Communication and Shared Memory Efficient Mapping Techniques of Real-Time DAGs upon Clustered Multicore Platforms

Matheus Schuh Claire Maiza Pascal Raymond
Bruno Ferres Joël Goossens
Benoît Dupont de Dinechin

This work

- optimize the mapping of DAG tasks
- onto clustered multicore
- limited memory and communication capacities
- problem and solution

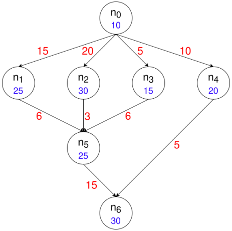
Fixed parameters in an industrial setting

- task model
- execution model
- platform

Optimising the mapping

Context

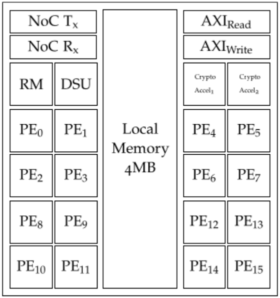
- application as a **DAG of tasks**
- execution model:
  local read / execute / remote write
- platform as **interconnection of clusters** (MPPA3 from Kalray)
  typically programmed using *List-Scheduling* algorithm



DAG to map

Target platform (Kalray MPPA3©)

Kalray MPPA3: varied communication latencies



NoC T_x

NoC R_x

RM DSU

PE_0 PE_1

PE_2 PE_3

PE_8 PE_9

PE_{10} PE_{11}

Local Memory 4MB

AXI_{Read}

AXI_{Write}

Crypto $Accel_1$ Crypto $Accel_2$

PE_4 PE_5

PE_6 PE_7

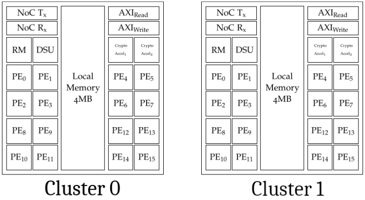
PE_{12} PE_{13}

PE_{14} PE_{15}

Cluster from the MPPA3

MPPA3

Kalray MPPA3: varied communication latencies



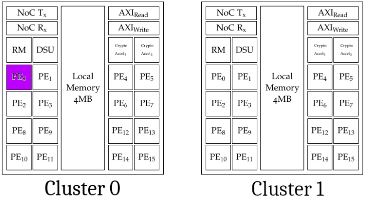
Two clusters from the MPPA3

Three types of communications

- intracore
- intracluster
- intercluster

Source \ Target	CL_0	CL_1
CL_0	23	108
CL_1	108	23

Kalray MPPA3: varied communication latencies



NoC T_x | NoC R_x | RM | DSU | PE_0 | PE_1 | PE_2 | PE_3 | PE_8 | PE_9 | PE_{10} | PE_{11}

Local Memory 4MB

AXI_{Read} | AXI_{Write} | Crypto $Accel_1$ | Crypto $Accel_2$ | PE_4 | PE_5 | PE_6 | PE_7 | PE_{12} | PE_{13} | PE_{14} | PE_{15}

Cluster 0

Cluster 1

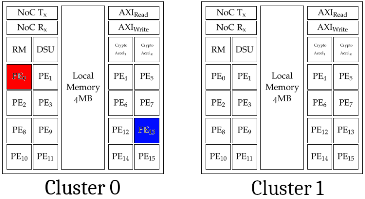
Two clusters from the MPPA3

Three types of communications

- intracore ($CL_0.PE_0 \rightarrow CL_0.PE_0$)
- intracluster
- intercluster

Target / Source	CL_0	CL_1
CL_0	23	108
CL_1	108	23

Kalray MPPA3: varied communication latencies



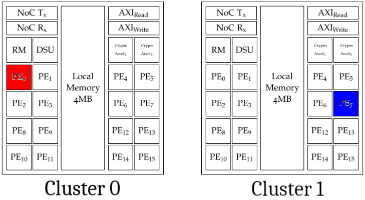
Two clusters from the MPPA3

Three types of communications

- intracore
- intracluster ($CL_0.PE_0 \rightarrow CL_0.PE_{13}$)
- intercluster

Target / Source	CL_0	CL_1
CL_0	23	108
CL_1	108	23

Kalray MPPA3: varied communication latencies



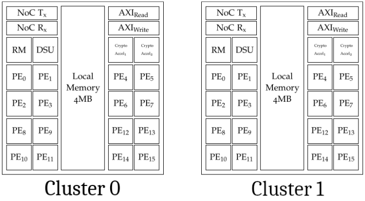
Two clusters from the MPPA3

Three types of communications

- intracore
- intracluster
- intercluster ($CL_0.PE_0 \rightarrow CL_1.PE_7$)

Source \ Target	CL_0	CL_1
CL_0	23	108
CL_1	108	23

Kalray MPPA3: varied communication latencies



Cluster 0

Cluster 1

Two clusters from the MPPA3

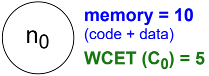
Three types of communications

- intracore
- intracluster
- intercluster

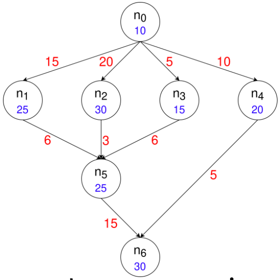
Target Source	CL_0	CL_1
CL_0	23	108
CL_1	108	23

Our goal is to minimize inter-cluster communication by mapping tasks accordingly

The DAG application model



Single node of a DAG



DAG with precedence constraints (→) and both memory and communication costs

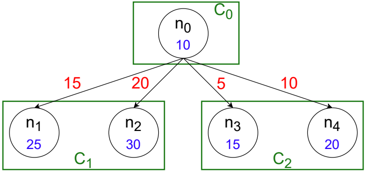
Constraints on tasks

- tasks are **non-preemptive**
- a task can only run on a core if its data & code are stored in the local memory
  ↪ tasks are periodic ⇒ code+data remain in local memory

Model of execution

3-phase model of execution

1. **loading phase:** the data is read from main local memory
2. **execution phase:** the task is executed
3. **storing phase:** results are written in successor tasks' memories

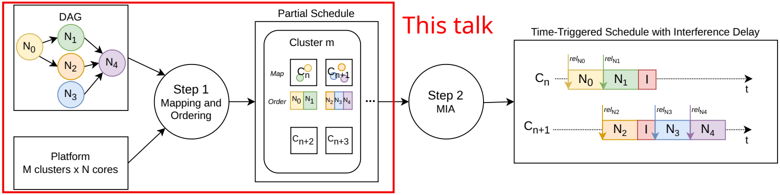


Example (with arbitrary mapping)

1. n_0 read from local memory (c_0)
2. n_0 execute on its core (c_0)
3. n_0 writes to its successors
   - n_1 and n_2 on c_1
   - n_3 and n_4 on c_2

↪ the cost of the storing phase depends on the mapping

Mapping & ordering problem



Deployment flow of DAGs to the MPPA platform

MIA: *Multicore Interference Analysis* [1]

- input: **DAG** + **mapping** on cores
- output **time-triggered scheduling** with **interference delays**

[1] `https://www-verimag.imag.fr/multi-core-interference-analysis.html`

Two-phase solution

Phase 1

Determine the mapping of tasks to cores

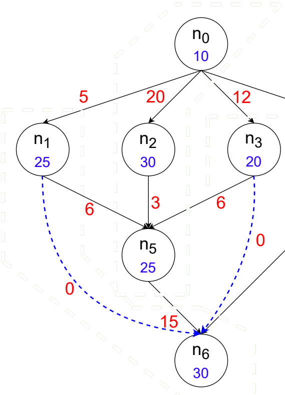
1. try to minimize **inter-core communication**
2. constraint: local memory limits

Phase 2

Determine the mapping of cores to clusters

1. try to minimize **inter-cluster communication**
2. constraint: limited number of cores per cluster

Phase 1: Mapping tasks to cores



constraint: 80 units of memory per core

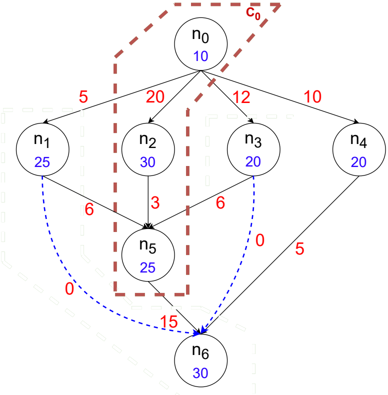
Communication aware mapping

Main idea is to map “greatest” successors “close” to their predecessors,

↪ map sequences to the same core

↪ exploit potential parallelism as much as possible

Phase 1: Mapping tasks to cores



constraint: 80 units of memory per core

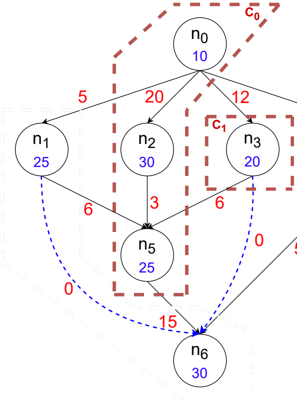
Communication aware mapping

Main idea is to map "greatest" successors "close" to their predecessors,

↪ map sequences to the same core

↪ exploit potential parallelism as much as possible

Phase 1: Mapping tasks to cores



constraint: 80 units of memory per core

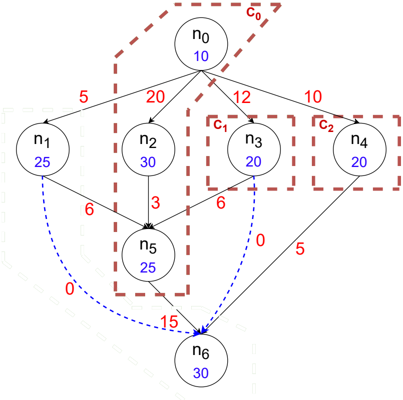
Communication aware mapping

Main idea is to map “greatest” successors “close” to their predecessors,

↪ map sequences to the same core

↪ exploit potential parallelism as much as possible

Phase 1: Mapping tasks to cores



constraint: 80 units of memory per core

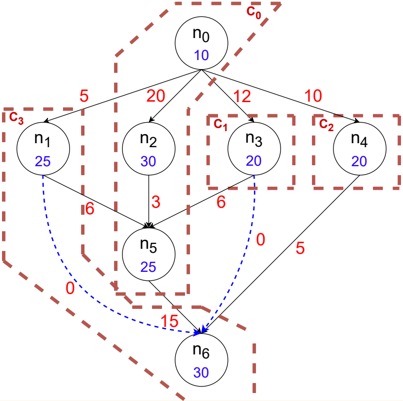
Communication aware mapping

Main idea is to map “greatest” successors “close” to their predecessors,

↪ map sequences to the same core

↪ exploit potential parallelism as much as possible

Phase 1: Mapping tasks to cores



constraint: 80 units of memory per core

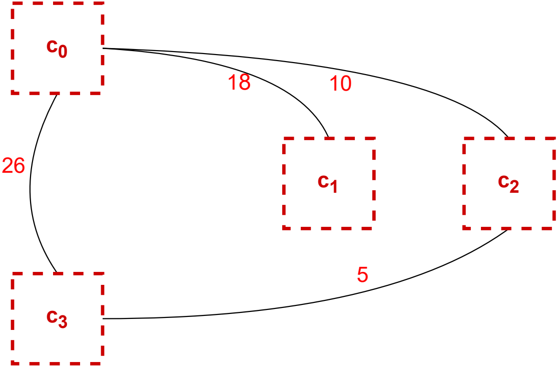
Communication aware mapping

Main idea is to map “greatest” successors “close” to their predecessors,

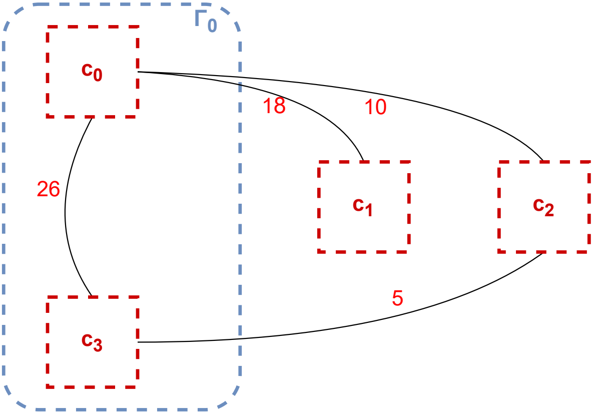
↪ map sequences to the same core

↪ exploit potential parallelism as much as possible

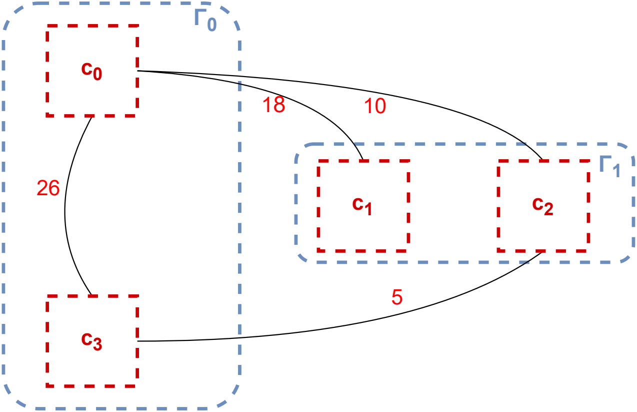
Phase 2a: Mapping cores to clusters



Phase 2a: Mapping cores to clusters



Phase 2a: Mapping cores to clusters



Phase 2b: Mapping virtual clusters to physical clusters

- map these virtual clusters to physical clusters, considering the actual inter-cluster communication latencies
- we perform an exhaustive search to find the optimal mapping.

Key point: **Handling latency heterogeneity on the Kalray MPPA3**

Minimize high cost communications between clusters

Synthetic benchmark & experimental methodology

3240 DAGs with various characteristics

- small *(10 tasks)* to large *(280 tasks)*
- parallelism level *(mostly sequential to mostly parallel)*
- inter-task communications *(light to heavy)*

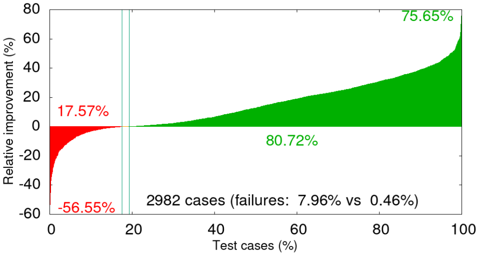
Experimental methodology

- Compare with the SoA list-scheduling
- Criterion: global Worst Case Response Time (**WCRT**)
  ↪ computed by **MIA**
- Results: relative improvement (in %)
  ↪ + failures (not enough memory and/or cores)

Results against a state of the art algorithm

Alternative Algo. (adapted to integrate **memory constraints**)

- *List Scheduling-Based Mapping Algorithm* (**LSA**)
  - ↪ using *Highest Level First with Estimated Time* (**HLFET**) heuristic



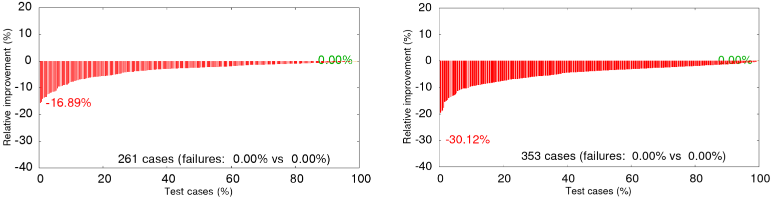
Comparing our approach (Proposed Algorithm) to LSA in term of WCRT

Comparison to a lower bound

Lower-bound mapping

Cumulative WCET of critical path ≡ a lower-bound to WCRT

- not realistic (especially in multicluster)
  ↪ interference not integrated
- but still a lower bound — can be used to compare PA *vs.* LSA



Comparing PA (left) and LSA (right) to lower bound, on a single cluster

Key takeaways

Given fixed parameters in an industrial setting

- task model: **DAGs with precedence/communication costs**
- execution model: **Local read/execute/remote write**
- platform: **Kalray MPPA3 (clusters, memory banks, heterogeneous laxities)**

Our contribution

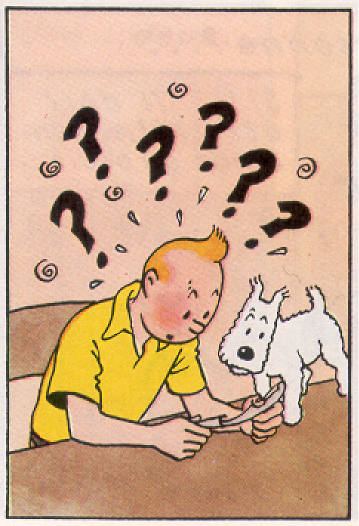
- **communication-aware mapping**: Minimize inter-cluster/core overhead
- **memory-constrained**: limited memory capacities
- **reduce the WCRT by up to 75%** vs. classic list-scheduling

Key takeaways (cont.)

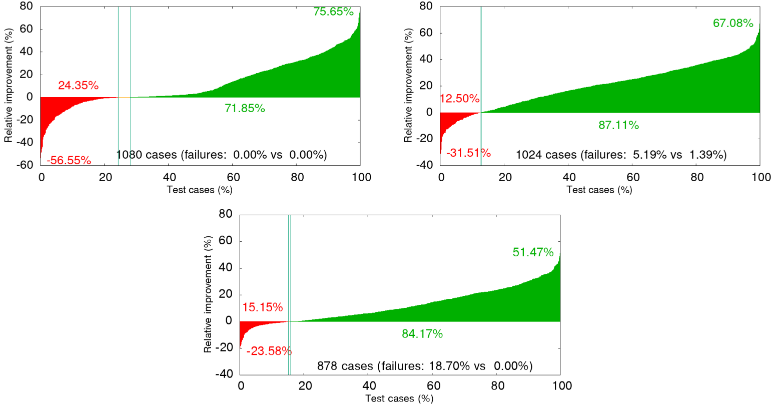
Why it matters for industry

- No platform/model changes needed → **seamless integration**
- Predictable timing → **hard real-time systems**
- Better resource use → **cost-effective scaling**

Questions



More results



LSA vs proposed algorithm: small, medium and big DAGs