

Research Review 2021

Using All Processor Cores While Being Confident about Timing

November 2021

Bjorn Andersson

Copyright 2021 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM21-0810

Complex, cyber-physical DoD systems depend on correct timing—any timing failure could be disastrous. What's more, while these systems drive demand for use of multicore processors, concern about timing has led to disabling all processor cores except one—limiting system capability.

We aim to develop a solution to overcome this obstacle.

DoD Systems Interact with Their Physical Environment



DoD Systems Include Software



DoD Systems Include Software That Interacts with the Physical Environment



DoD Systems Include Software That Has Real-Time Requirements



Satisfying Real-Time Requirements Is a Challenge for the DoD in General



Satisfying Real-Time Requirements Is Challenging for Upgrading the Blackhawk UH-60 Helicopter



Satisfying Real-Time Requirements Is Challenging for Upgrading the Blackhawk UH-60 Helicopter

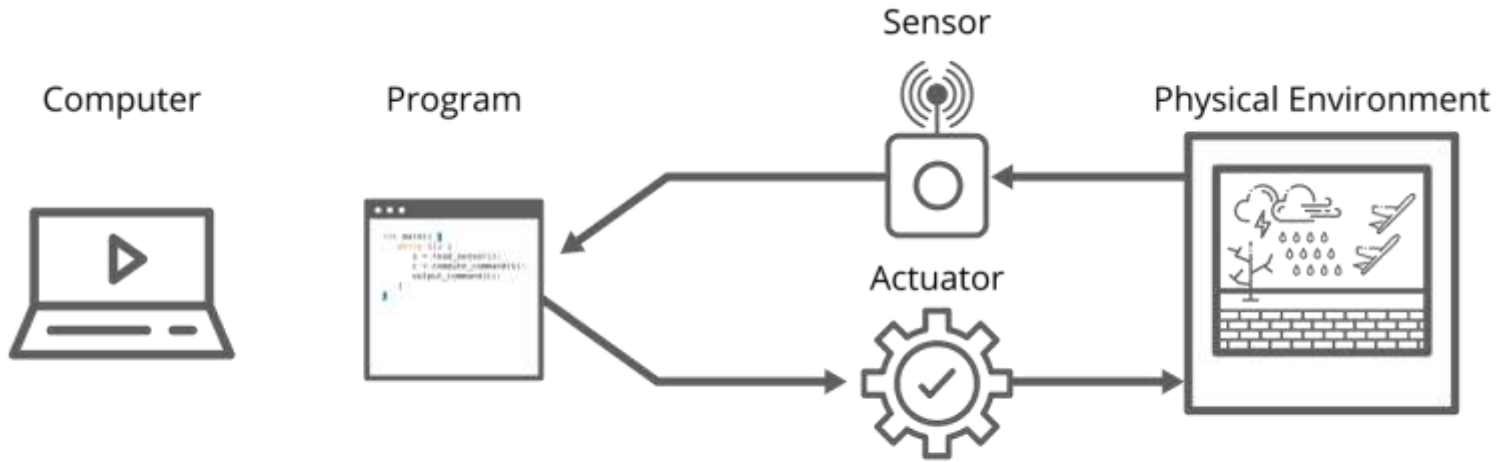


“The trick there, when you’re processing flight critical information, it has to be a deterministic environment, meaning we know exactly where a piece of data is going to be exactly when we need to — no room for error,” Langhout says. “On a multi-core processor there’s a lot of sharing going on across the cores, so right now we’re not able to do that.”

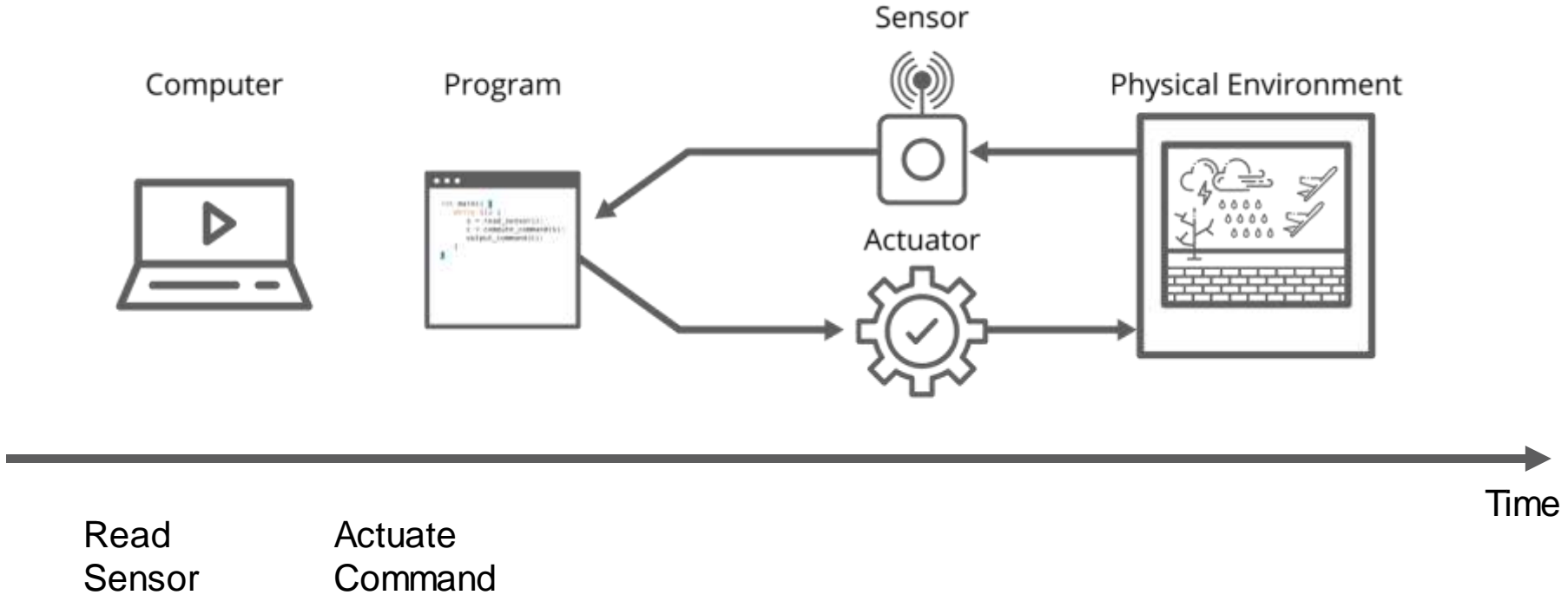
- Jeff Langhout, Acting Director, U.S. Army Aviation and Missile Research Development and Engineering Center (AMRDEC)

Source: “Army still working on multi-core processor for UH-60V,” May 2017, Available at <https://www.flightglobal.com/news/articles/army-still-working-on-multi-core-processor-for-uh-6-436895/>.

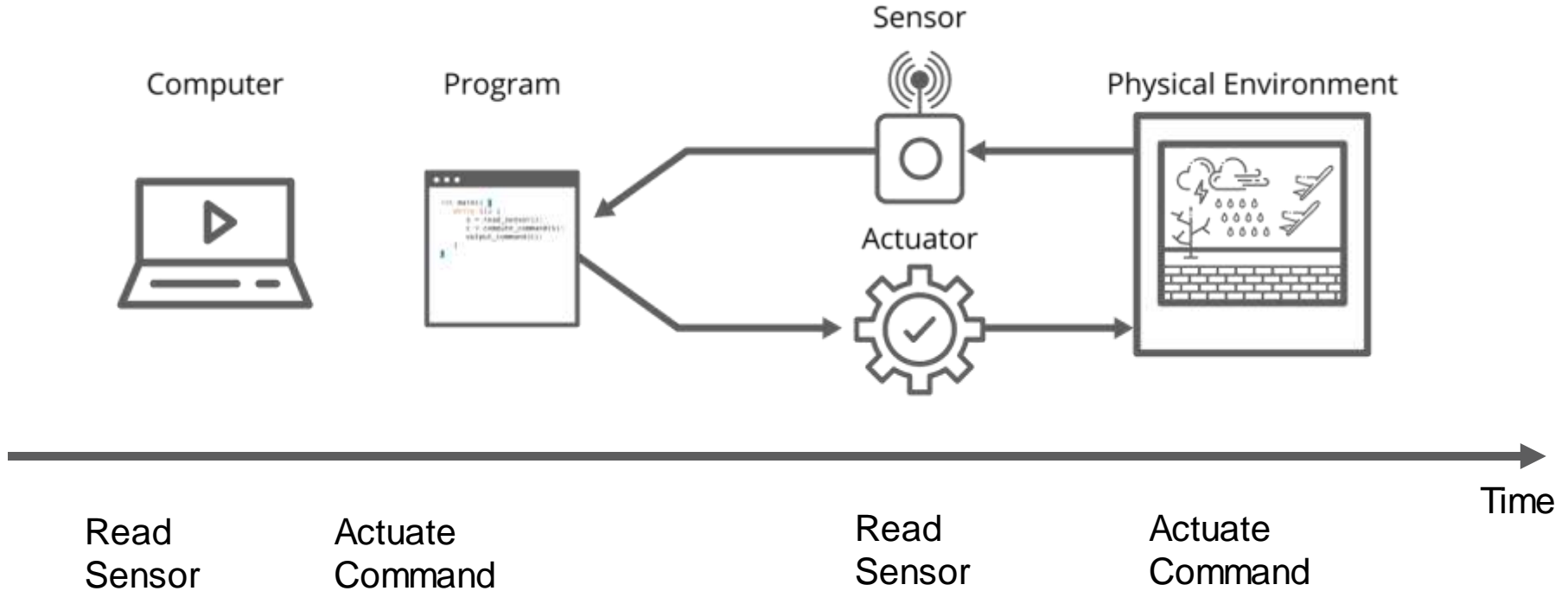
Commonality of DoD Systems



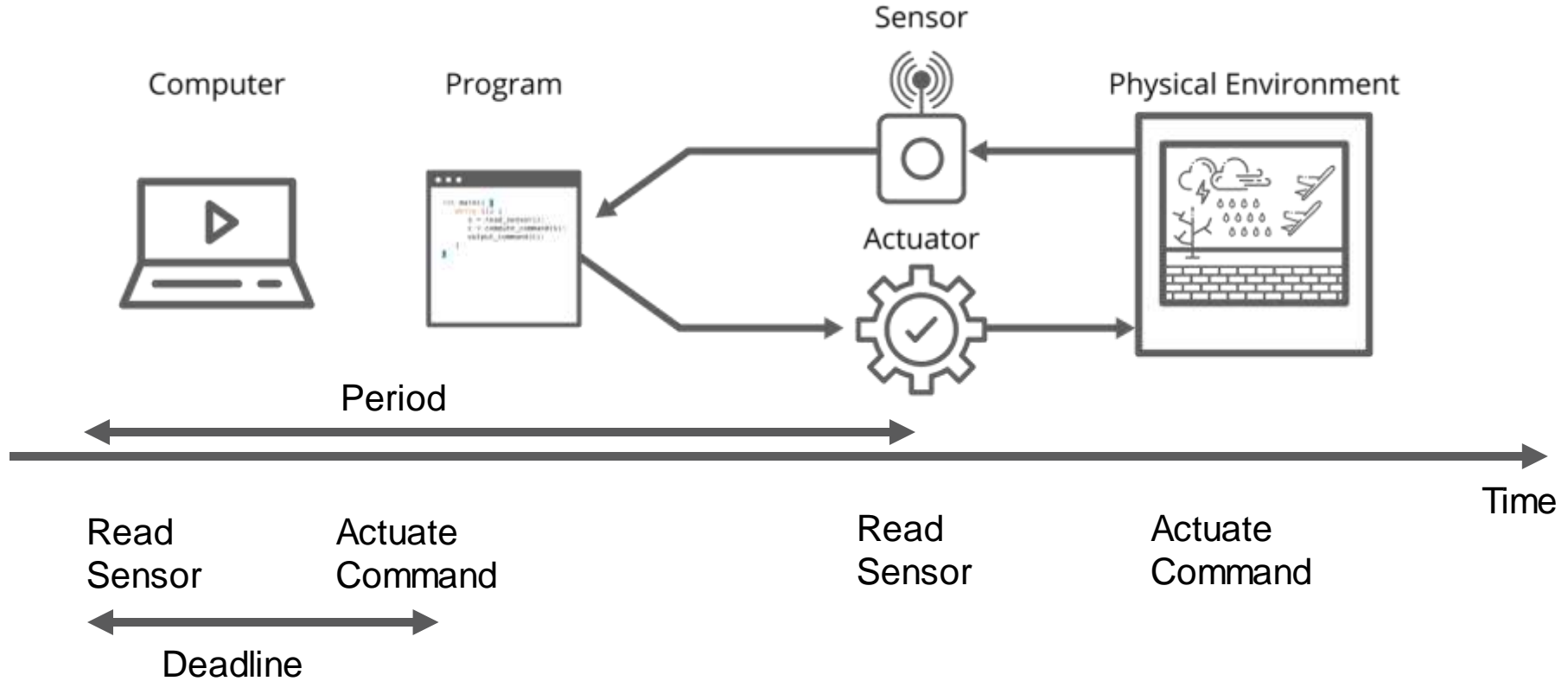
Commonality of DoD Systems



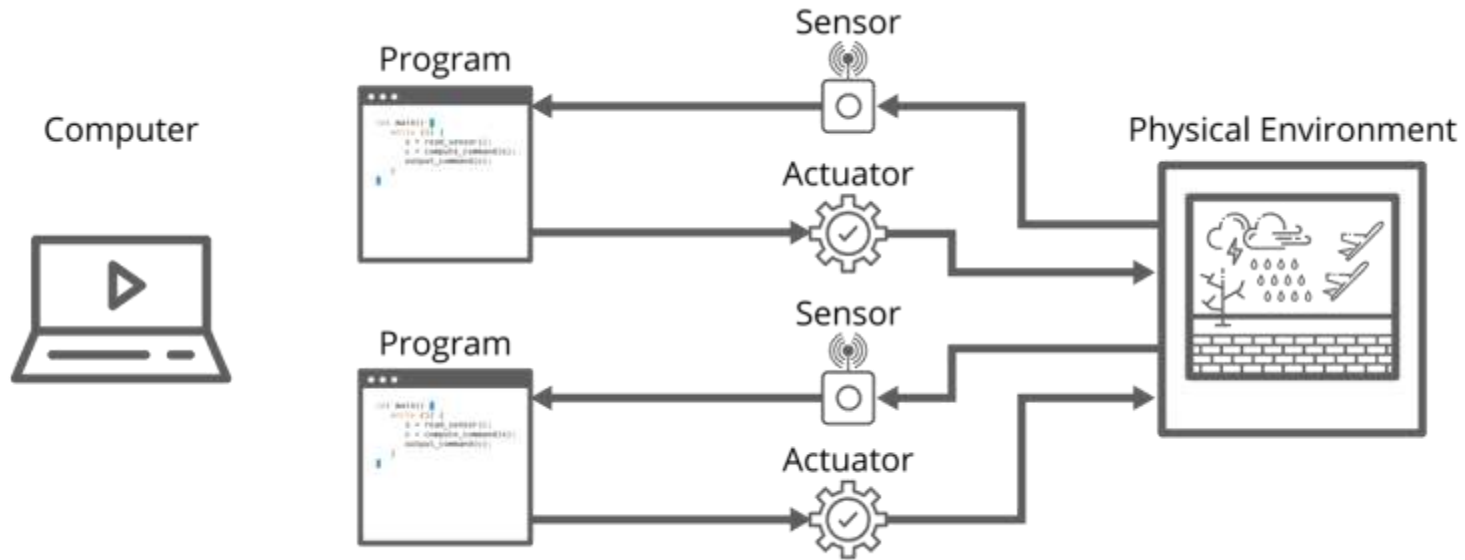
Commonality of DoD Systems



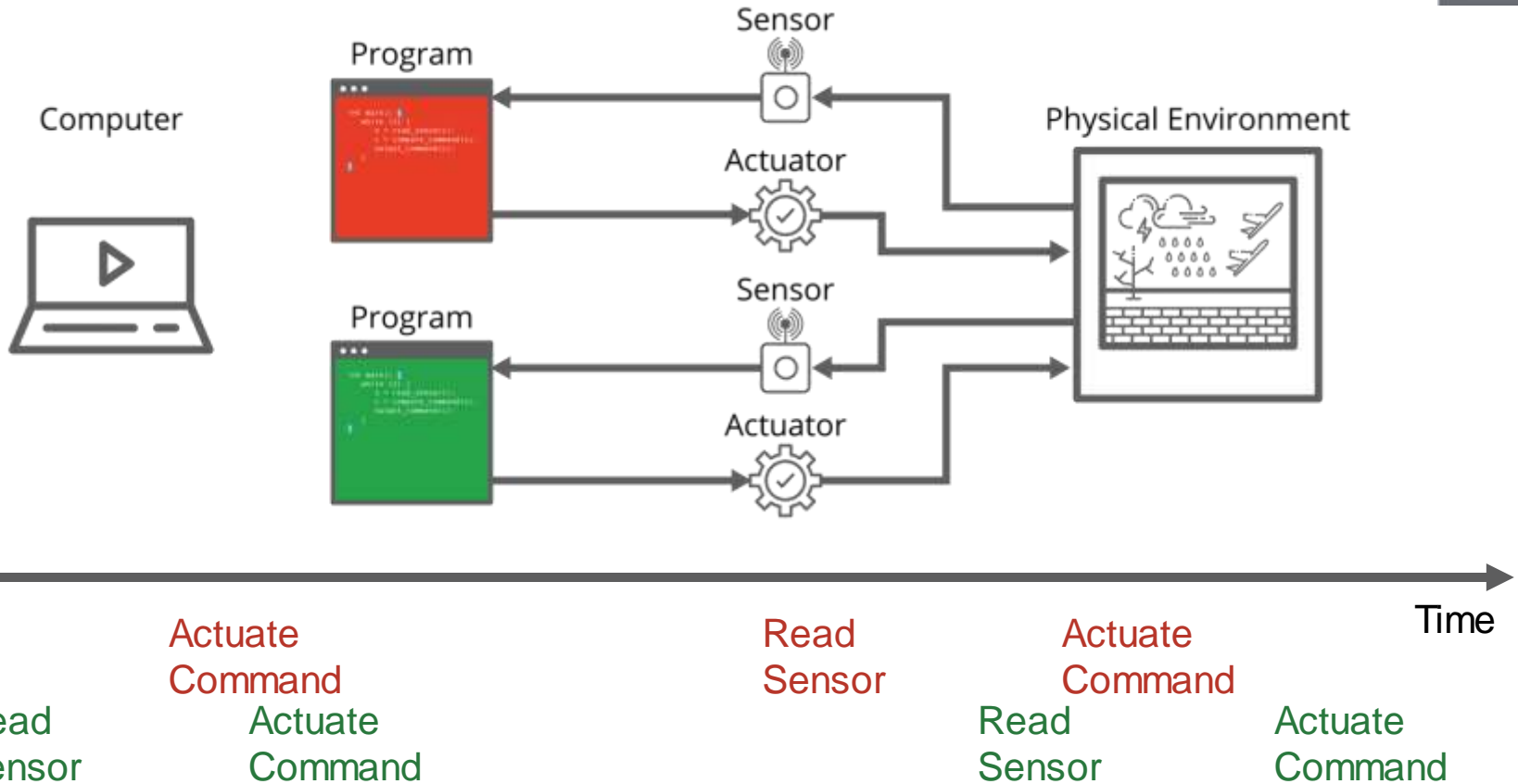
Commonality of DoD Systems



Commonality of DoD Systems



Commonality of DoD Systems



What Makes it Challenging to Satisfy Real-Time Requirements?

What Causes Delay of Software?

What Causes Delay of Software?



Time

What Causes Delay of Software?



What Causes Delay of Software?

Thread executes
one path



Time when one thread
in the software system arrives

Deadline Time

What Causes Delay of Software?

Thread executes another path



What Causes Delay of Software?

Preemption:
Another thread uses
the processor.



What Causes Delay of Software?

Interrupt
service
routine
executes



What Causes Delay of Software?

Thread requests a
critical section held by
another thread



What Causes Delay of Software?

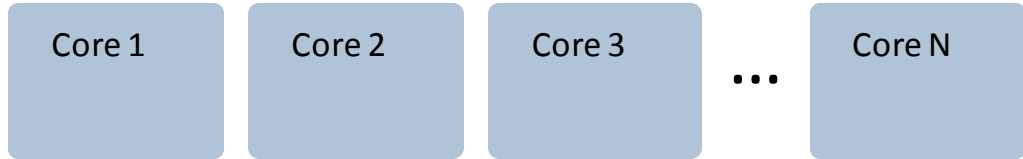
Thread experiences extra cache misses because of preemptions



Real-Time Requirements of Software Executing on a Multicore Processor

Hardware Trends

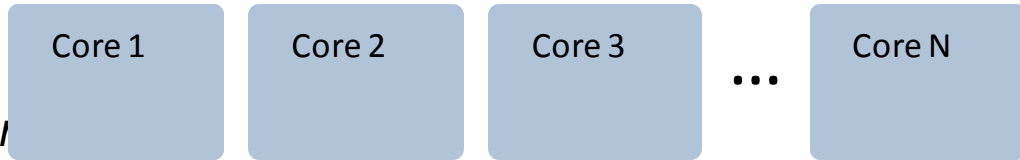
- *All computers are multicores.*



Real-Time Requirements of Software Executing on a Multicore Processor

Hardware Trends

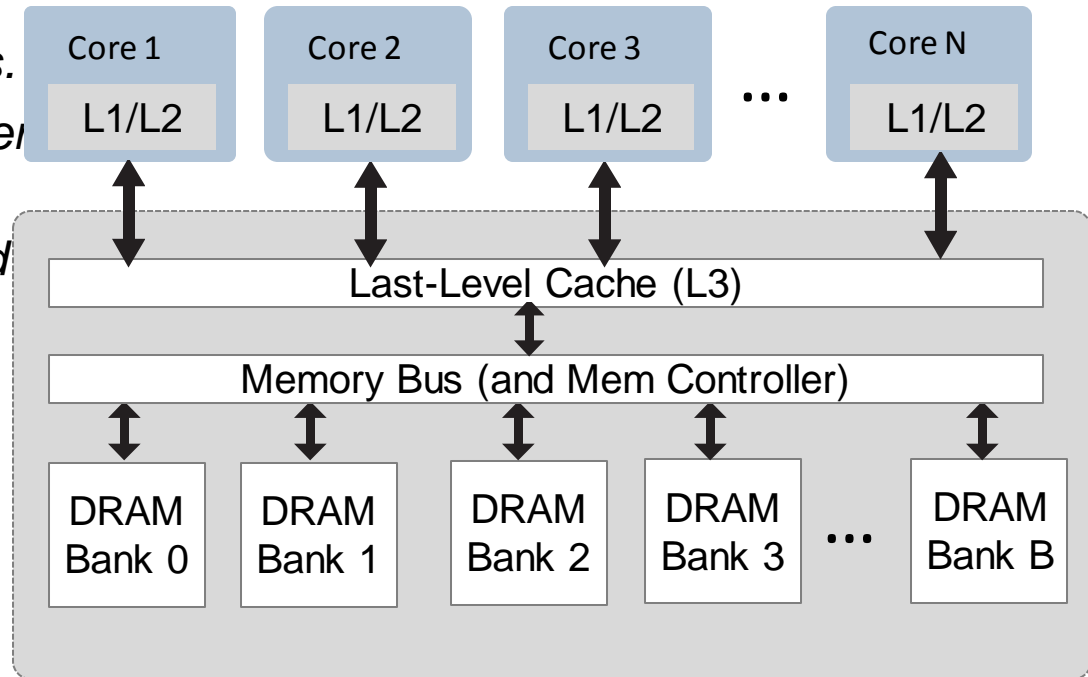
- *All computers are multicores.*
- *Most chip makers do not offer single core.*

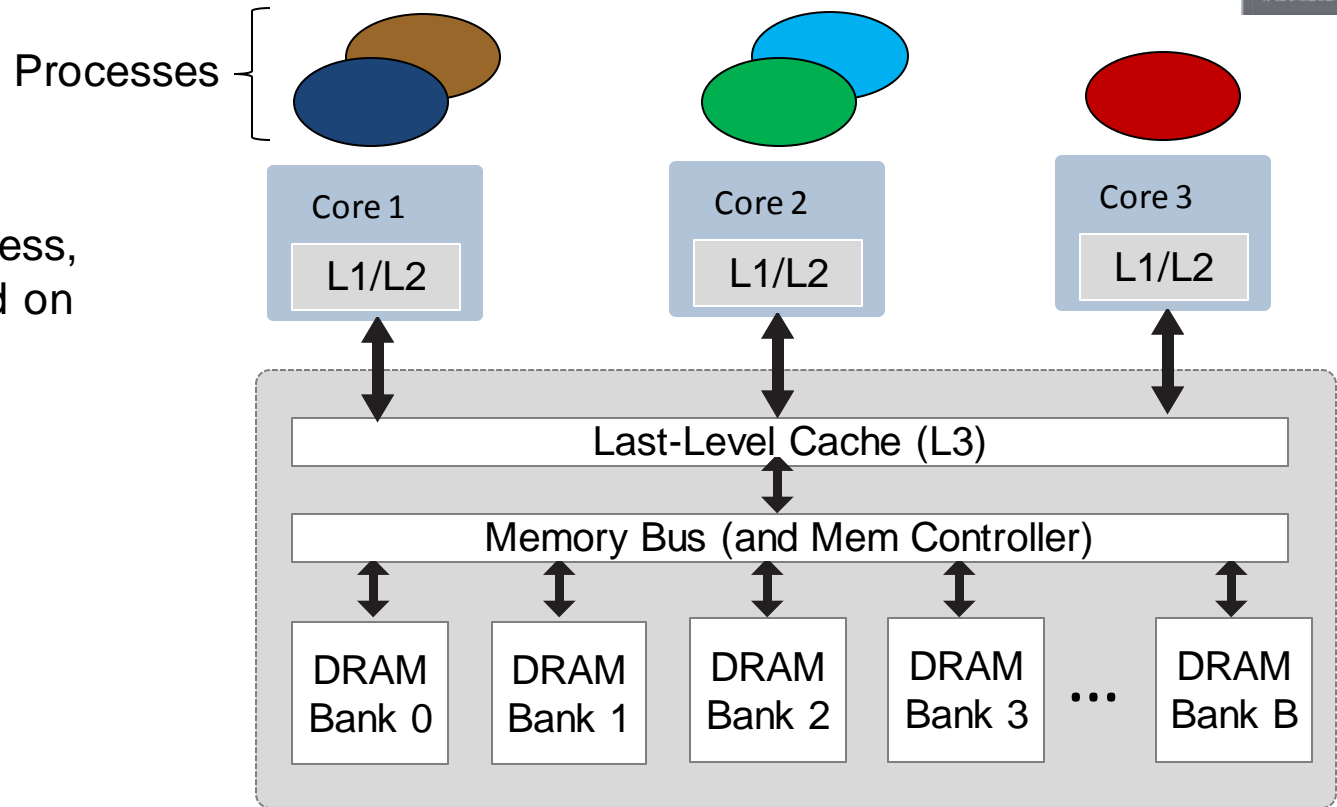


Real-Time Requirements of Software Executing on a Multicore Processor

Hardware Trends

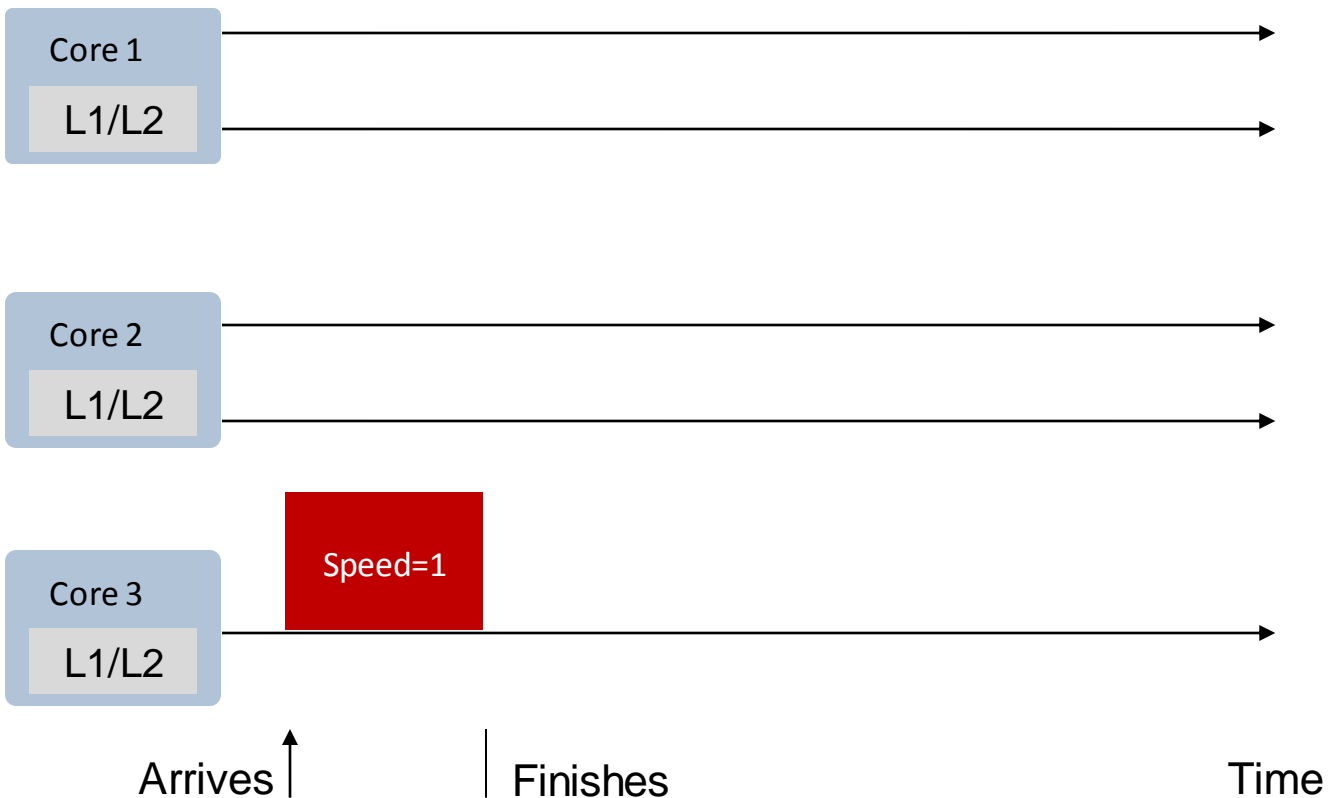
- *All computers are multicores.*
- *Most chip makers do not offer single core.*
- *Most multicores have shared memory.*



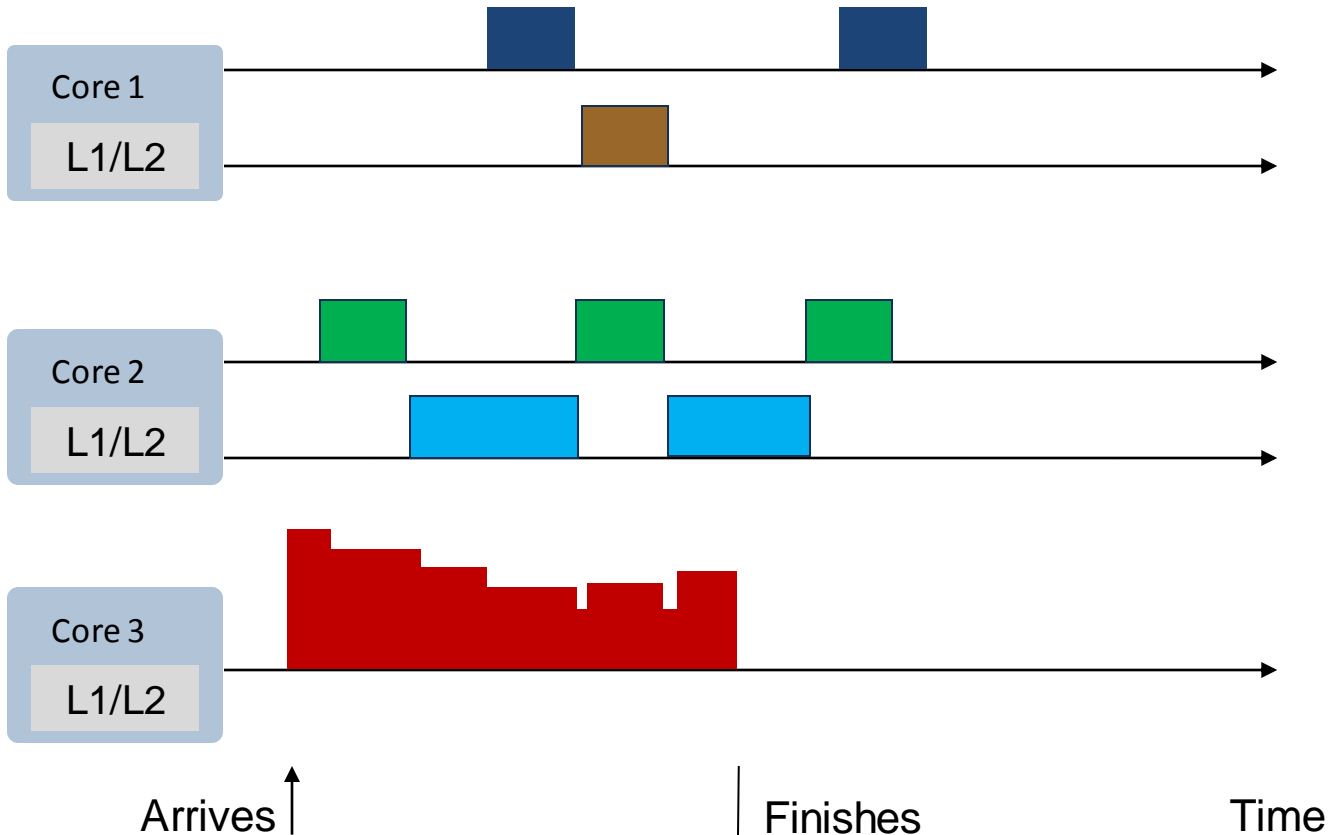


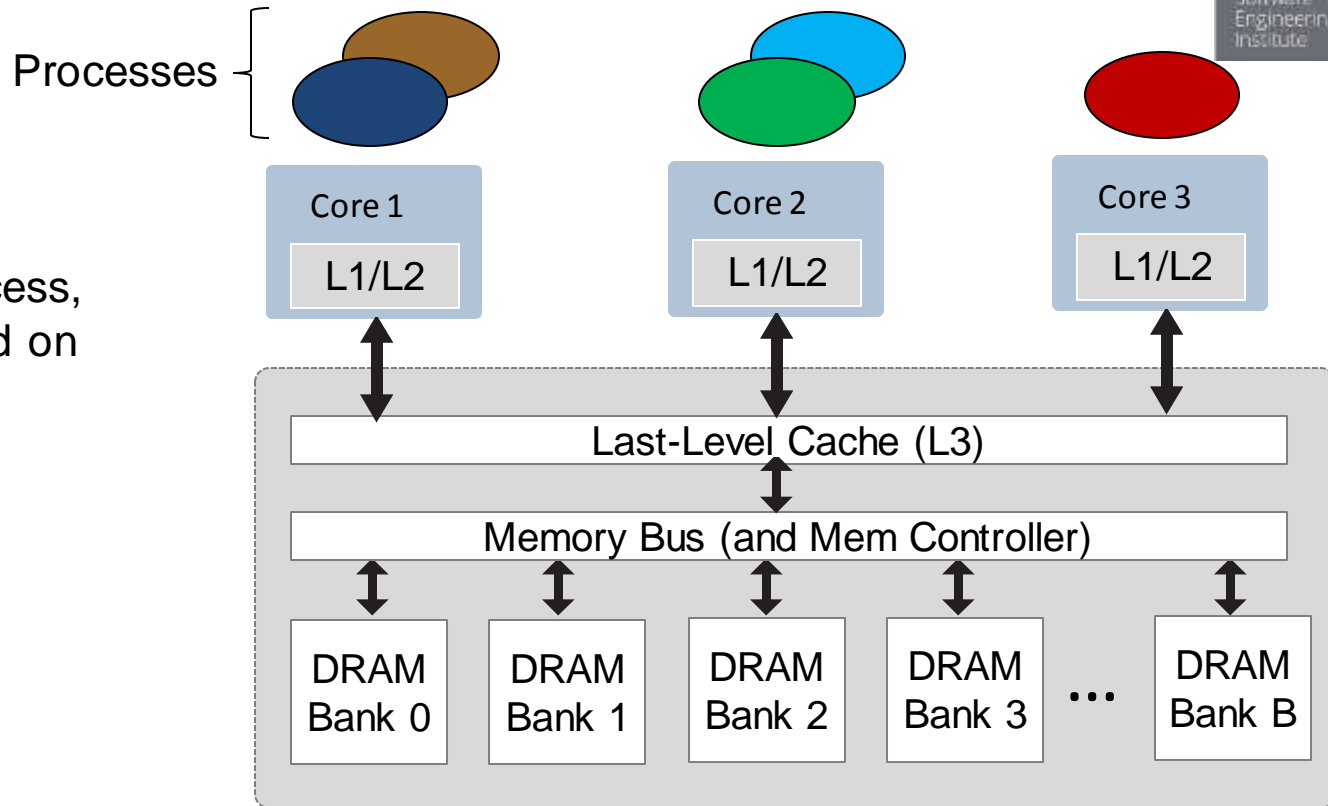
Problem: For each process, compute an upper bound on its response time.

How Co-Runners Impact Speed of Execution



How Co-Runners Impact Speed of Execution



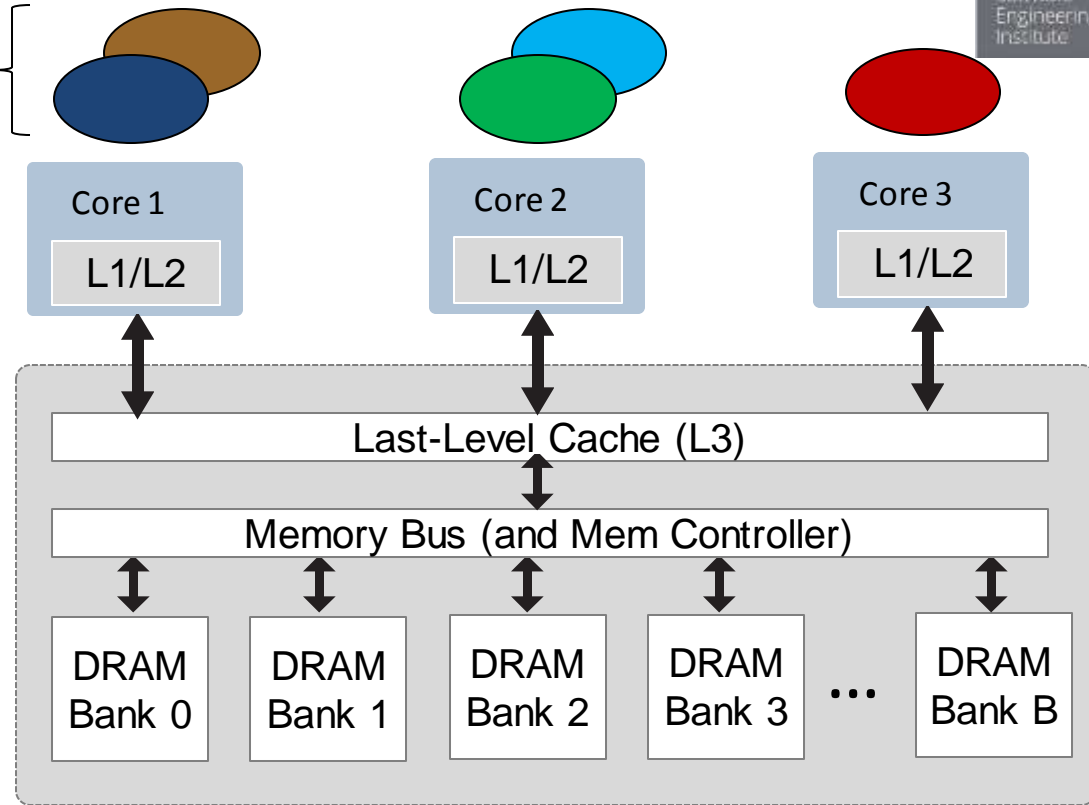


Problem: For each process, compute an upper bound on its response time.

Issues

- *Shared hardware resources impact timing.*

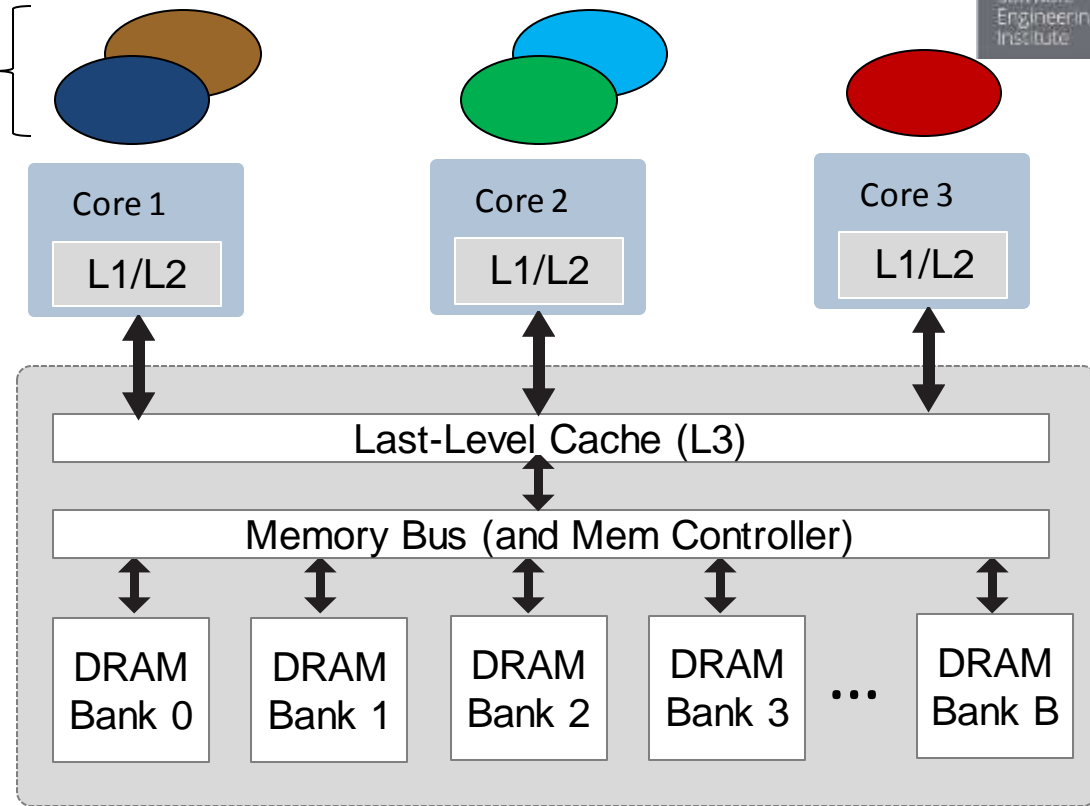
Processes



Issues

- *Shared hardware resources impact timing.*
- *103 times slowdown has been observed.**

Processes

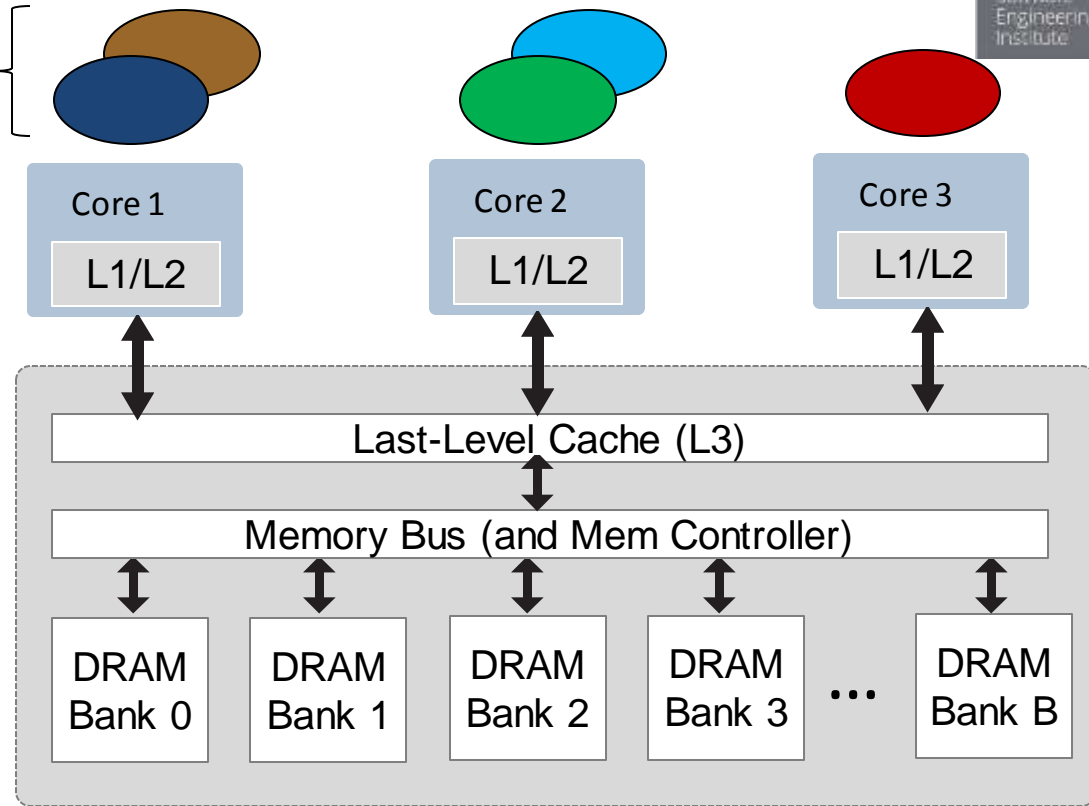


*H. Yun and P. K. Valsan, "Evaluating the Isolation Effect of Cache Partitioning on COTS Multicore Platforms," OSPERT, 2015.

Issues

- *Shared hardware resources impact timing.*
- *103 times slowdown has been observed [Yun15].*
- *Current methods cannot deal with undocumented resources.*

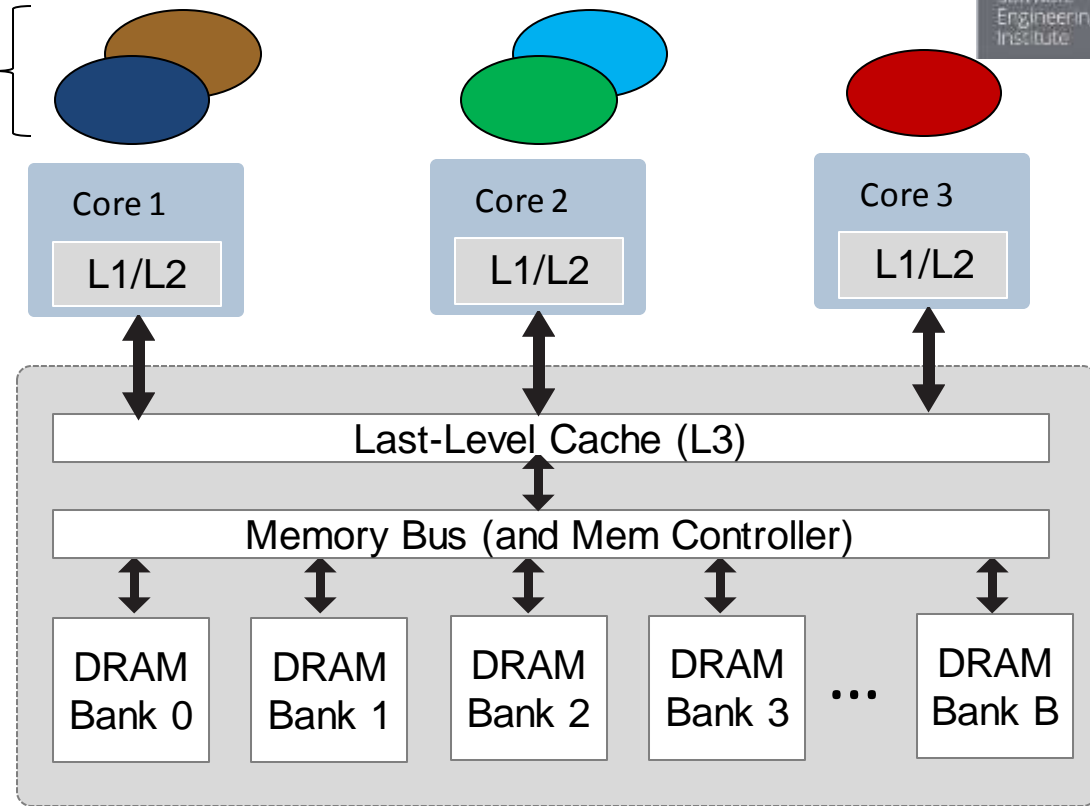
Processes



Issues

- *Shared hardware resources impact timing.*
- *103 times slowdown has been observed [Yun15].*
- *Current methods cannot deal with undocumented resources.*
- *Even for the case that resources are documented, current methods can only analyze/manage a small set of them.*

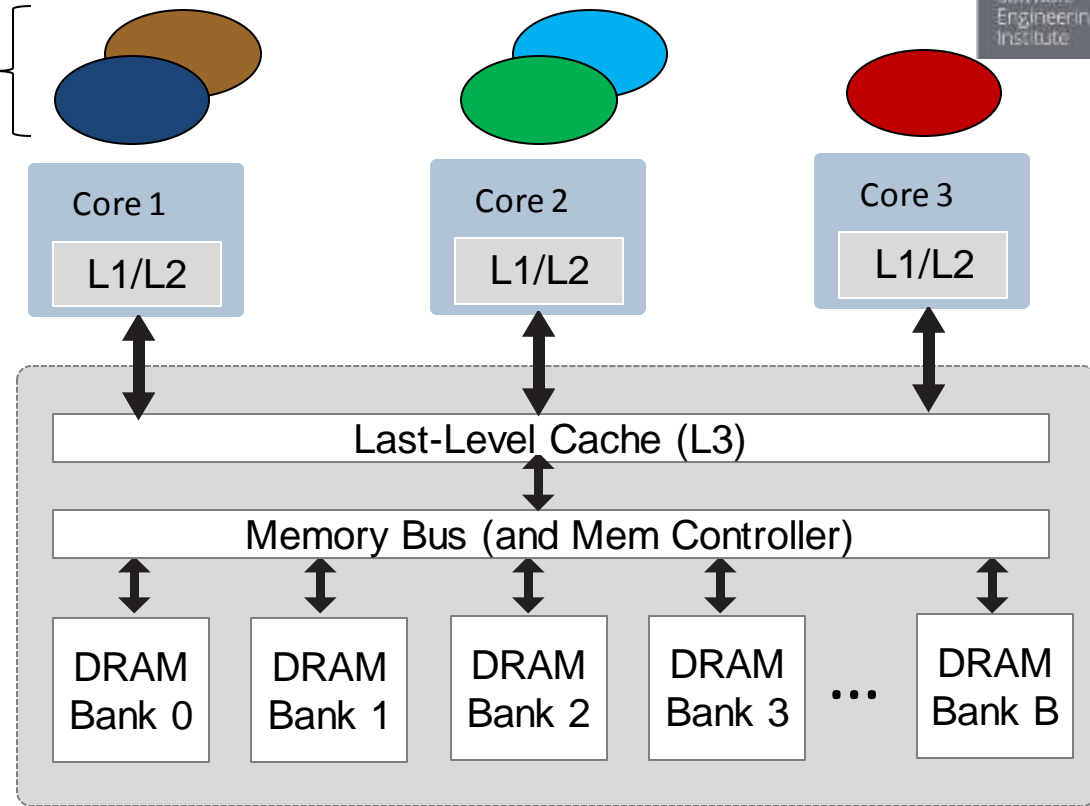
Processes



Issues

- *Shared hardware resources impact timing.*
- *103 times slowdown has been observed [Yun15].*
- *Current methods cannot deal with undocumented resources.*
- *Even when resources are documented, current methods can only analyze/manage a small set of them.*
- *The problem is getting worse:*
 - * *Slowdown increasing*
 - * *More undocumented h/w*

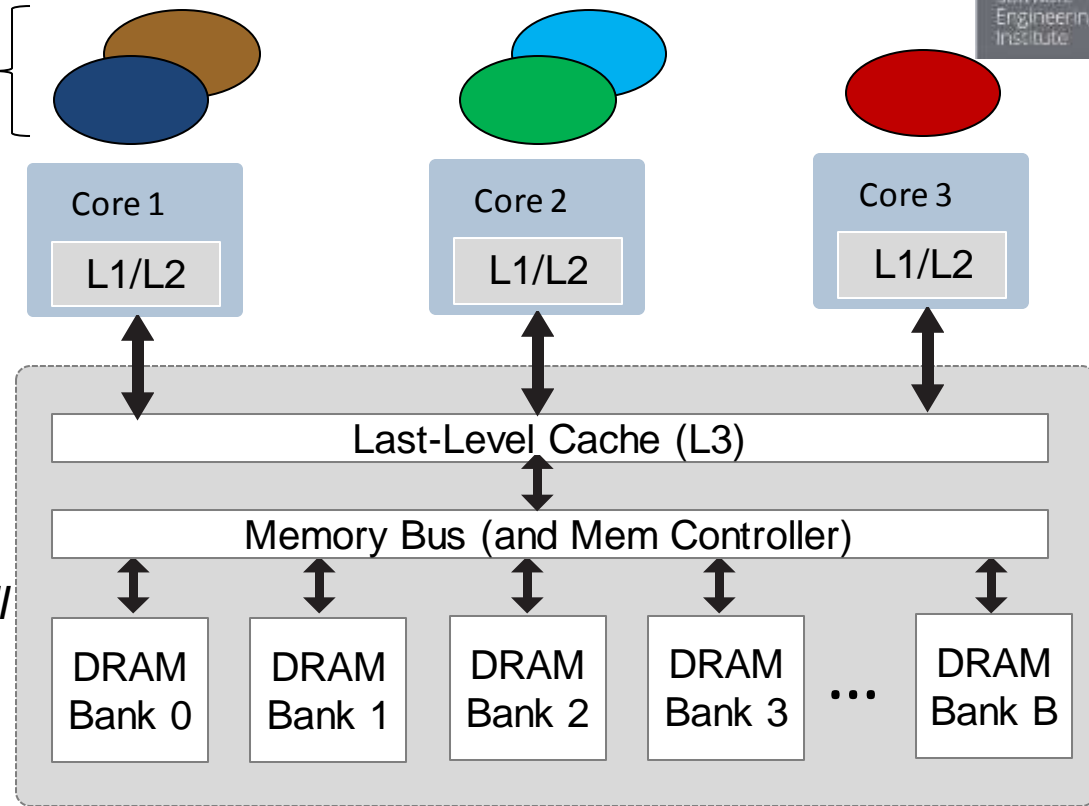
Processes



Issues

- *Shared hardware resources impact timing.*
- *103 times slowdown has been observed [Yun15].*
- *Current methods cannot deal with undocumented resources.*
- *Even when resources are documented, current methods can only analyze/manage a small set of them.*
- *The problem is getting worse:*
 - * *Slowdown increasing*
 - * *More undocumented h/w*

Processes



We need a new method to compute response times of processes.

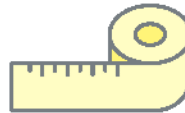
Project Objective

Many real-time systems within the DoD have all processor cores except one disabled in order to be confident about timing.

Therefore, the objective of this project is to develop a solution to overcome this obstacle.



New verification procedure



Method to obtain abstractions



Configuration

A Look at Our Analysis Tools - 1

Schedulability analysis of tasks with co-runner dependent execution times

This program implements the schedulability test in B. Andersson et al., "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times," ACM TECS, 2018.

Number of tasks: 5 Number of processors: 2

	Minimum inter-arrival time	Deadline	Number of segments	Priority	Processor	Execution requirement	Default speed	Co-runner specification
Task 1	1.5	1.5	1	3	1			
Segment 1						0.25	0.5	{{[3, 1]}, [1.0]}, {[4, 1]}, 0.5}, {[5, 1]}, 1.0}, {[5, 2]}, 1.0}}
Segment 2								
Segment 3								
Task 2	2.0	2.0	1	2	1			
Segment 1						0.25	0.5	{{[3, 1]}, 0.5}, {[4, 1]}, 1.0}, {[5, 1]}, 1.0}, {[5, 2]}, 1.0}}
Segment 2								
Segment 3								
Task 3	2.0	2.0	1	3	2			
Segment 1						0.25	0.5	{{[1, 1]}, 1.0}, {[2, 1]}, 0.5}}
Segment 2								
Segment 3								
Task 4	2.0	2.0	1	2	2			
Segment 1						0.25	0.5	{{[1, 1]}, 0.5}, {[2, 1]}, 1.0}}
Segment 2								
Segment 3								
Task 5	2.25	2.25	2	1	2			
Segment 1						0.5	1.0	{{[1, 1]}, 1.0}, {[2, 1]}, 1.0}}
Segment 2						0.125	0.5	{{[1, 1]}, 0.5}, {[2, 1]}, 1.0}}
Segment 3								

Do schedulability analysis

A Look at Our Analysis Tools - 2

Schedulability analysis of tasks with co-runner dependent execution times

This program implements the schedulability test in B. Andersson et al., "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times," ACM TECS, 2018.

Number of tasks: 5 Number of processors: 2

	Minimum inter-arrival time	Deadline	Number of segments	Priority	Processor	Execution requirement	Default speed	Co-runner specification
Task 1	1.5	1.5	1	3	1			
Segment 1						0.25	0.5	[[[3, 1]], 1.0], [[4, 1]], 0.5], [[5, 1]], 1.0], [[5, 2]], 1.0]]
Segment 2								
Segment 3								
Task 2	2.0	2.0	1	2				
Segment 1]], 1.0], [[5, 1]], 1.0], [[5, 2]], 1.0]]
Segment 2								
Segment 3								
Task 3	2.0	2.0	1	3]], 0.5]]
Segment 1								
Segment 2								
Segment 3								
Task 4	2.0	2.0	1	2]], 1.0]]
Segment 1								
Segment 2								
Segment 3								
Task 5	2.25	2.25	2	1	2			
Segment 1						0.5	1.0	[[[1, 1]], 1.0], [[2, 1]], 1.0]]
Segment 2						0.125	0.5	[[[1, 1]], 0.5], [[2, 1]], 1.0]]
Segment 3								

Do schedulability analysis

Taskset is schedulable

Upper bounds on the response times of task are as follows:

- For task 1: 0.5
- For task 2: 1.0
- For task 3: 0.5
- For task 4: 1.0
- For task 5: 1.75

OK

A Look at Our Analysis Tools - 3

```

corunnerinfopropertyset::corunnerstring => "[[task3],0.727217]]";
end mythread1.imp;

= thread mythread2
end mythread2;

= thread implementation mythread2.imp
properties
Priority           => 1;
Period            => 28734 ms;
Deadline          => 14147 ms;
Compute_Execution_Time => 40 ms,.40 ms;
corunnerinfopropertyset::taskid => "task2";
corunnerinfopropertyset::pd      => 0.05;
corunnerinfopropertyset::corunnerstring => "[[task3],0.616589]]";
end mythread2.imp;

= thread mythread3
end mythread3;

= thread implementation mythread3.imp
properties
Priority           => 2;
Period            => 2319 ms;
Deadline          => 1232 ms;
Compute_Execution_Time => 40 ms,.40 ms;
corunnerinfopropertyset::taskid => "task3";
corunnerinfopropertyset::pd      => 0.05;
corunnerinfopropertyset::corunnerstring => "[[task1],0.965866],[[task2],0.915646]]";
end mythread3.imp;

end mypakwithtaskset;

```

A Look at Our Analysis Tools - 4

The screenshot shows the Eclipse IDE with the AADL Navigator on the left and the AADL Editor on the right. A 'Message' dialog box is open in the center, displaying the following text:

```

Message
Taskset is schedulable
Upper bounds on the response times of task are as follows:
For task 1: 6.87526837243904
For task 2: 61.749311415110206
For task 3: 43.685004903641804
  
```

The background AADL code shows a taskset with three threads and their implementations:

```

corunnerinfopropertyset.aadl
corunnerinfopropertyset::corunnerstring => "[[[[task3],0.72717]]]";
end mythread1.imp;

thread mythread2
end mythread2;

thread implementation mythread2.imp

mythread2::mythread2::corunnerstring => "[[[[task3],0.616589]]]";
end mythread2.imp;

thread implementation mythread3.imp
properties
Priority => 2;
Period => 2319 ms;
Deadline => 1232 ms;
Compute_Execution_Time => 40 ns,.40 ns;
corunnerinfopropertyset::taskid => "task3";
corunnerinfopropertyset::pd => 0.05;
corunnerinfopropertyset::corunnerstring => "[[[[task1],0.965866],[[task2],0.915646]]]";
end mythread3.imp;
end mypakwithtaskset;
  
```

A Look at Our Analysis Tools - 5

```
ba@ba-desktop: ~/ga_find_wcet
File Edit View Search Terminal Help
ba@ba-desktop:~/ga_find_wcet$ more myconf
0
262144
ba@ba-desktop:~/ga_find_wcet$ ./ga_find_wcet myconf ./bubblesort
0.306359
ba@ba-desktop:~/ga_find_wcet$ time ./bubblesort < ascending_integers.dat

real    0m0.002s
user    0m0.001s
sys     0m0.001s
ba@ba-desktop:~/ga_find_wcet$ time ./bubblesort < descending_integers.dat

real    0m0.275s
user    0m0.275s
sys     0m0.000s
ba@ba-desktop:~/ga_find_wcet$ time ./bubblesort < worstcaseinput_GA_inputfile.dat

real    0m0.308s
user    0m0.308s
sys     0m0.000s
ba@ba-desktop:~/ga_find_wcet$ █
```

Publications

B. Andersson, H. Kim, D. de Niz, M. Klein, R. (Raj) Rajkumar, and J. Lehoczky, "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times, " ACM Transactions on Embedded Computing Systems, Vol. 17, No. 3, Article 71. May 2018.

B. Andersson, D. de Niz, and M. Klein, "The Multicore Challenge in Assured Autonomy," AUVSI Exponential 2021.

H. Kim, D. de Niz, B. Andersson, M. Klein and J. Lehoczky, "Addressing Multi-Core Timing Interference using Co-Runner Locking," IEEE Real-Time Systems Symposium, 2021.

V. Petrucci, B. Andersson, E. Massa, G. Lima, "Heterogeneous Quasi-Partitioned Scheduling," IEEE Real-Time Systems Symposium, 2021.

Tools

<http://www.andrew.cmu.edu/user/banderss/projects.html>

Team Photos



Bjorn Andersson (SE/CMU)



Dionisio de Niz (SE/CMU)



Mark Klein (SE/CMU)



John Lehoczky (CMU)



Hyoseung Kim (UCR)



Bill Anderson (SE/CMU)



Anton Dimov Hristozov (SE/CMU)

Contact Information

Presenter / Point of Contact match to Information Sheets

Bjorn Andersson, Ph.D.

Principal Researcher

Telephone: +1 412.268.9243

Email: baandersson@sei.cmu.edu

Thanks!