The One-Out-of-m Multicore Problem

Jim Anderson, Kenan Professor
University of North Carolina at Chapel Hill

Work supported by NSF and AFOSR

Outline

- Problems caused by multicore.
  - “The one-out-of-m problem.”
  - Why this is an important problem.
- Basic solution strategy.
  - MC^2 (mixed-criticality on multicore).
  - Hardware management in MC^2.
- Brief overview of recent work.
  - Key focus: features of real-world task systems that break hardware isolation.

The One-Out-Of-m Multicore Problem

» In many safety-critical domains, we would like to be able to exploit the computational capacity of multicore. ***However:***

Image source: http://www.as.northropgrumman.com/products/nucasx47b/assets/lgm_UCAS_3_0911.jpg

– When using an m-core platform in a safety-critical domain, analysis pessimism can be so great, the capacity of the "additional" m – 1 cores is entirely negated.

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

– In avionics, this problem has led to the common practice of simply disabling all but one core if highly critical system components exist.

Roots of the problem:

- Shared hardware that is not predictably managed.
  - See the FAA position paper “CAST 32” for an extensive discussion of problems caused by multicore.
- Excessive pessimism in provisioning tasks.
  - Mixed-criticality analysis seeks to address this.

» In

dc

able to exploit the computational capacity multicore. ***However:***

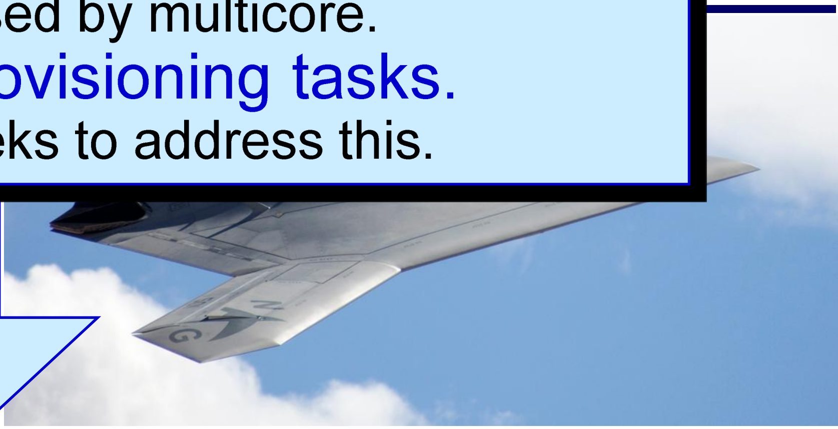


Image source: http://www.as.northropgrumman.com/products/nucasx47b/assets/lgm_UCAS_3_0911.jpg

– When using an m-core platform in a safety-critical domain, analysis pessimism can be so great, the capacity of the “additional” m – 1 cores is entirely negated.

» We call this the “one-out-of-m” problem.

– In avionics, this problem has led to the common practice of simply disabling all but one core if highly critical system components exist.

What is Mixed-Criticality Analysis?

(Vestal [RTSS ‘07])

- Each task is assigned a criticality level.
- Each task has provisioned execution time (PET) specified at each criticality level.
  - » PETs at higher levels are (typically) larger.
- **Example:** Assuming criticality levels A (highest), B, C, etc., task τ_i might have PETs $C_i^A = 20$, $C_i^B = 12$, $C_i^C = 5$, …
- **Rationale:** Will use more pessimistic analysis at high levels, more optimistic at low levels.

What is Mixed-Criticality Analysis?

(Vestal [RTSS '07])

- Each task is assigned a criticality level.
- Each task has provisioned execution time (PET) specified at each criticality level.
  - » PETs at higher levels are (typically) larger.
- The task system is *correct at Level X* iff all Level-X tasks meet their timing requirements assuming all tasks have Level-X PETs.

What is Mixed-Criticality Analysis?

(Vestal [RTSS ‘07])

Some “weirdness” here: Not just one system anymore, but several: the Level-A system, Level-B,…

-
-
  - » PETs at higher levels typically) larger.
- The task system is *correct at Level X* iff all Level-X tasks meet their timing requirements assuming all tasks have Level-X PETs.

Outline

- Problems caused by multicore.
  - » "The one-out-of-m problem."
  - » Why this is an important problem.
- Basic solution strategy.
  - » MC^2 (mixed-criticality on multicore).
  - » Hardware management in MC^2.
- Brief overview of recent work.
  - » Key focus: features of real-world task systems that break hardware isolation.

Our Solution Strategy

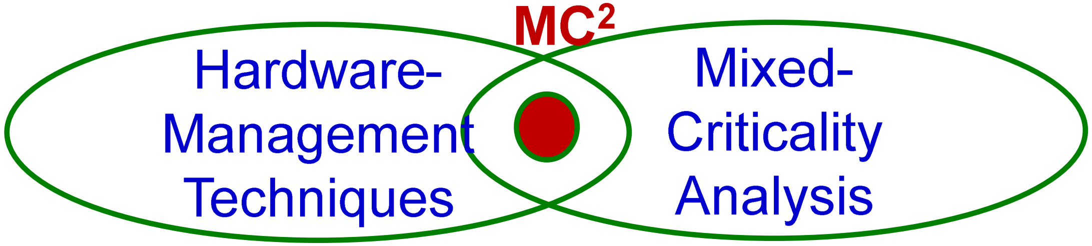
- W.r.t. lessening capacity loss generally (even on uniprocessors), two orthogonal approaches have been investigated previously:
  - » **Hardware-management techniques** that reduce hardware interference.
  - » **Mixed-criticality analysis techniques** that enable less critical tasks to be provisioned less pessimistically.

Hardware-Management Techniques

Mixed-Criticality Analysis

Our Solution Strategy

- Our work focuses broadly on research questions that arise when applying both approaches together.
  - » We are addressing such questions in the context of a resource-allocation and analysis framework developed by us called **MC^2 (mixed criticality on multicore)**.



MC^2: Starting Assumptions

- Modest core count (e.g., 2-8).
  - » Quad-core in avionics would be a tremendous innovation.

MC^2: Starting Assumptions

- Modest core count (e.g., 2-8).
- Modest number of criticality levels (e.g., 2-5).
  - » 2 may be too constraining
  - » ∞ isn't practically interesting.
  - » These levels may not necessarily match DO-178B/C.

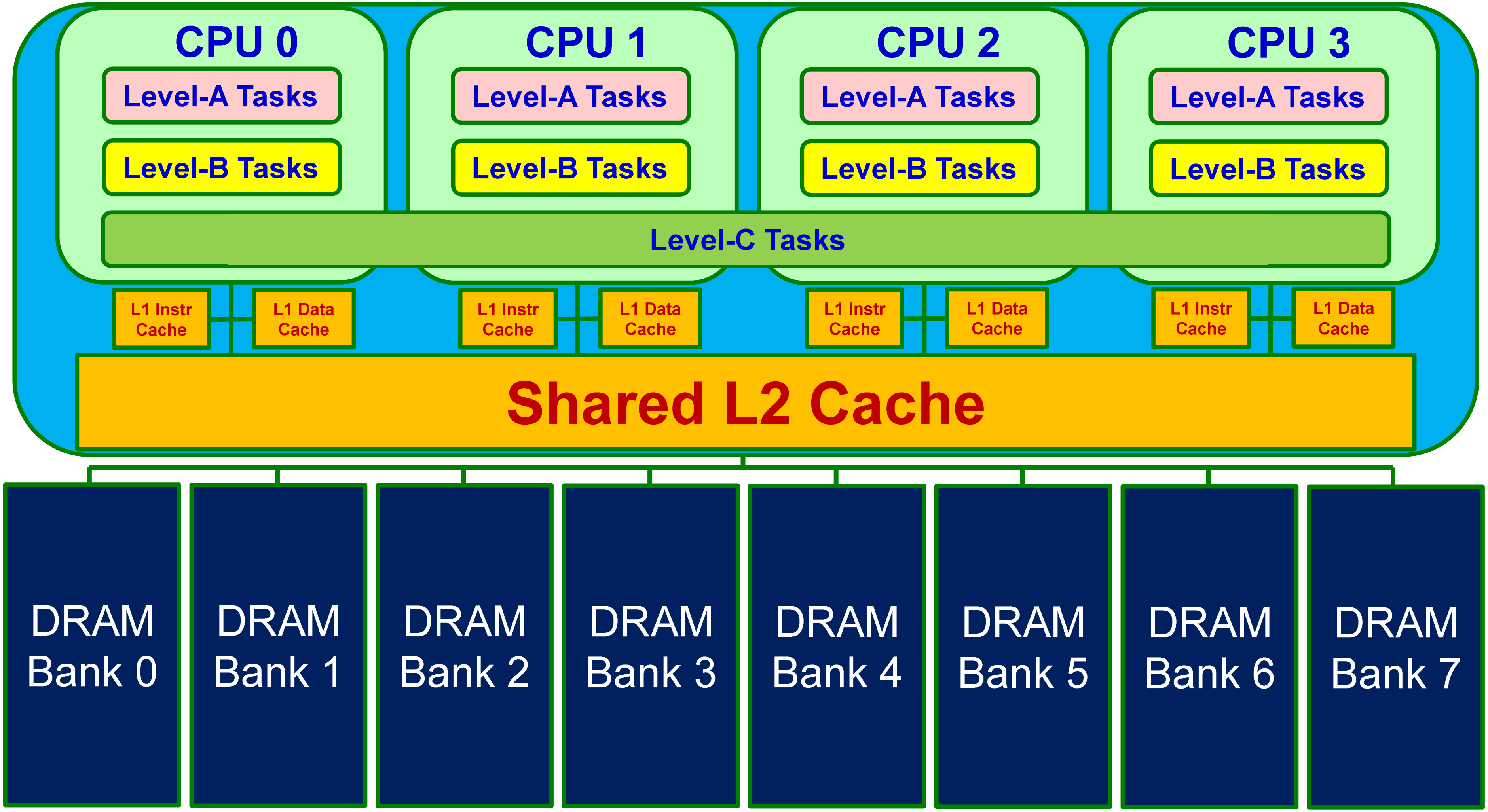
MC^2: Starting Assumptions

- Modest core count (e.g., 2-8).
- Modest number of criticality levels (e.g., 2-5).

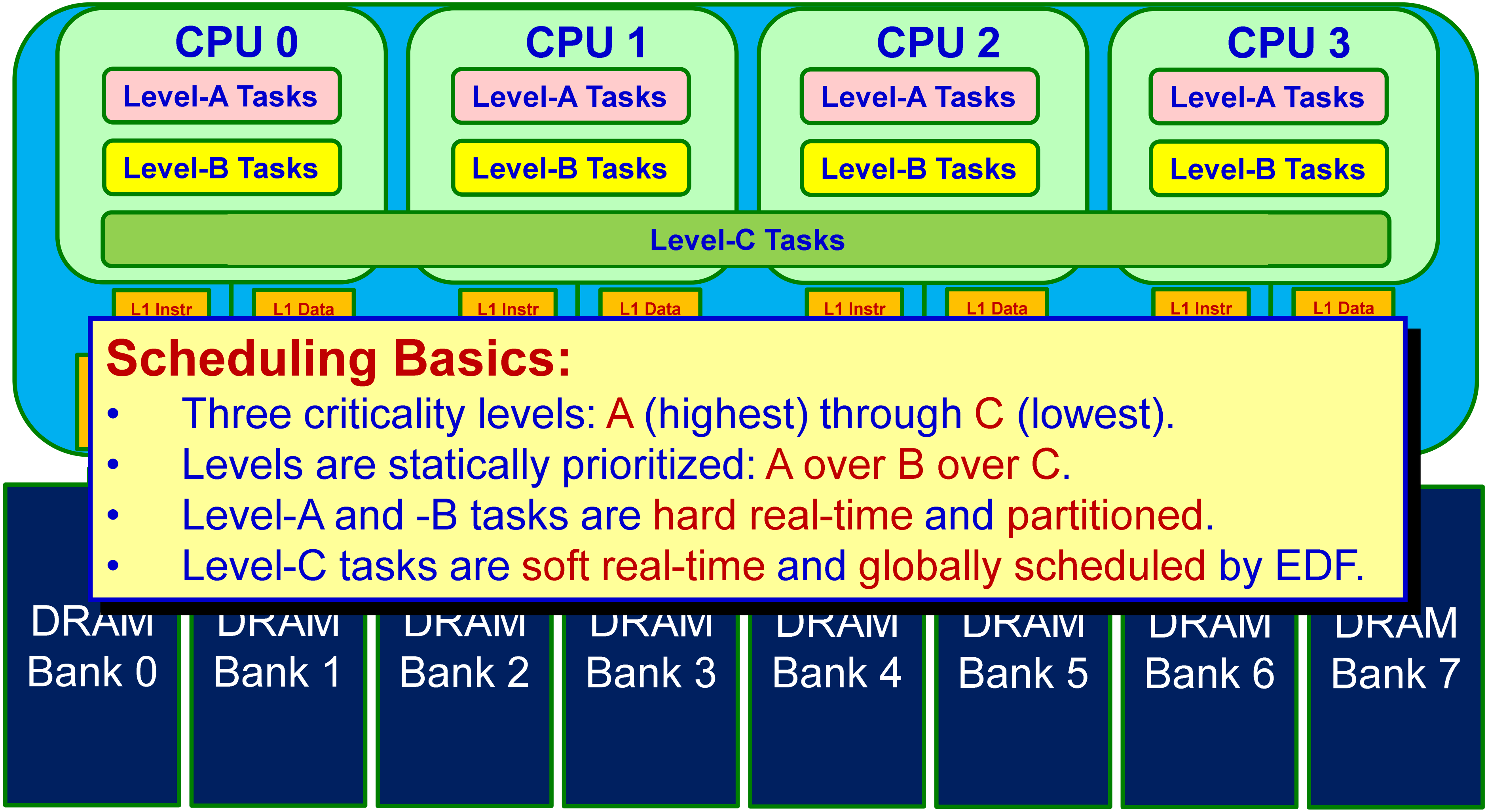
Main motivation: To develop a framework that allows interesting design tradeoffs to be investigated that is reasonably plausible from an avionics point of view.

A Non-Goal: Developing a framework that could really be used in avionics today.

MC^2 on Our Quad-Core Test Platform



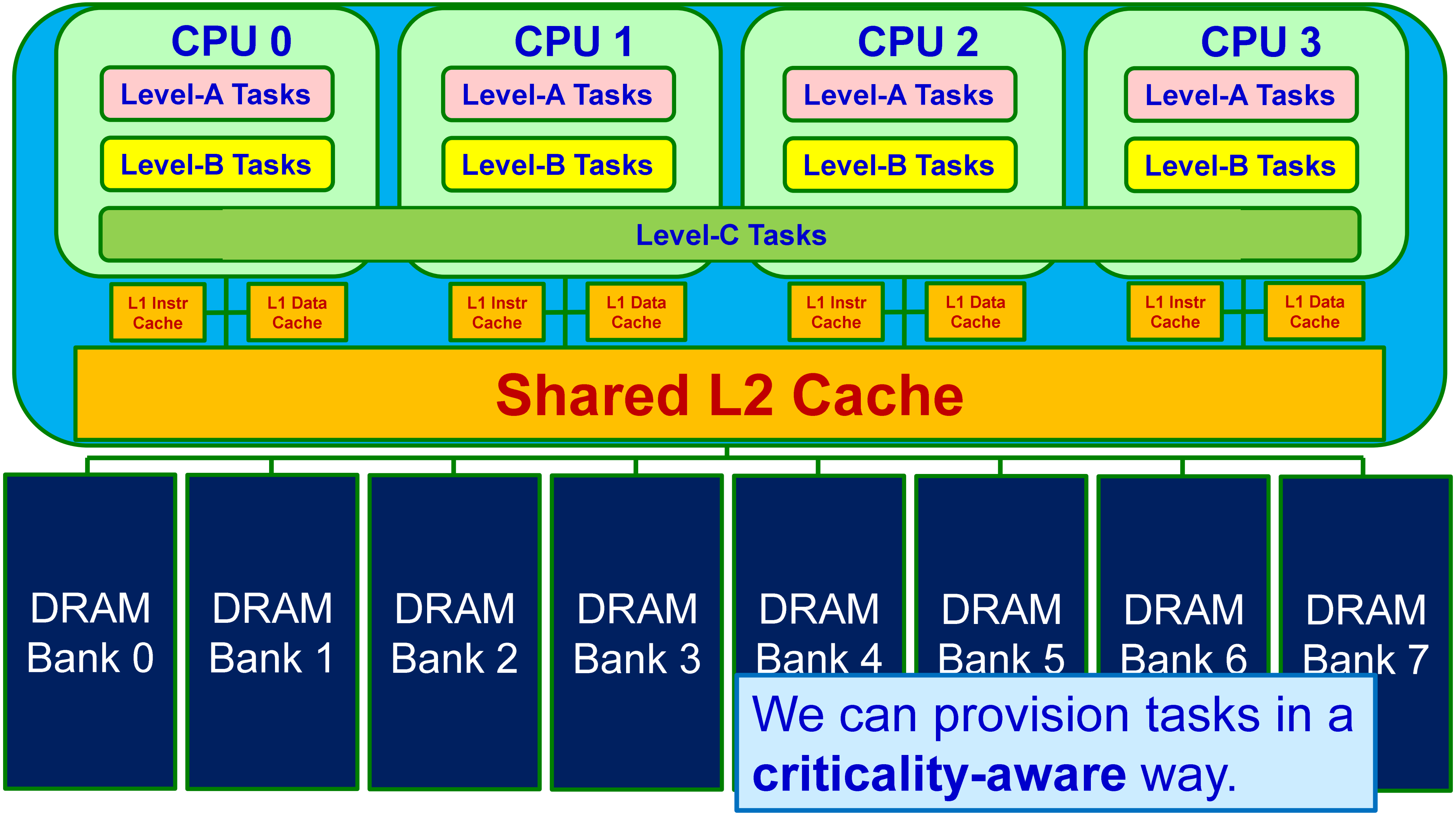
MC^2 on Our Quad-Core Test Platform



Scheduling Basics:

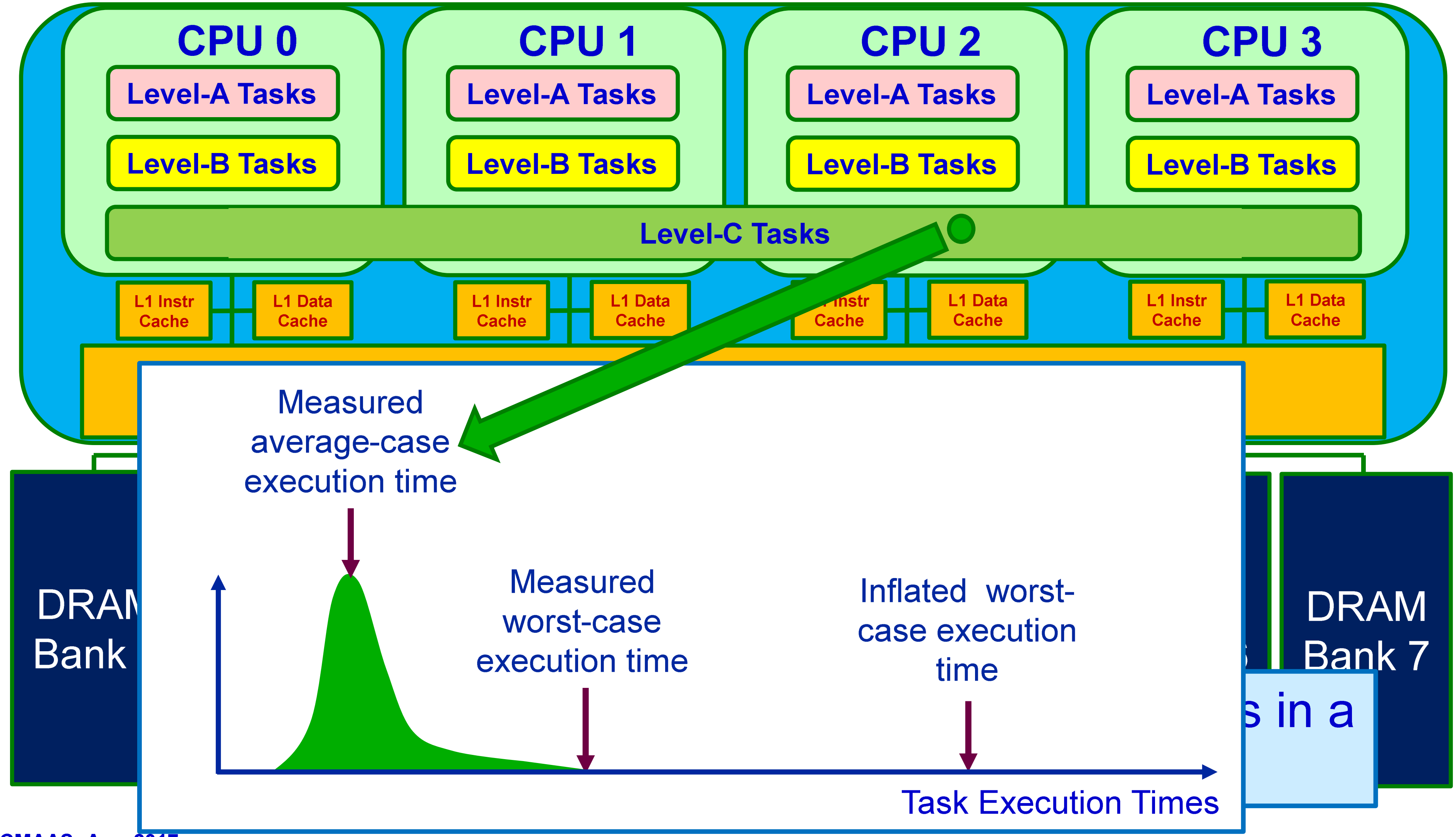
- Three criticality levels: A (highest) through C (lowest).
- Levels are statically prioritized: A over B over C.
- Level-A and -B tasks are hard real-time and partitioned.
- Level-C tasks are soft real-time and globally scheduled by EDF.

MC^2 on Our Quad-Core Test Platform

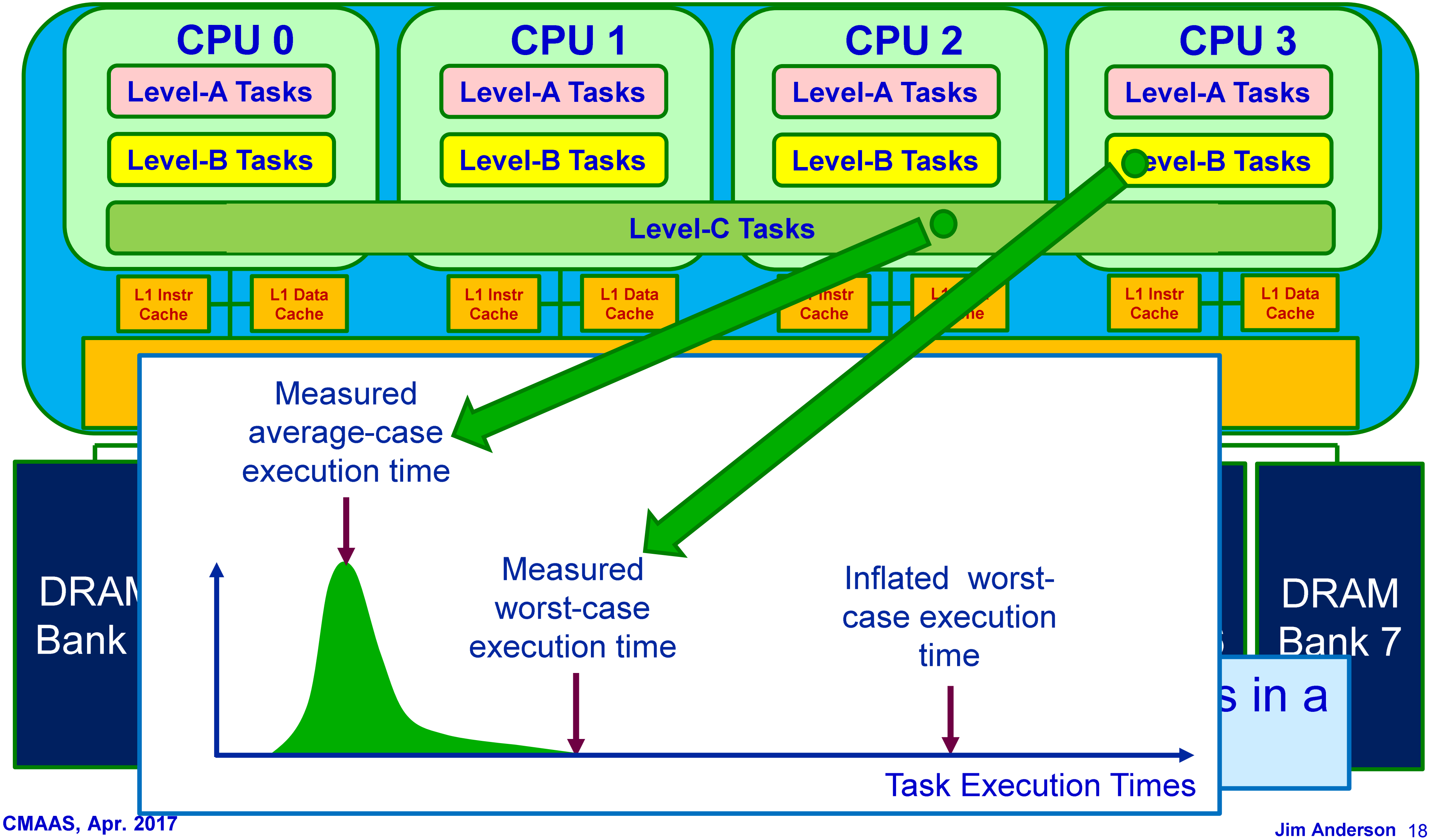


We can provision tasks in a **criticality-aware** way.

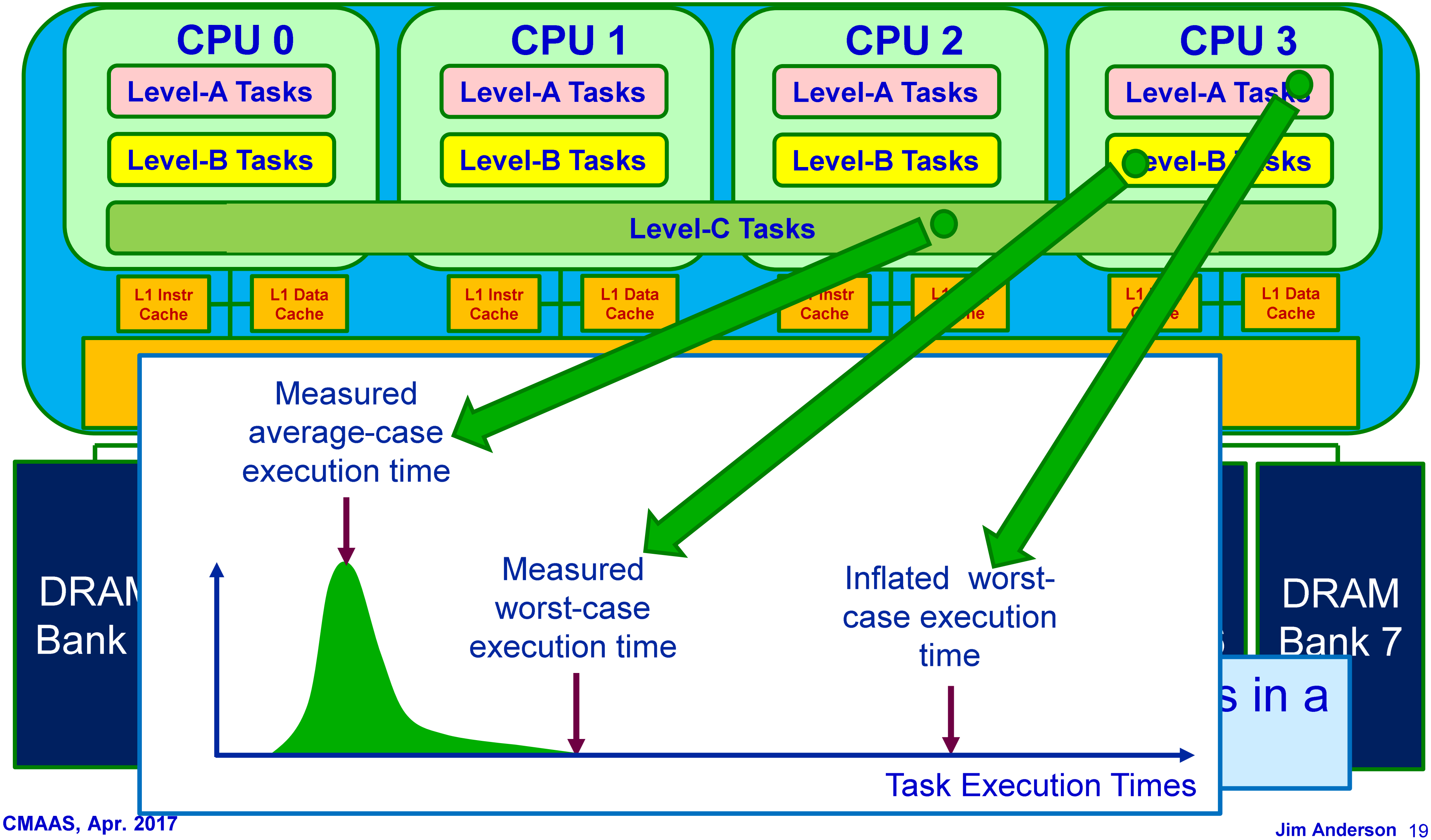
MC^2 on Our Quad-Core Test Platform



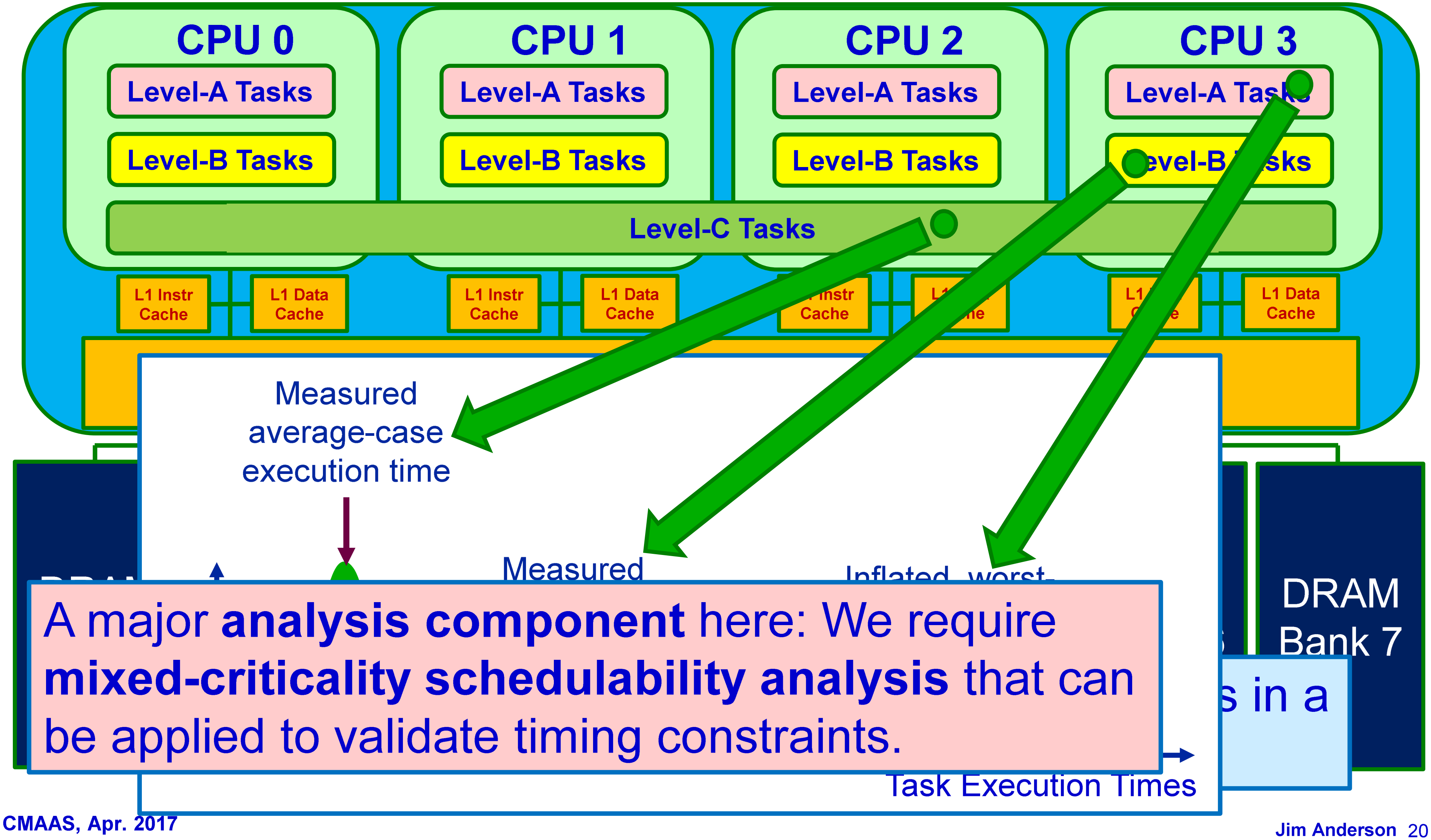
MC^2 on Our Quad-Core Test Platform



MC^2 on Our Quad-Core Test Platform

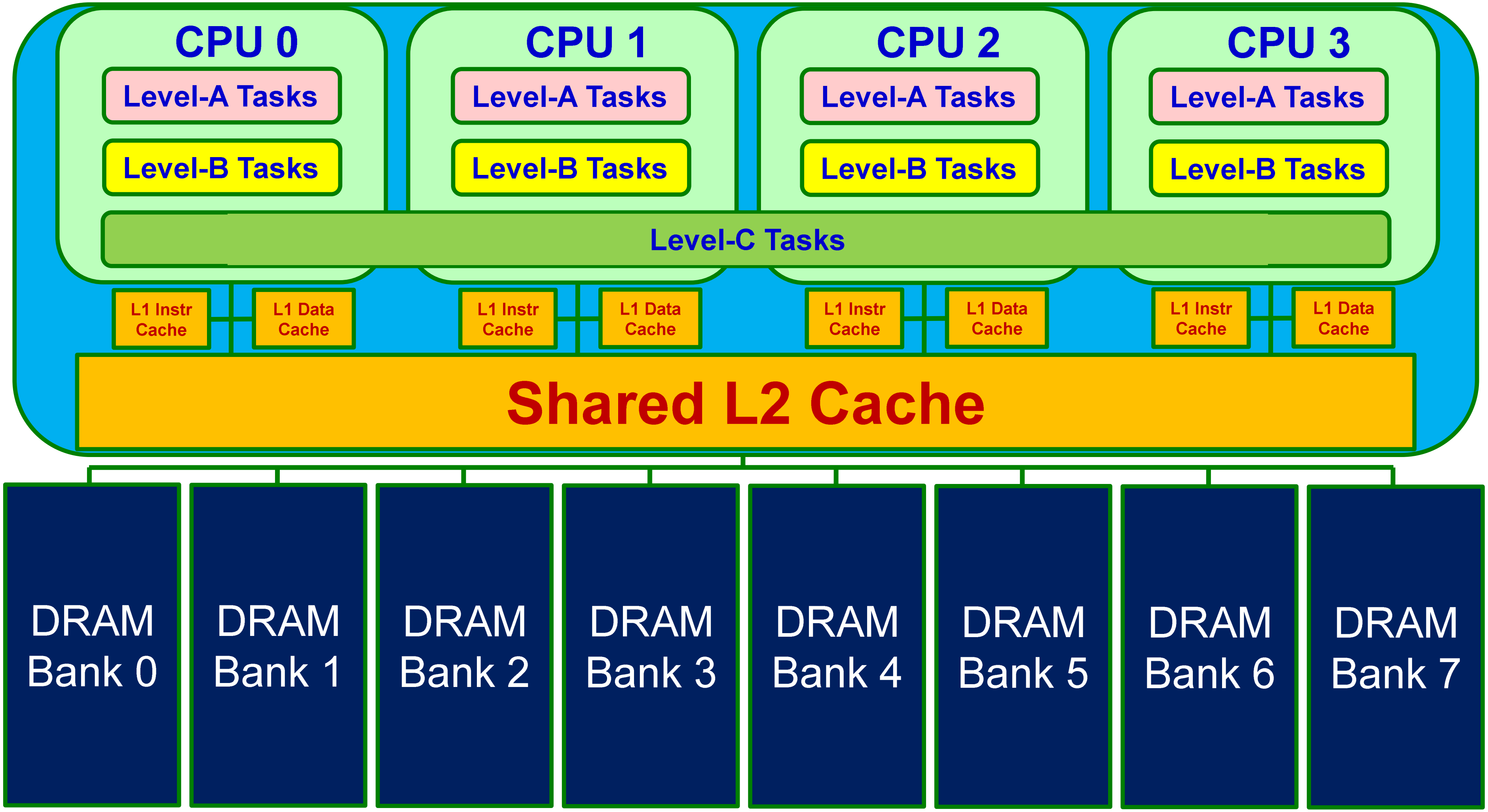


MC^2 on Our Quad-Core Test Platform

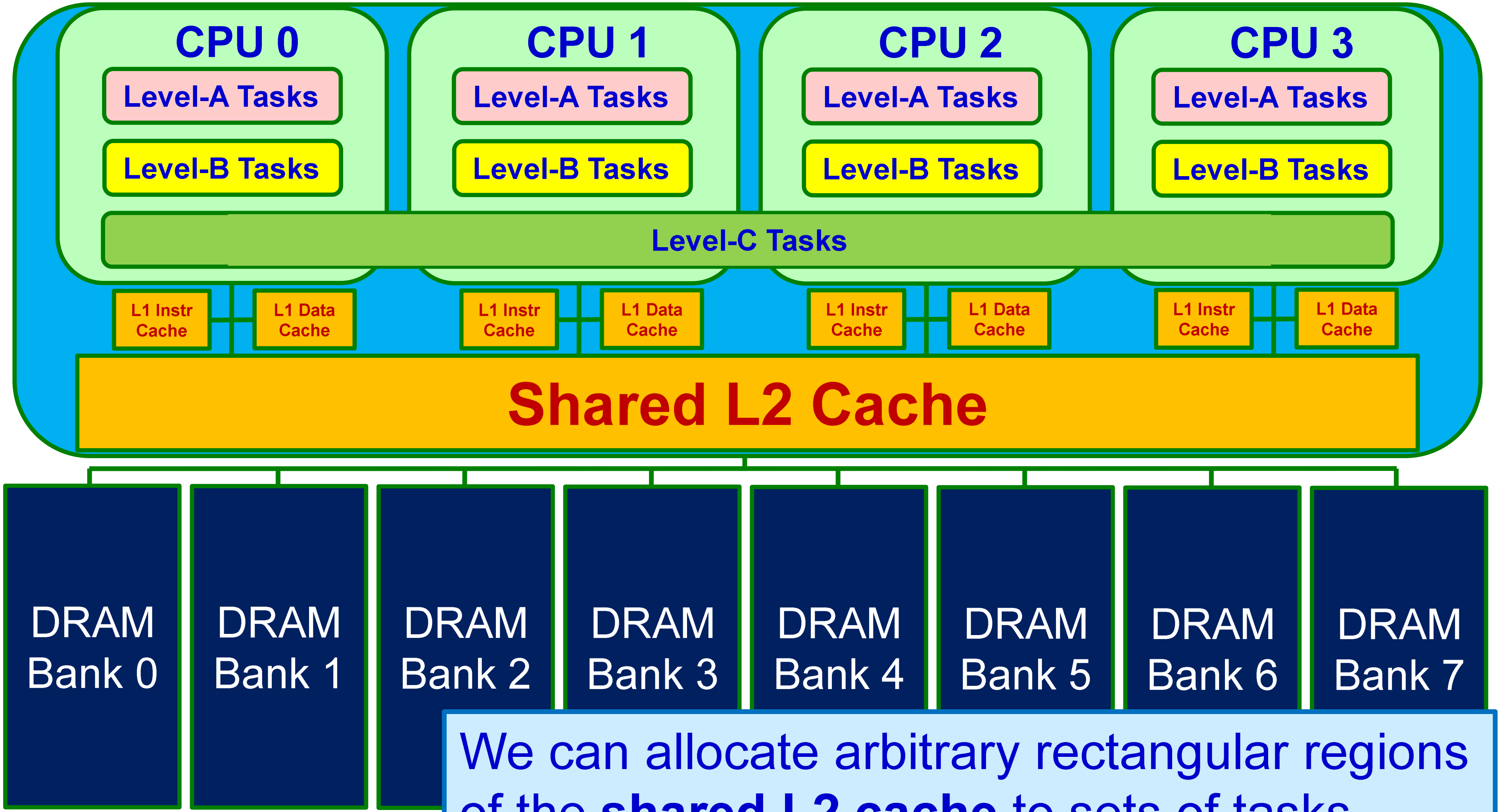


A major **analysis component** here: We require **mixed-criticality schedulability analysis** that can be applied to validate timing constraints.

MC^2 on Our Quad-Core Test Platform

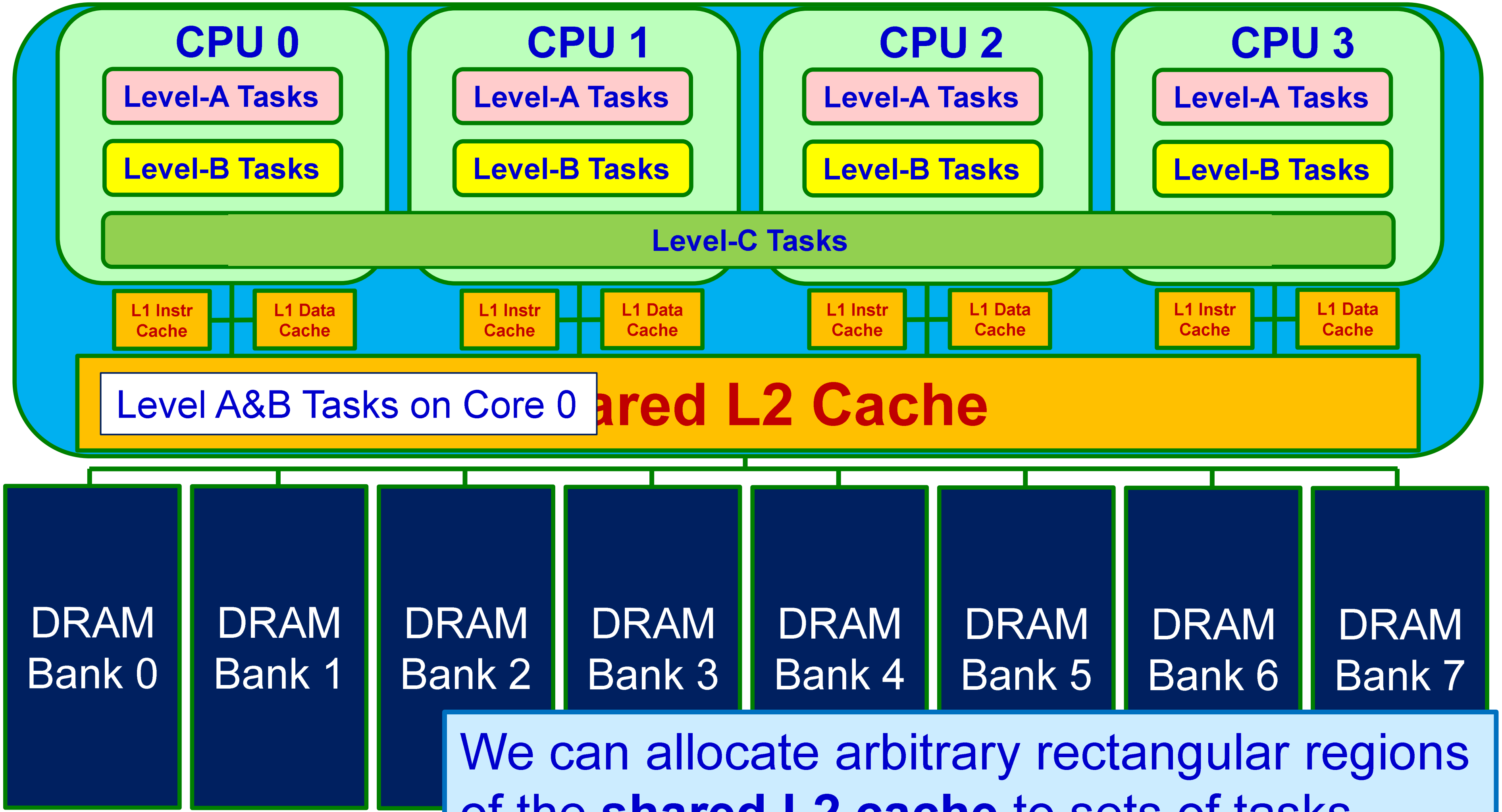


MC^2 on Our Quad-Core Test Platform



We can allocate arbitrary rectangular regions of the **shared L2 cache** to sets of tasks.

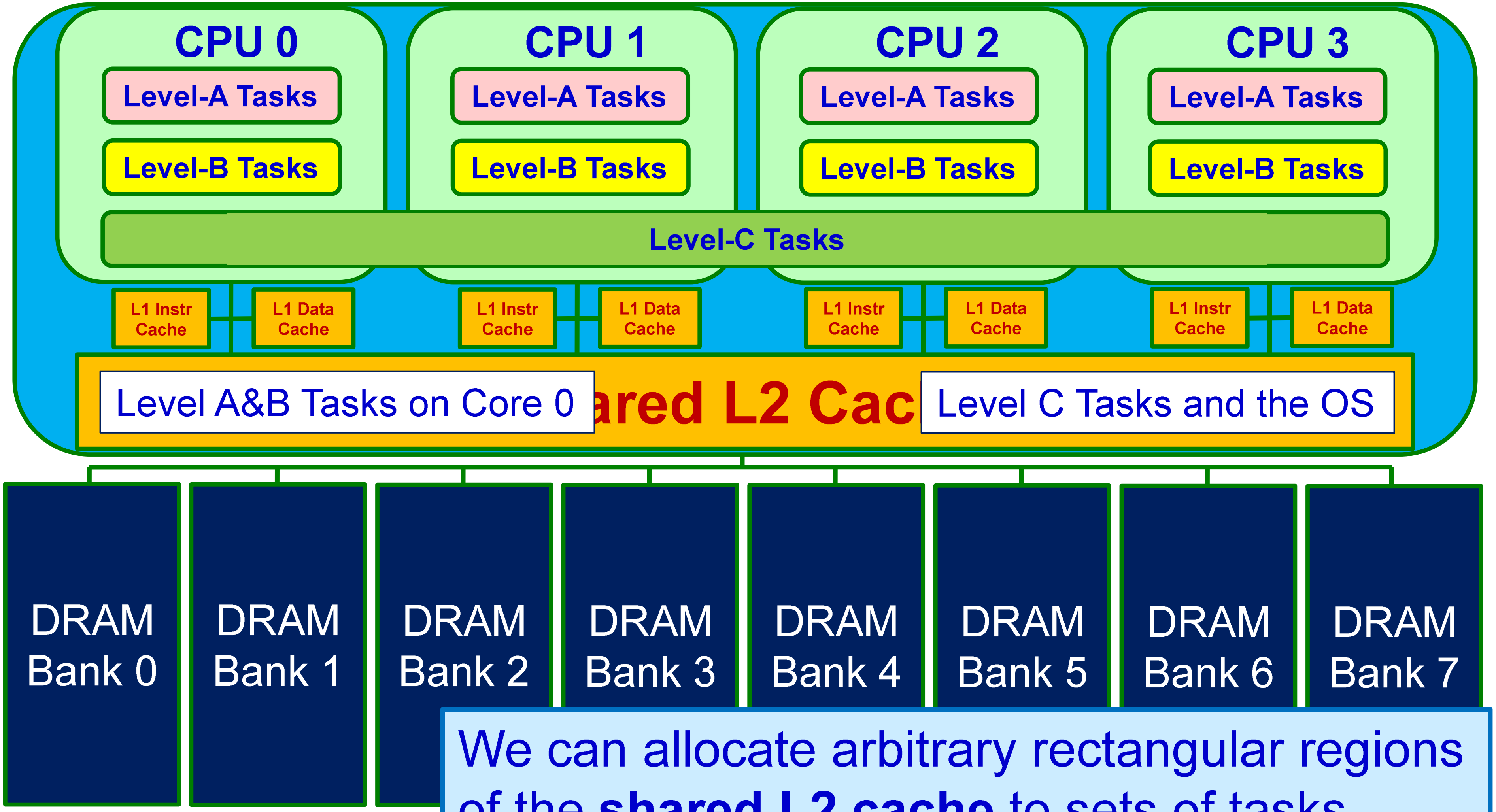
MC^2 on Our Quad-Core Test Platform



Level A&B Tasks on Core 0

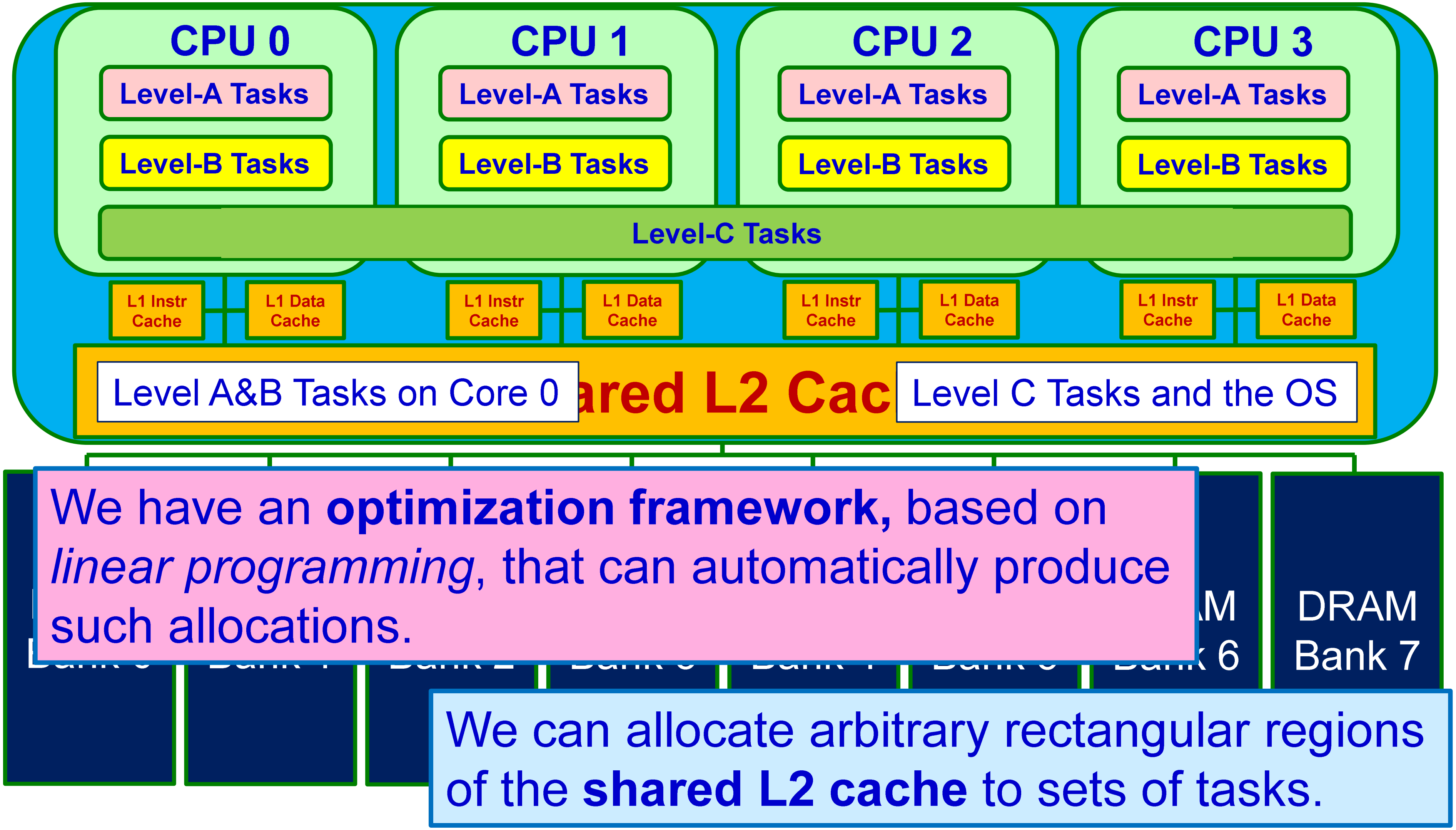
We can allocate arbitrary rectangular regions of the **shared L2 cache** to sets of tasks.

MC^2 on Our Quad-Core Test Platform



We can allocate arbitrary rectangular regions of the **shared L2 cache** to sets of tasks.

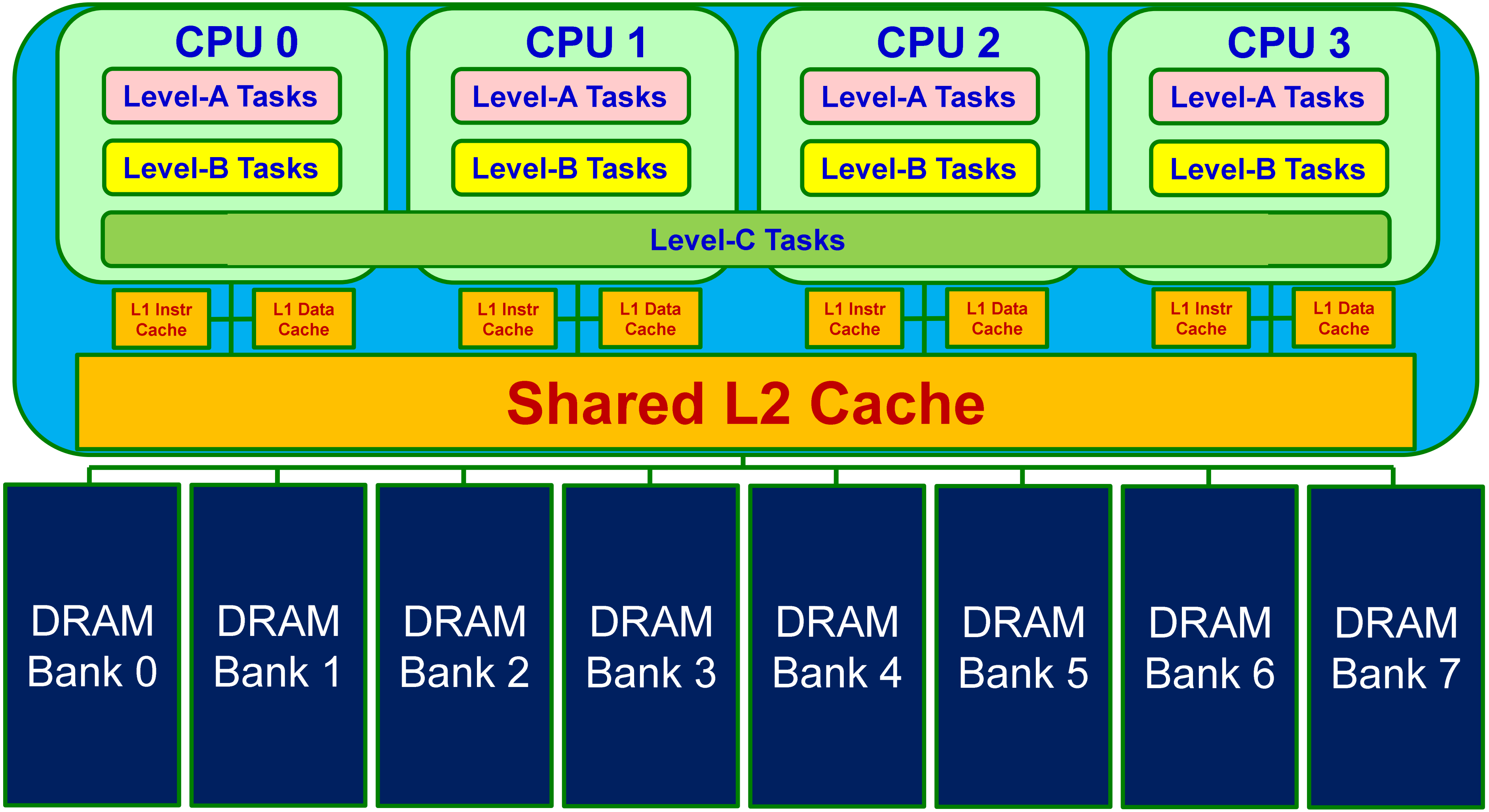
MC^2 on Our Quad-Core Test Platform



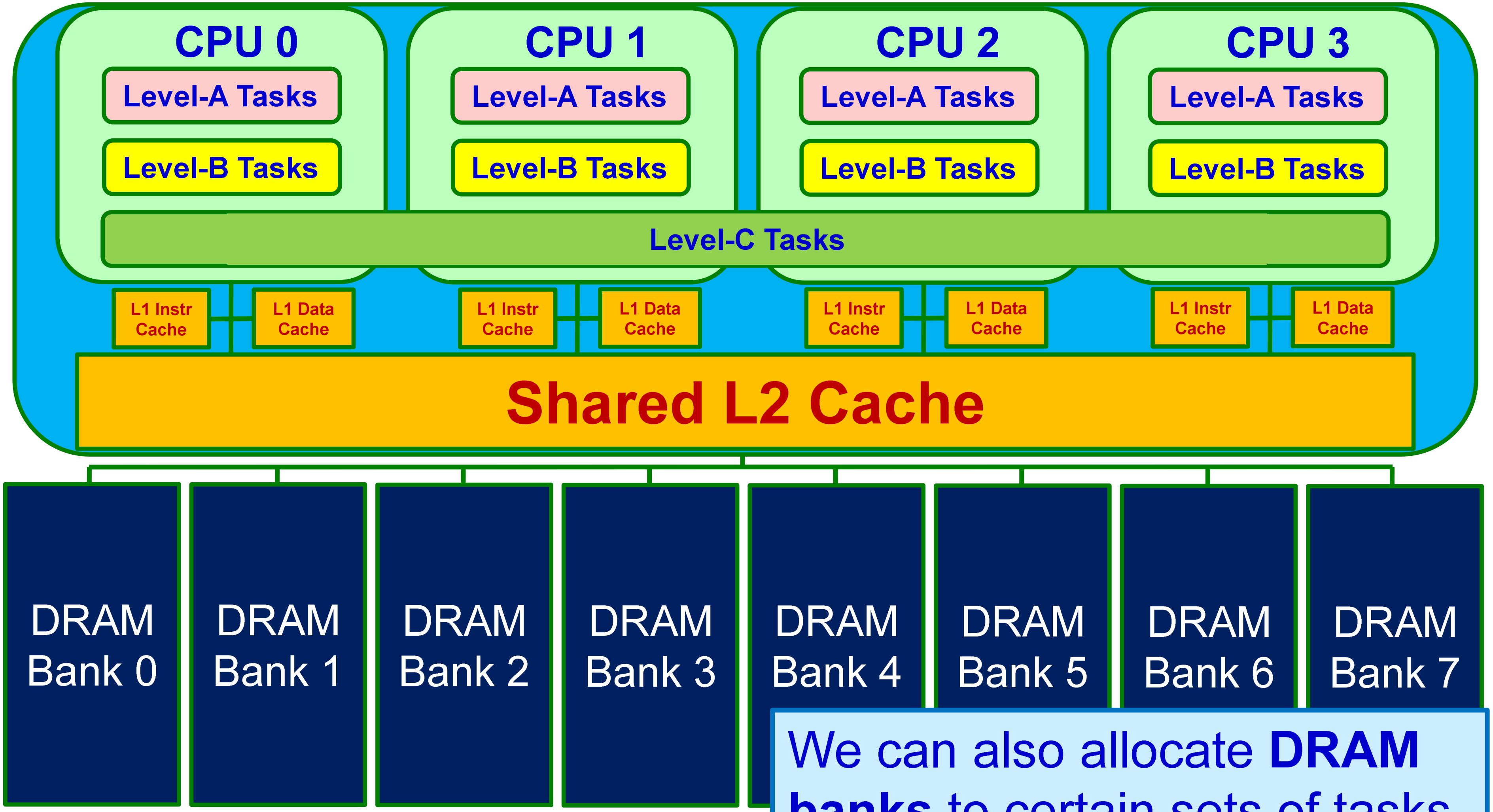
We have an **optimization framework,** based on *linear programming*, that can automatically produce such allocations.

We can allocate arbitrary rectangular regions of the **shared L2 cache** to sets of tasks.

MC^2 on Our Quad-Core Test Platform

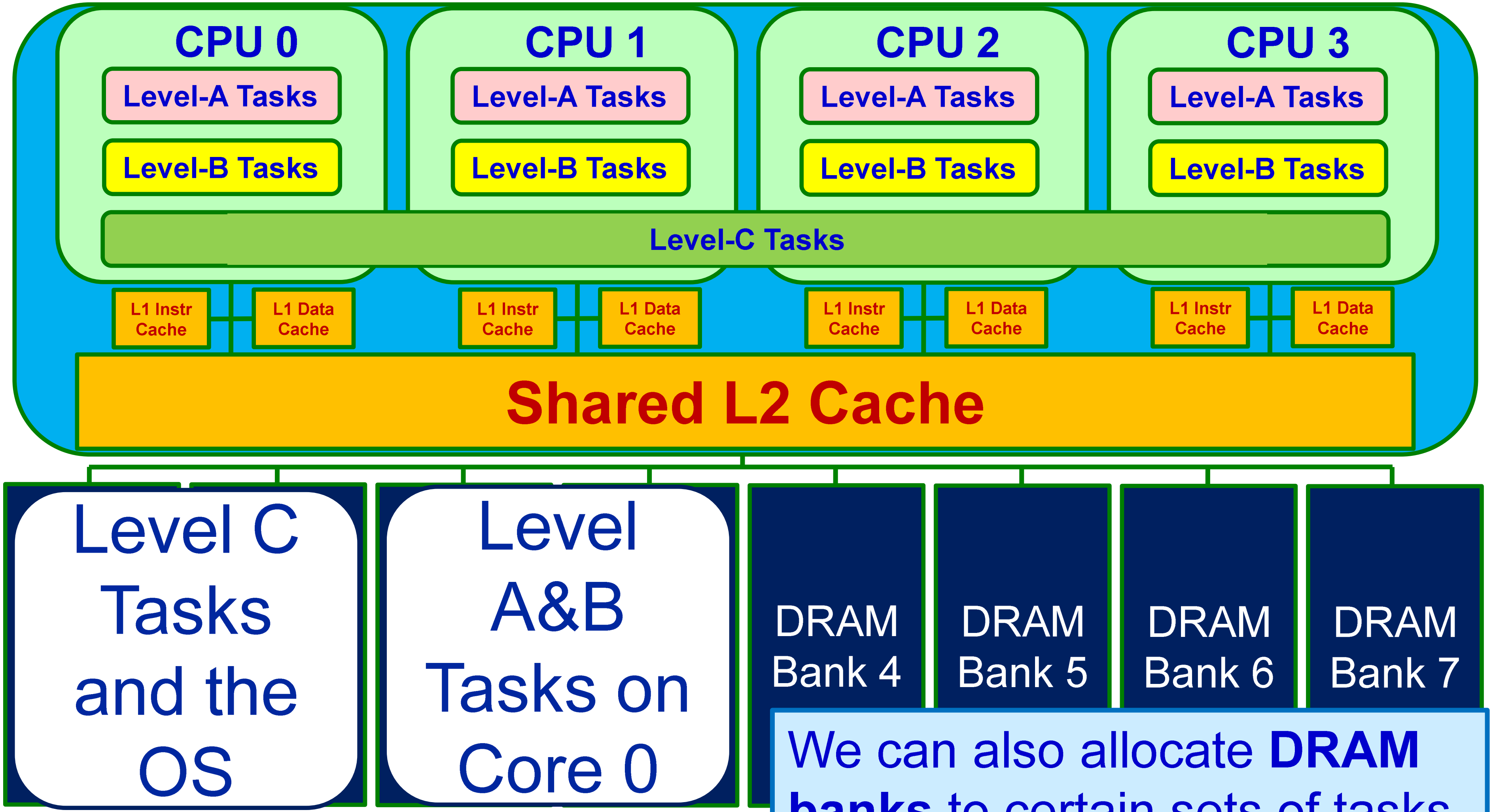


MC^2 on Our Quad-Core Test Platform



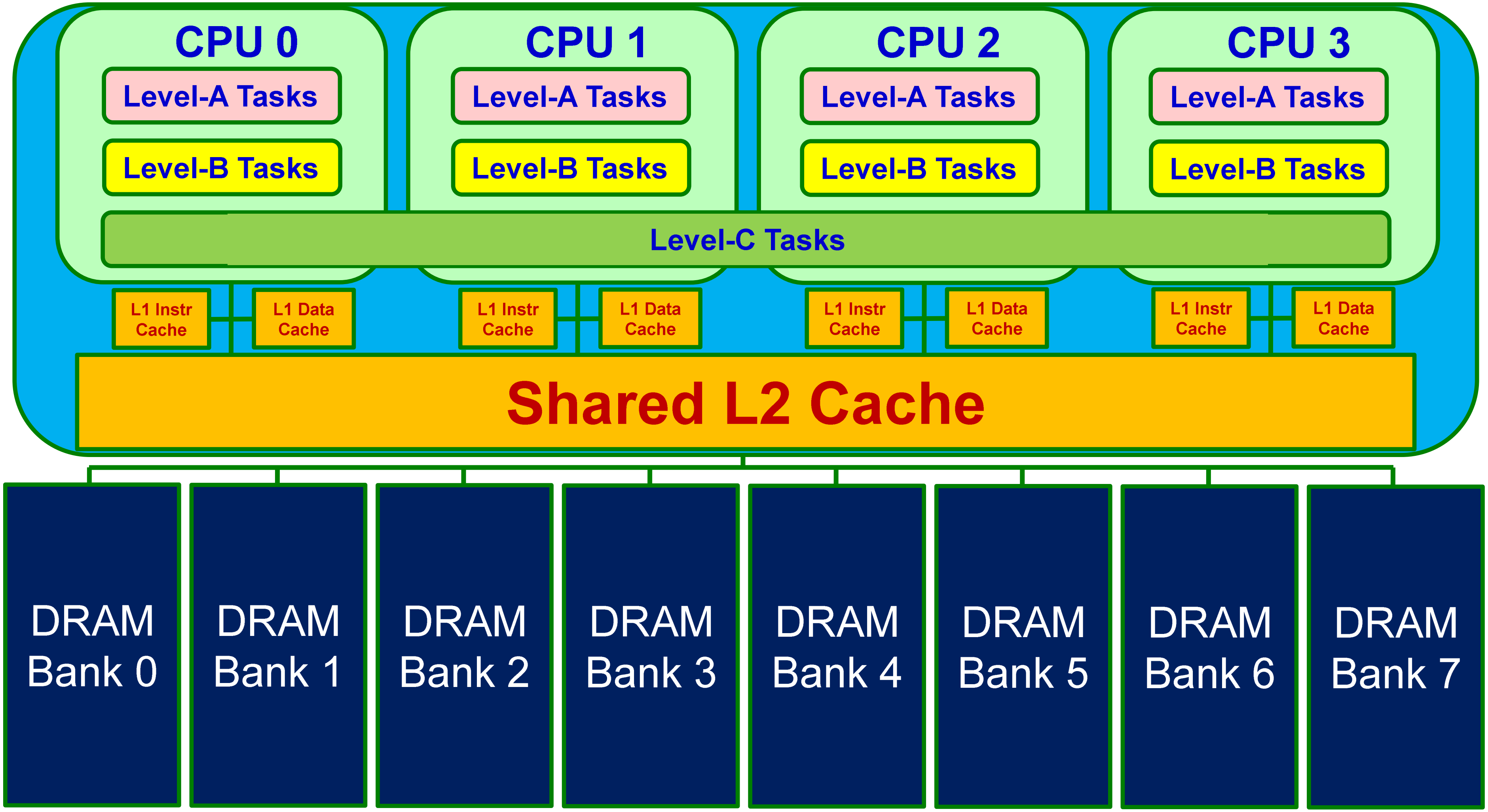
We can also allocate **DRAM banks** to certain sets of tasks.

MC^2 on Our Quad-Core Test Platform

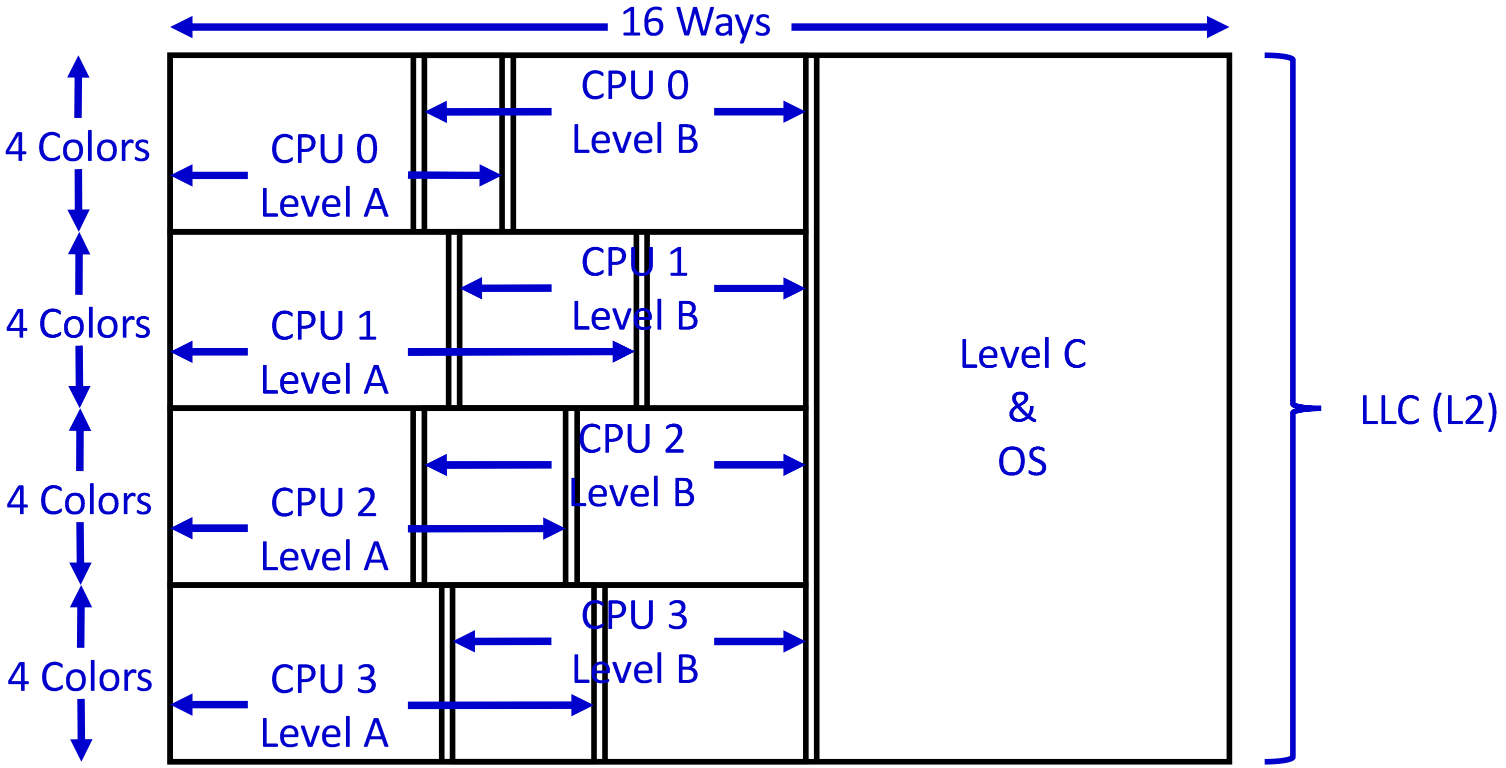


We can also allocate **DRAM banks** to certain sets of tasks.

MC^2 on Our Quad-Core Test Platform

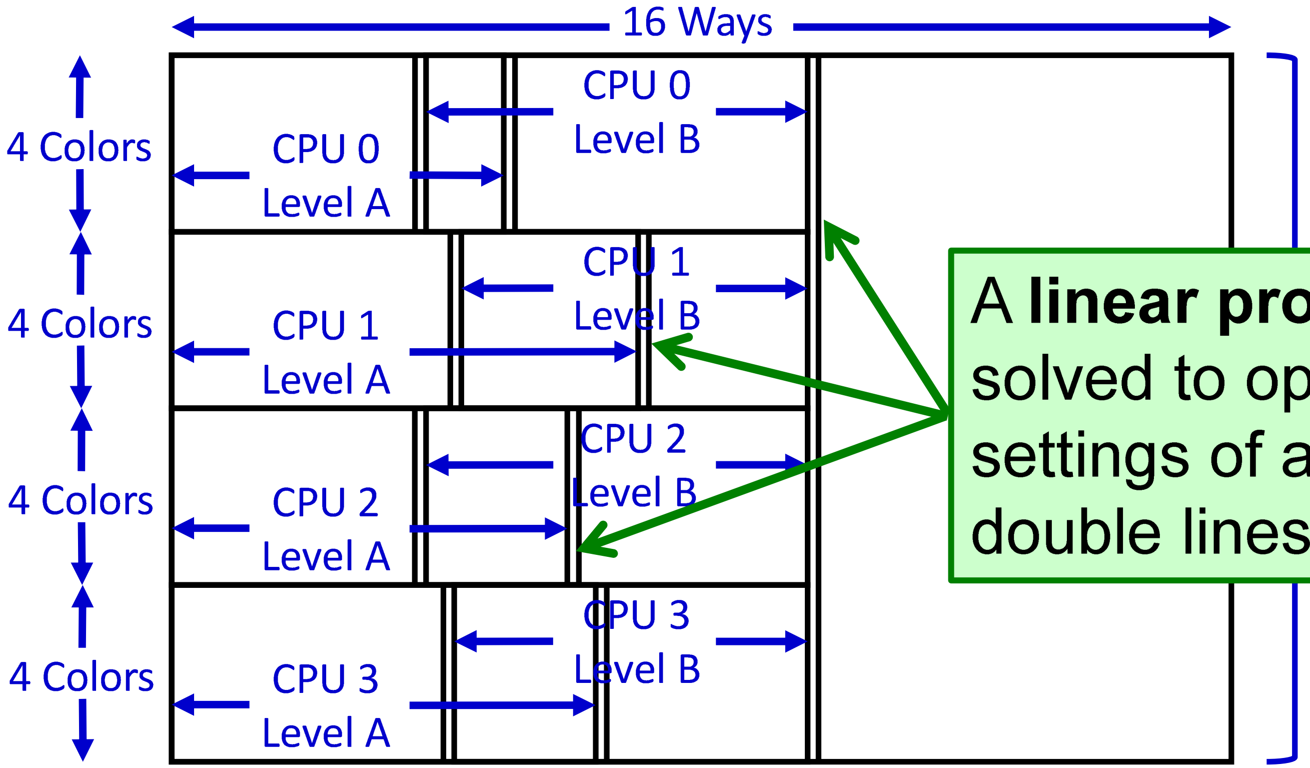


Our Actual Allocation Scheme



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

Our Actual Allocation Scheme



A **linear program** is solved to optimize the settings of all the double lines.

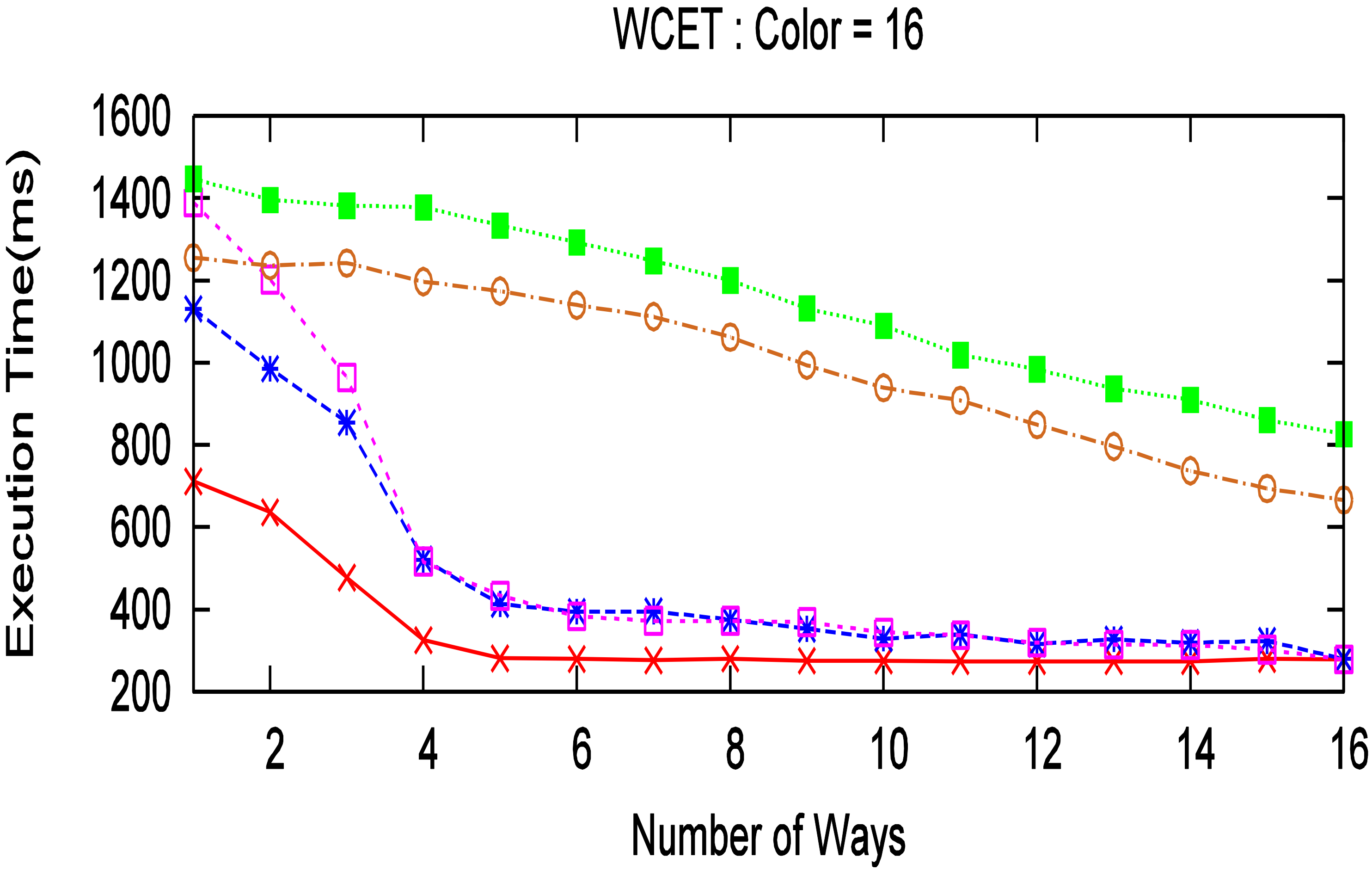
DRAM Bank 0 Level C & OS	DRAM Bank 1 Level C & OS	DRAM Bank 2 Level C & OS	DRAM Bank 3 CPU 0 A & B	DRAM Bank 4 CPU 1 A & B	DRAM Bank 5 CPU 2 A & B	DRAM Bank 6 CPU 3 A & B	DRAM Bank 7 Level C & OS

Experimental Evaluations

- We have assessed the value of hardware management w.r.t.
  - » individual tasks through experiments involving benchmark programs,
  - » entire task systems from a schedulability point of view.

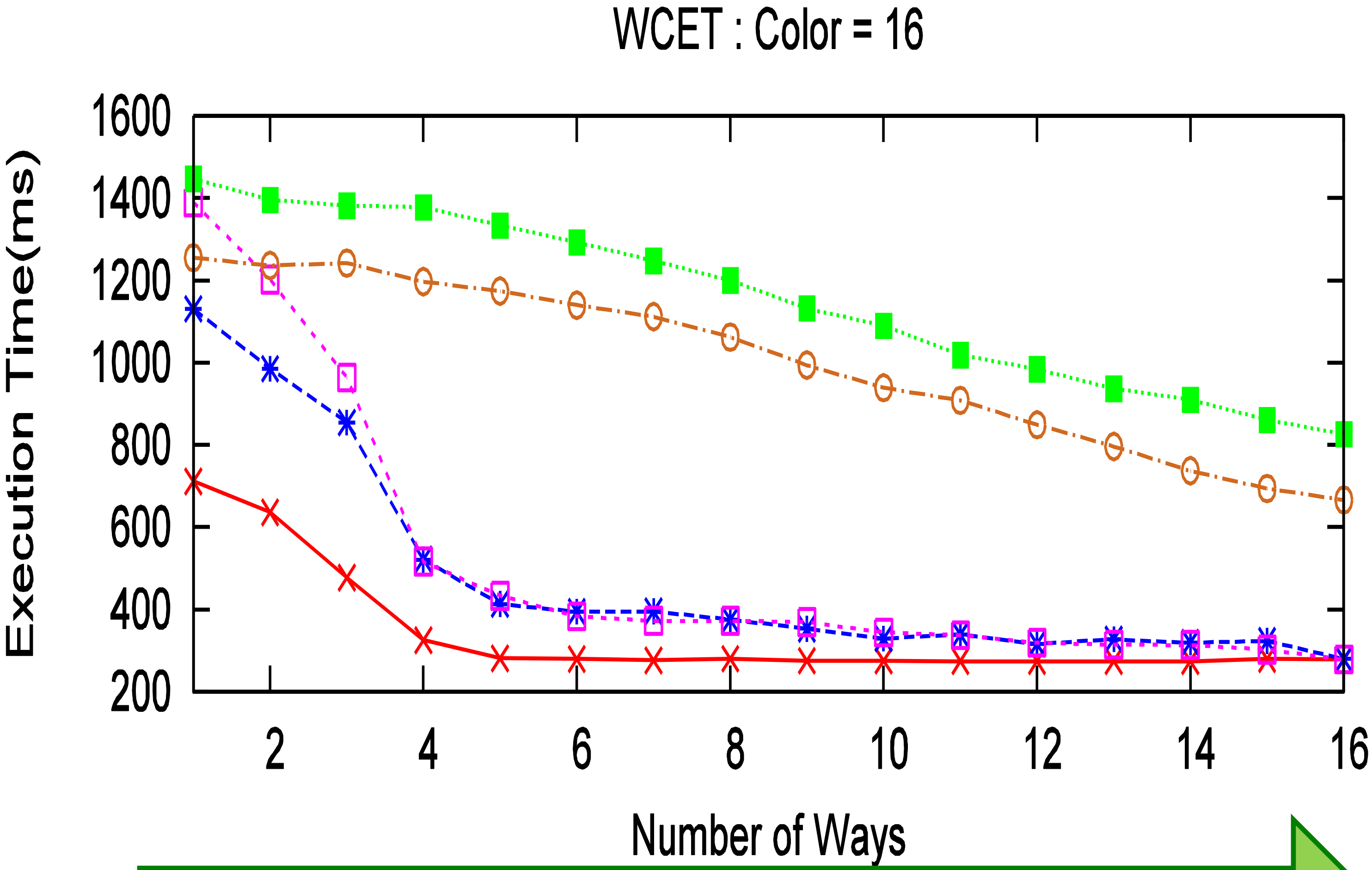
WCETs for a Benchmark Task

As a Function of Allocated LLC Area



WCETs for a Benchmark Task

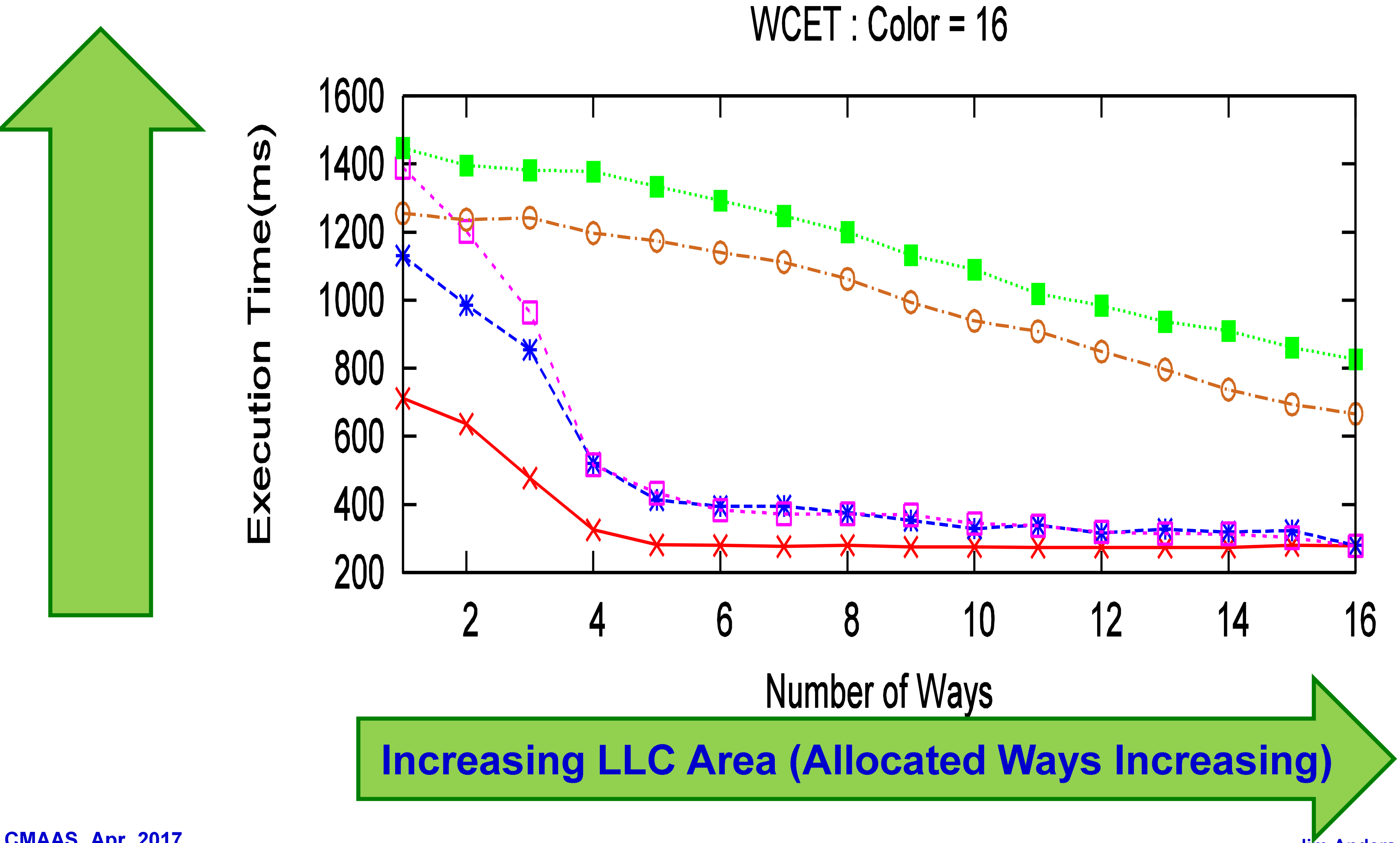
As a Function of Allocated LLC Area



Increasing LLC Area (Allocated Ways Increasing)

WCETs for a Benchmark Task

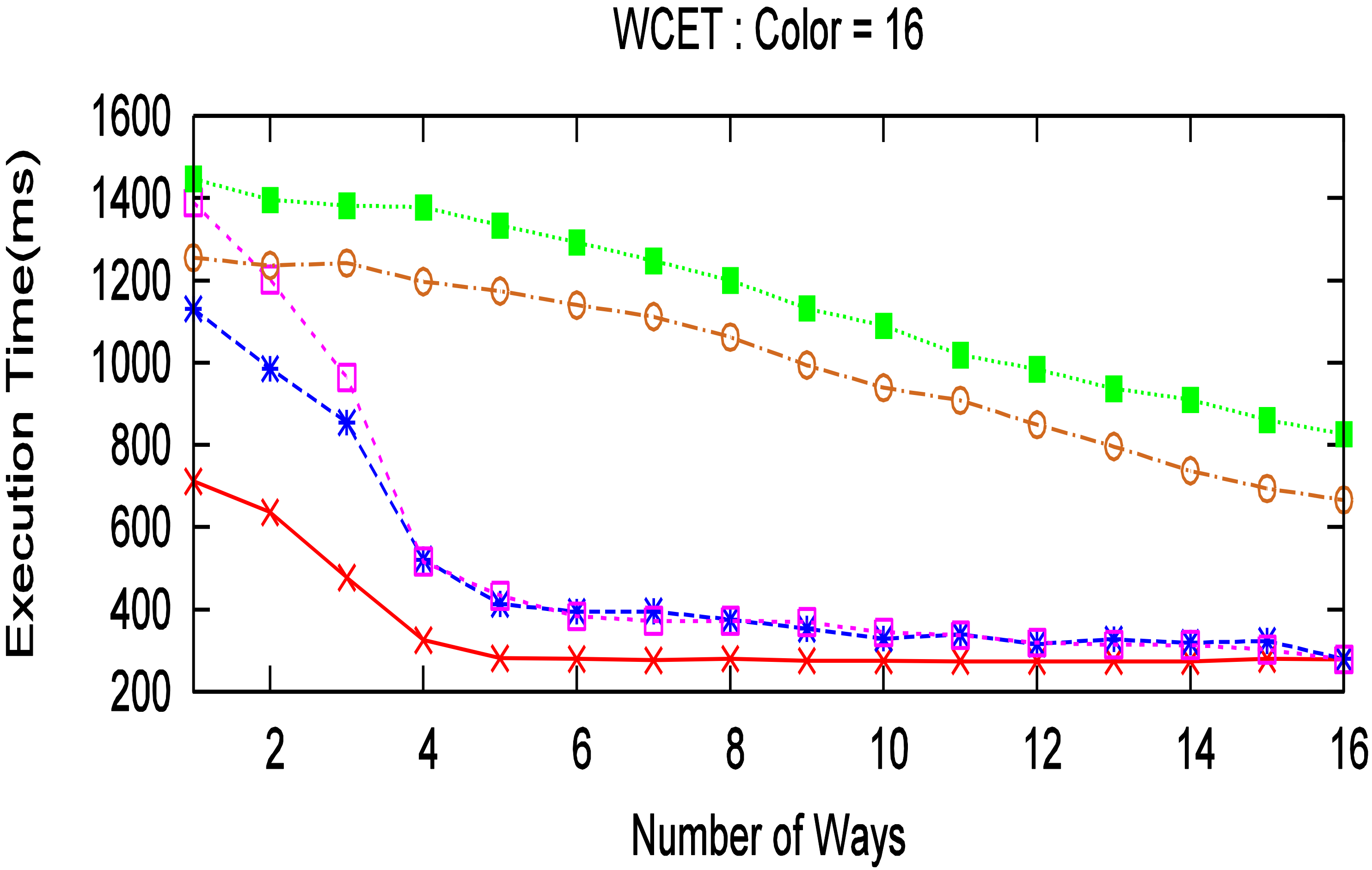
As a Function of Allocated LLC Area



Increasing LLC Area (Allocated Ways Increasing)

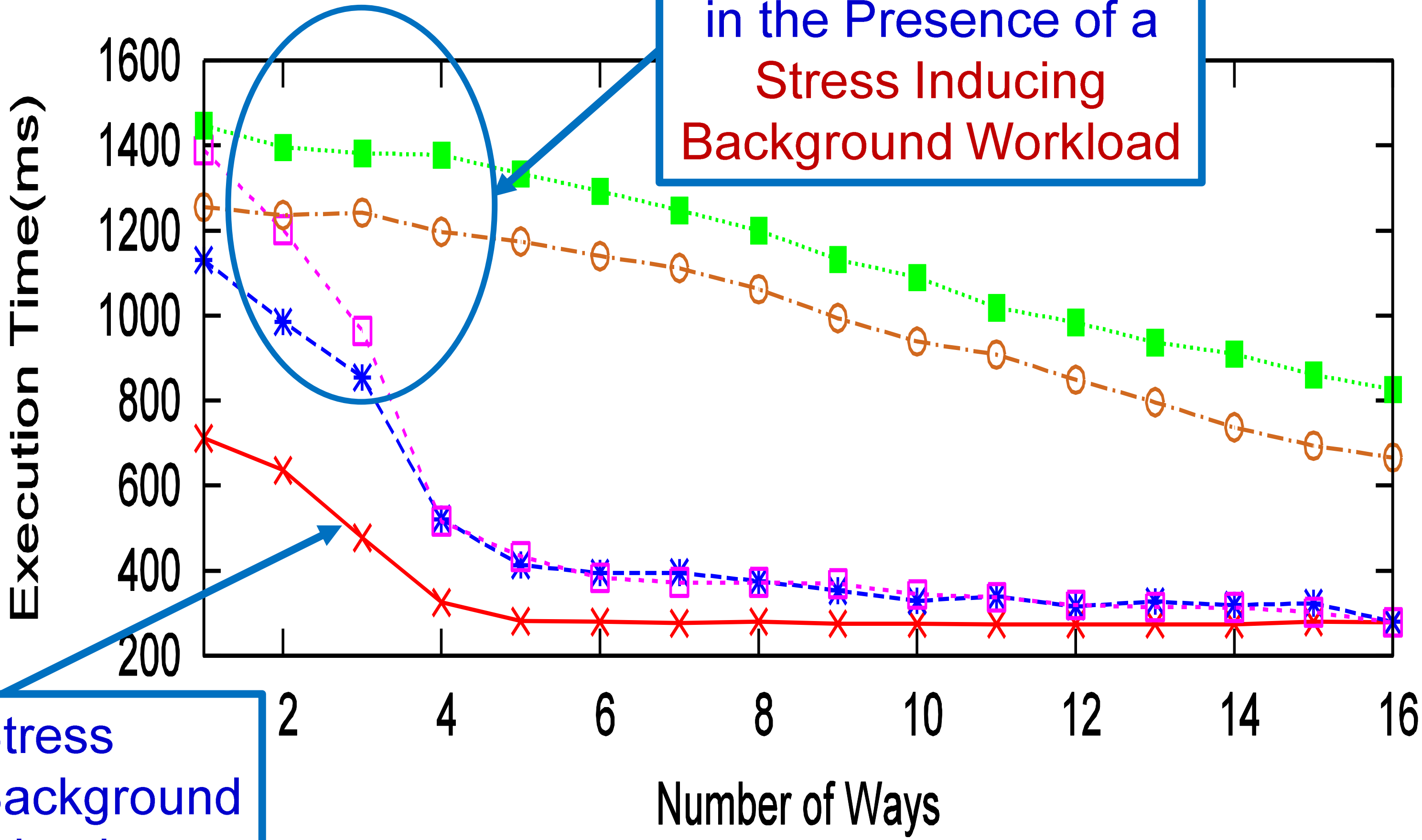
WCETs for a Benchmark Task

As a Function of Allocated LLC Area



WCETs for a Benchmark Task

As a Function of Allocated LLC Area

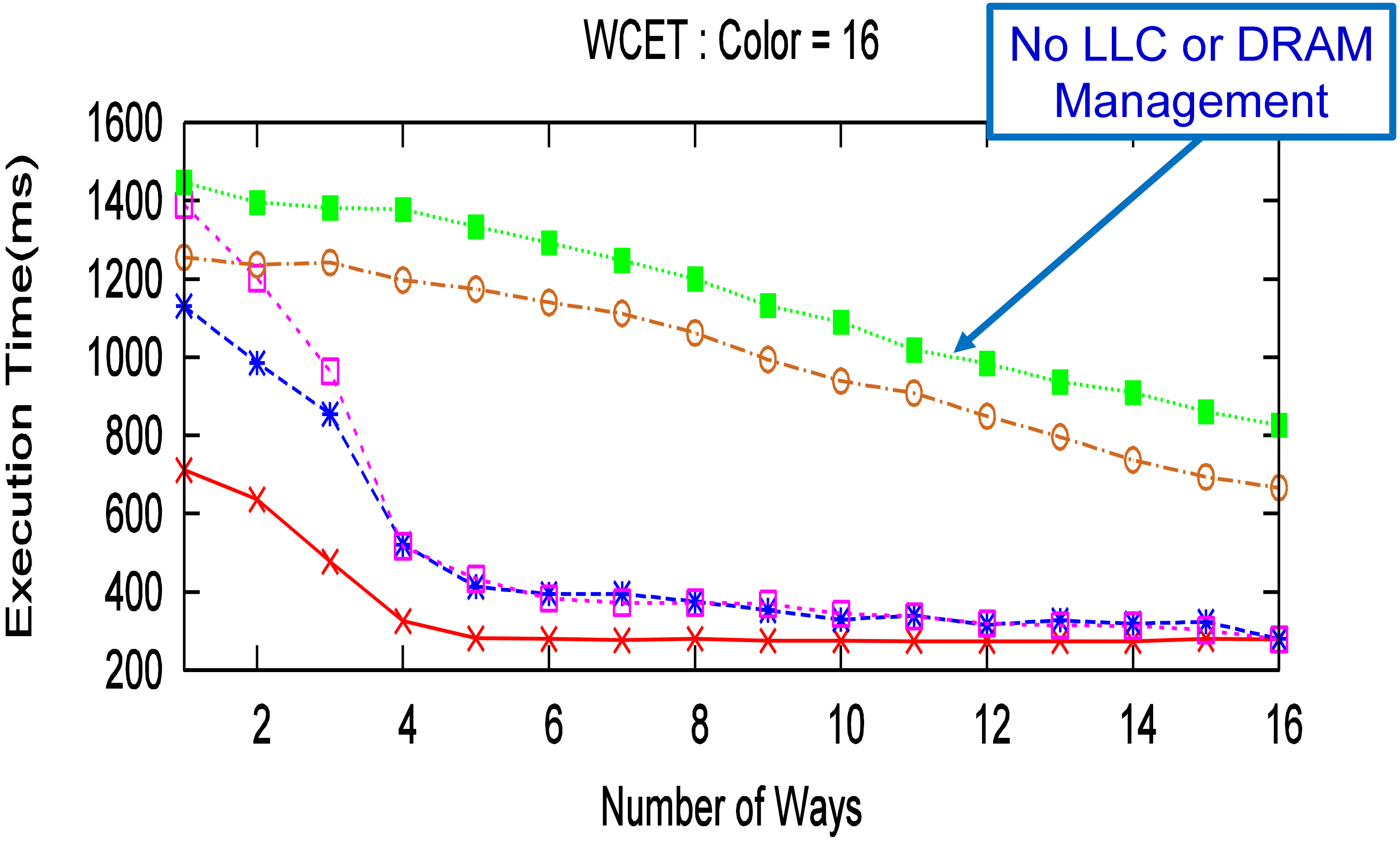


Measurements Taken in the Presence of a Stress Inducing Background Workload

No Stress Inducing Background Workload

WCETs for a Benchmark Task

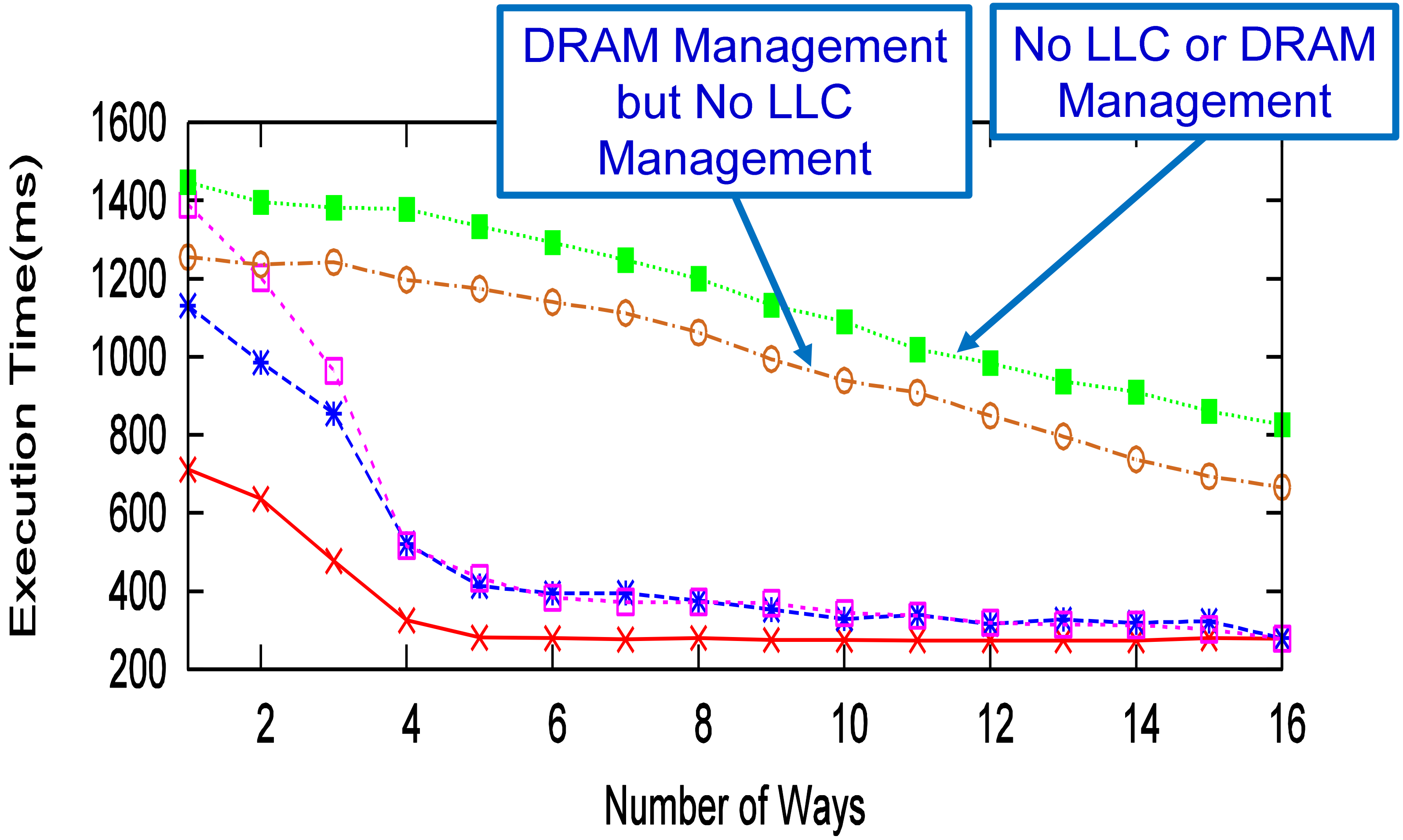
As a Function of Allocated LLC Area



No LLC or DRAM Management

WCETs for a Benchmark Task

As a Function of Allocated LLC Area

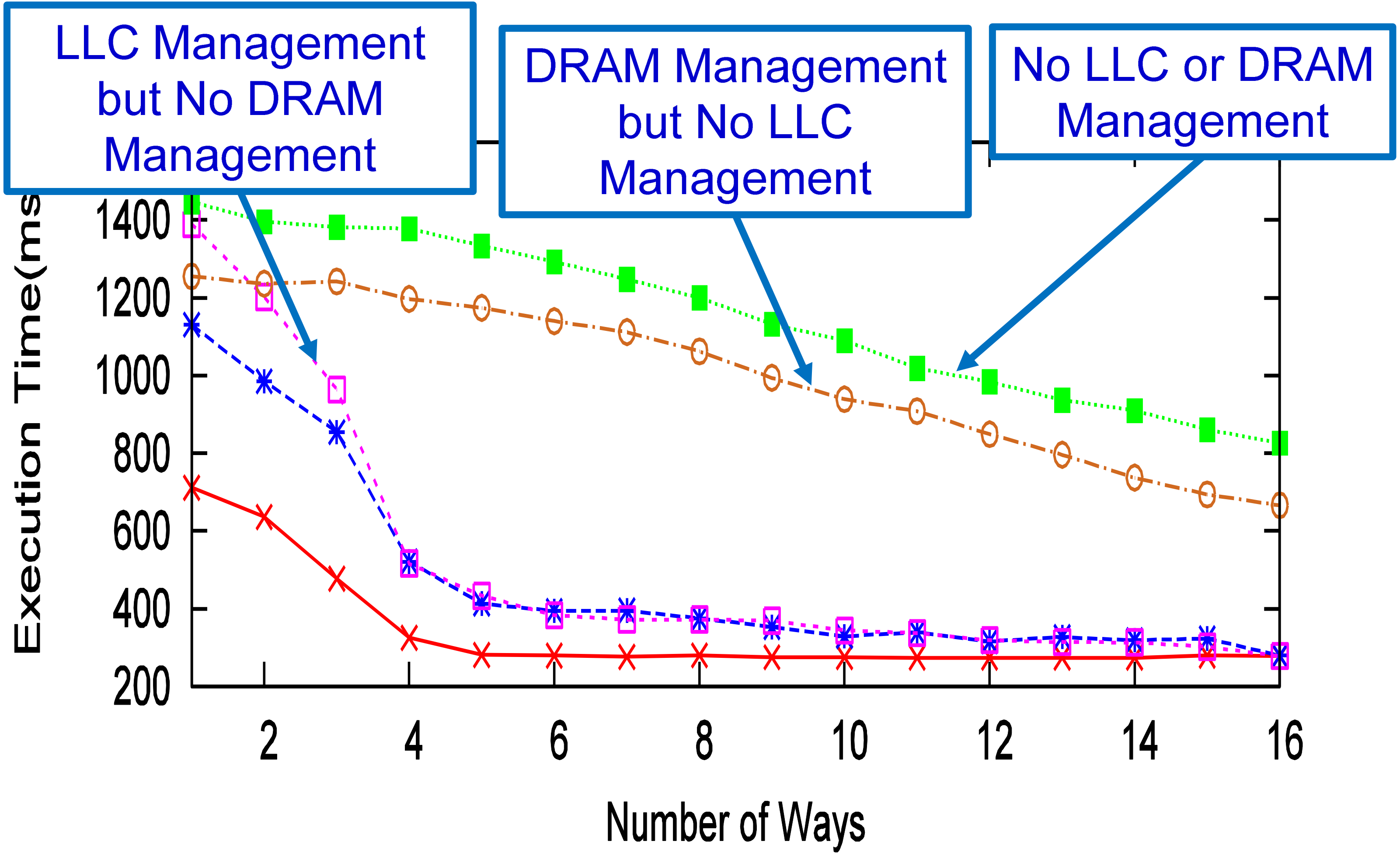


DRAM Management but No LLC Management

No LLC or DRAM Management

WCETs for a Benchmark Task

As a Function of Allocated LLC Area



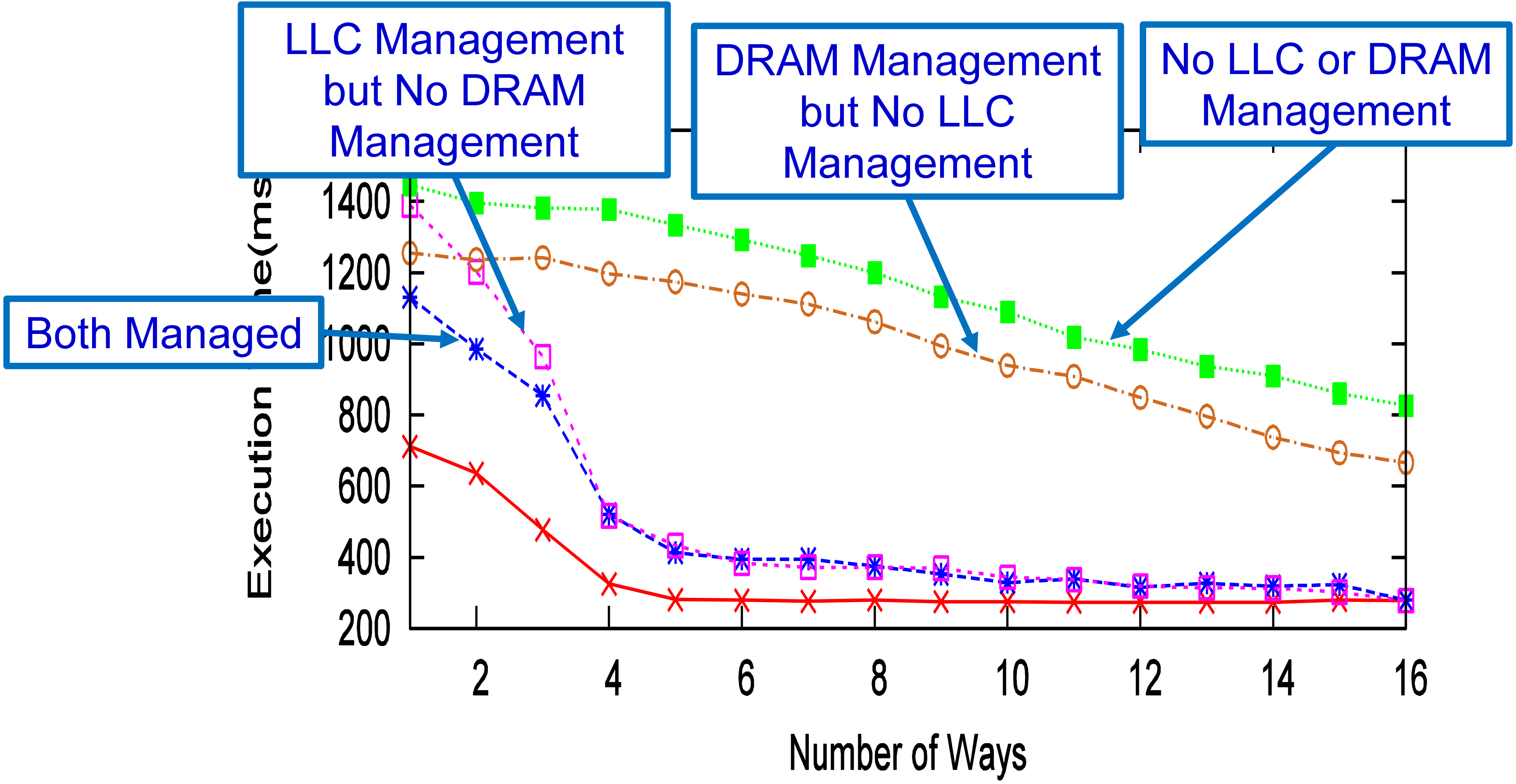
LLC Management but No DRAM Management

DRAM Management but No LLC Management

No LLC or DRAM Management

WCETs for a Benchmark Task

As a Function of Allocated LLC Area



LLC Management but No DRAM Management

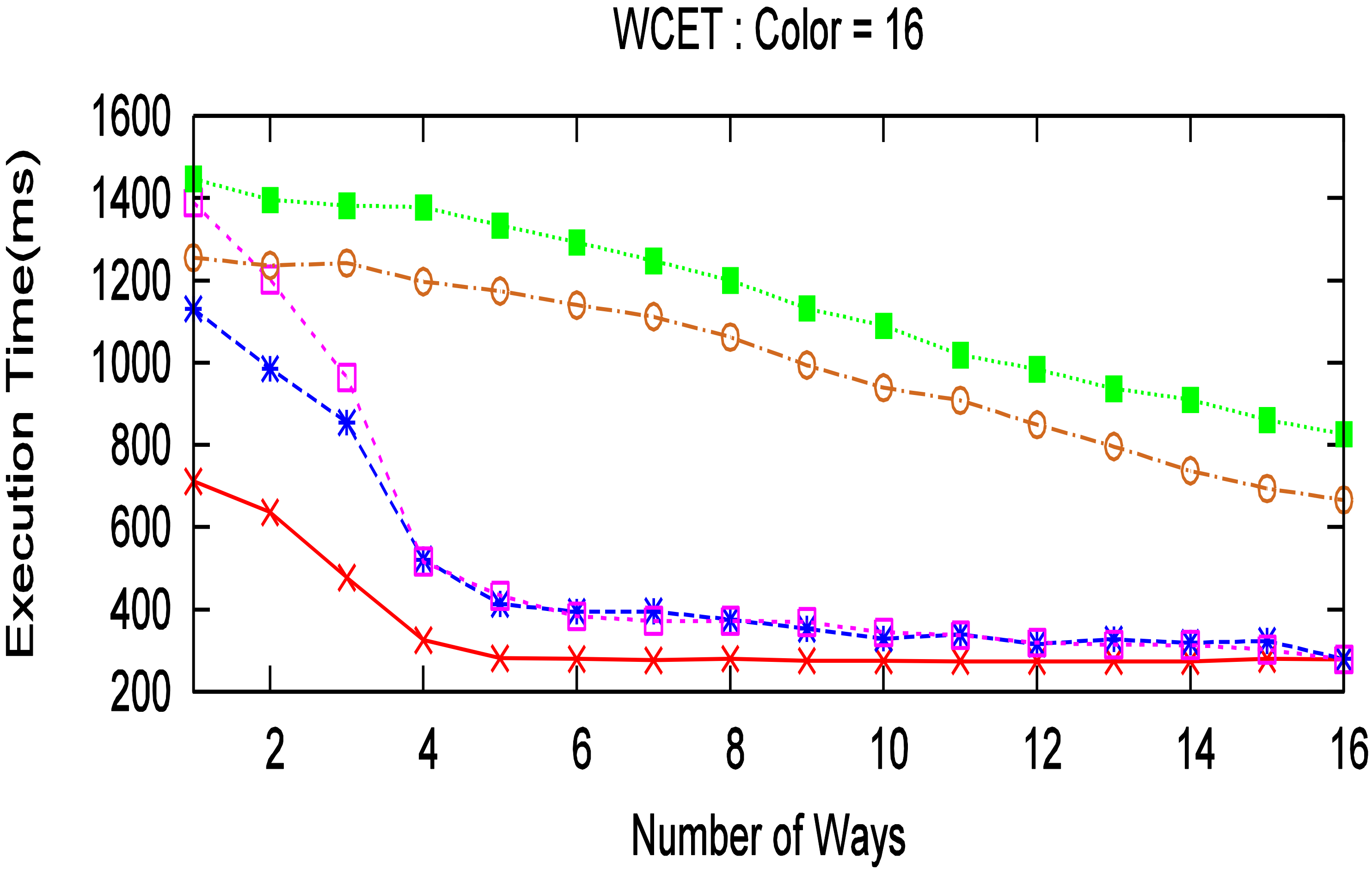
DRAM Management but No LLC Management

No LLC or DRAM Management

Both Managed

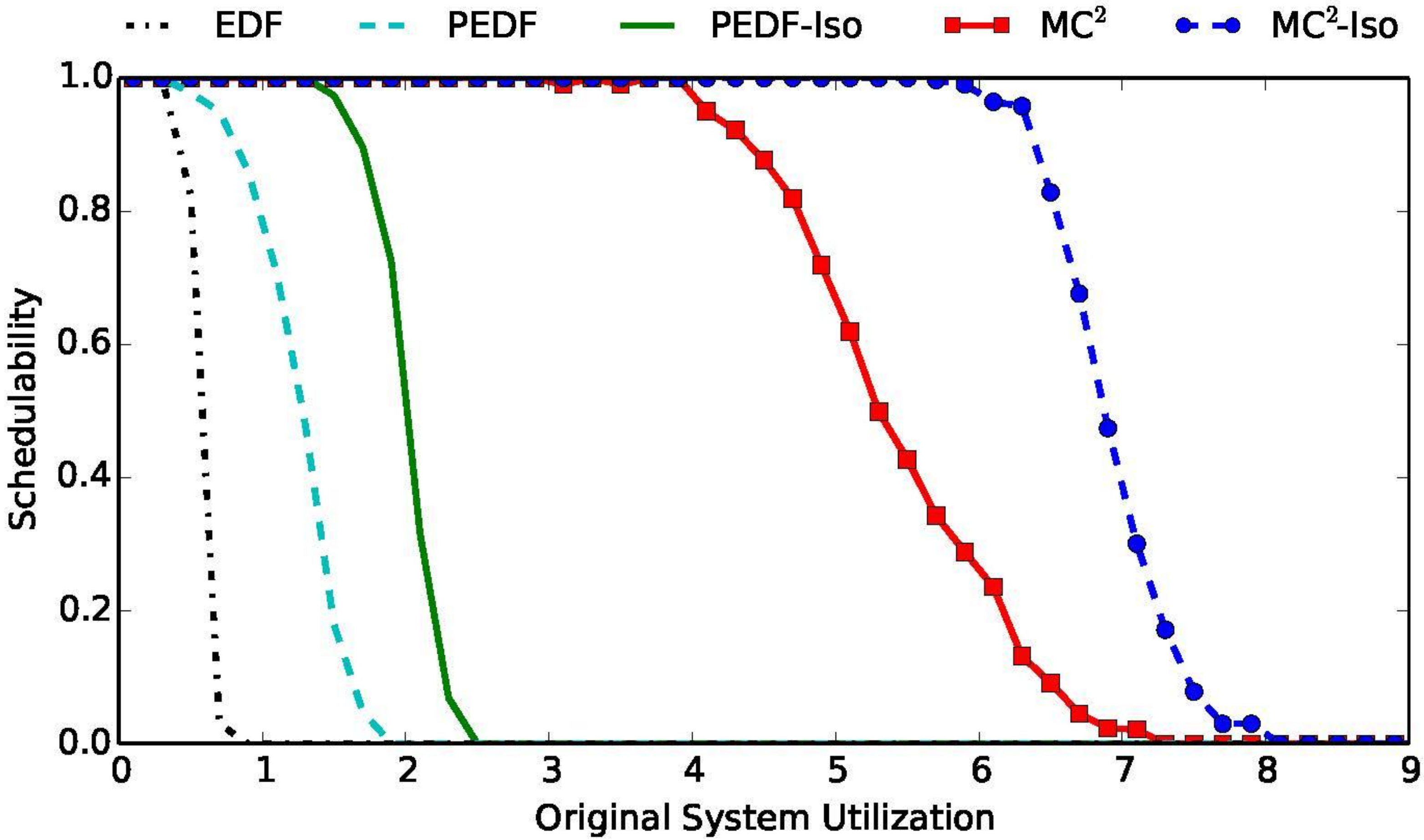
WCETs for a Benchmark Task

As a Function of Allocated LLC Area



Overhead-Aware Schedulability Study

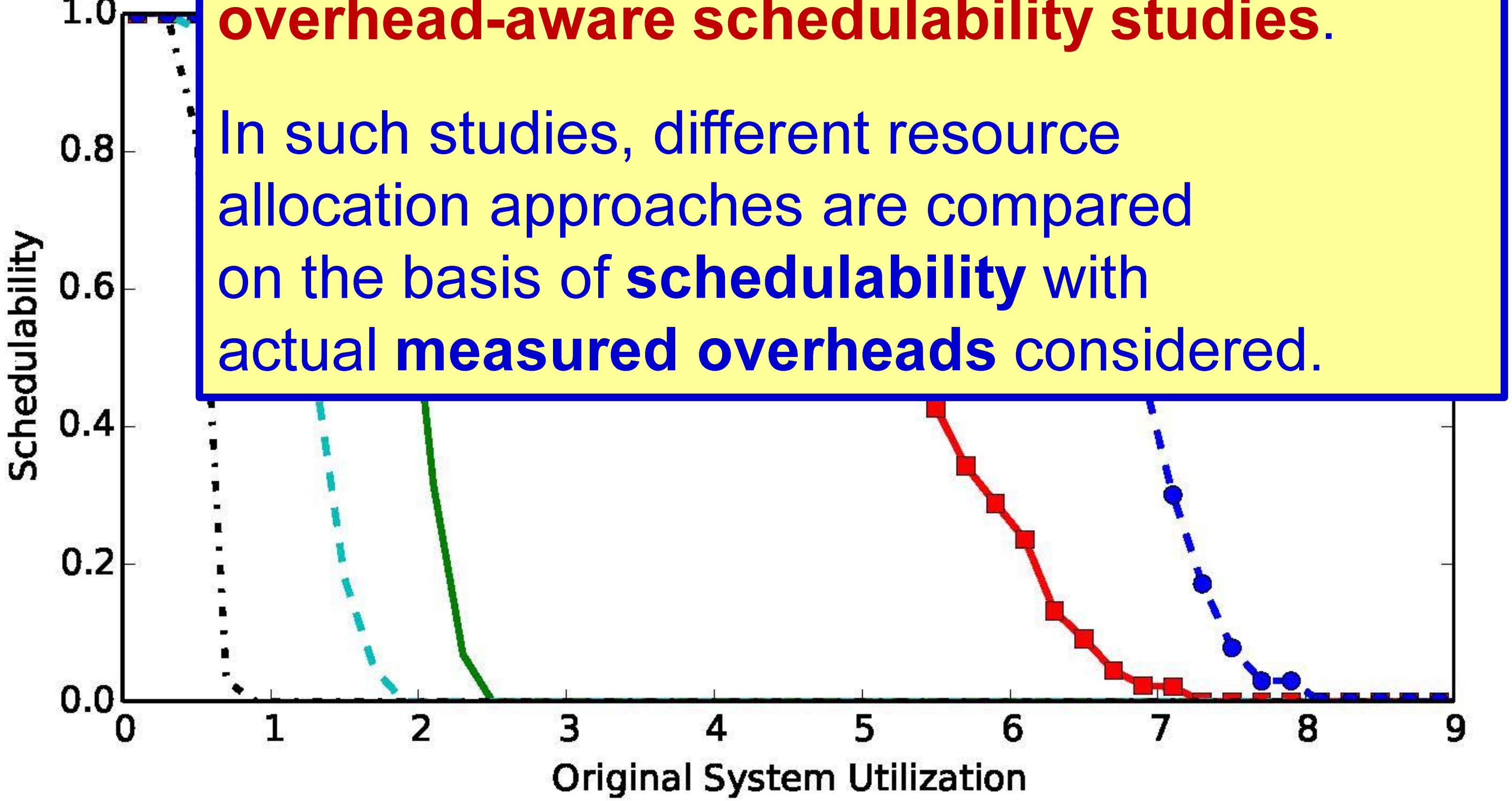
This is One Out of About 500 Graphs



Overhead-Aware Schedulability Study

This is One Out of About 500 Graphs

…

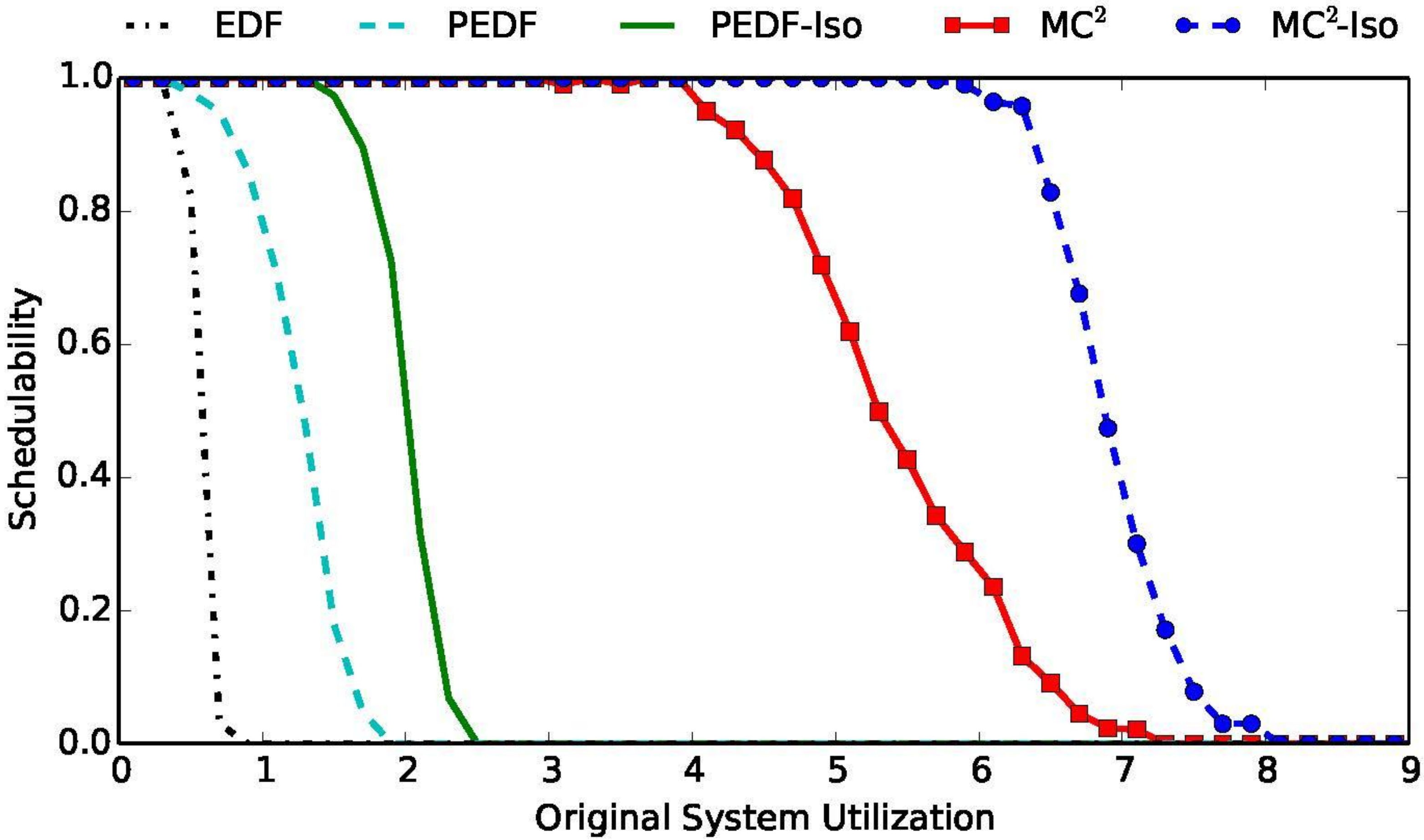


We assess system-wide impacts by conducting **overhead-aware schedulability studies**.

In such studies, different resource allocation approaches are compared on the basis of **schedulability** with actual **measured overheads** considered.

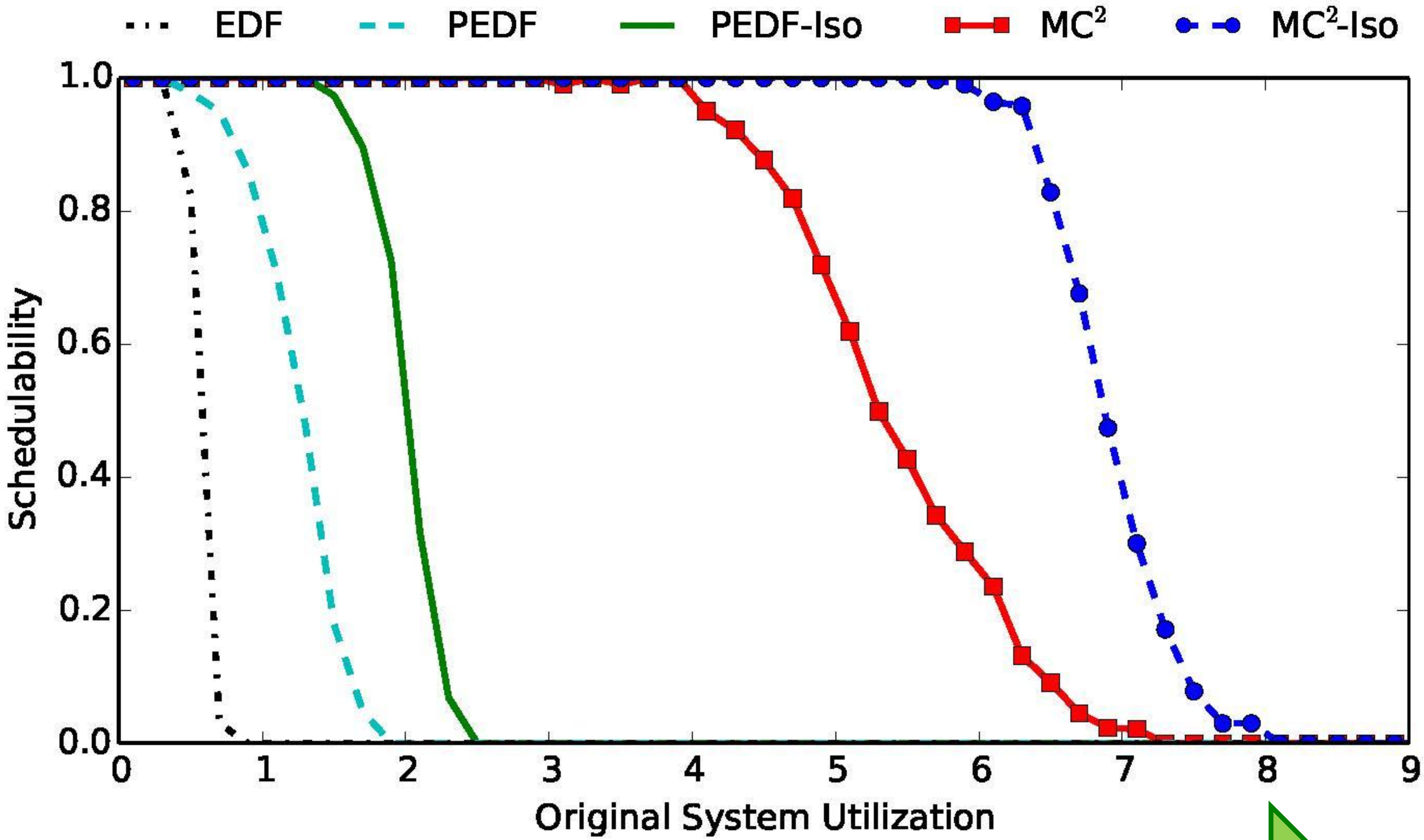
Overhead-Aware Schedulability Study

This is One Out of About 500 Graphs



Overhead-Aware Schedulability Study

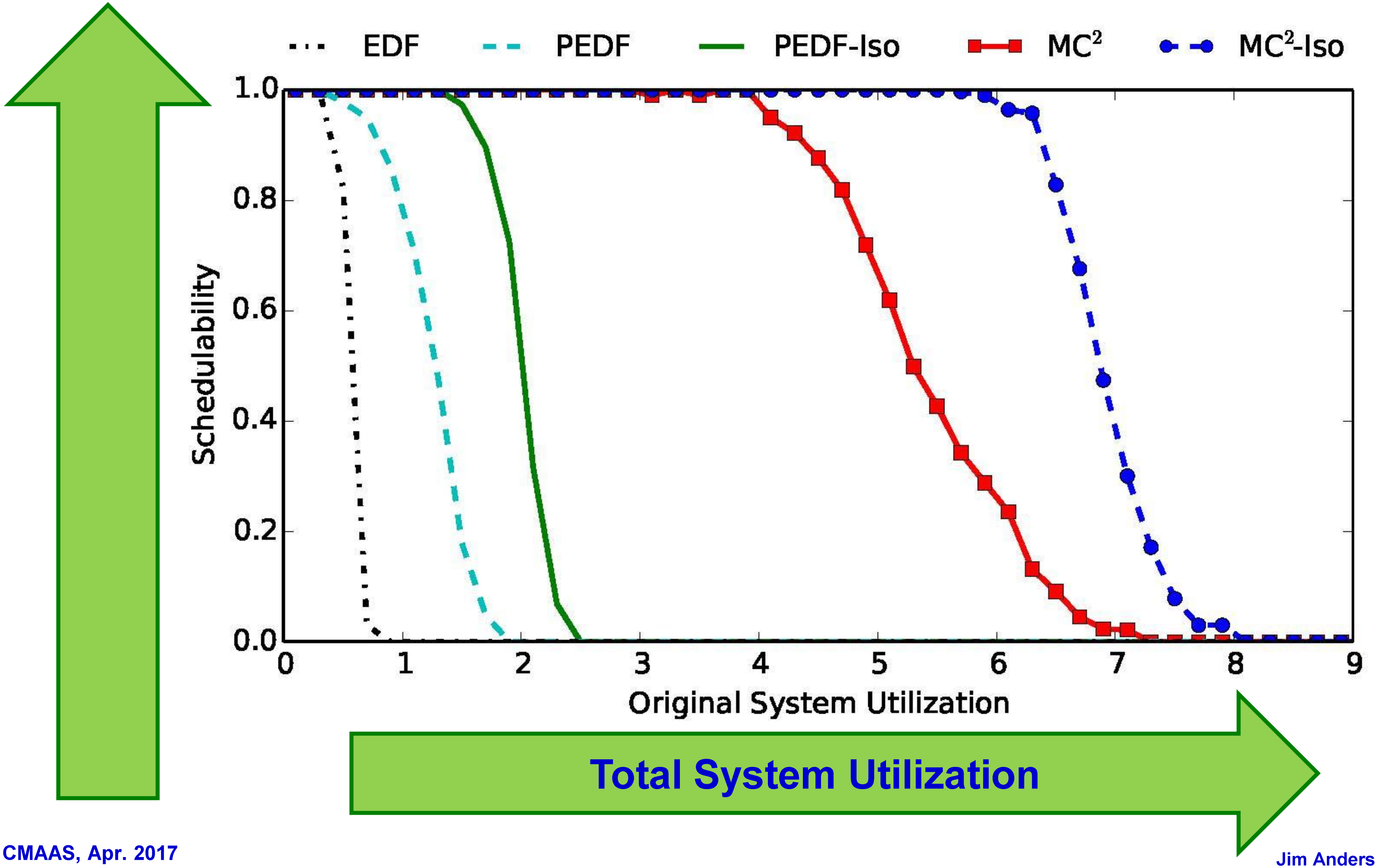
This is One Out of About 500 Graphs



Total System Utilization

Overhead-Aware Schedulability Study

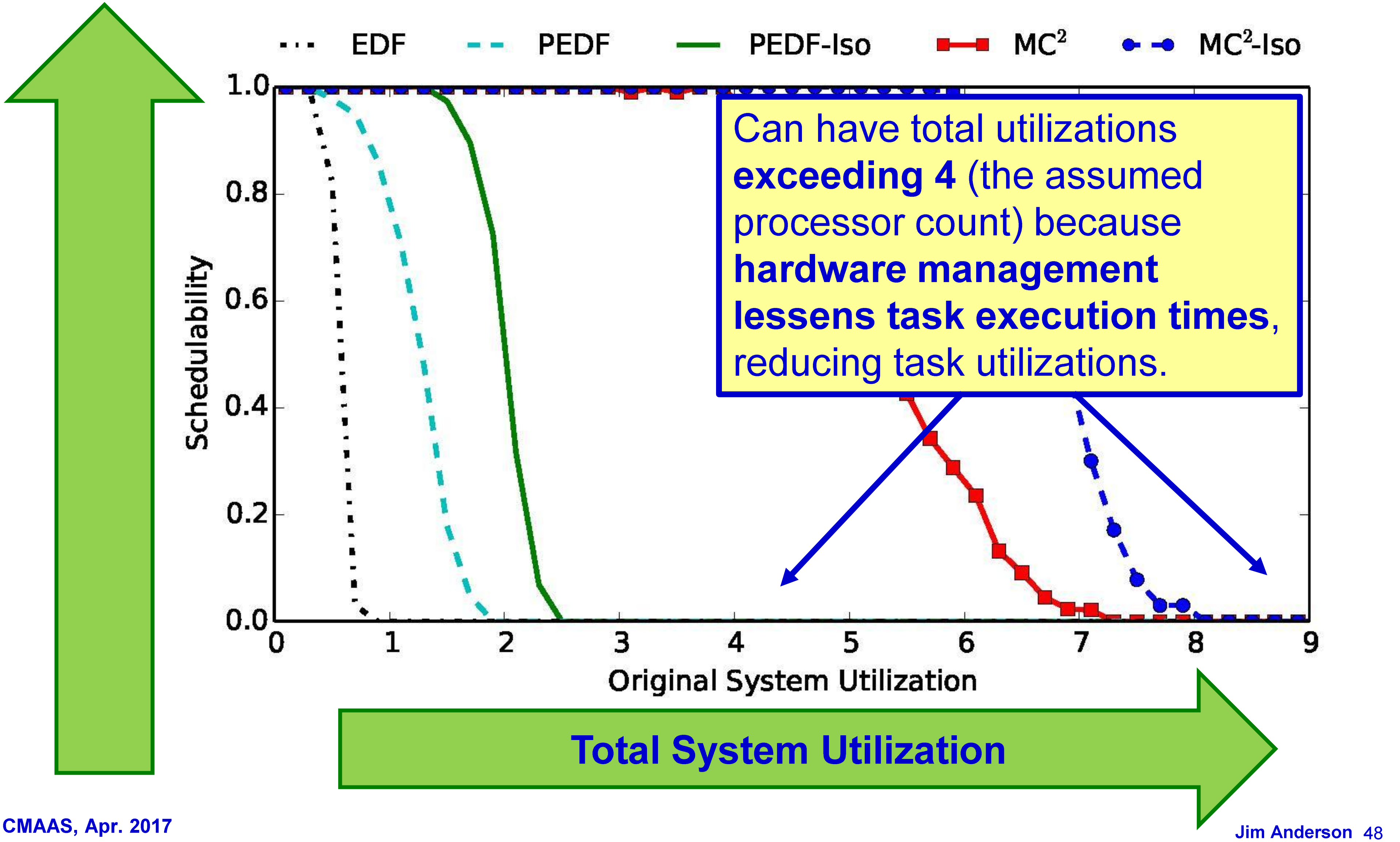
This is One Out of About 500 Graphs



Total System Utilization

Overhead-Aware Schedulability Study

This is One Out of About 500 Graphs

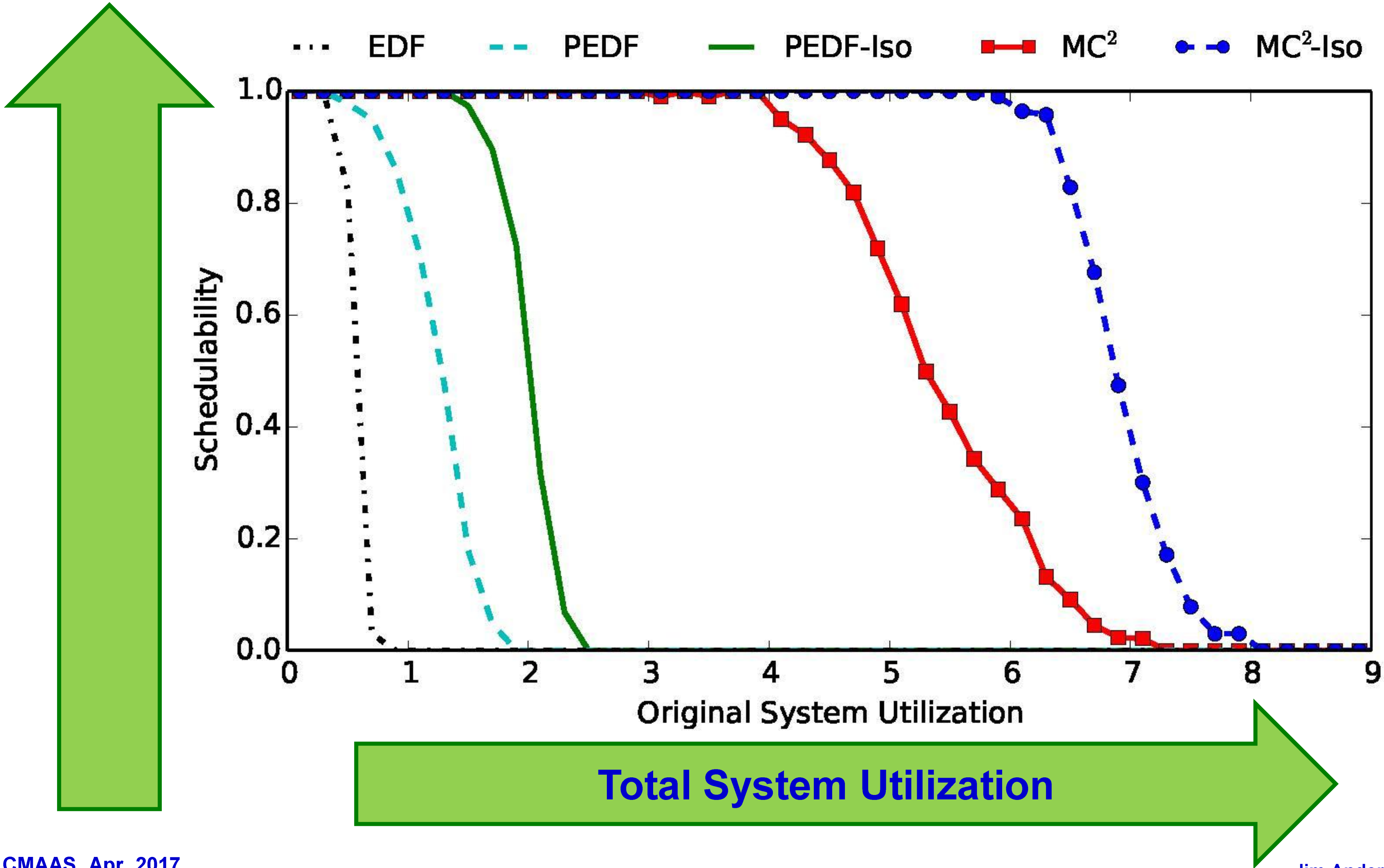


Can have total utilizations **exceeding 4** (the assumed processor count) because **hardware management lessens task execution times**, reducing task utilizations.

Total System Utilization

Overhead-Aware Schedulability Study

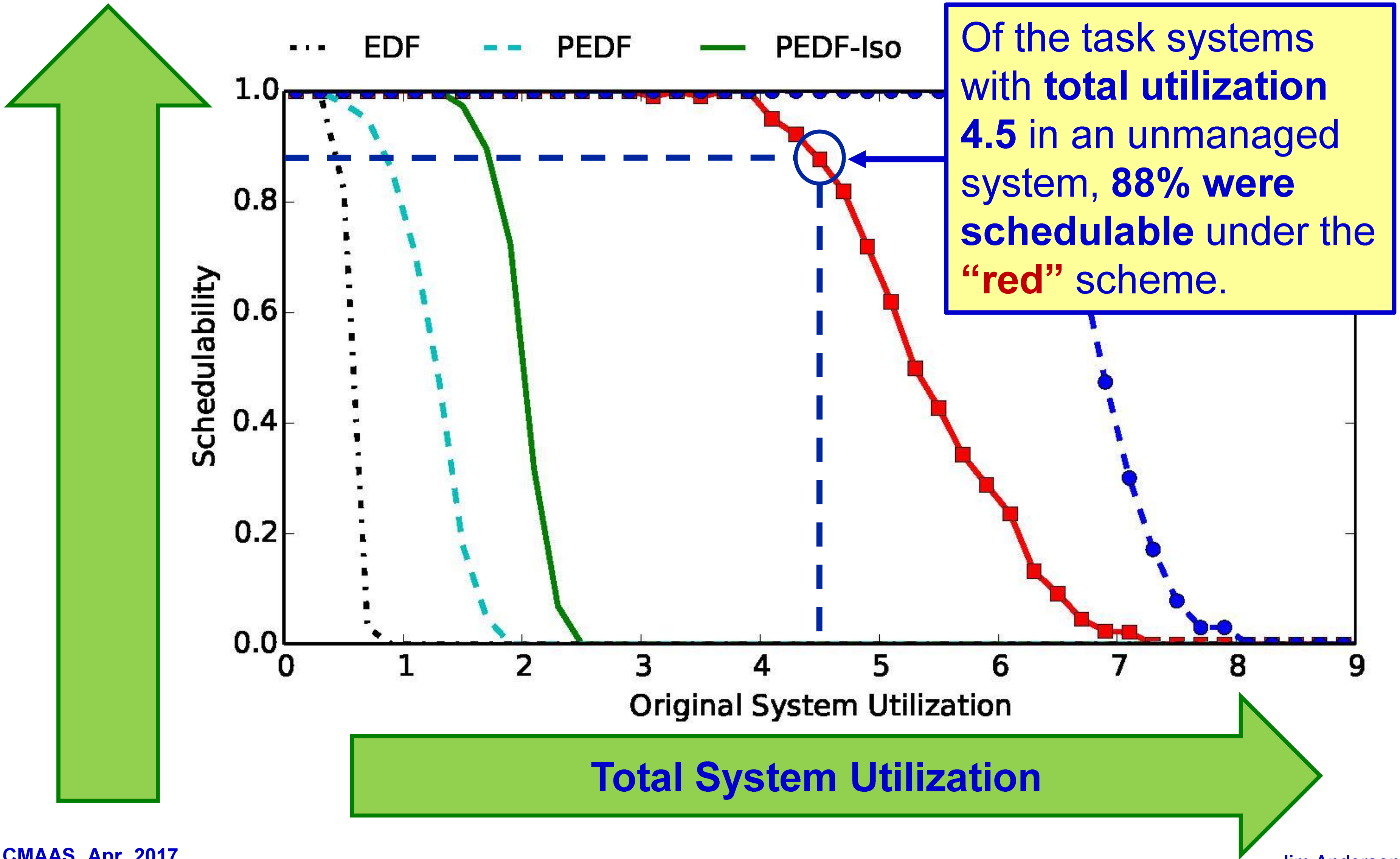
This is One Out of About 500 Graphs



Total System Utilization

Overhead-Aware Schedulability Study

This is One Out of About 500 Graphs

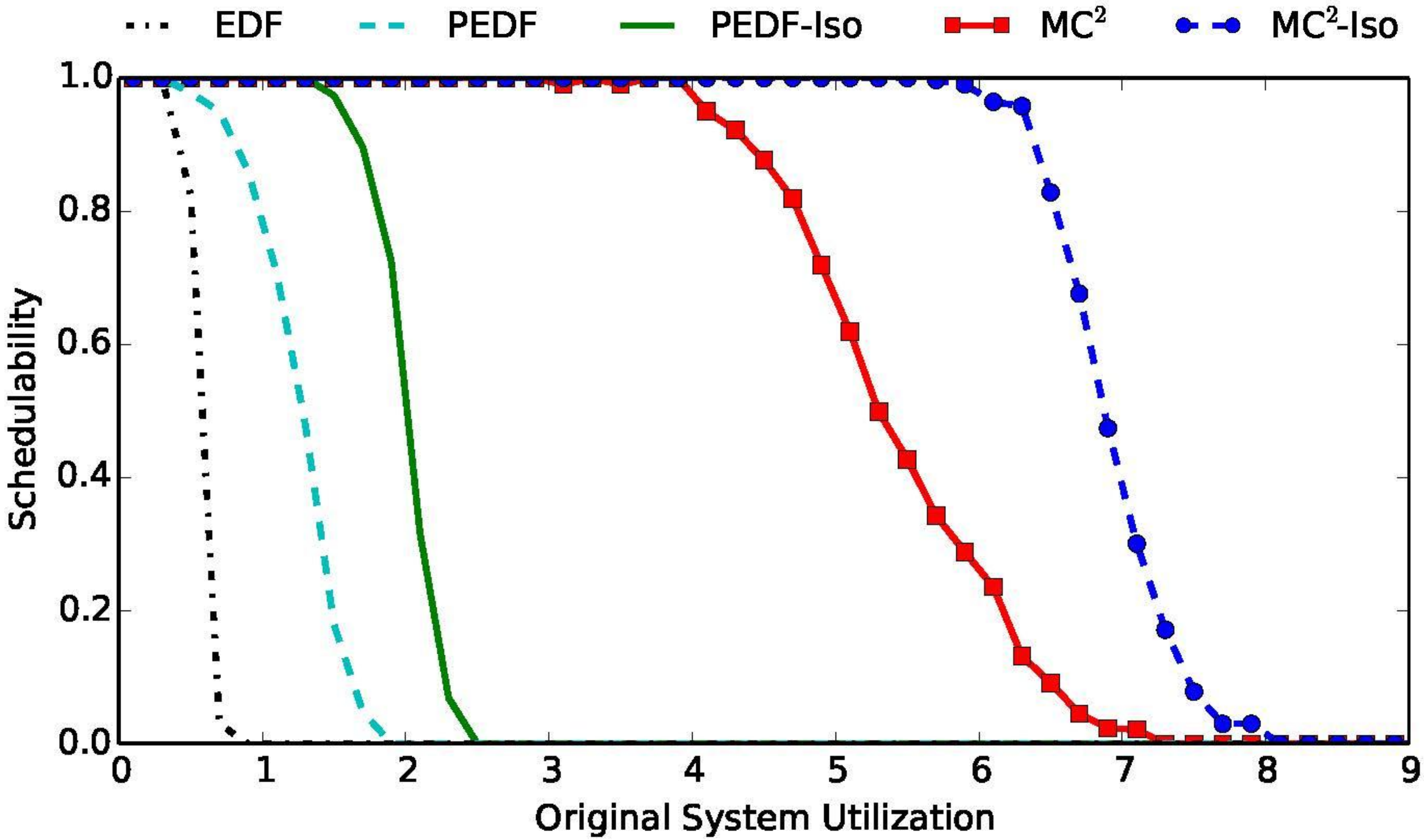


Of the task systems with **total utilization 4.5** in an unmanaged system, **88% were schedulable** under the **"red"** scheme.

Total System Utilization

Overhead-Aware Schedulability Study

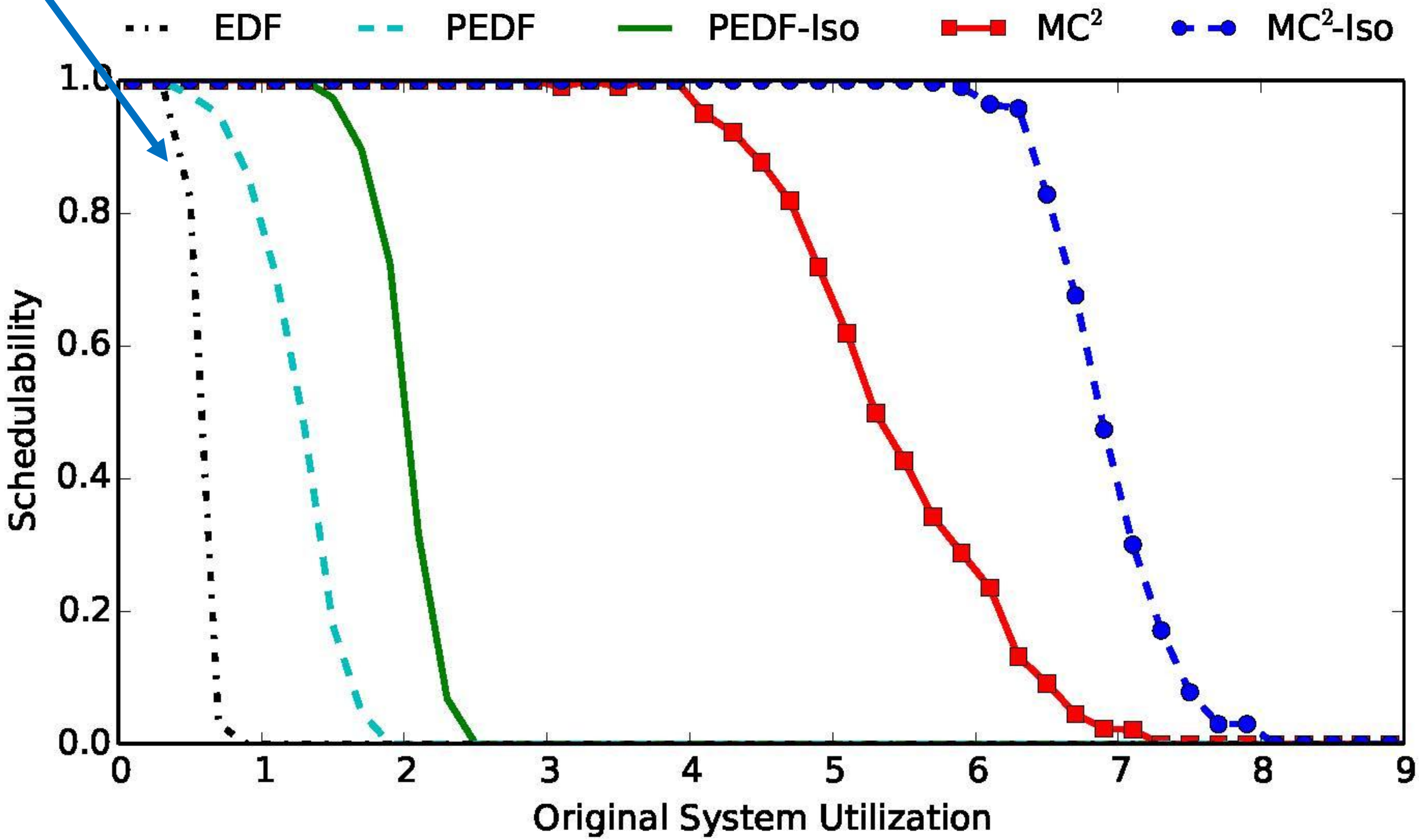
This is One Out of About 500 Graphs



d-Aware Schedulability Study

s is One Out of About 500 Graphs

Uniprocessor EDF (the current de facto standard)

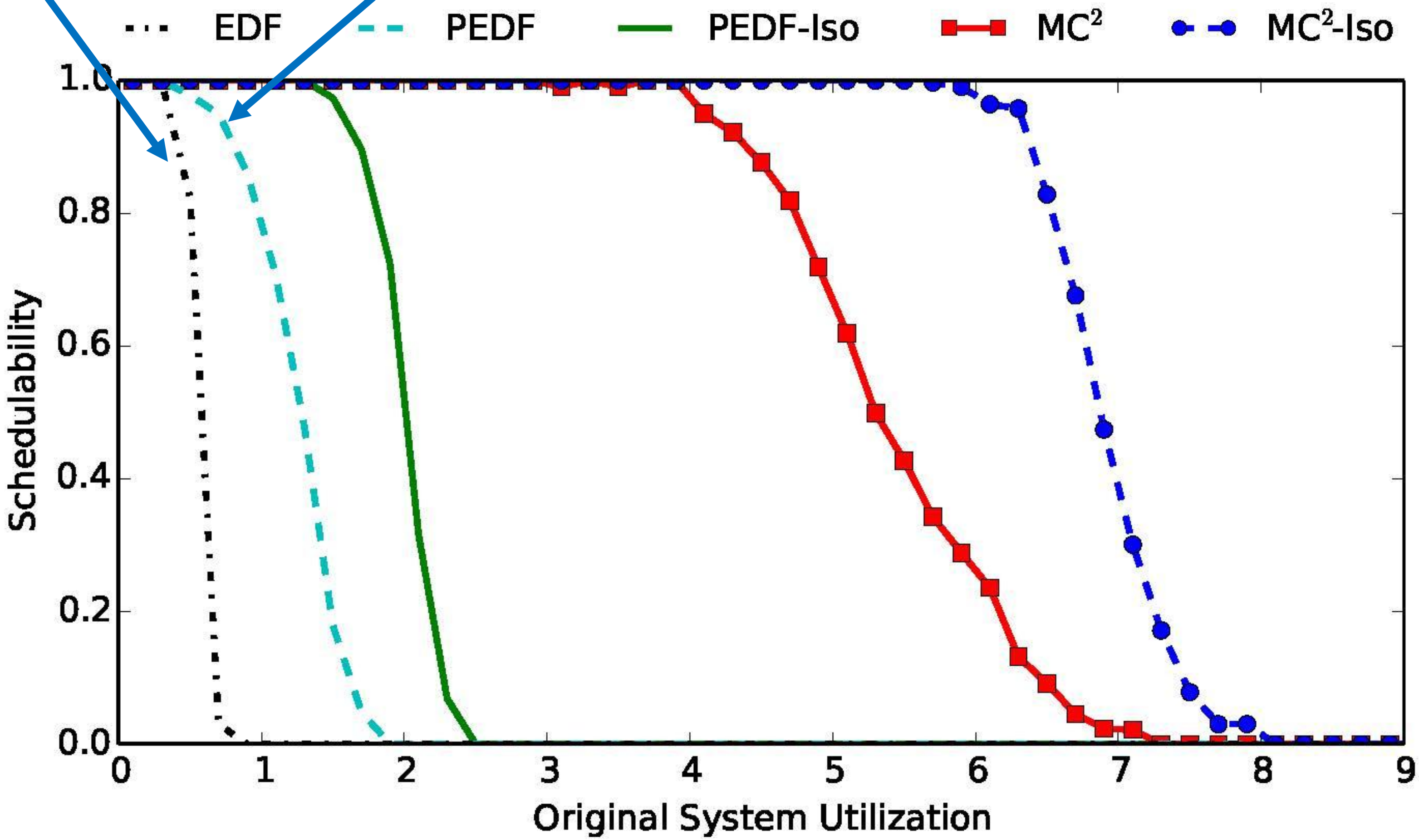


d-Aware Schedulability Study

s is One Out of About 500 Graphs

Uniprocessor EDF (the current de facto standard)

Partitioned EDF



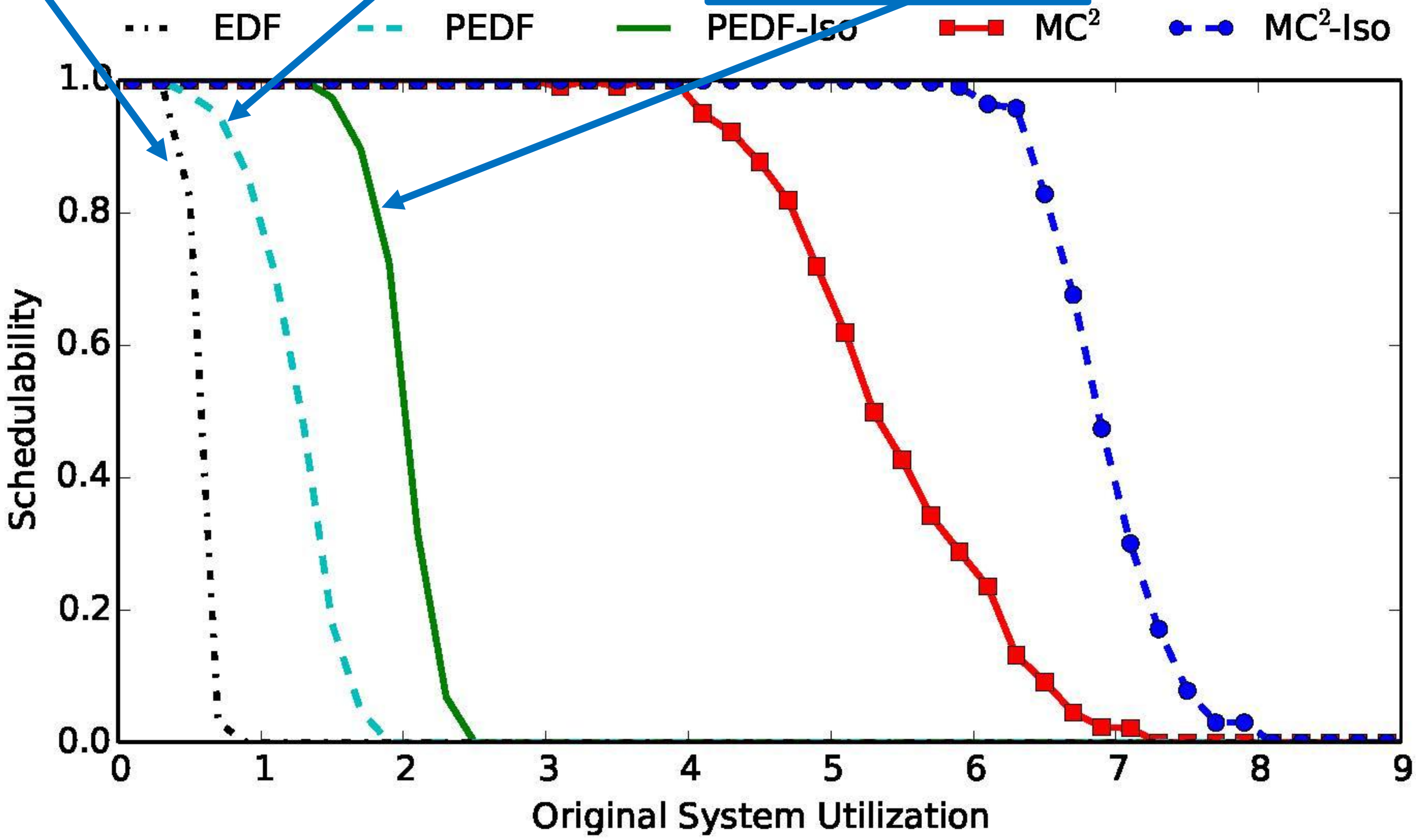
d-Aware Schedulability Study

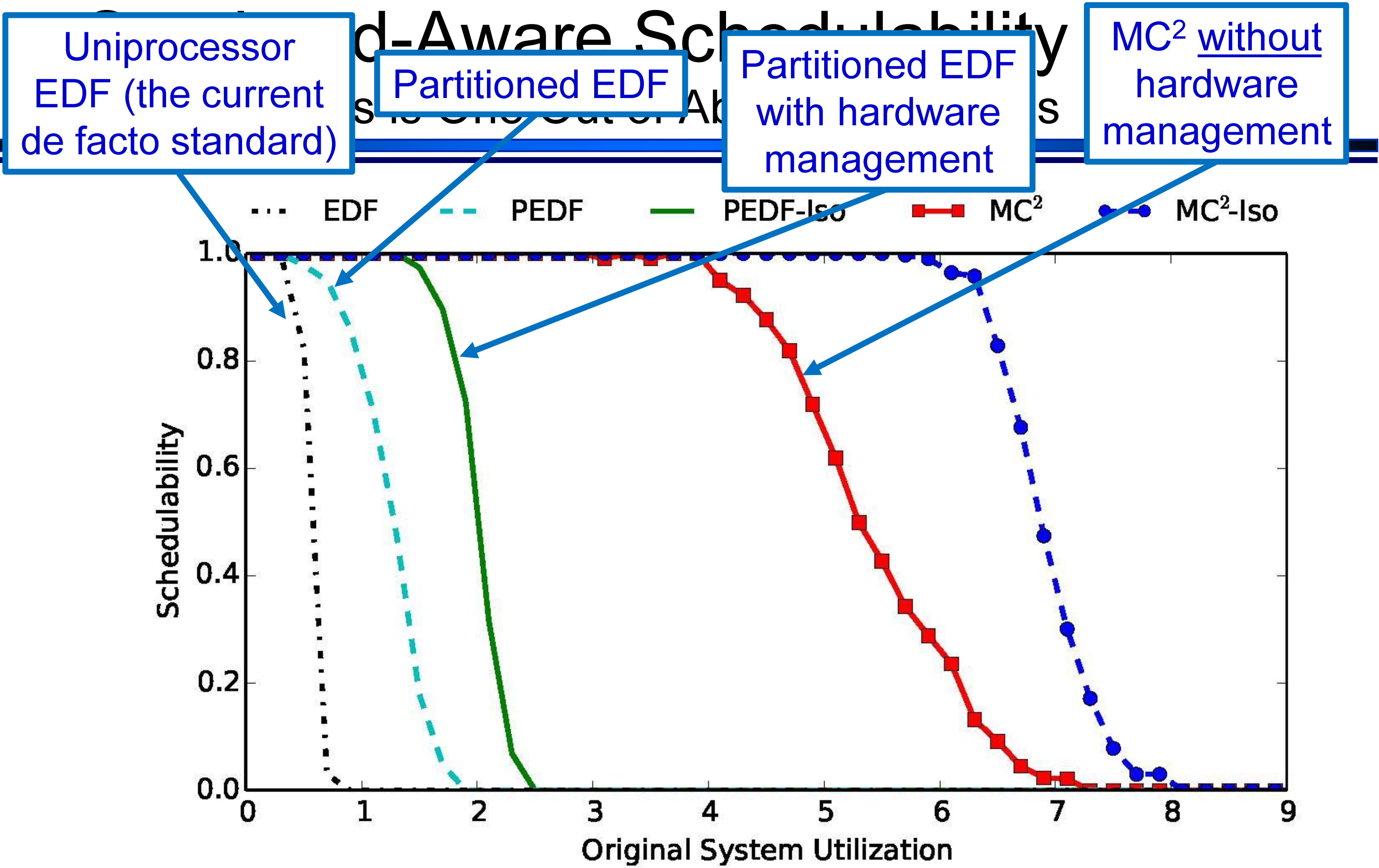
s is One Out of Ab s

Uniprocessor EDF (the current de facto standard)

Partitioned EDF

Partitioned EDF with hardware management





Uniprocessor EDF (the current de facto standard)

Partitioned EDF

Partitioned EDF with hardware management

MC^2 without hardware management

d-Aware Schedulability

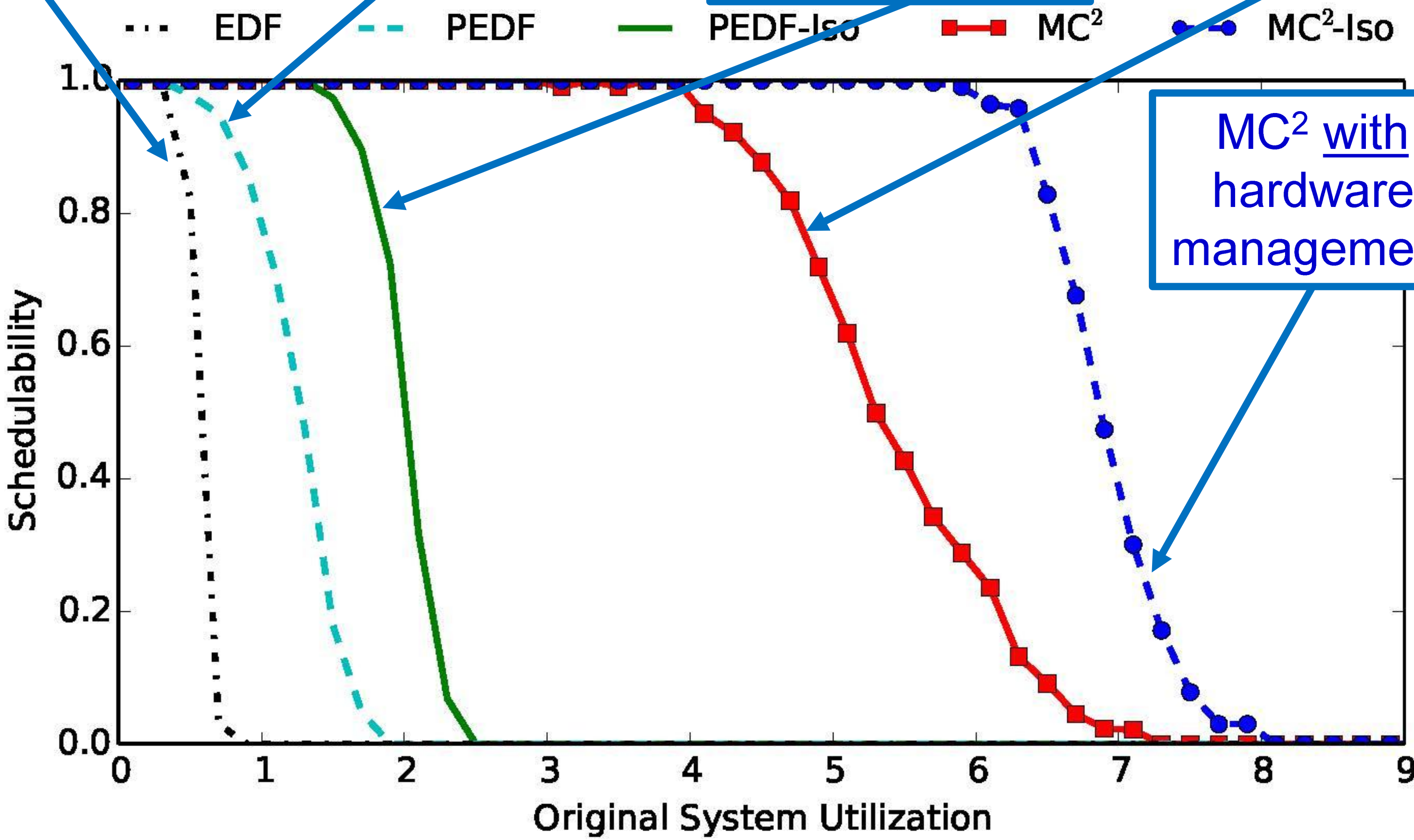
s is One Out of Ab s

Uniprocessor EDF (the current de facto standard)

Partitioned EDF

Partitioned EDF with hardware management

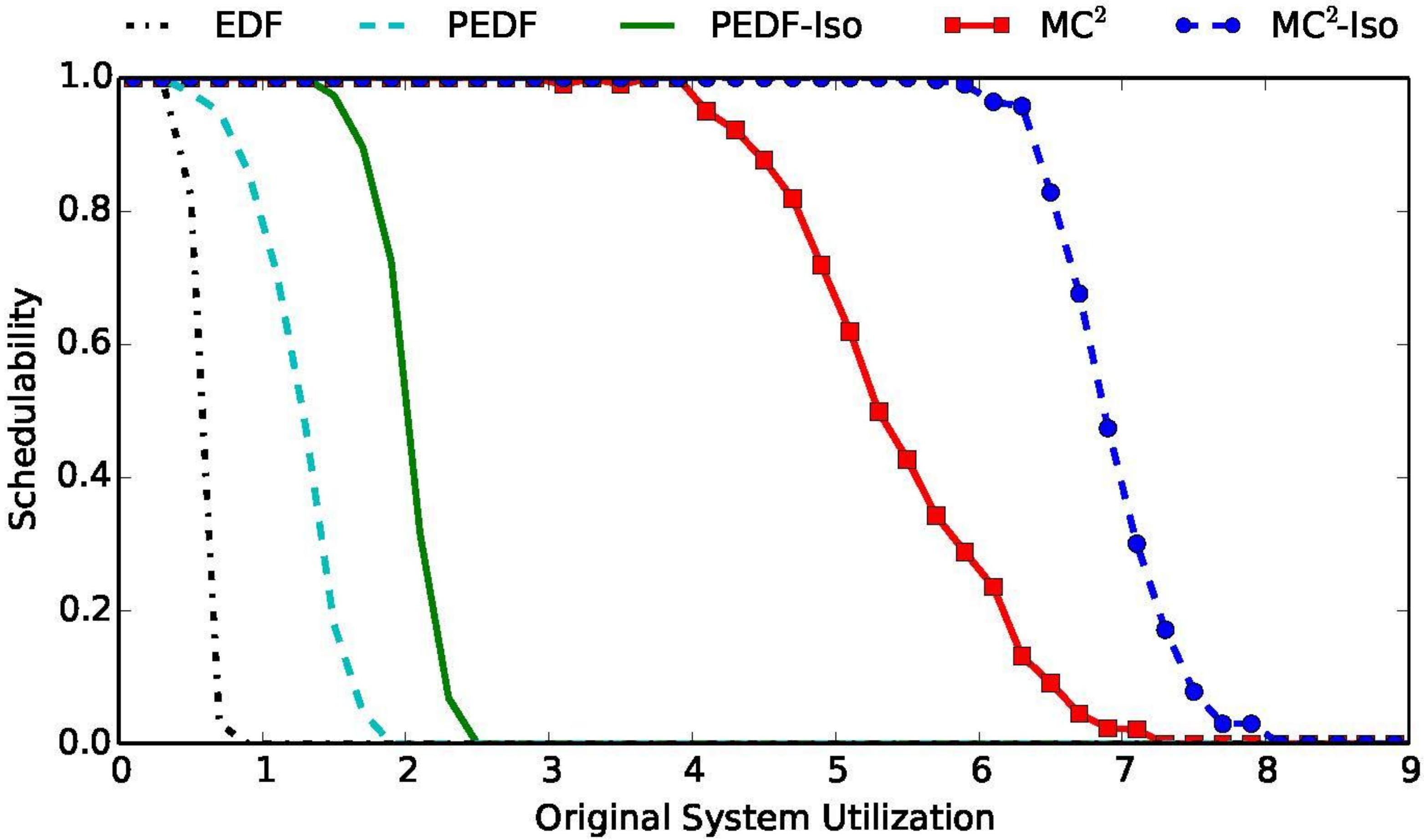
MC^2 <u>without</u> hardware management



MC^2 <u>with</u> hardware management

Overhead-Aware Schedulability Study

This is One Out of About 500 Graphs



Outline

- Problems caused by multicore.
  - » "The one-out-of-m problem."
  - » Why this is an important problem.
- Basic solution strategy.
  - » MC^2 (<u>m</u>ixed-<u>c</u>riticality on <u>m</u>ulti<u>c</u>ore).
  - » Hardware management in MC^2.
- Brief overview of recent work.
  - » Key focus: features of real-world task systems that break hardware isolation.

Recent Work

Dealing with Shared Pages

- Real-world task systems share memory pages.
- In recent work, we've dealt with these sources of sharing:
  - » "Explicit" read/write sharing due to producer/consumer relationships [RTSS'16].
  - » "Implicit" read-only sharing due to shared libraries [RTAS'17].
  - » Sharing due to interrupt-driven I/O [under construction].
- We've also investigated:
  - » Applications that must support mode changes [under construction].

MC^2 Papers

(Available at http://www.cs.unc.edu/~anderson/papers.html)

- J. Anderson, S. Baruah, and B. Brandenburg, “Multicore Operating-System Support for Mixed Criticality,” *Proc. of the Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification*, 2009.
  - » A **“precursor” paper** that discusses some of the design decisions underlying MC^2.
- M. Mollison, J. Erickson, J. Anderson, S. Baruah, and J. Scoredos, “Mixed Criticality Real-Time Scheduling for Multicore Systems,” *Proc. of the 7th IEEE International Conf. on Embedded Software and Systems*, 2010.
  - » Focus is on **schedulability**, i.e., how to check timing constraints at each level and “shift” slack.
- J. Herman, C. Kenna, M. Mollison, J. Anderson, and D. Johnson, “RTOS Support for Multicore Mixed-Criticality Systems,” *Proc. of the 18th RTAS*, 2012.
  - » Focus is on **RTOS design**, i.e., how to reduce the impact of RTOS-related overheads on high-criticality tasks due to low-criticality tasks.
- B. Ward, J. Herman, C. Kenna, and J. Anderson, “Making Shared Caches More Predictable on Multicore Platforms,” *Proc. of the 25th ECRTS*, 2013.
  - » Adds **shared cache management** to a two-level variant of MC^2. The approach in today’s talk is different.
- J. Erickson, N. Kim, and J. Anderson, “Recovering from Overload in Multicore Mixed-Criticality Systems,” *Proc. of the 29th IPDPS*, 2015.
  - » Adds **virtual-time-based scheduling** to Level C.

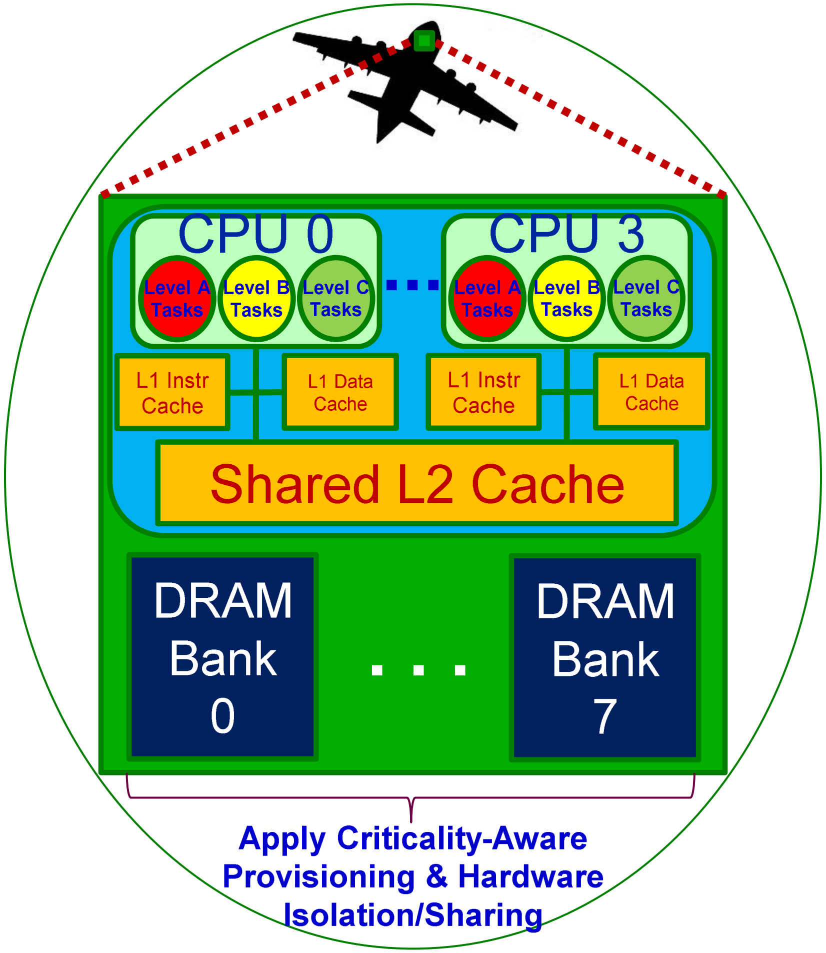
MC² Papers

(Available at http://www.cs.unc.edu/~anderson/papers.html)

- M. Chisholm, B. Ward, N. Kim, and J. Anderson, "Cache Sharing and Isolation Tradeoffs in Multicore Mixed-Criticality Systems," *Proc. of the 36th RTSS*, 2015.
  - » Presents linear-programming-based techniques for **optimizing LLC area allocations**.
- N. Kim, B. Ward, M. Chisholm, C.-Y. Fu, J. Anderson, and F.D. Smith, "Attacking the One-Out-Of-m Multicore Problem by Combining Hardware Management with Mixed-Criticality Provisioning," *Proc. of the 22nd RTAS*, 2016.
  - » Adds **shared hardware management** to MC².
- M. Chisholm, N. Kim, B. Ward, N. Otterness, J. Anderson, and F.D. Smith, "Reconciling the Tension Between Hardware Isolation and Data Sharing in Mixed-Criticality, Multicore Systems," Proc. of the 37th RTSS, 2016.
  - » Adds support for **data sharing** to MC².
- N. Kim, M. Chisholm, N. Otterness, J. Anderson, and F.D. Smith, "Allowing Share Libraries while Supporting Hardware Isolation in Multicore Real-Time Systems," Proc. of the 23rd RTAS, 2017 (to appear).
  - » Adds **selective sharing of libraries** to MC².

Thanks!

- Questions?



Apply Criticality-Aware Provisioning & Hardware Isolation/Sharing