Allowing Shared Libraries while Supporting Hardware Isolation in Multicore Real-Time Systems

Namhoon Kim, Micaiah Chisholm, Nathan Otterness,
James H. Anderson, and F. Donelson Smith

April 19, 2017

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Motivation

- When certifying the real-time correctness of a m-core system, the capacity of the additional m-1 cores can easily be negated.

- We call this the "one-out-of-m" problem.

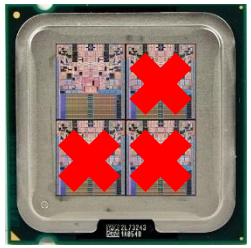


Image source: http://www.northropgrumman.com/Photos/pgM_UC-10028_026.jpg

Motivation

- The root of the problem is that shared hardware (caches, buses, and memory banks) resources are not predictably managed.

- See the FAA (Federal Aviation Administration) position paper “CAST 32A” for an extensive discussion of multicore-related certification difficulties.

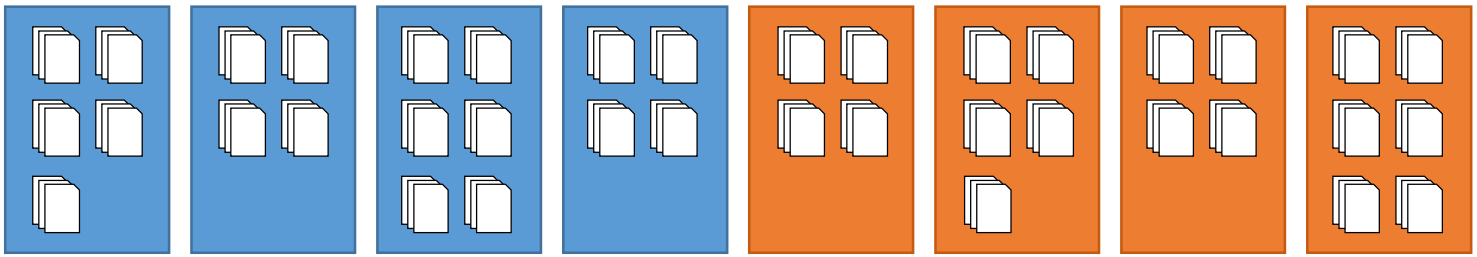
Background

- Two orthogonal approach have been investigated previously.
  - Mixed-criticality (MC) analysis techniques.
  - Hardware-management techniques.
    - Cache partitioning, DRAM bank partitioning.
- Tasks are not allowed to share anything across partitions.

Problems

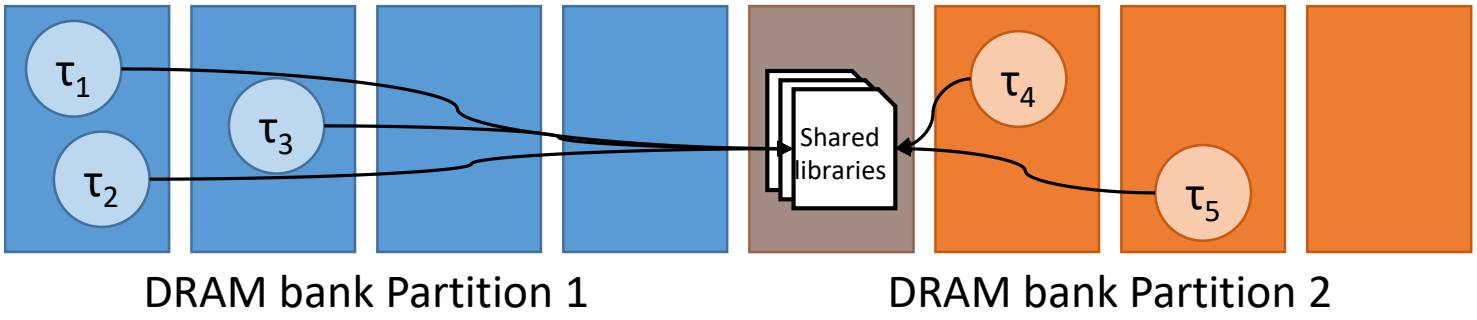
- How do we achieve hardware isolation for a task that shares libraries?
  - *Static linking* can solve this problem.

All libraries are replicated on a *per-task* basis to eliminate any sharing.



Memory Capacity

- Fully replicating libraries can degrade schedulablity *when memory is considered as a constrained resource*.
- Dynamic linking can save memory space, but *breaks* task isolation across DRAM partitions.



Challenge

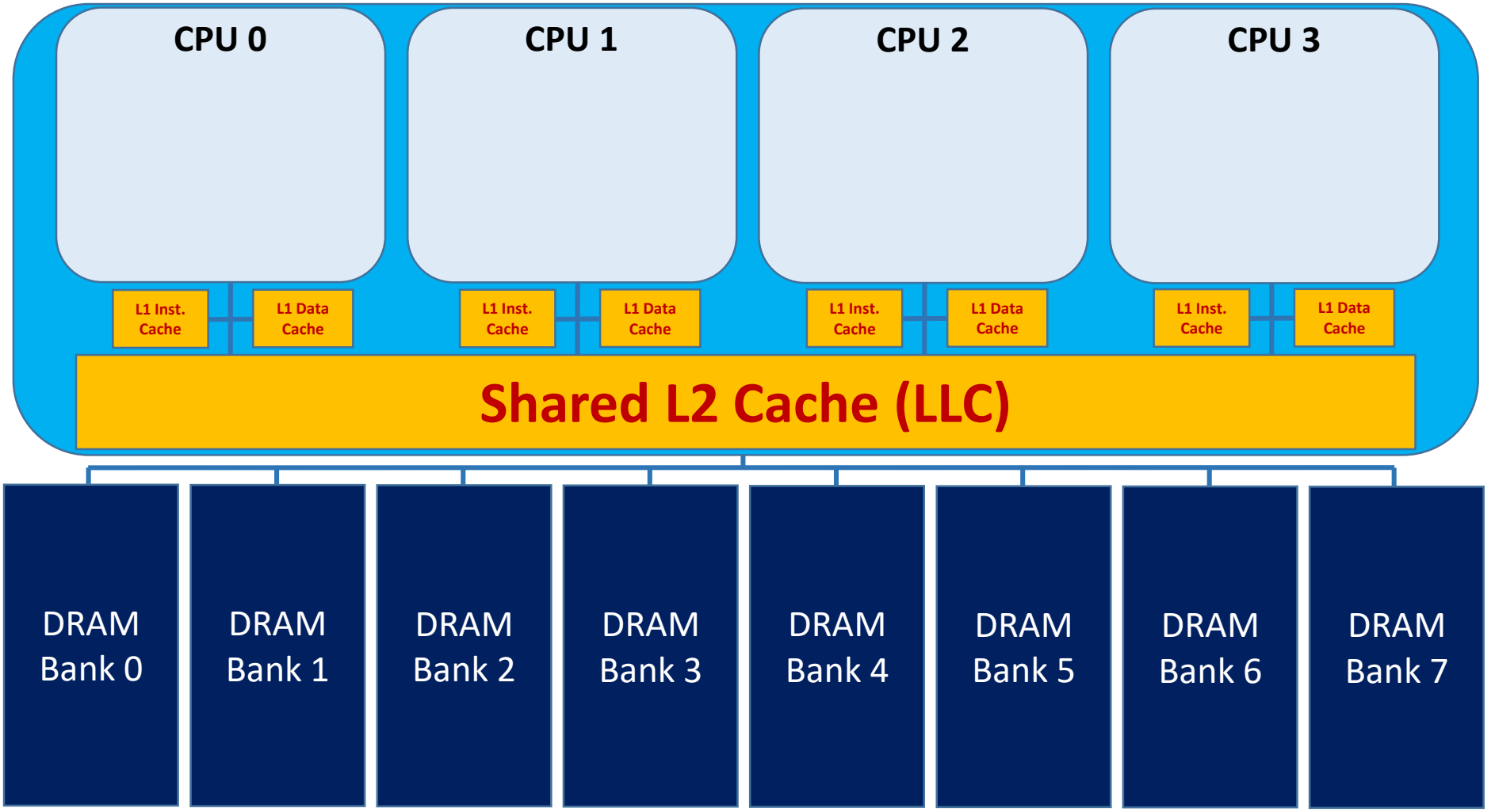
- Static linking can degrade schedulability and dynamic linking breaks isolation.

- **Can we improve schedulability by using shared libraries?**
- **How can we allow shared libraries while ensuring hardware isolation?**

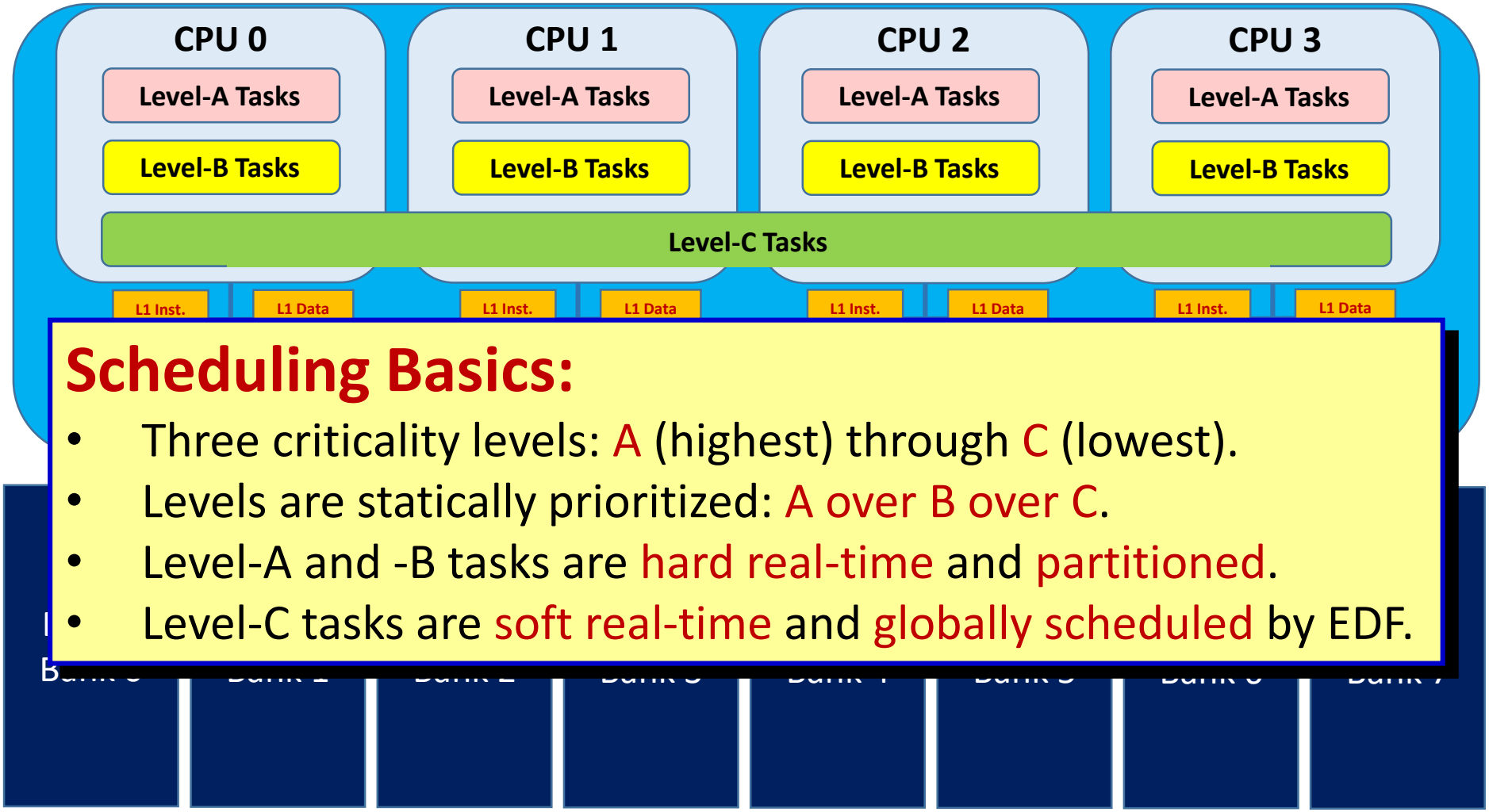
MC^2 (Mixed-Criticality on Multicore)

- The “one-out-of-m” problem can be effectively addressed by combining mixed-criticality provisioning and hardware management in MC^2 (Mixed-Criticality on Multicore) [RTAS16].
- We managed the last-level cache (LLC) and DRAM banks.

MC^2 (Mixed-Criticality on Multicore)



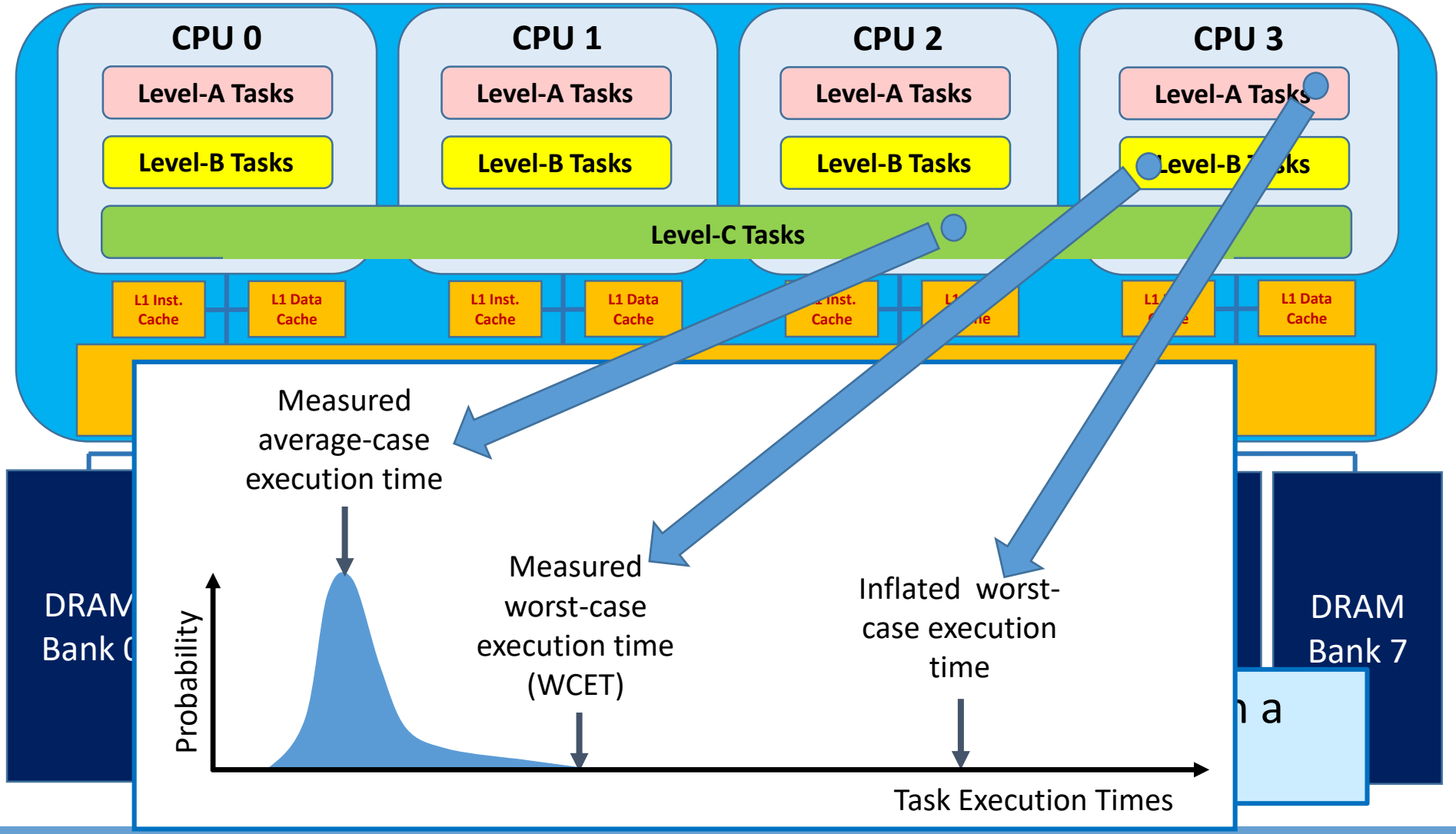
MC^2 on Our Quad-Core Platform



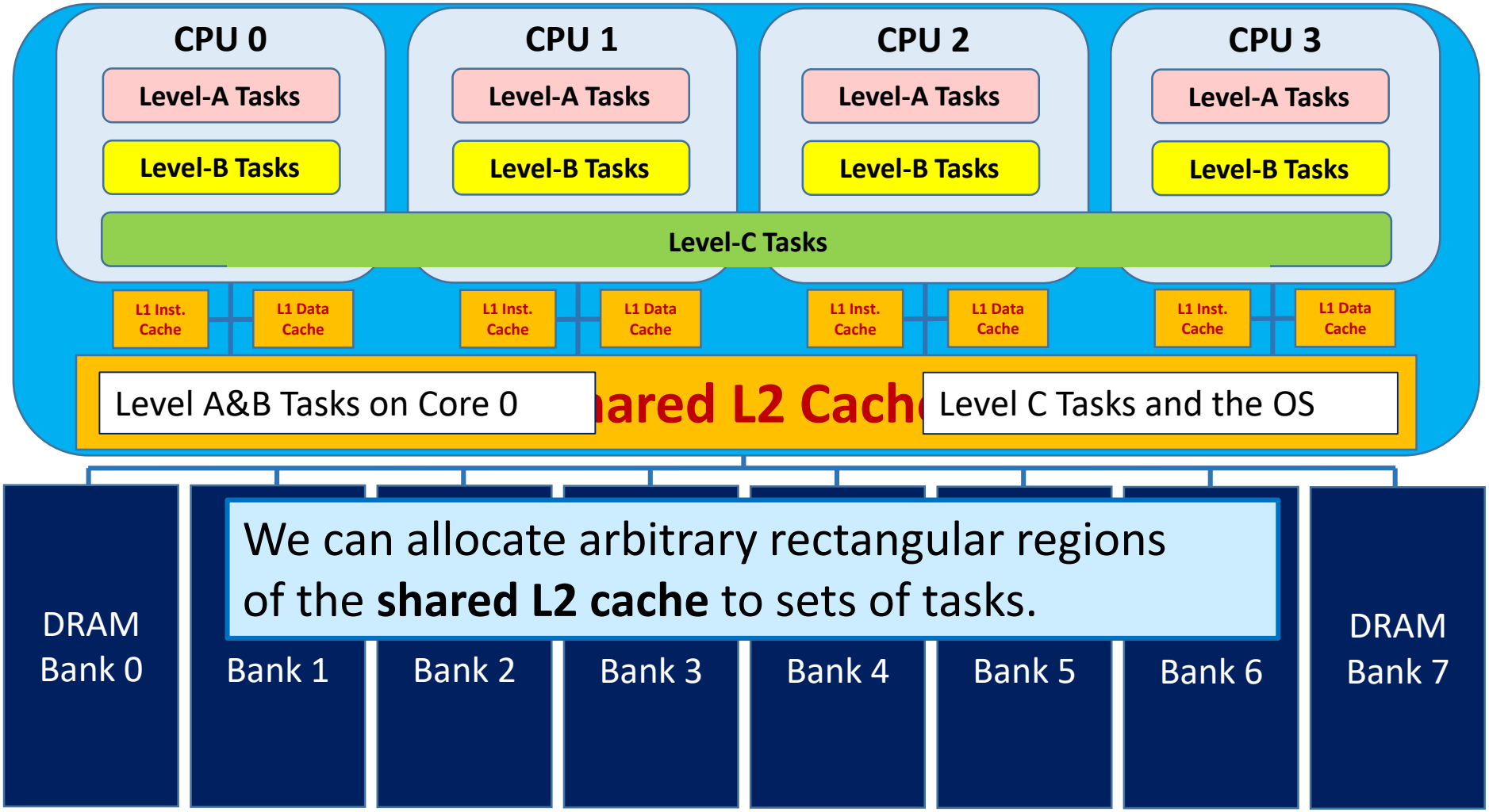
Scheduling Basics:

- Three criticality levels: A (highest) through C (lowest).
- Levels are statically prioritized: A over B over C.
- Level-A and -B tasks are hard real-time and partitioned.
- Level-C tasks are soft real-time and globally scheduled by EDF.

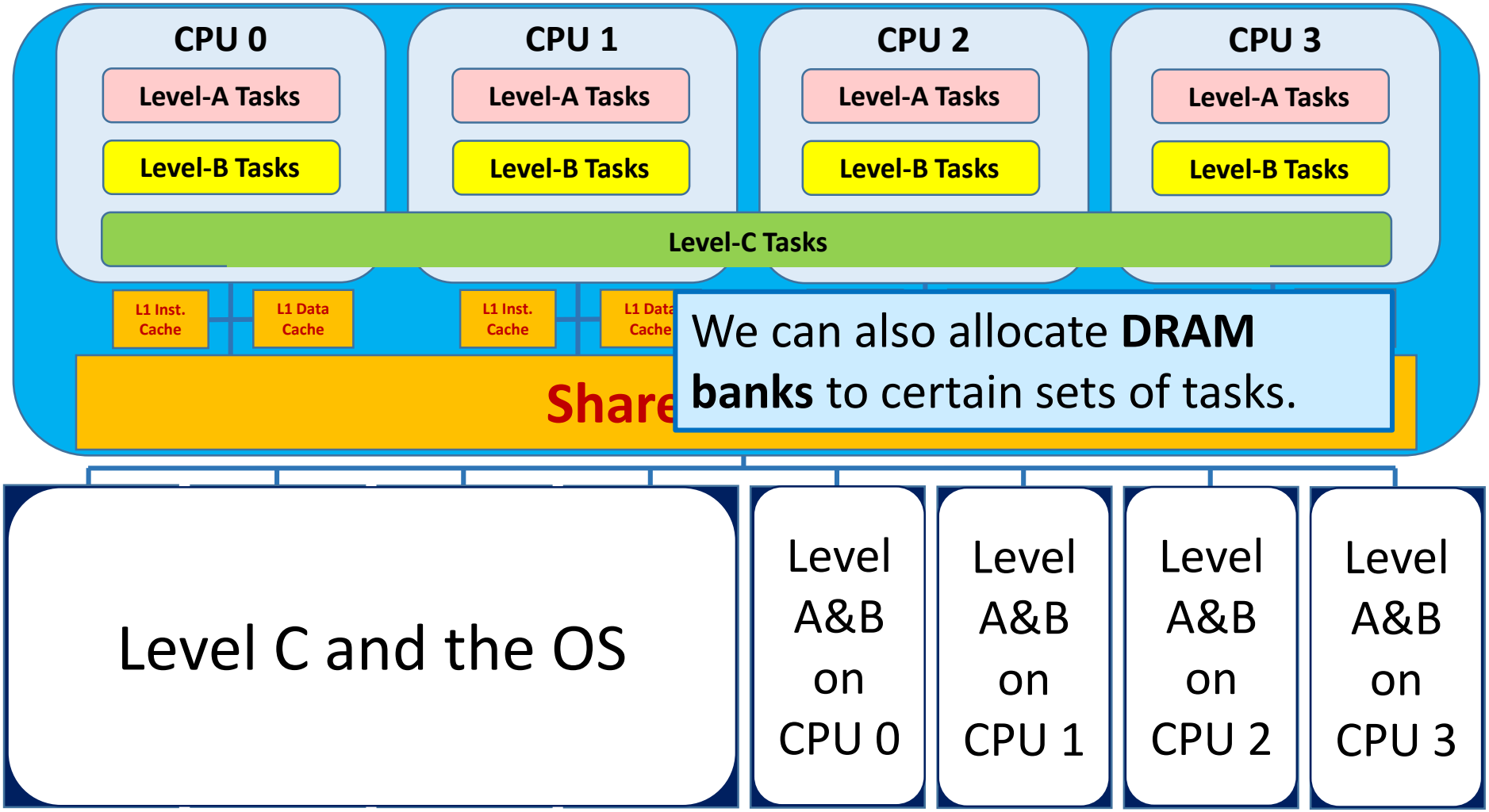
MC^2 on Our Quad-Core Platform



MC^2 on Our Quad-Core Platform

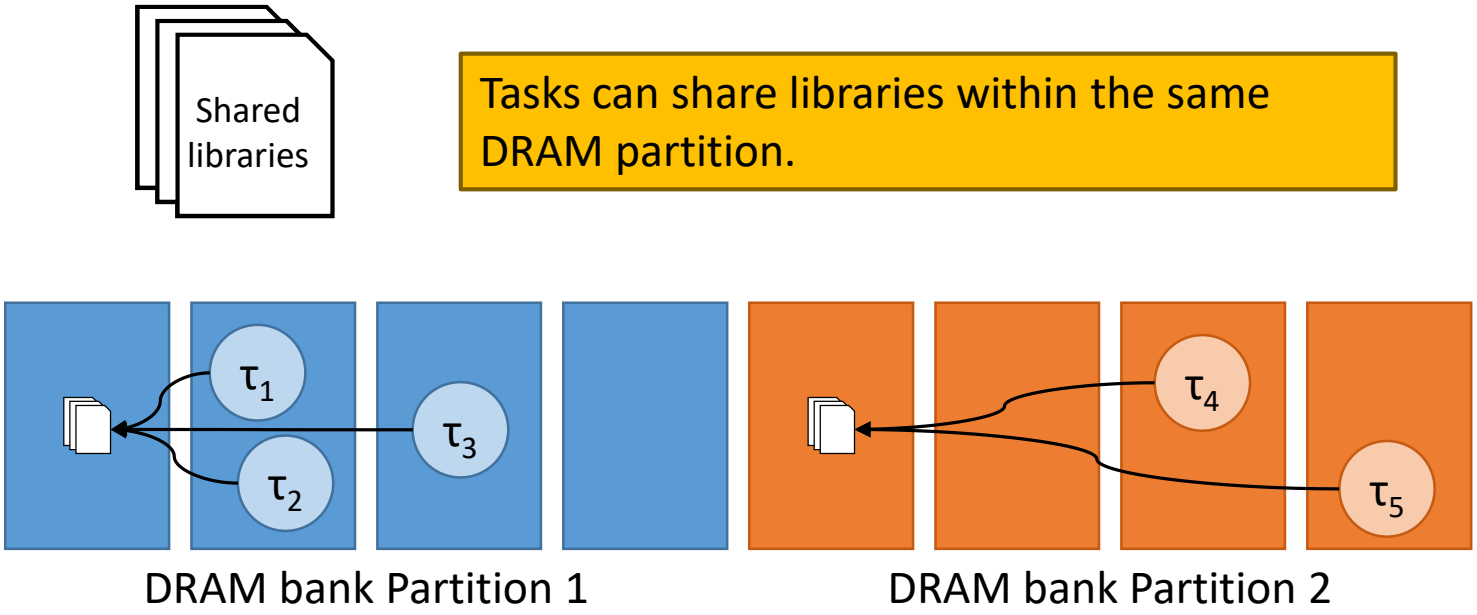


MC^2 on Our Quad-Core Platform



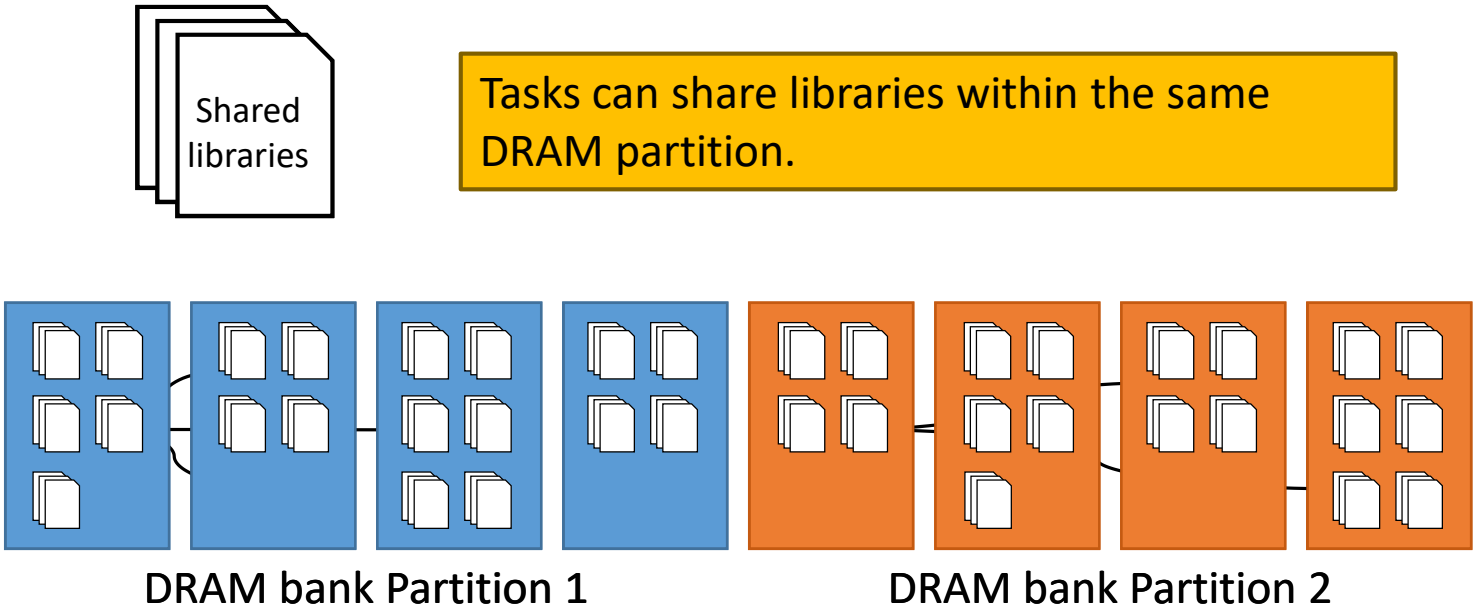
Allowing Shared Libraries

- We introduce per-partition library replicas.

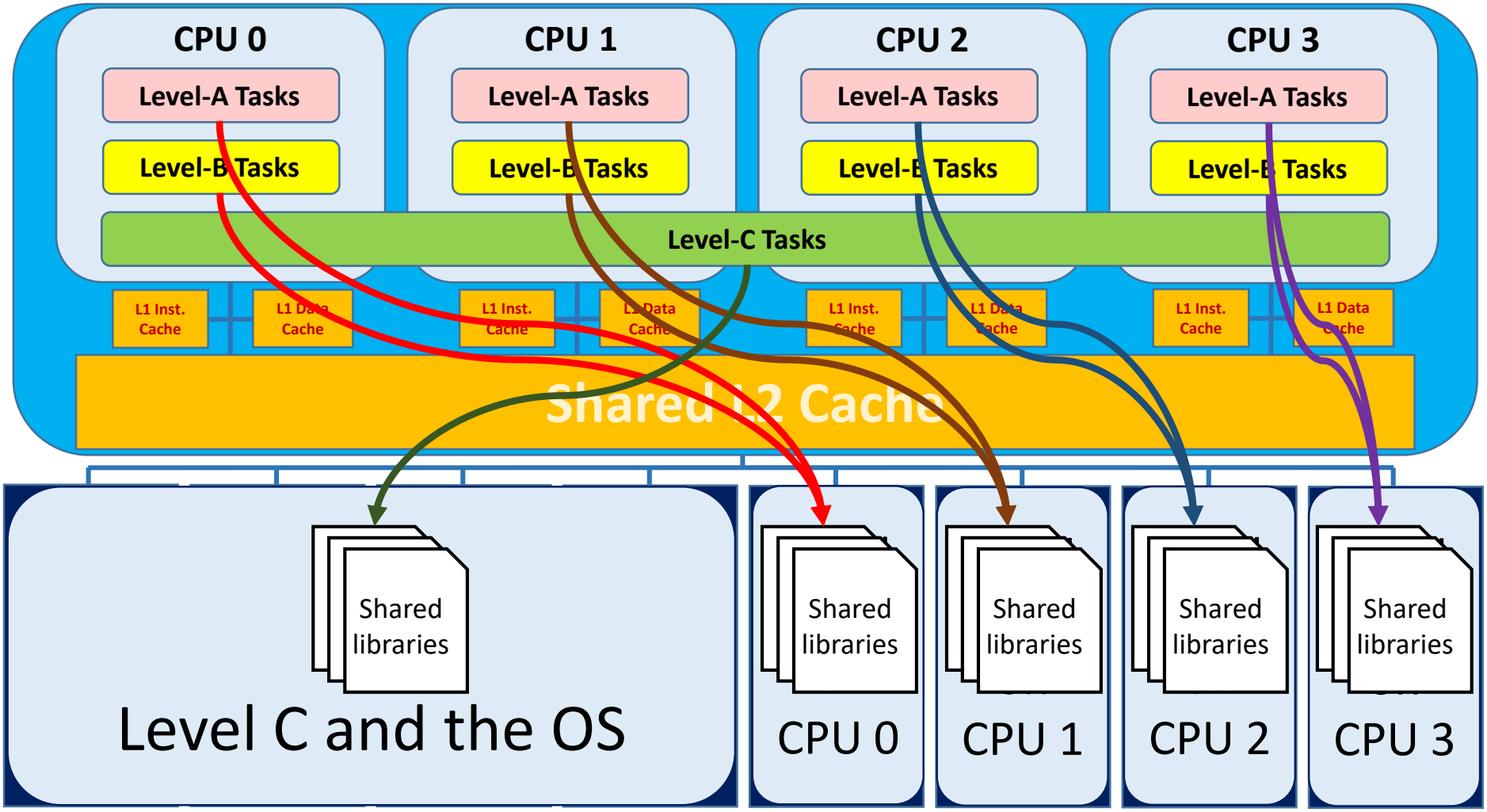


Allowing Shared Libraries

- We introduce per-partition library replicas.



Allowing Shared Libraries



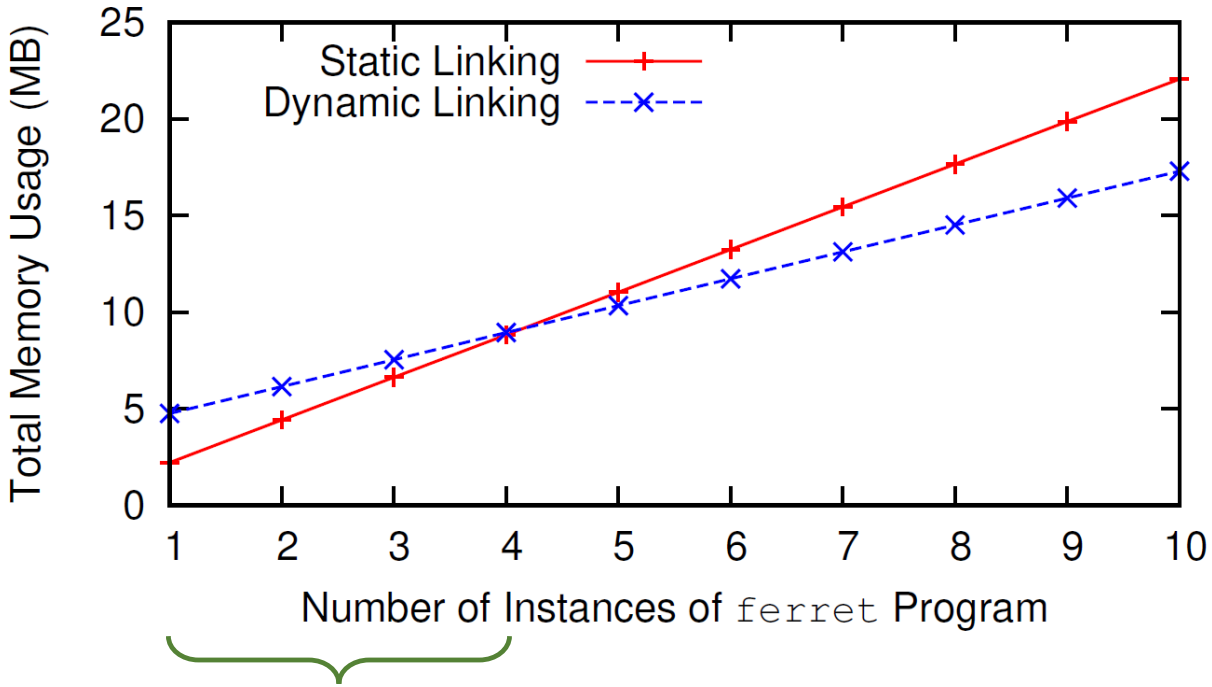
Selective Sharing

- Our major objective is to reduce the system's memory footprint while preserving all isolation properties of MC^2.

- *Selective-sharing* approach enables us to take advantage of dynamic linking's memory savings.

Selective Sharing

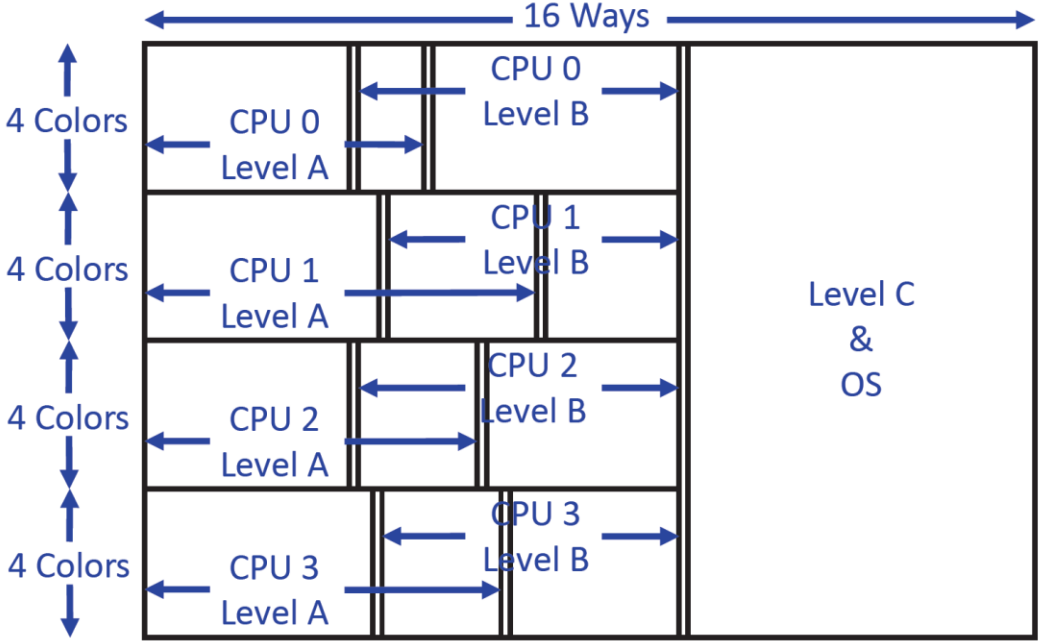
- If libraries are shared by few tasks, static linking may result in better schedulability.



Static linking requires less memory space.

LLC and DRAM Allocation

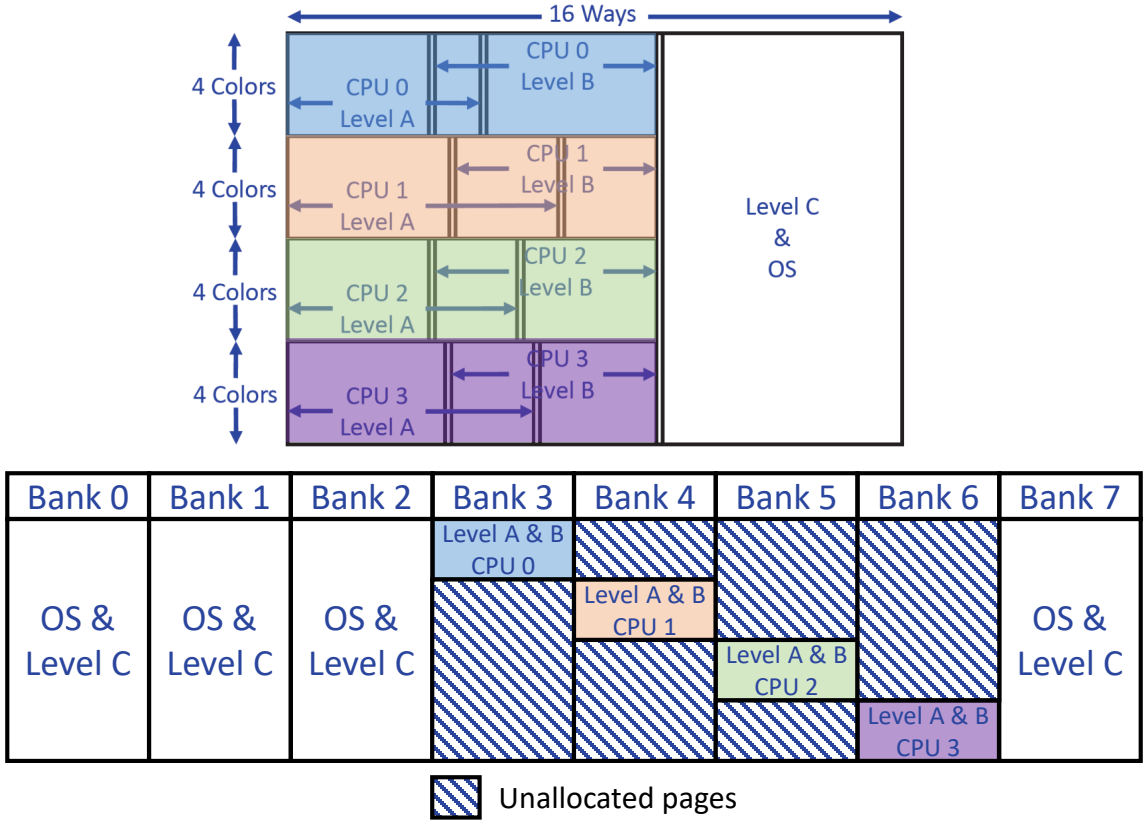
- The size of LLC partition is determined by an optimization framework based on linear programming [RTSS15].



LLC allocation

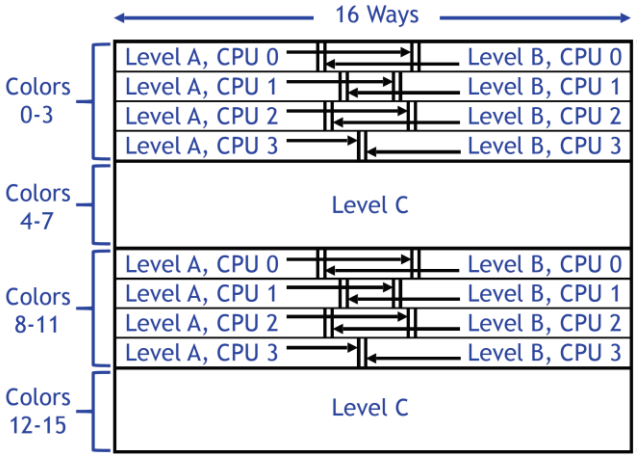
Non-interleaved vs. Interleaved Memory

- With non-interleaved memory, each bank contains 16 colors.



Non-interleaved vs. Interleaved Memory

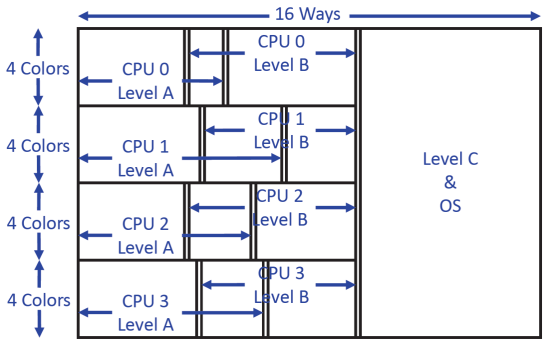
- With interleaved memory, each bank contains only two colors.

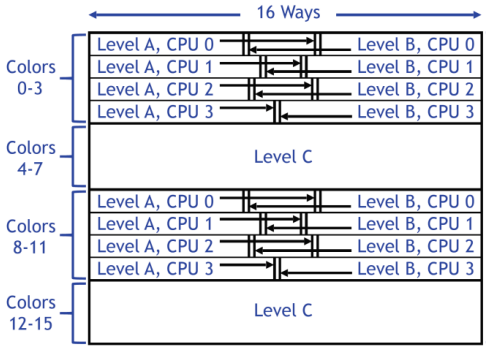


Bank 0	Bank 1	Bank 2	Bank 3	Bank 4	Bank 5	Bank 6	Bank 7
OS & Levels A & B CPU 0	OS & Levels A & B CPU 1	OS & Levels A & B CPU 2	OS & Levels A & B CPU 3	OS & Level C	OS & Level C	OS & Level C	OS & Level C

Evaluation

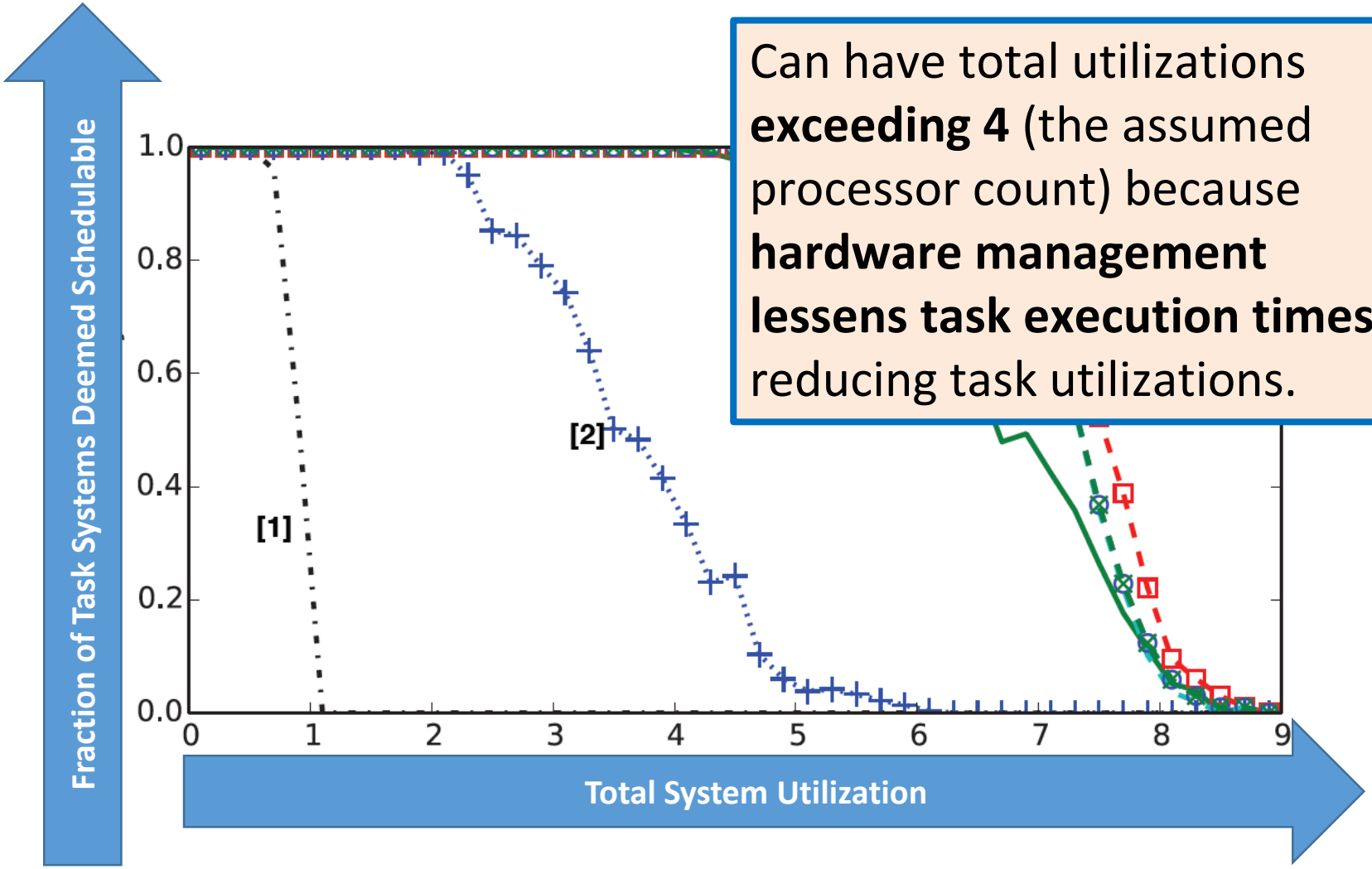
- We conducted a large-scale overhead-aware schedulability study.
- We used:
  - Millions of randomly generated task systems.
  - Optimized LLC allocations and linking types based on linear programming.





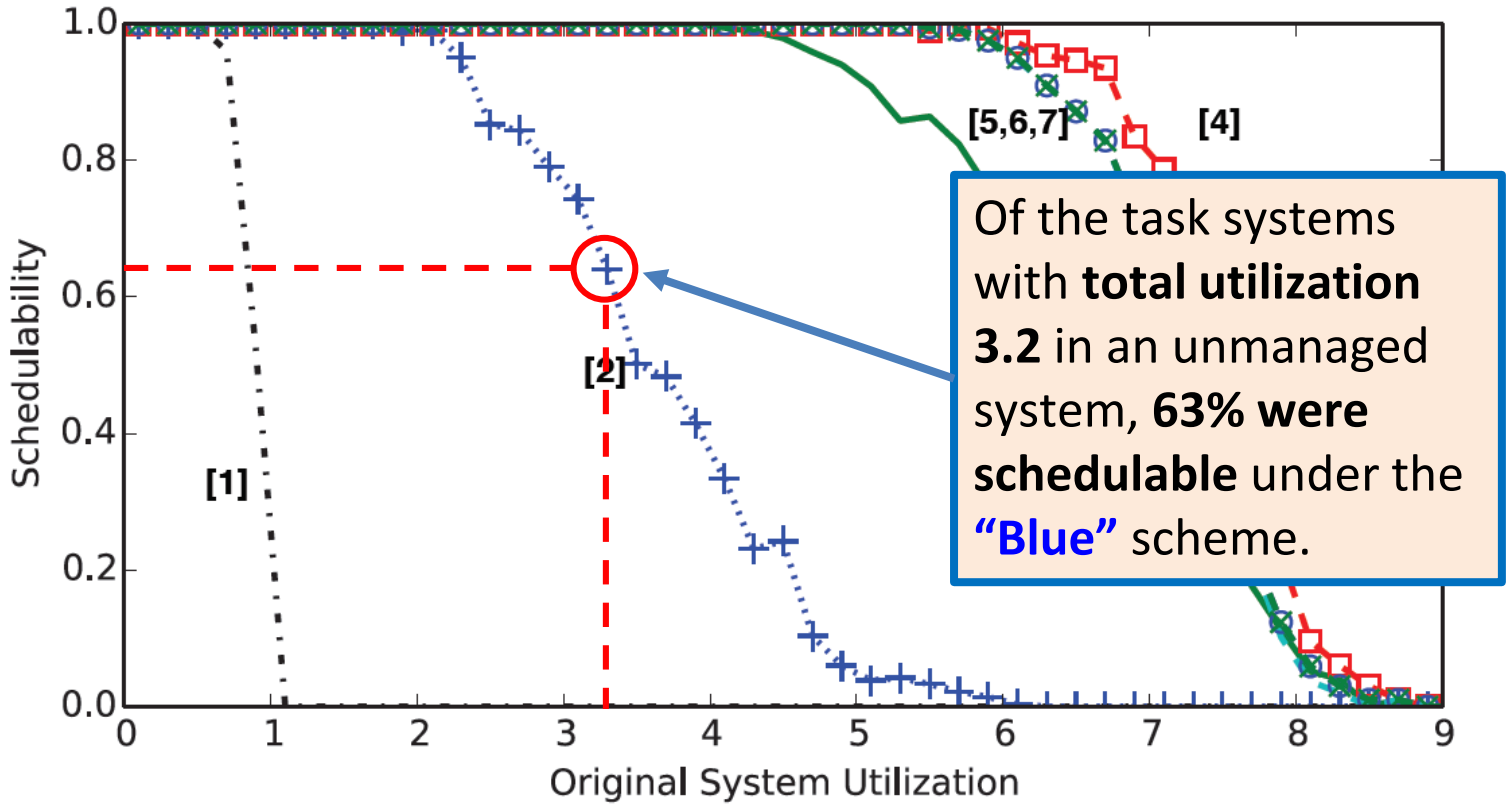
  - Measured overheads on NXP iMX6 quad-core ARM platform with LITMUS^{RT}.

Schedulability



Can have total utilizations **exceeding 4** (the assumed processor count) because **hardware management lessens task execution times**, reducing task utilizations.

Schedulability



Of the task systems with **total utilization 3.2** in an unmanaged system, **63% were schedulable** under the **"Blue"** scheme.

NI-STC: Non-Interleaved, static linking [2]
NI-SSH: Non-Interleaved, selective sharing [3]
NI-IDL: Non-Interleaved, no DRAM constraints [4]
I-STC: Interleaved, static linking [5]
I-SSH: Interleaved, selective sharing [6]
I-IDL: Interleaved, no DRAM constraints [7]

MC² with no DRAM contraints and non-interleaved memory.

MC² with selective sharing and non-interleaved memory

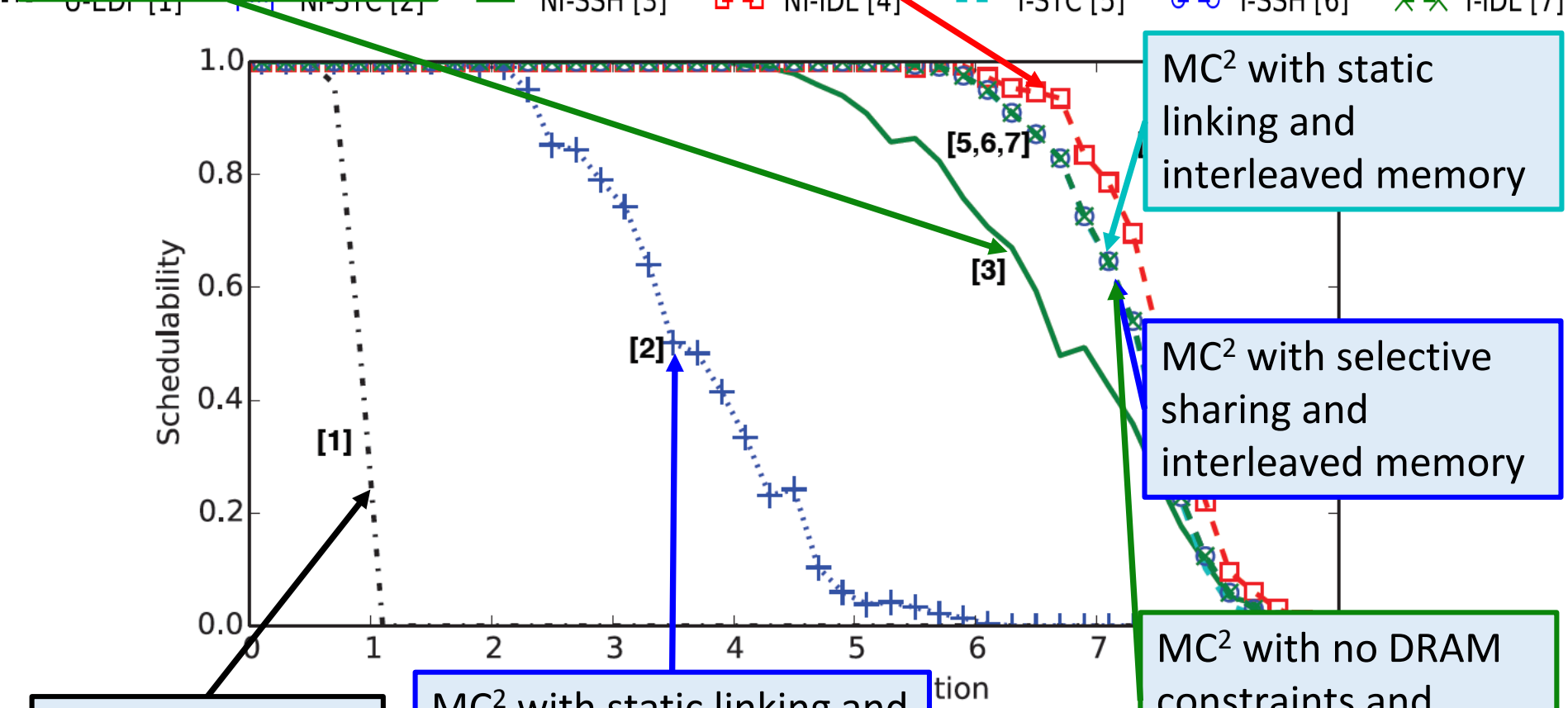
MC² with static linking and interleaved memory

MC² with selective sharing and interleaved memory

MC² with no DRAM constraints and interleaved memory

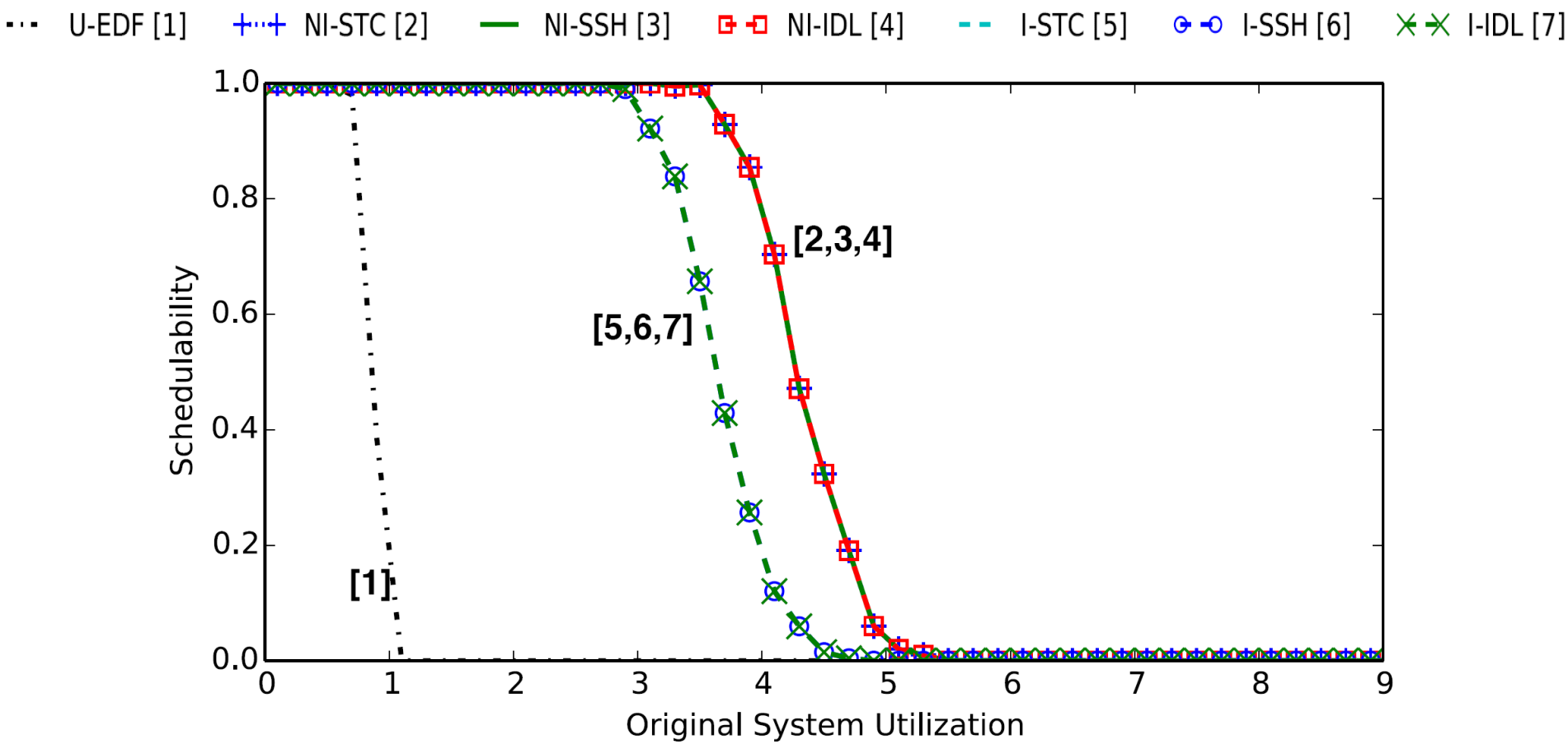
Uniprocessor EDF

MC² with static linking and non-interleaved memory.



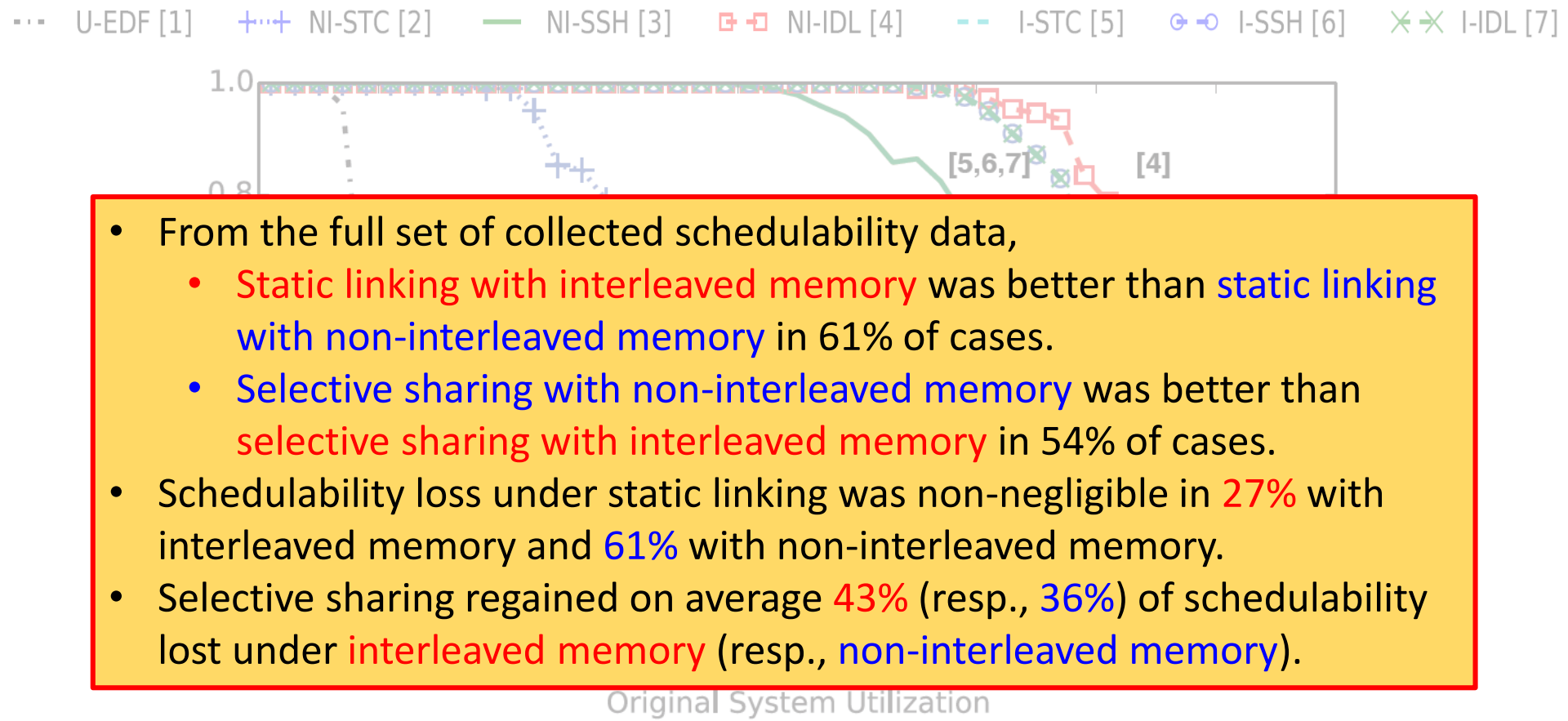
Schedulability

NI-STC: Non-Interleaved, static linking [2]
NI-SSH: Non-Interleaved, selective sharing [3]
NI-IDL: Non-Interleaved, no DRAM constraints [4]
I-STC: Interleaved, static linking [5]
I-SSH: Interleaved, selective sharing [6]
I-IDL: Interleaved, no DRAM constraints [7]



Schedulability

NI-STC: Non-Interleaved, static linking [2]
NI-SSH: Non-Interleaved, selective sharing [3]
NI-IDL: Non-Interleaved, no DRAM constraints [4]
I-STC: Interleaved, static linking [5]
I-SSH: Interleaved, selective sharing [6]
I-IDL: Interleaved, no DRAM constraints [7]

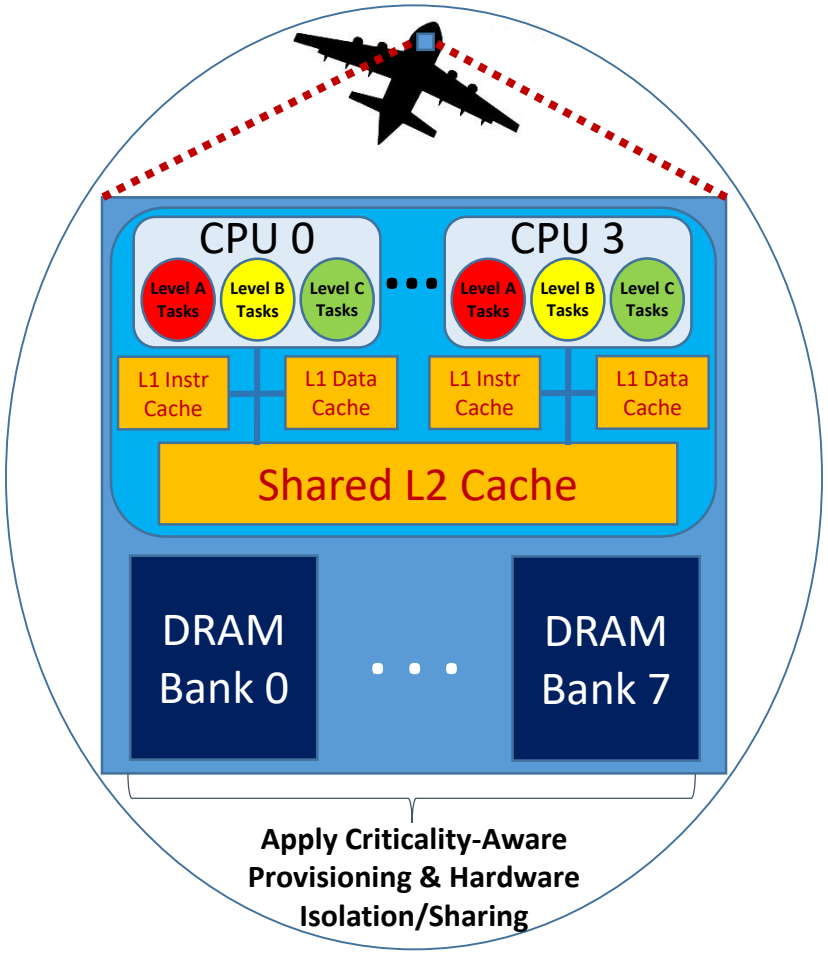


- From the full set of collected schedulability data,
  - Static linking with interleaved memory was better than static linking with non-interleaved memory in 61% of cases.
  - Selective sharing with non-interleaved memory was better than selective sharing with interleaved memory in 54% of cases.
- Schedulability loss under static linking was non-negligible in 27% with interleaved memory and 61% with non-interleaved memory.
- Selective sharing regained on average 43% (resp., 36%) of schedulability lost under interleaved memory (resp., non-interleaved memory).

Conclusion

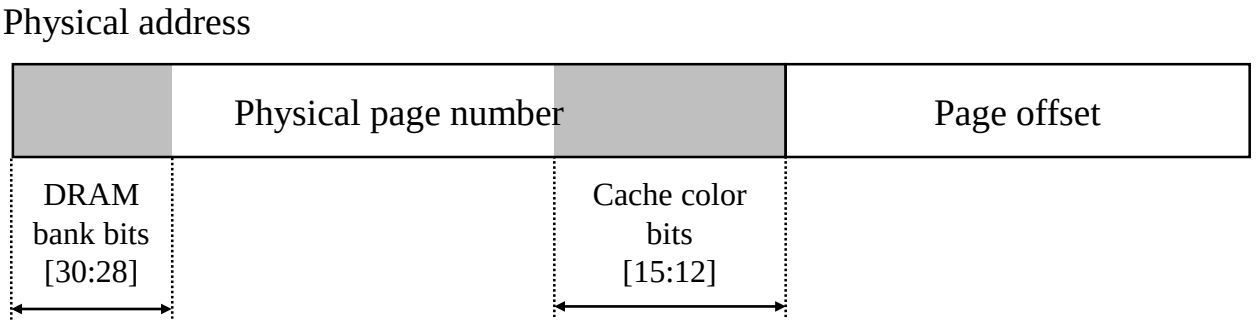
- We examined the issue of sharing, which directly breaks isolation for any hardware management approach.
- We considered the impact of memory limits on MC^2.
- We proposed per-partition library replicas for allowing shared library while supporting hardware isolation.
- We evaluated our approach with a large-scale overhead-aware scheduability study.

Thank You!


CPU 0
Level A Tasks
Level B Tasks
Level C Tasks
CPU 3
Level A Tasks
Level B Tasks
Level C Tasks
L1 Instr Cache
L1 Data Cache
L1 Instr Cache
L1 Data Cache
Shared L2 Cache
DRAM Bank 0
DRAM Bank 7
Apply Criticality-Aware Provisioning & Hardware Isolation/Sharing

LLC and DRAM Allocation

- Non-interleaved memory.



- Interleaved memory