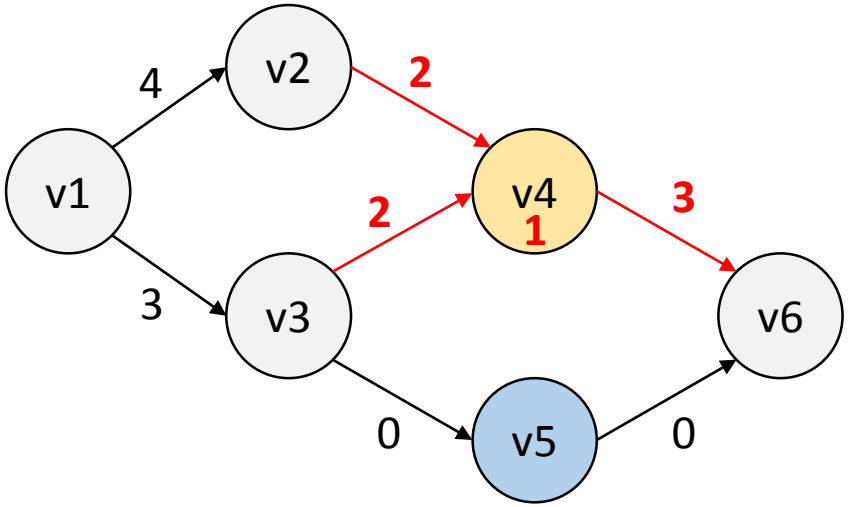


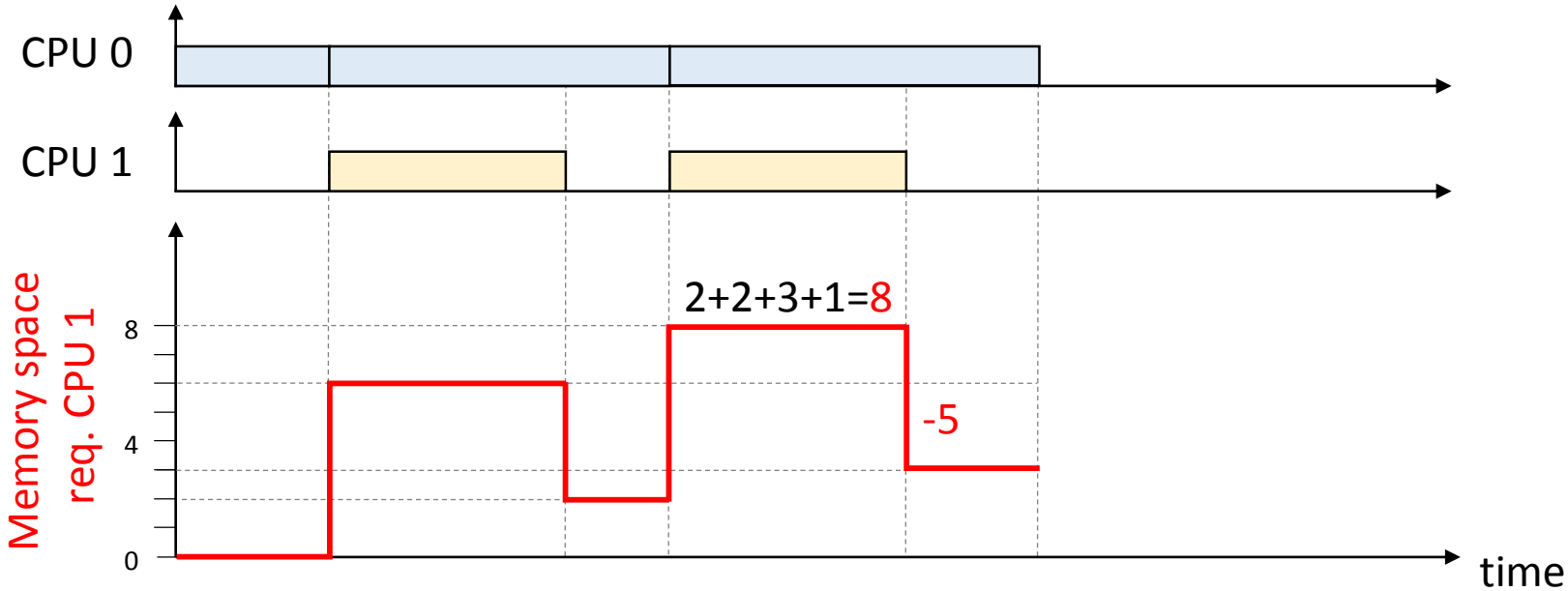
Memory Space Requirement



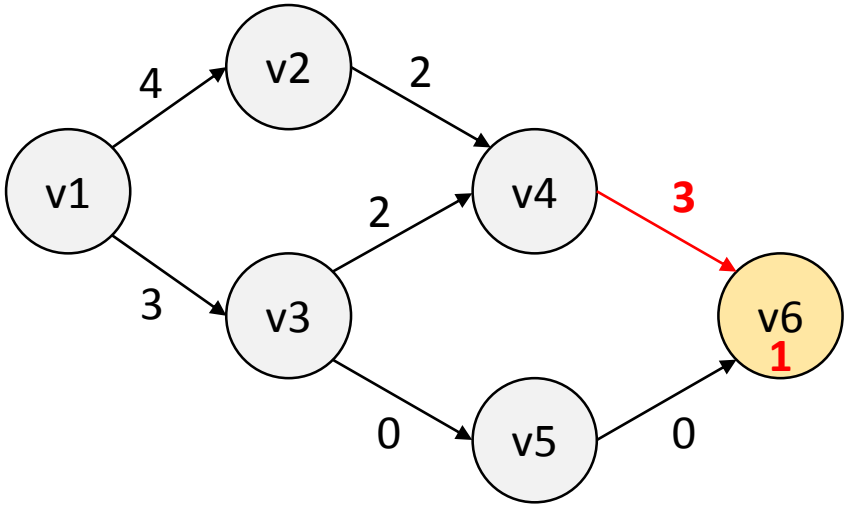


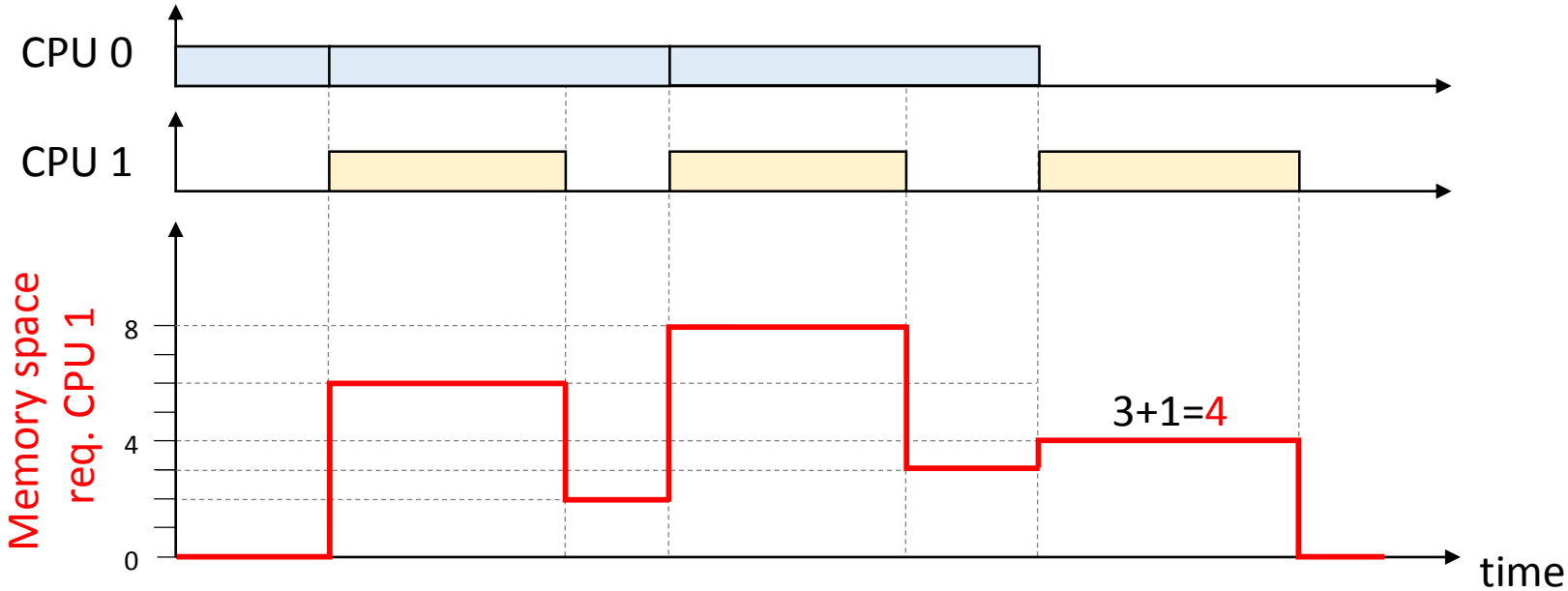
Memory Space Requirement



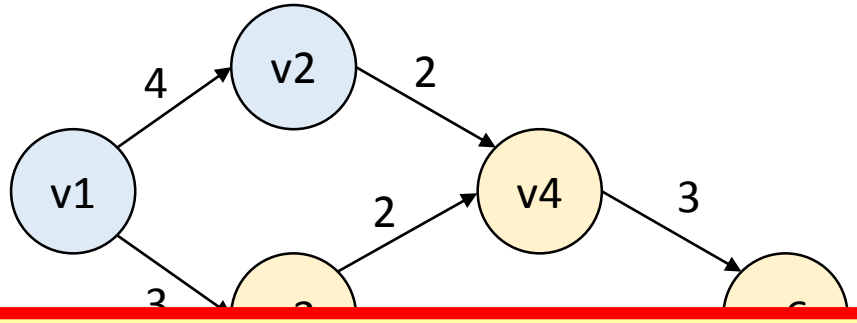


Memory Space Requirement

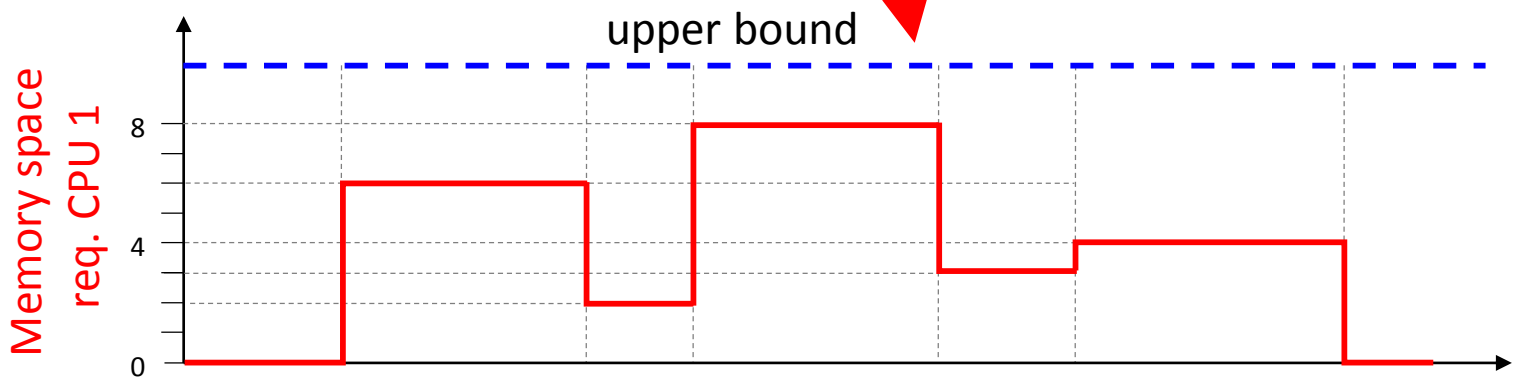




Objective of this Paper: a closer look

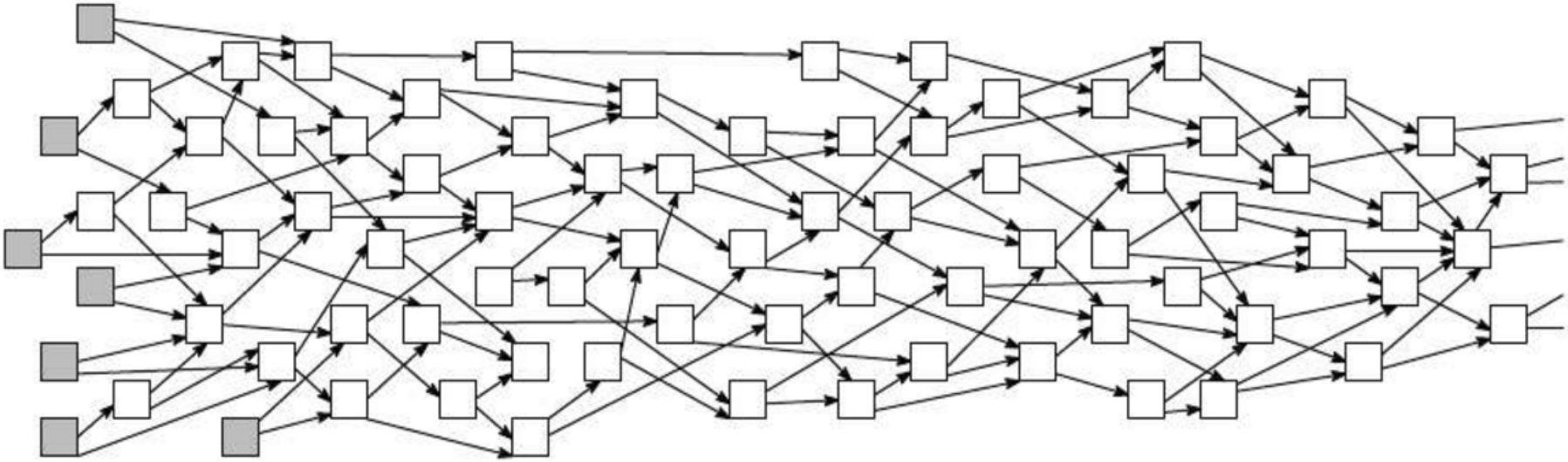


This paper: how to compute an upper-bound on the maximum memory space requirement generated on each core



Objective of this Paper: a closer look

Bounding the maximum memory space requirement in the presence of **complex** precedence constraints is not trivial…



Automotive software (complex inter-runnable dependencies)

Deep neural networks

PROPOSED SOLUTIONS

Proposed Solutions

- This work presents **two** approaches to compute a bound on the memory space requirement generated by a set of parallel tasks

$$f(x) = \ldots$$

Closed-form bound via a top-down decomposition of the memory space requirement

Algorithmic methods based on

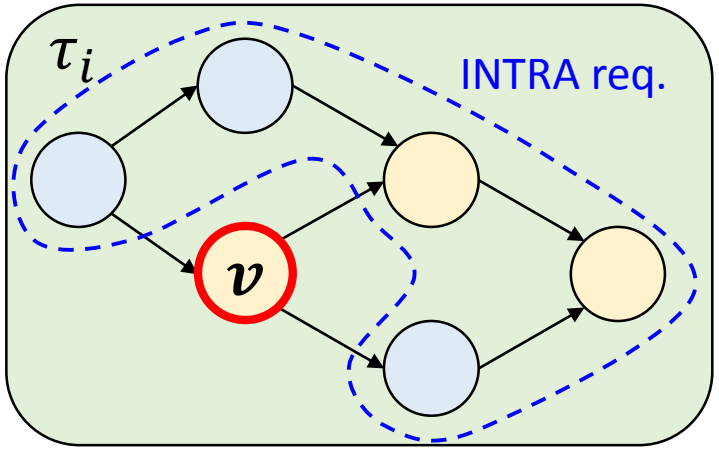
- max-plus algebra for fork-join tasks
  (very efficient)
- mapping to a max-flow cut problem for arbitrary DAG tasks
  (solvable with both MILP or graph transformations)

Closed-form Bound

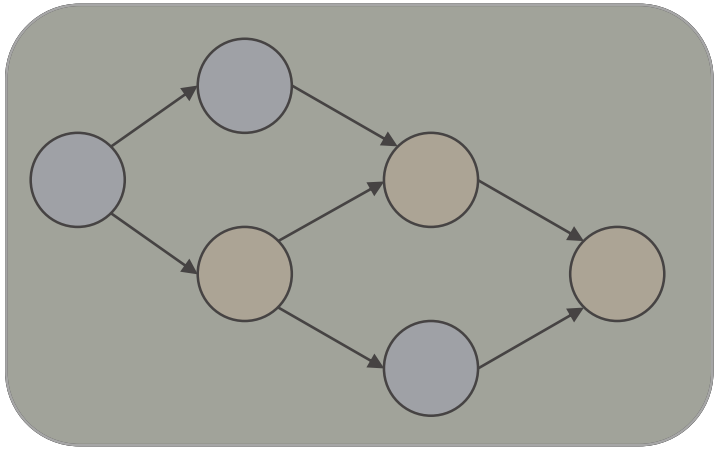
- **Top-down decomposition** of the memory space requirement
  - For each task…
  - Each task may be executing or not…
  - Due to non-preemptive scheduling, when it is executing there is **only one** node started but not completed…

$$\begin{cases} M_i^{\text{EX}} = max_v\{M_{i,v}^{\text{NODE}} + M_{i,v}^{\text{INTRA}}\} \\ M_i^{\text{NEX}} = \ldots \end{cases}$$

Approximations (difficult to bound)

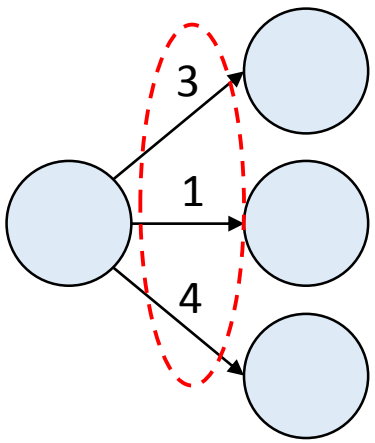


…

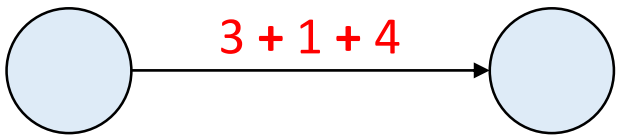


Algorithmic Methods: Max-plus Algebra

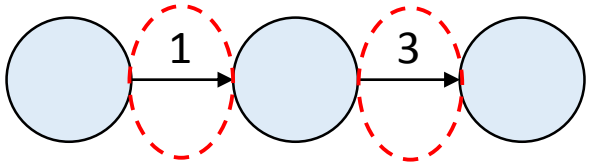
- The memory space requirement of **fork-join** tasks can be computed with a max-plus algebra (for nodes that are <u>not executing</u>)



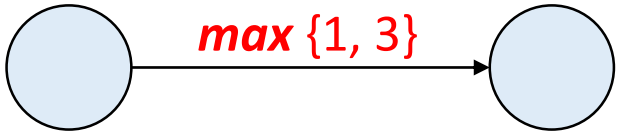
- The three communications can be pending at the same time



mutually exclusive

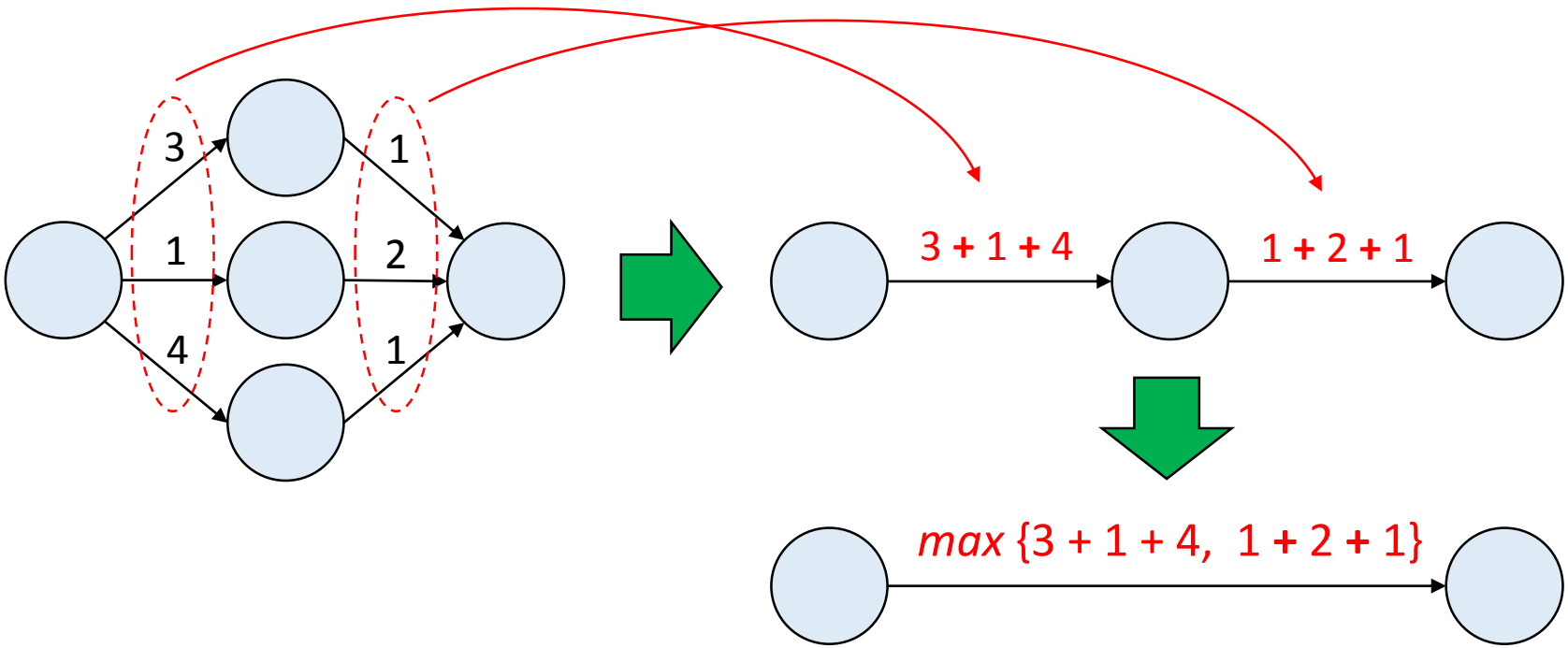


- The two communications <u>cannot</u> be pending at the same time (serialized)



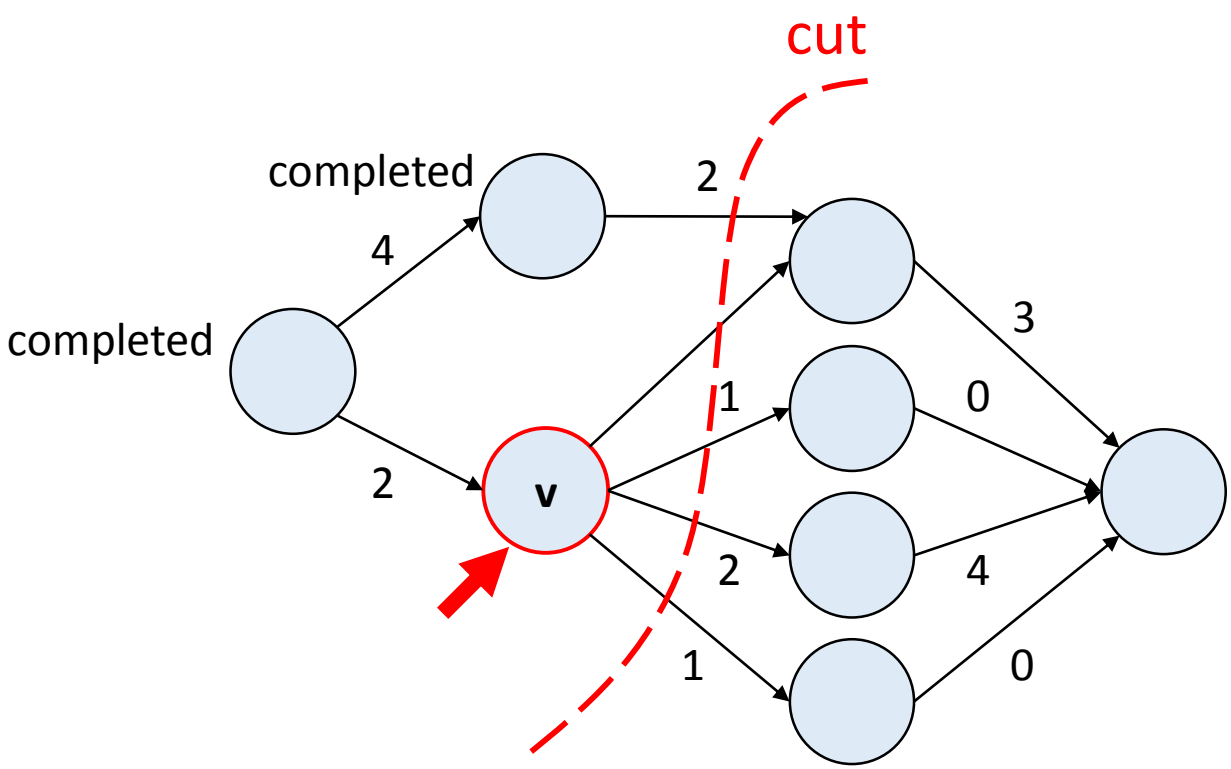
Algorithmic Methods: Max-plus Algebra

- Efficient solution for fork-join tasks
- By repeatedly applying the two rules, it is possible to express the memory requirement with a formula, which, for instance, may be useful to implement a sensitivity analysis



Algorithmic Methods: Max-flow Cut

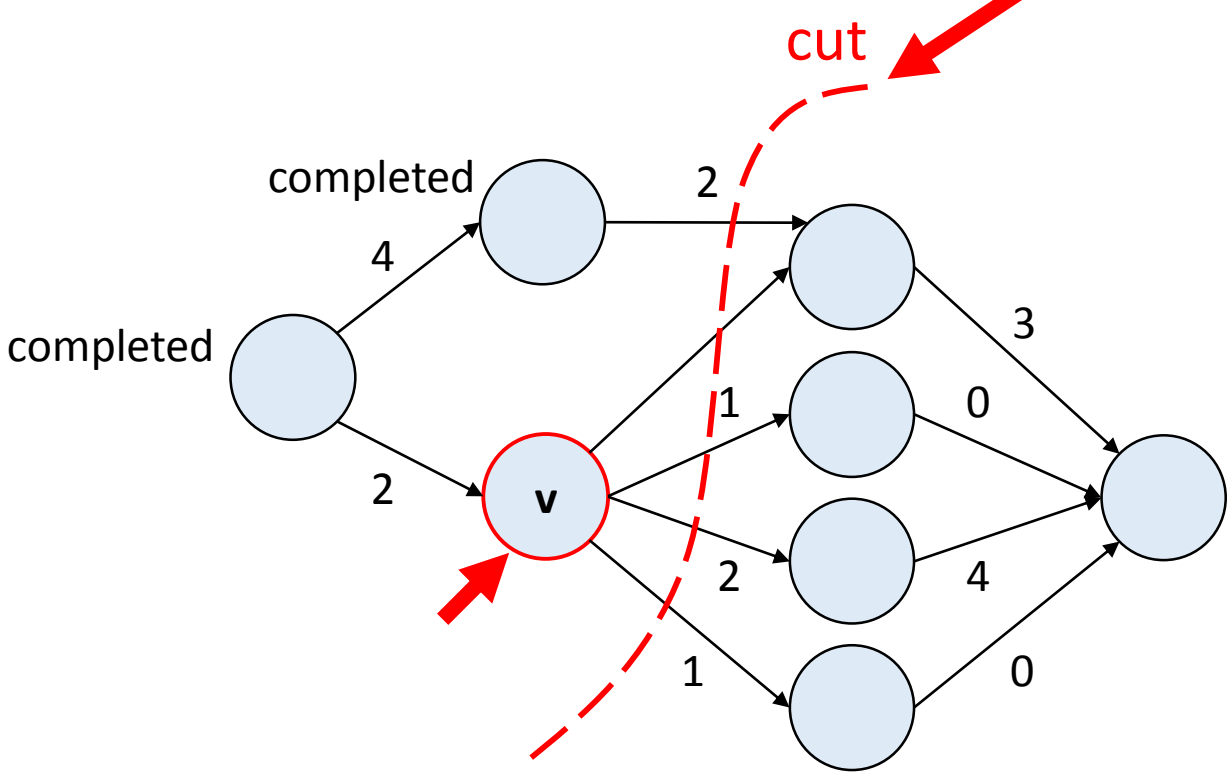
- Applies to arbitrary **DAG** structures
- Consider a node **v**. At any time, due to non-preemptive execution, all the other nodes can either be
  - completed; or
  - not yet started



Algorithmic Methods: Max-flow Cut

- Applies to arbitrary **DAG** structures
- Consider a node **v**. At any time, due to non-preemptive execution, all the other nodes c
  - completed; or
  - not yet started

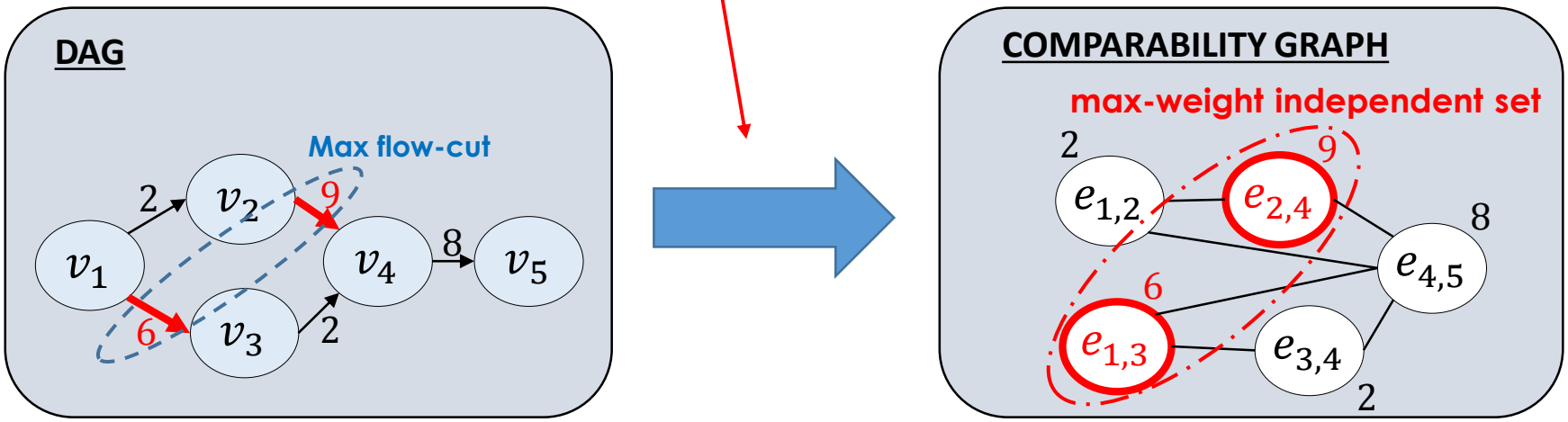
Edges on the **cut** represent communications pending at the same time



Algorithmic Methods: Max-flow Cut

- Finding the maximum memory space requirement is hence <u>equivalent</u> to finding the cut traversing edges such that the sum of their weights is **maximal**
- This max-flow cut problem can be solved either with
  - Mixed-Integer Linear Programming (MILP)
  - Polynomial-time algorithmic approach

- create one node per edge
- connect edges that cannot be on the same path



EVALUATION

Evaluation

- The proposed approaches have been evaluated with both
  - **Synthetic workload** based on randomly-generated DAG tasks
  - **STR2RTS benchmarks** (*digital signal processing* applications)
  - … on 4-core and 8-core platforms

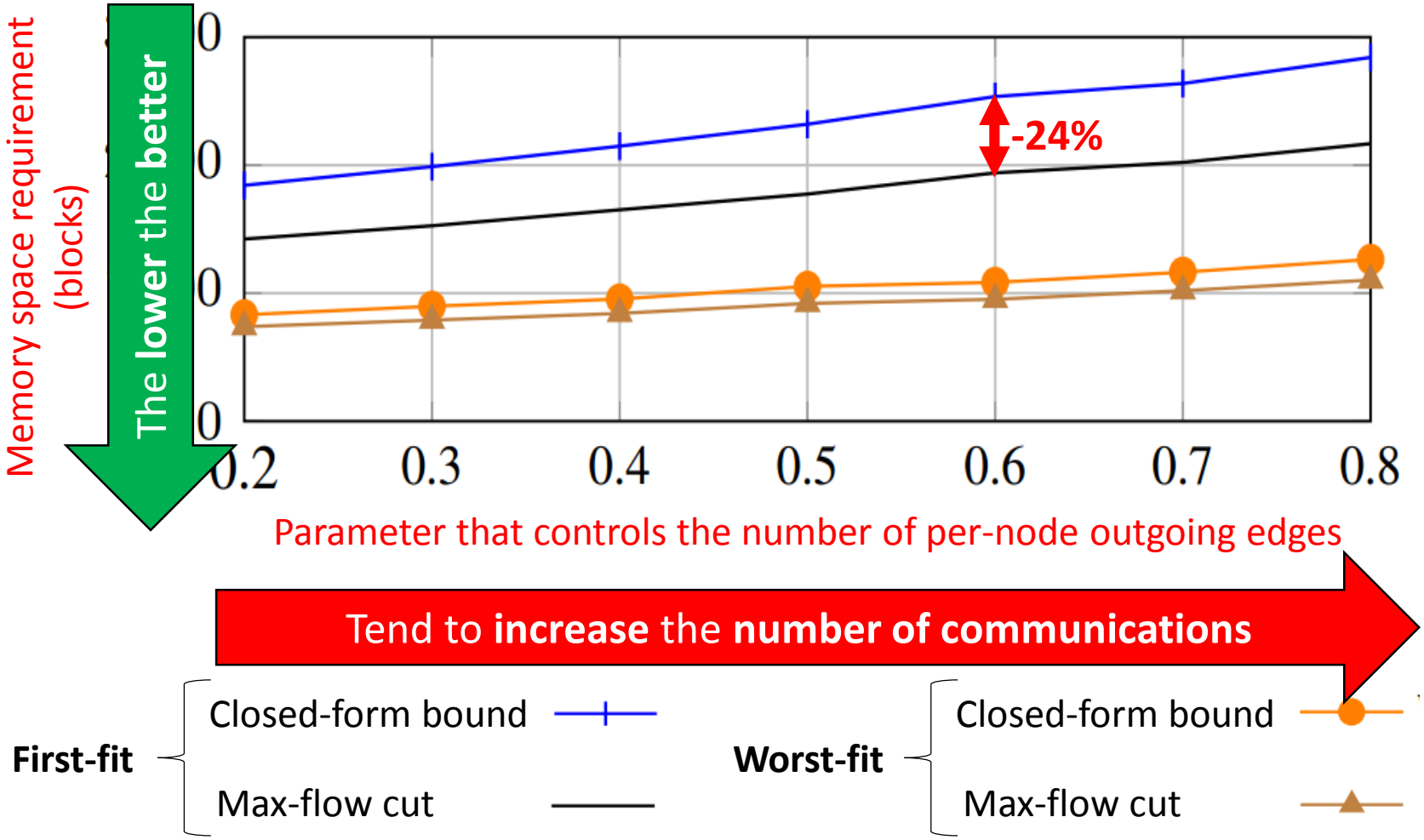
- Different node-to-core partitioning strategies have been tested
  - Worst-fit, First-fit, Best-fit heuristics (e.g., based on the WCETs)
  - Rank-based (kind of topological order)

STR2RTS benchmarks

Id	Name	Description
B1	FFT4	Precise Fast Fourier Transform
B2	FilterBankNew	Multi-rate signal processing
B3	FMRadio	FM radio
B4	AudioBeam	Audio beam-forming
B5	Beamformer	Beam-forming
B6	CFAR	Constant False Alarm Detection
B7	FFT2	Fast Fourier Transform

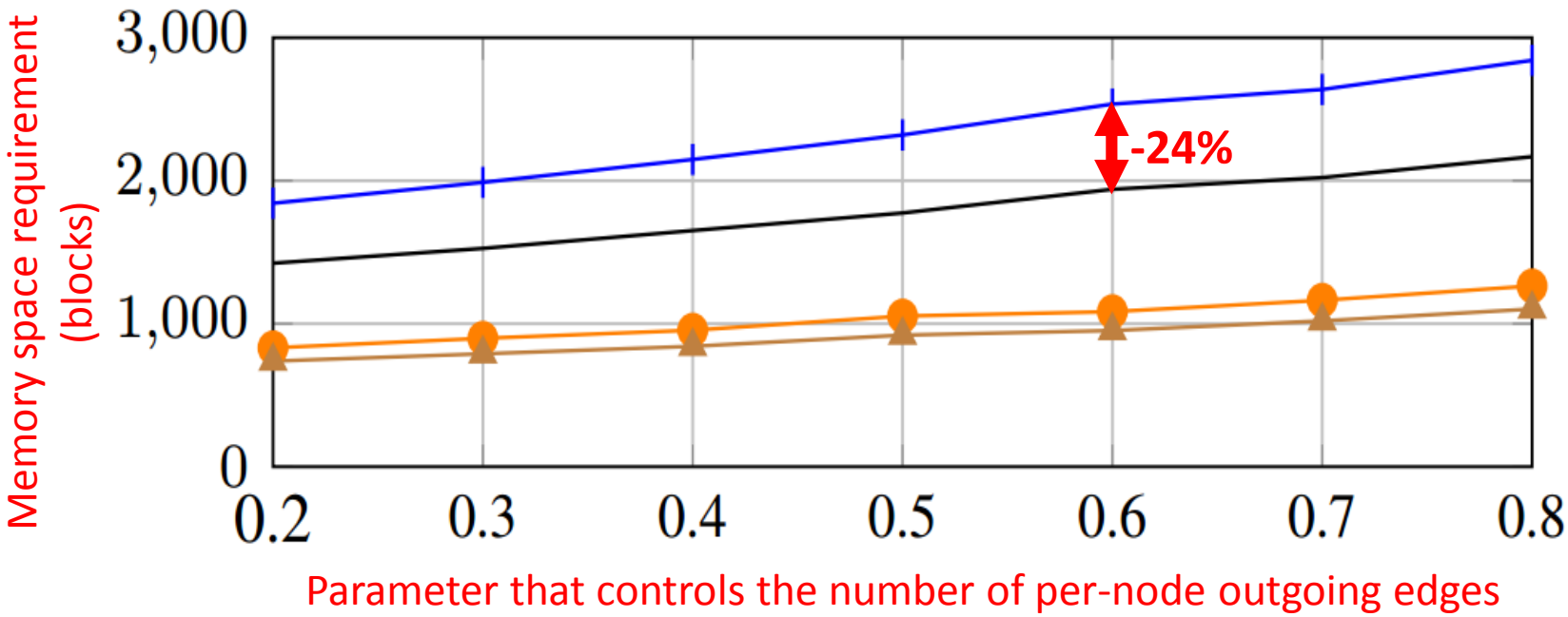
Experimental Results (1)

Representative configuration with **4** processors and **5** DAG tasks
(synthetic workload)



Experimental Results (1)

Representative configuration with **4** processors and **5** DAG tasks
(synthetic workload)

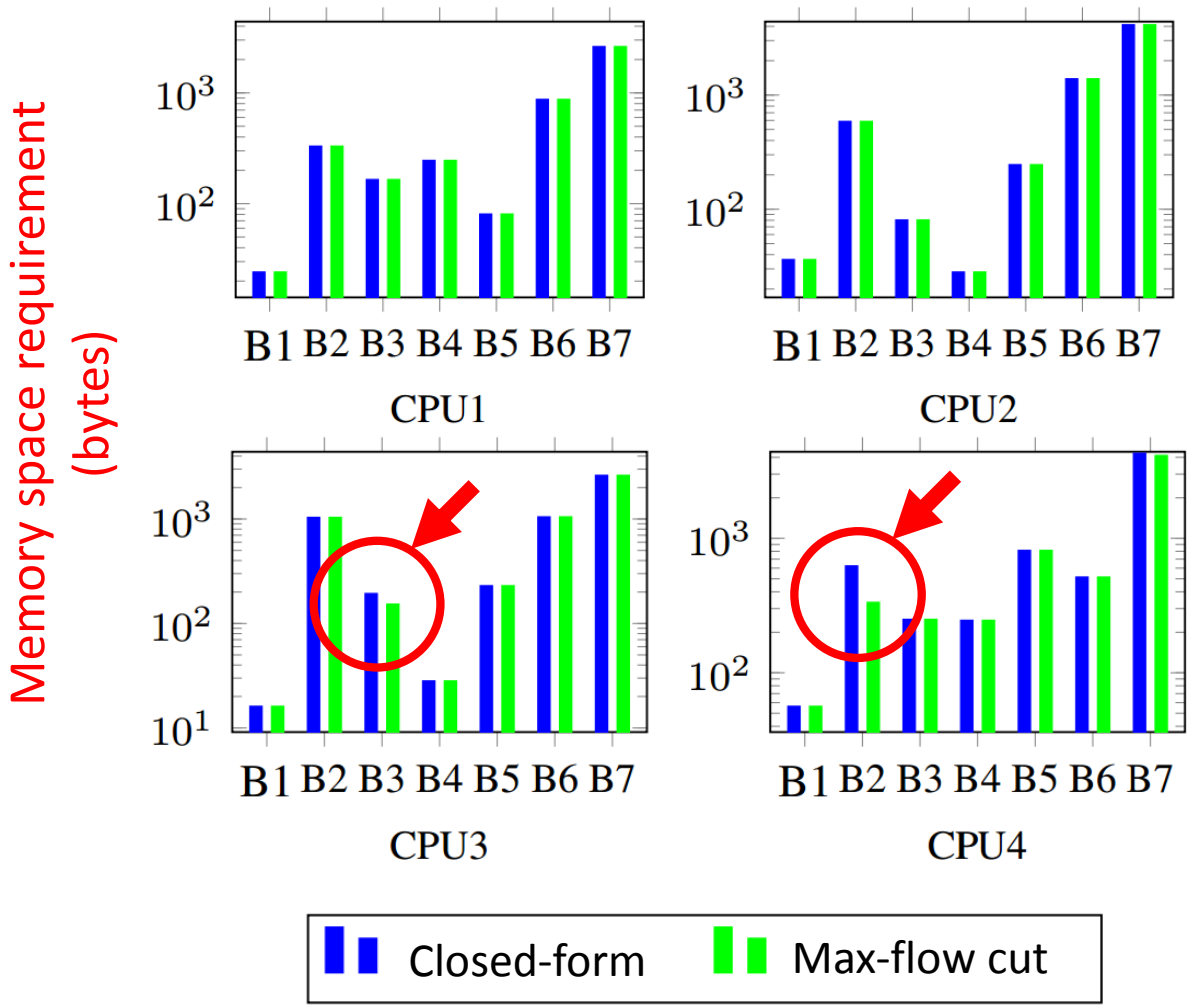


First-fit
- Closed-form bound
- Max-flow cut

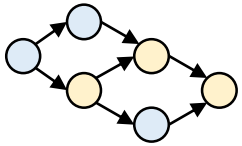
Worst-fit
- Closed-form bound
- Max-flow cut

Experimental Results (2)

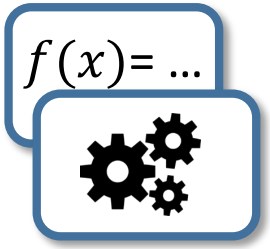
- Seven (B1, …, B7) benchmarks from **STR2RTS**
- Worst-fit partitioning based on WCETs



Conclusions

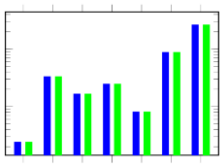


We proposed techniques to **bound** the maximum memory space requirement of **parallel tasks** running upon scratchpad-based multicore platforms



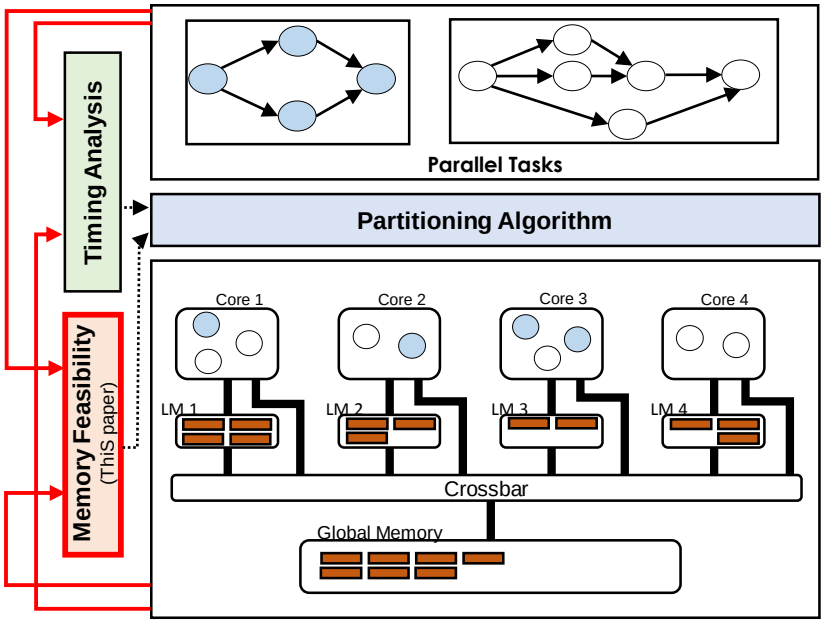
Both **closed-form** and **algorithmic techniques** have been presented

- Top-down decomposition
- Max-plus algebra (applies to fork-join tasks)
- Max-flow cut problem (applies to DAG tasks)

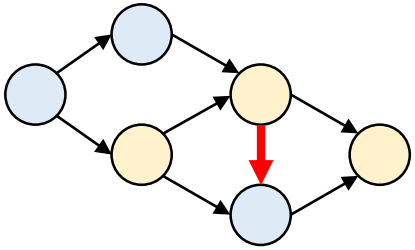


Experimental results showed that closed-form bounds are close to those provided by algorithmic techniques, especially for real-world workload (STR2RTS benchmarks)

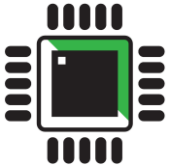
Future Work



- Integrate with **timing analysis**
- ...also accounting for memory contention
- ... to develop a holistic partitioning methodology



Synthetize additional precedence constraints to minimize the memory space requirement while ensuring schedulability



Application to hardware accelerators

Thank you!

Alessandro Biondi
alessandro.biondi@sssup.it