Compositional Mixed-Criticality Scheduling

Arvind Easwaran[1] and Insik Shin[2]

[1]Nanyang Technological University, Singapore
[2]Korea Advanced Institute of Science and Technology, Korea

December 2, 2014

NANYANG TECHNOLOGICAL UNIVERSITY

Mixed-Criticality Systems – Some Background

Systems with applications at different levels of importance “criticality”, sharing the same hardware platform

Mixed-Criticality Systems – Some Background

Systems with applications at different levels of importance "criticality", sharing the same hardware platform

- Nothing new, these systems have always been around (e.g., IMA in avionics)

Mixed-Criticality Systems – Some Background

Systems with applications at different levels of importance "criticality", sharing the same hardware platform

- Nothing new, these systems have always been around (e.g., IMA in avionics)
- Worst-case resource reservation was used as a mechanism to achieve partitioning/isolation
  - Uses highly reliable (and potentially pessimistic) Worst-Case Execution Time (WCET) estimates for reservation

Mixed-Criticality Systems – Some Background

Systems with applications at different levels of importance "criticality", sharing the same hardware platform

- Nothing new, these systems have always been around (e.g., IMA in avionics)
- Worst-case resource reservation was used as a mechanism to achieve partitioning/isolation
  - Uses highly reliable (and potentially pessimistic) Worst-Case Execution Time (WCET) estimates for reservation
- Ideal for achieving isolation, but not very efficient in resource utilization

Mixed-Criticality Systems – Recent Trends

- In 2007 a new Mixed-Criticality (MC) task model based on multiple execution time estimates was introduced
  - A less reliable estimate denoting a common scenario, and a more reliable estimate denoting a critical scenario

Mixed-Criticality Systems – Recent Trends

- In 2007 a new Mixed-Criticality (MC) task model based on multiple execution time estimates was introduced
  - A less reliable estimate denoting a common scenario, and a more reliable estimate denoting a critical scenario
  - Improves resource utilization
  - No mechanisms to deal with critical scenarios (assumed to be a rare event)

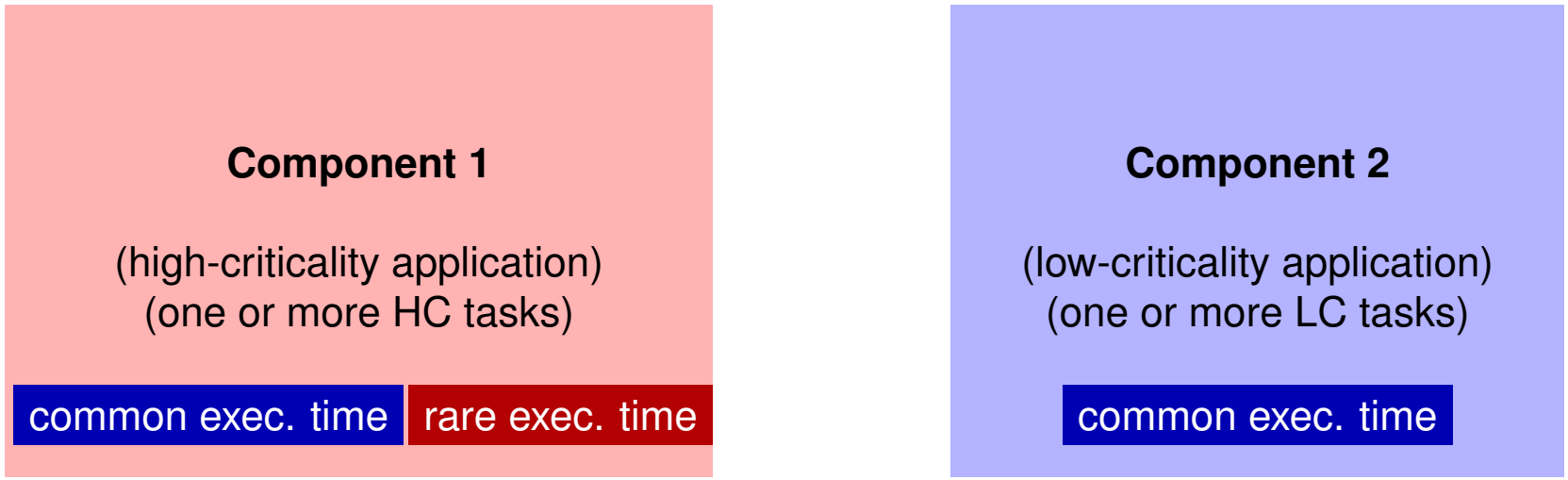
Mixed-Criticality Systems – Recent Trends

- In 2007 a new Mixed-Criticality (MC) task model based on multiple execution time estimates was introduced
  - A less reliable estimate denoting a common scenario, and a more reliable estimate denoting a critical scenario
  - Improves resource utilization
  - No mechanisms to deal with critical scenarios (assumed to be a rare event)
- Several studies have used (and extended) the above model to develop mixed-criticality scheduling theory
  - Assume all high-criticality tasks will simultaneously exhibit critical scenarios, and consequently suspend all low-criticality tasks

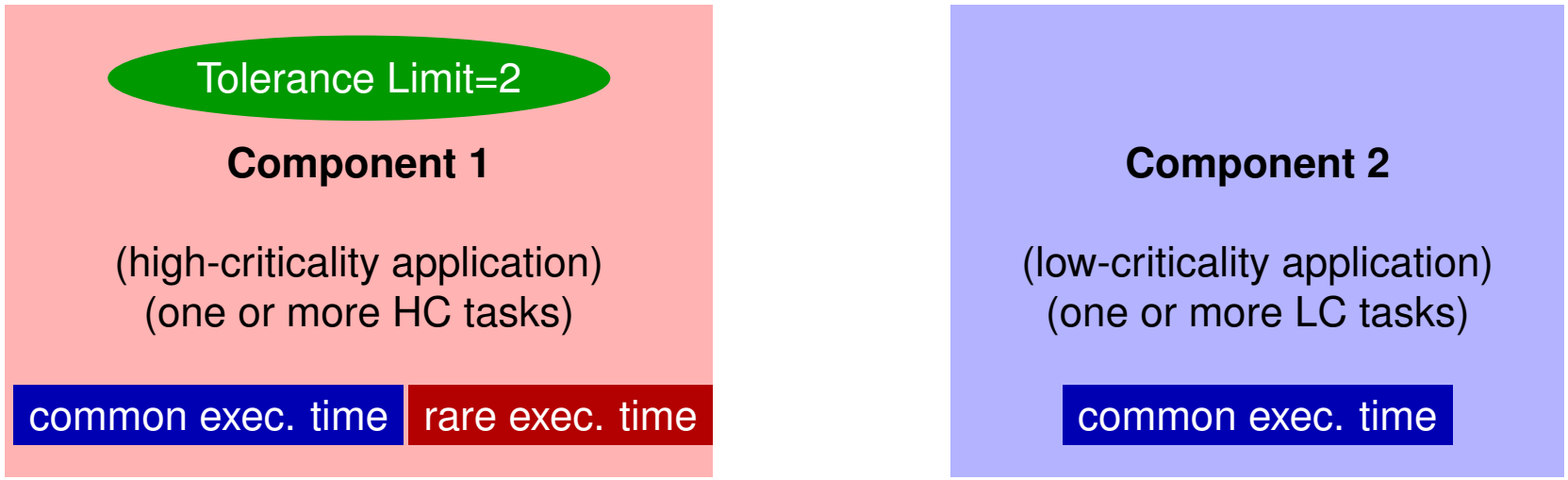
Shortcomings of Many Existing Studies

- Main issues identified in a workshop article in 2014 (WMC)
  - Why is it reasonable to assume that all high-criticality tasks will simultaneously exhibit critical scenarios?
  - Why suspend all low-criticality tasks even when a single high-criticality task exhibits critical scenario?
  - What is the consequence of abruptly suspending low-criticality tasks?
- Some research towards addressing these issues: elastic low-criticality releases, explicit dependency specification between high- and low-criticality tasks, . . .

Our Approach – Mixed-Criticality Components



Our Approach – Mixed-Criticality Components



Tolerance Limit denotes how many simultaneous critical scenarios should the component tolerate (designer tunable)

Our Approach – Mixed-Criticality Components

Execution time estimate for common scenario (crit. tasks <= 2)

Execution time estimate for common scenario

Tolerance Limit=2

Component 1

(high-criticality application)
(one or more HC tasks)

common exec. time | rare exec. time

Component 2

(low-criticality application)
(one or more LC tasks)

common exec. time

Tolerance Limit denotes how many simultaneous critical scenarios should the component tolerate (designer tunable)

NANYANG TECHNOLOGICAL UNIVERSITY

Our Approach – Mixed-Criticality Components

Execution time estimate for common scenario (crit. tasks <= 2)

Execution time estimate for critical scenario (crit. tasks > 2)

Execution time estimate for common scenario

Tolerance Limit=2

Component 1

(high-criticality application)
(one or more HC tasks)

common exec. time | rare exec. time

Component 2

(low-criticality application)
(one or more LC tasks)

common exec. time

Tolerance Limit denotes how many simultaneous critical scenarios should the component tolerate (designer tunable)

NANYANG TECHNOLOGICAL UNIVERSITY

Mixed-Criticality Component Model

1. Tolerance limit is a natural way to capture dependencies between HC tasks and provide isolation to LC tasks
   - Designers can use probability of simultaneous occurrence of task critical scenarios to determine this limit
   - Component boundaries provide isolation to other LC tasks as long as this limit is not violated
   - Offers trade-off between resource reservation for rare events and LC preservation through partitioning

Mixed-Criticality Component Model

1. Tolerance limit is a natural way to capture dependencies between HC tasks and provide isolation to LC tasks
   - Designers can use probability of simultaneous occurrence of task critical scenarios to determine this limit
   - Component boundaries provide isolation to other LC tasks as long as this limit is not violated
   - Offers trade-off between resource reservation for rare events and LC preservation through partitioning

2. Generalizes both worst-case reservation as well as existing popular academic approaches

Mixed-Criticality Component Model

1. Tolerance limit is a natural way to capture dependencies between HC tasks and provide isolation to LC tasks
   - Designers can use probability of simultaneous occurrence of task critical scenarios to determine this limit
   - Component boundaries provide isolation to other LC tasks as long as this limit is not violated
   - Offers trade-off between resource reservation for rare events and LC preservation through partitioning

2. Generalizes both worst-case reservation as well as existing popular academic approaches

3. Can be naturally extended to support hierarchical scheduling / compositional analysis

Open Problems

1. Can the proposed model bring mixed-criticality research closer to reality?
2. What interface models are suitable for mixed-criticality components?
3. What is the mechanism to convey scenario change across components?
4. How does one determine the tolerance limit? Is there some notion of optimality with respect to schedulability?
5. How can we link probability of scenario change to tolerance limit, and what guarantees can be provide then?