Using All Processor Cores While Being Confident about Timing

November 2021

Bjorn Andersson
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)

Commonality of DoD Systems

Computer

Program

Sensor

Physical Environment

Actuator

Research Review 2021

Commonality of DoD Systems

Computer

Program

Sensor

Physical Environment

Actuator

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.
Commonality of DoD Systems

Computer

Program

Sensor

Physical Environment

Actuator

Time

Read
Sensor

Actuate
Command
Commonality of DoD Systems
Commonality of DoD Systems

Computer

Program

Physical Environment

Period

Read Sensor

Actuate Command

Actuate Command

Read Sensor

Sensor

Actuator

Deadline

Time

Research Review 2021

Using All Processor Cores While Being Confident about Timing
© 2021 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.
Commonality of DoD Systems
Commonality of DoD Systems

<table>
<thead>
<tr>
<th>Read Sensor</th>
<th>Actuate Command</th>
<th>Read Sensor</th>
<th>Actuate Command</th>
<th>Read Sensor</th>
<th>Actuate Command</th>
</tr>
</thead>
</table>

Time

Computer

Program

Sensor

Actuator

Physical Environment
Why is Satisfying Real-Time Requirements Challenging?
What Causes Delay of Software?
What Causes Delay of Software?
What Causes Delay of Software?

Time when one thread in the software system arrives

Deadline

Time
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
- Time when one thread in the software system arrives
- Deadline
- Time
What Causes Delay of Software?

Preemption:
Another thread uses
the processor.

Time when one thread
in the software system arrives

Deadline

Time
What Causes Delay of Software?

Thread requests a critical section held by another thread

Time when one thread in the software system arrives

Deadline

Time
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

Core 1
L1/L2

Core 2
L1/L2

Core 3
L1/L2

Speed=1

Arrives
Finishes
Time
How Co-Runners Impact Speed of Execution

Core 1
L1/L2

Core 2
L1/L2

Core 3
L1/L2

Arrives    Finishes    Time
**Problem:** For each process, compute an upper bound on its response time.
Issues

• Shared hardware resources impact timing.
Issues

• Shared hardware resources impact timing.
• 103 times slowdown has been observed [Yun15].

Issues

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

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

Does it hold that for all scenarios, for each task, all its deadlines are met?
Overview of Our Solution

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?
Overview of Our Solution

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?

Design document / Source code
Period of task 1 is $T_1$

Software requirement specification
Task 1 must finish within $D_1$ time units from its arrival
Overview of Our Solution

- Execution time of task 1 is $C_1$
- Design document / Source code
  Period of task 1 is $T_1$
- Software requirement specification
  Task 1 must finish within $D_1$ time units from its arrival
- Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?
Overview of Our Solution

Determine Worst-Case Execution Time

- Execution time of task 1 is $C_1$
- Design document / Source code
  Period of task 1 is $T_1$
- Software requirement specification
  Task 1 must finish within $D_1$ time units from its arrival

Determine Speed as Function of Co-Runners

- Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?

Task 1 must finish within $D_1$ time units from its arrival

Determine Worst-Case Execution Time

- Execution time of task 1 is $C_1$
- Design document / Source code
  Period of task 1 is $T_1$
- Software requirement specification
  Task 1 must finish within $D_1$ time units from its arrival

Determine Speed as Function of Co-Runners

- Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?

Task 1 must finish within $D_1$ time units from its arrival

Determine Worst-Case Execution Time

- Execution time of task 1 is $C_1$
- Design document / Source code
  Period of task 1 is $T_1$
- Software requirement specification
  Task 1 must finish within $D_1$ time units from its arrival

Determine Speed as Function of Co-Runners

- Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?

Task 1 must finish within $D_1$ time units from its arrival

Determine Worst-Case Execution Time

- Execution time of task 1 is $C_1$
- Design document / Source code
  Period of task 1 is $T_1$
- Software requirement specification
  Task 1 must finish within $D_1$ time units from its arrival

Determine Speed as Function of Co-Runners

- Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

Schedulability analysis

Does it hold that for all scenarios, for each task, all its deadlines are met?

Task 1 must finish within $D_1$ time units from its arrival
Overview of Our Solution

Determine Worst-Case Execution Time
- Execution time of task 1 is $C_1$
- Period of task 1 is $T_1$
- Software requirement specification
  - Task 1 must finish within $D_1$ time units from its arrival

Determine Speed as Function of Co-Runners
- Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

Schedulability analysis
- Does it hold that for all scenarios, for each task, all its deadlines are met?

Executable code
- Actual hardware

yes/no/undecided
response times
Our Tools Perform These Tasks

- Determine Worst-Case Execution Time
- Determine Speed as Function of Co-Runners
- Schedulability analysis
  Does it hold that for all scenarios, for each task, all its deadlines are met?
Our Abstraction Speed as Function of Co-Runners Is New

- Executable code
- Actual hardware
- Determine Speed as Function of Co-Runners

Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

...
Our Abstraction Speed as Function of Co-Runners Allows undoc hw

Executable code

Actual hardware

Determine Speed as Function of Co-Runners

Speed of execution of task 1 is 0.6 if it executes in parallel with task 2

....
### A Look at Our Tools - 1

#### Main Idea

Fixed-point iteration where each iteration solves a linear program
A Look at Our Tools - 2

Taskset is schedulable. Upper bounds on the response times of task are as follows:
- For task1: 6.875526837243904
- For task2: 61.740314131532026
- For task3: 43.6950994903641804

OK
Main Idea

Use genetic algorithms

Fitness = execution time

Gene = input to program

Maximize fitness

= Maximize execution time
A Look at Our Tools - 4

Main Idea

Use genetic algorithms

Fitness = slowdown

Gene = input to program

Maximize fitness

= Maximize slowdown
Publications


Tools

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

Transition success

Tools have been used in internal seminars at Army CCDC Aviation & Missile Center (AvMC)
Team Members

Bjorn Andersson
Principal Researcher
Software Solutions Division
Software Engineering Institute

Dionisio Di Niz
Technical Director, Assuring CyberPhysical Systems
Software Solutions Division
Software Engineering Institute

Mark Klein
Principal Technical Advisor
Software Solutions Division
Software Engineering Institute

John Lehoczky
Thomas Lord University
Professor of Statistics
Carnegie Mellon University

Hyoseung Kim
Associate Professor
Department of Electrical and Computer Engineering
University of California, Riverside

Bill Anderson
Senior Member of the Technical Staff (retired)
Software Solutions Division
Software Engineering Institute

Anton Dimov Hristozov
Software Engineer
Software Solutions Division
Software Engineering Institute
Contact Information

Point of Contact
Bjorn Andersson, Ph.D.
Principal Researcher
Telephone: +1 412.268.9243
Email: info@sei.cmu.edu
Thanks!