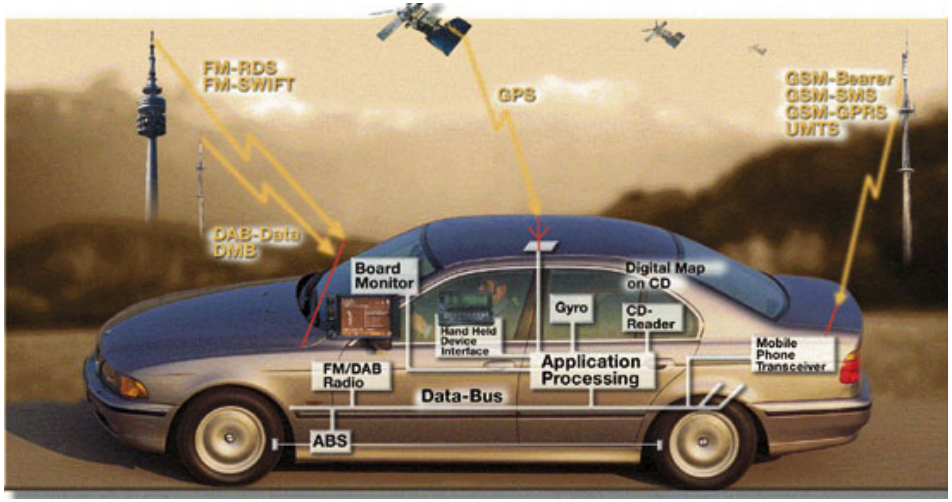
Cache-Aware Compositional Analysis of Real-Time Multicore Virtualization Platforms

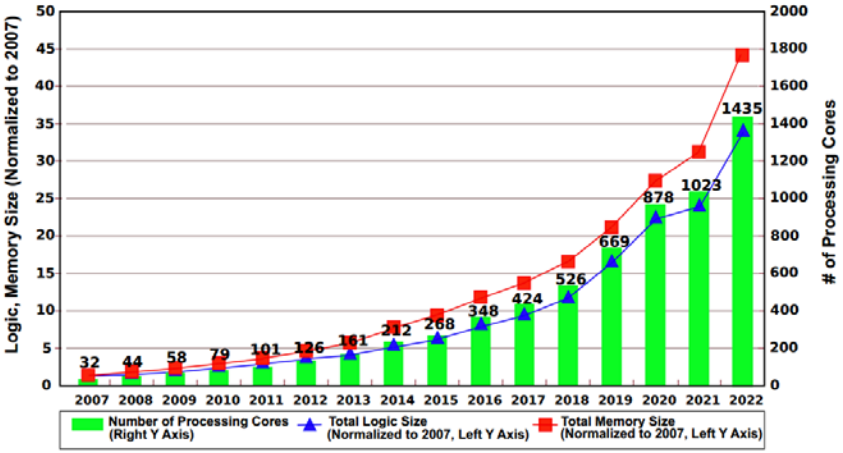
Meng Xu, Linh T.X. Phan, Insup Lee, Oleg Sokolsky,
Sisu Xi, Chenyang Lu and Christopher D. Gill

Complex Systems on Multicore Platforms

- Embedded systems
  - Become more and more complex
  - Consist of multiple sub-systems
- Multicore platforms
  - Number of cores keeps increasing



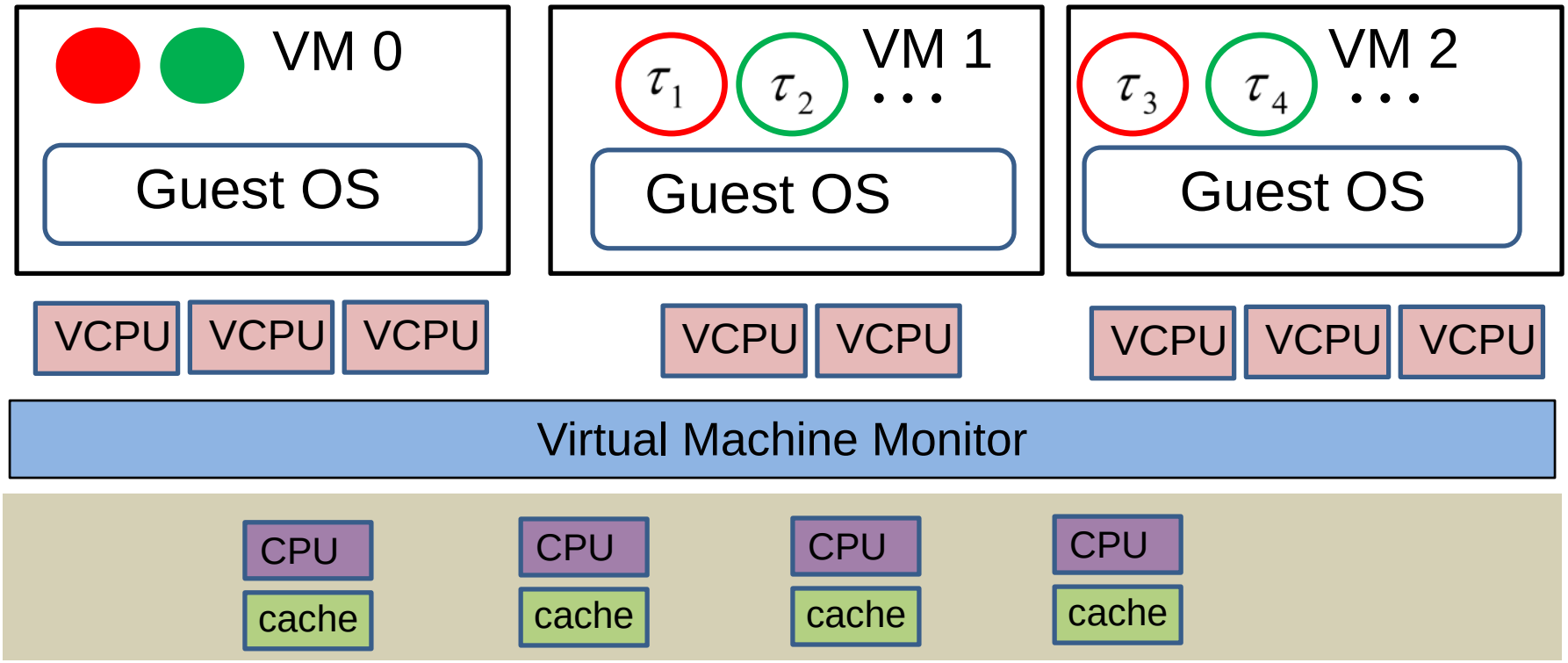
http://www.codeproject.com/Articles/16165/Robotics-Embedded-Systems-Part-I



International technology roadmap for semiconductors 2007 edition: System drivers

Virtualization

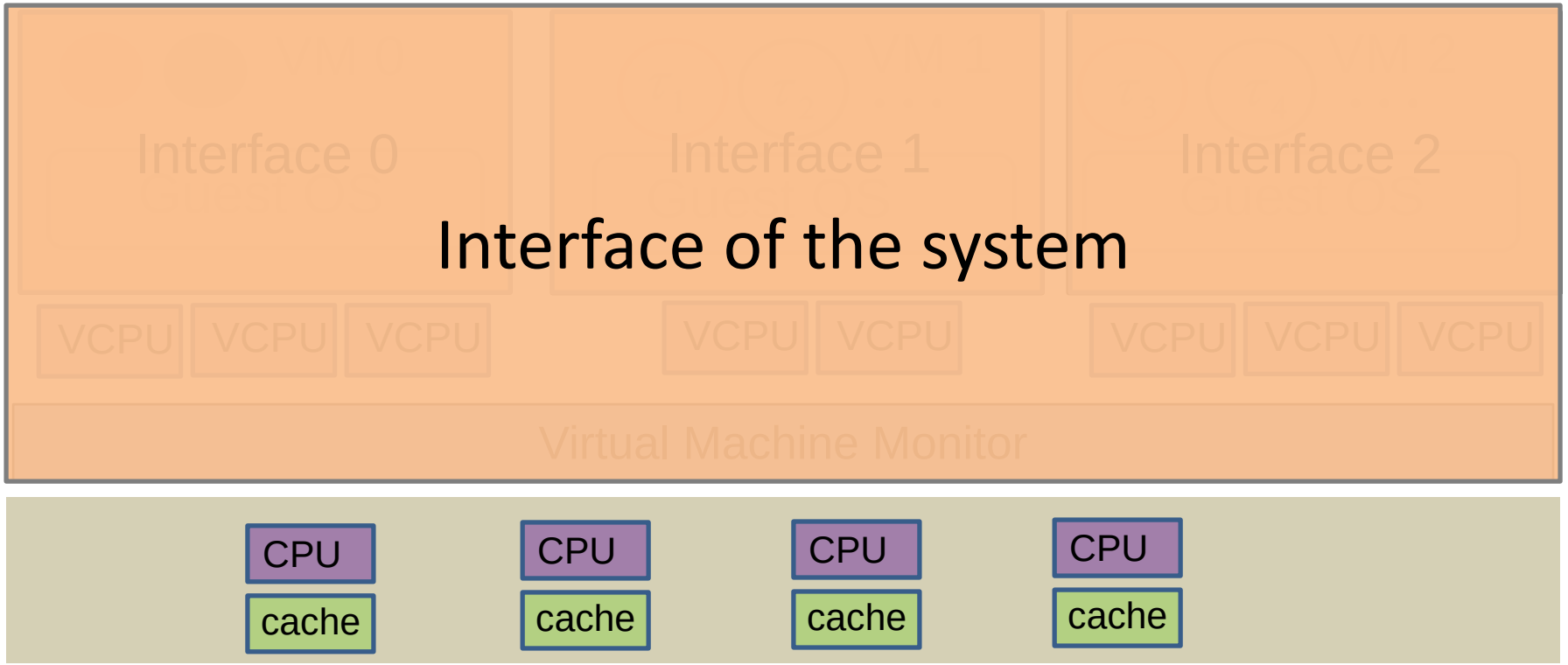
- The benefits of virtualization
  - Consolidate legacy systems
  - Integrate large, complex systems



Compositional Analysis for RT Guarantees

- Step 1: Abstract each component (VM) into an interface
- Step 2: Transform each interface into a set of VCPUs
- Step 3: Abstract the VCPUs of all VMs to the system's interface

VCPU: (Period, Budget)



Limitations of Existing Multicore Compositional Analysis

- Existing multicore compositional analysis does not consider platform overhead
- In practice, platform overhead is not negligible
  - Example: cache overhead
- Result: unsafe analysis!
  - Reason: analysis does not consider the effect of cache overhead in virtualization and under-estimates resource
  - Examples: cache overhead due to task preemption, VCPU preemption and VCPU completion

Contributions

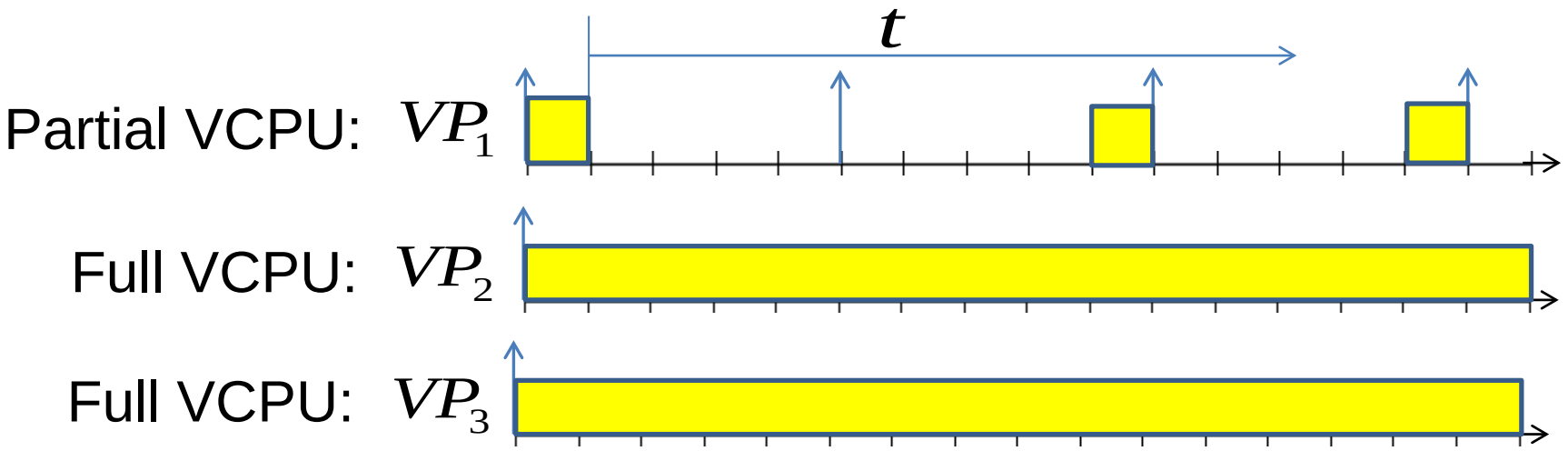
- Introduce overhead-free compositional analysis
  - DMPR: improved MPR resource model
- Quantify events that cause cache overhead
  - Task-preemption events, VCPU-preemption events, VCPU-completion events
- Propose cache-aware compositional analysis
  - Hybrid analysis: combination of task-centric analysis and model-centric analysis

Deterministic Multi-Processor Resource Model (DMPR)

DMPR $\mu = \langle \Pi, \Theta, m \rangle$ Interface Bandwidth = $m + \frac{\Theta}{\Pi}$

m full VCPUs (i.e., with bandwidth 1)

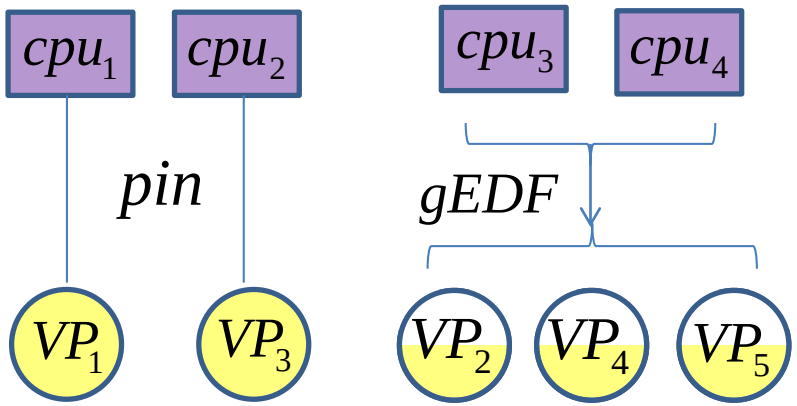
$\langle \Pi, \Theta \rangle$ one partial VCPU, with period Π and budget Θ



Worst-case resource supply of a DMPR $\mu = \langle 5, 1, 2 \rangle$

Assumptions

- Each core has a private cache; no shared cache
- Period of each component's interface is given by designers
- Maximum cache overhead per task preemption or migration in the system is upper bounded by Δ^{crpmd}
- Virtual machine monitor uses hybrid EDF (hEDF)



hEDF scheduling of VCPUs

Outline

- Introduction
- **Events that cause cache overhead**
- Cache-aware compositional analysis
- Evaluation

Event 1: Task Preemption Event

Definition: A task-preemption event happens when a task preempts another task within the same VM.

Example

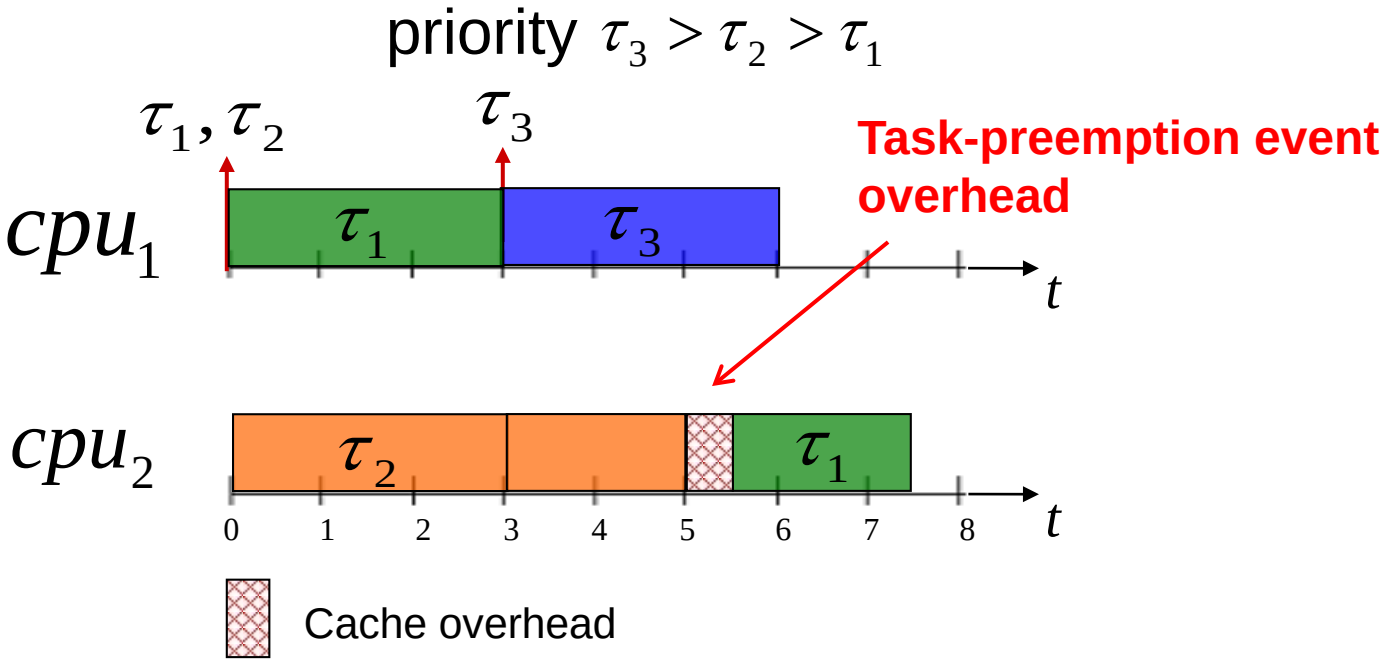
$\tau = \{\tau_1, \tau_2, \tau_3\}$

$\tau_1 = (12,5)$

$\tau_2 = (8,5)$

$\tau_3 = (4,3)$

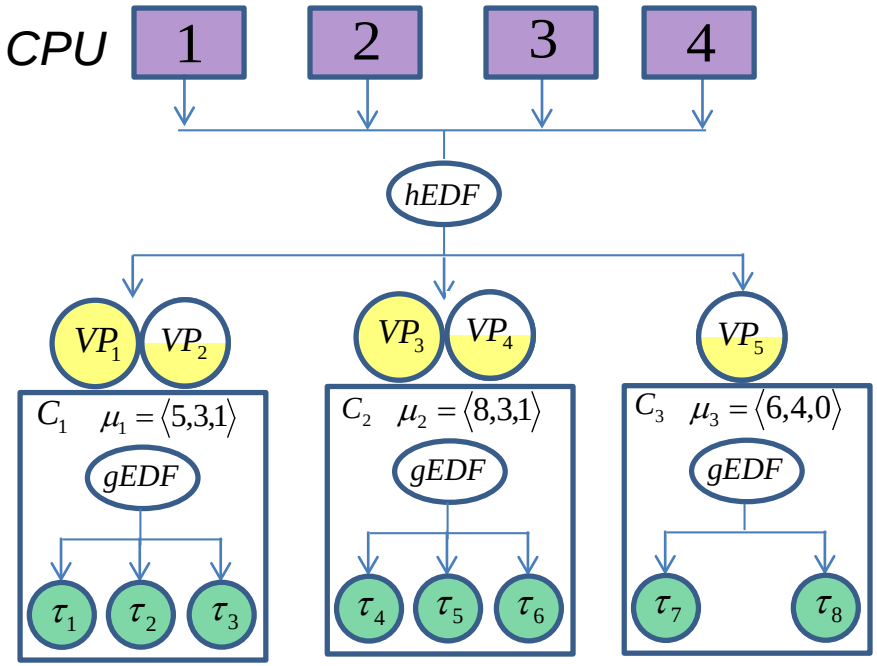
$gEDF$



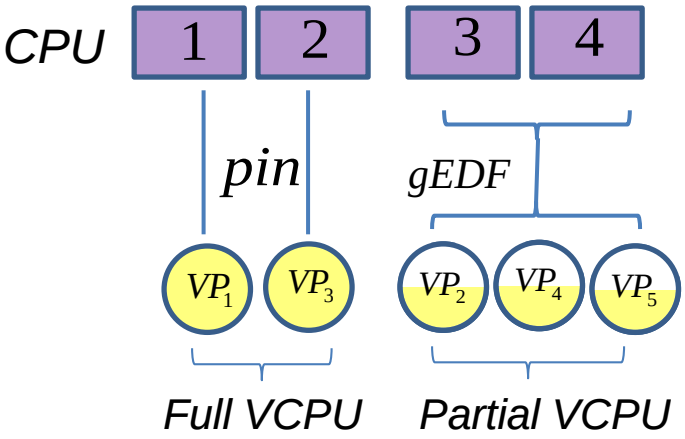
Event 2: VCPU-Preemption Event

Definition: A VCPU-preemption event occurs when a *VCPU* is preempted by another *VCPU* of *another VM*.

Example:

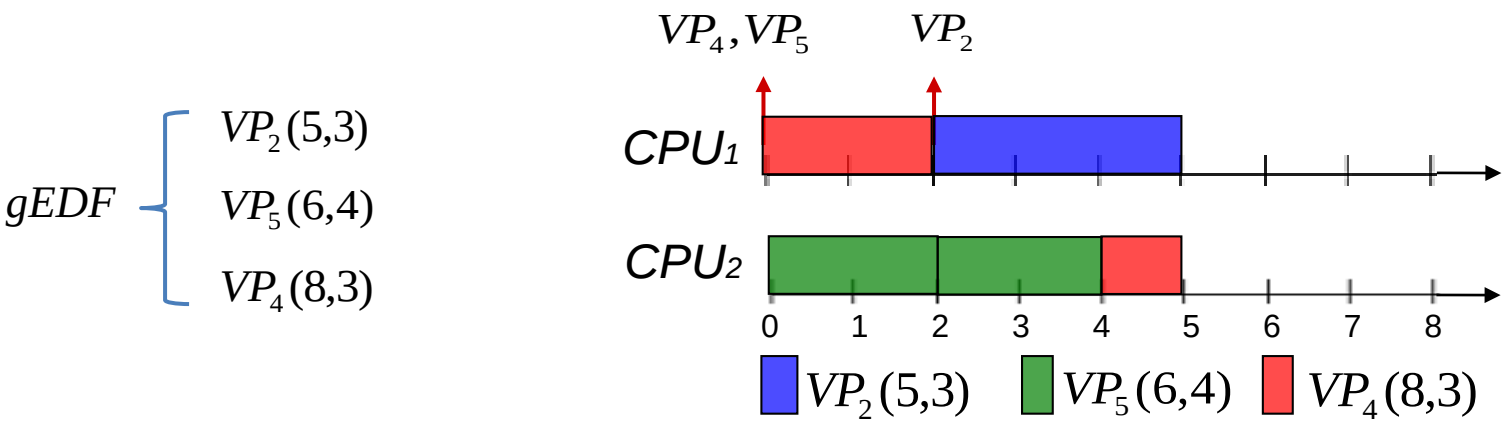


(a) VMs' configuration

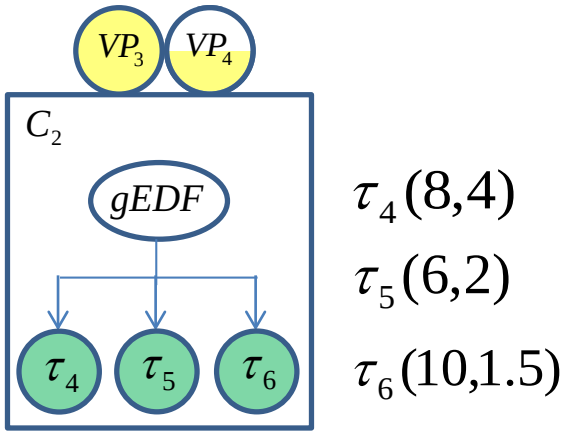


(b) VCPUs' configuration

Event 2: VCPU-Preemption Event



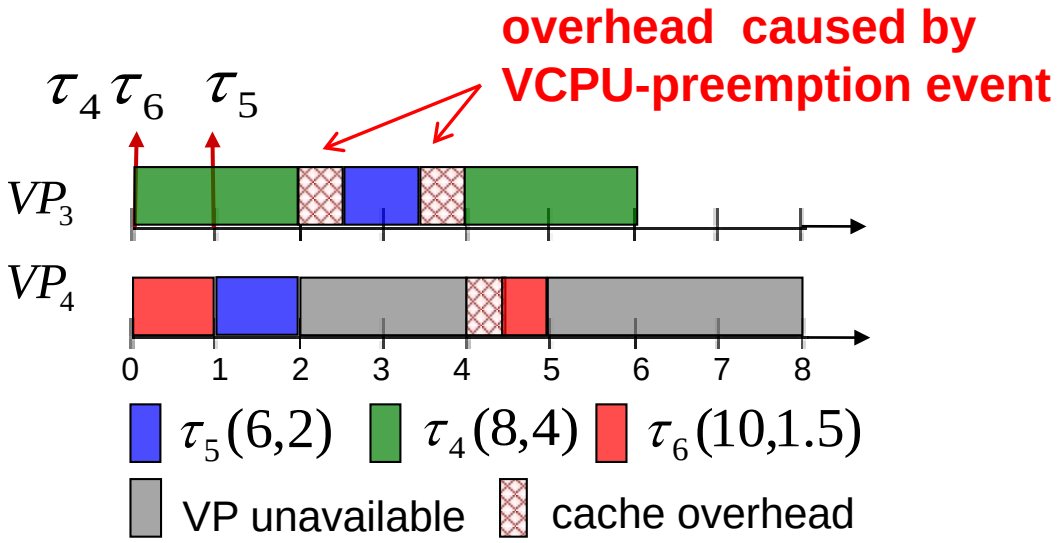
(c) Scheduling of partial VCPUs



$\tau_4(8,4)$

$\tau_5(6,2)$

$\tau_6(10,1.5)$

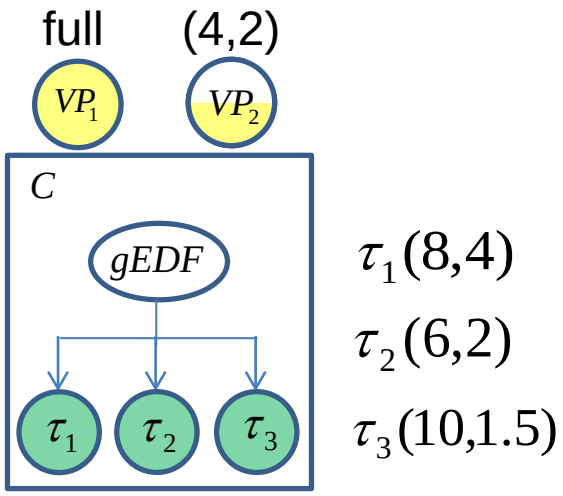


(d) Cache overhead of tasks in component 2

Event 3: VCPU-Completion Event

Definition: A VCPU-completion event of a *VCPU* happens when the *VCPU* exhausts its budget in a period and stops its execution.

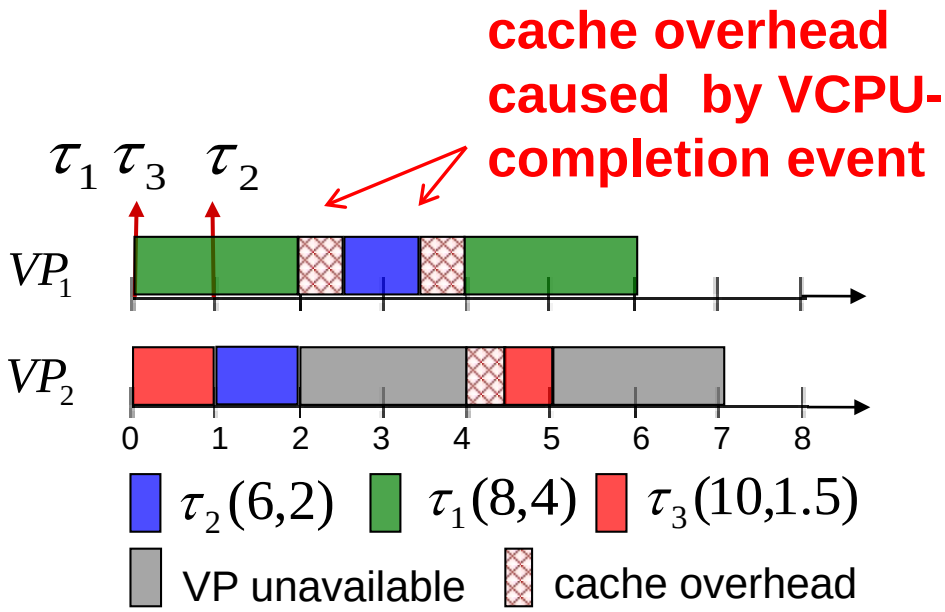
Example:



$\tau_1(8,4)$

$\tau_2(6,2)$

$\tau_3(10,1.5)$

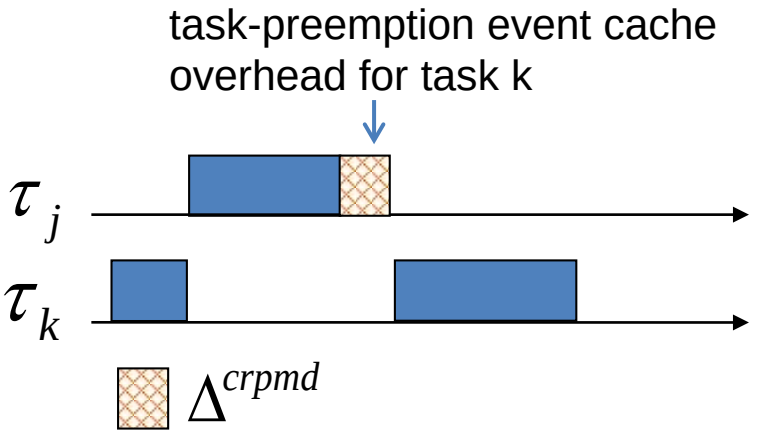


Outline

- Introduction
- Events that cause cache overhead
- **Cache-aware compositional analysis**
- Evaluation

Task-Centric Analysis

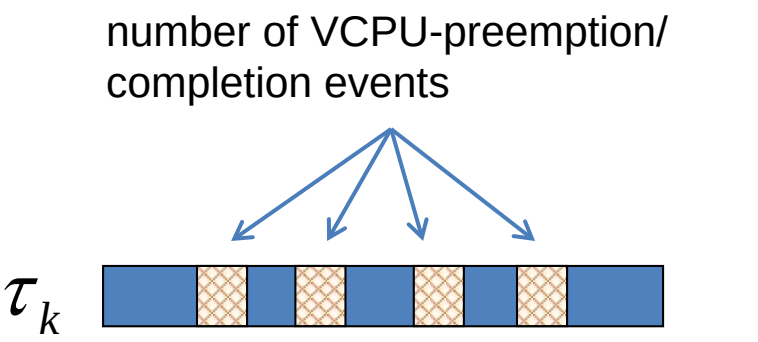
- Task-preemption event
  - Inflate higher priority task with one cache overhead
  - $e'_k = e_k + \Delta^{crpmd}$ (1)



(a) Task-preemption event overhead

- VCPU-preemption/completion event
  - Inflate task with the number of cache overhead caused by VCPU-preemption/completion events during a task's period

$$e'_k = e_k + \Delta^{crpmd}\left(N^2_{VP_i,\tau_k} + N^3_{VP_i,\tau_k}\right) \quad (2)$$



(b) VCPU-preemption event overhead during a period of task k

See paper for how to compute number of VCPU-preemption/completion events

Task-Centric Analysis

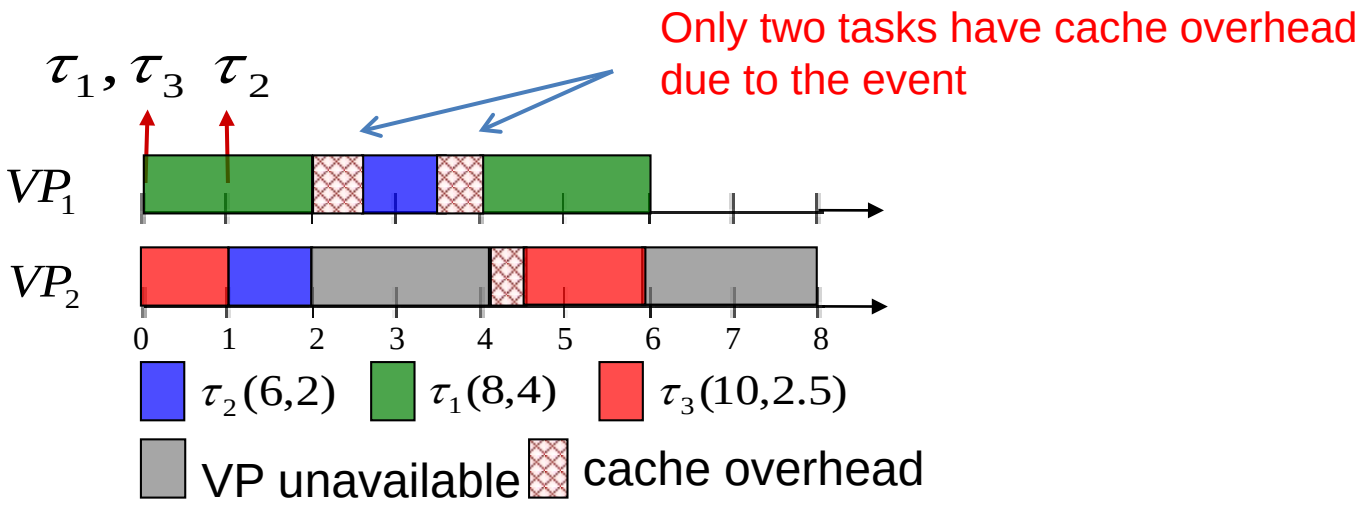
- Inflated WCET of each task

$$e'_k = e_i + \Delta^{crpmd} + \Delta^{crpmd}\left(N^{3}_{VP_i,\tau_k} + N^{2}_{VP_i,\tau_k}\right)$$

- System is schedulable under cache overhead ***if*** the inflated workload is schedulable

Pessimistic When Number of Tasks Is Large

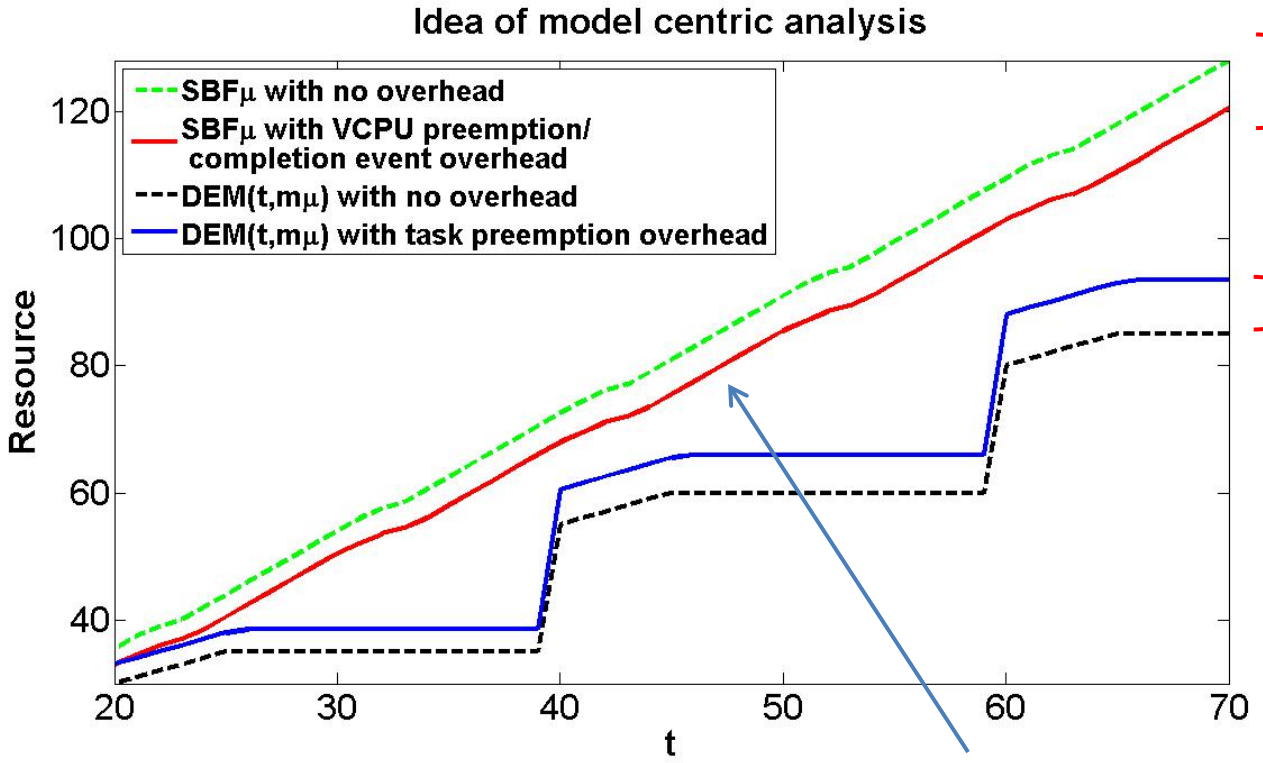
- Only two tasks have cache overhead in a VCPU-preemption/completion event
  - But don't know which two tasks have cache overhead
  - To be safe: have to inflate all tasks' WCET with one cache overhead per VCPU-preemption/completion event



Cache overhead in VCPU-completion event

Model-Centric Approach

- Subtract the overhead due to VCPU-preemption/completion events from the original resource supply of the interface to obtain its effective resource supply .



VCPU-preemption/ completion event overhead

Task-preemption event overhead

How to compute effective resource supply (red line)?

Effective SBF of DMPR Interface

Effective SBF of the partial VCPU

Effective SBF of m full VCPUs

Effective SBF of the interface

Reason: A DMPR interface provides resource with one partial VCPU and m full VCPUs

Worst Case Scenario of Effective Resource Supply of Partial VCPU:

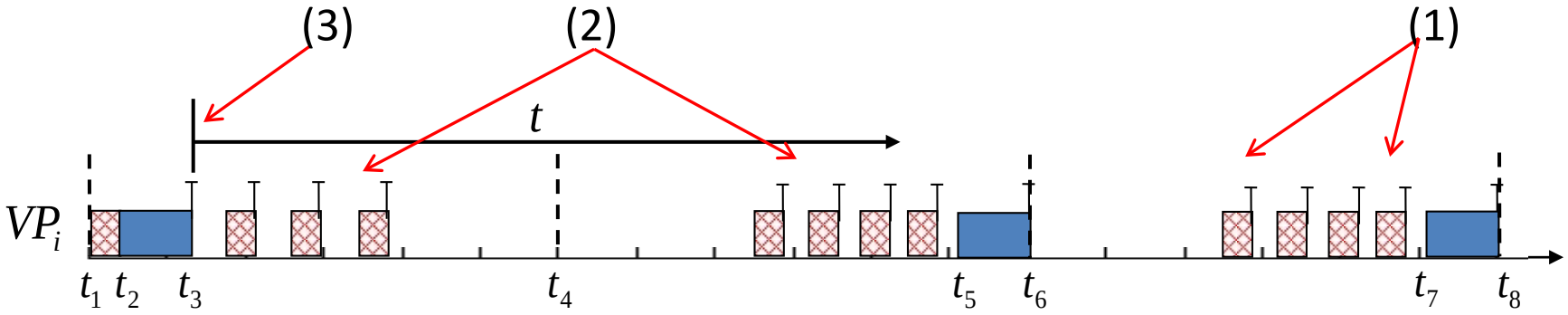
The worst case happens when:

(1) The partial *VCPU* has all VCPU-preemption/completion events

(2)The partial *VCPU* incurs the overhead as late as possible in the first period and as early as possible in the rest of periods

(3) The time interval *t* begins when the *VCPU* finishes supplying its effective resource in the first period.

Maximum number of VCPU-preemption/completion events during a partial VCPU's period is computed in the paper



Worst-case effective resource supply of *the partial VCPU*

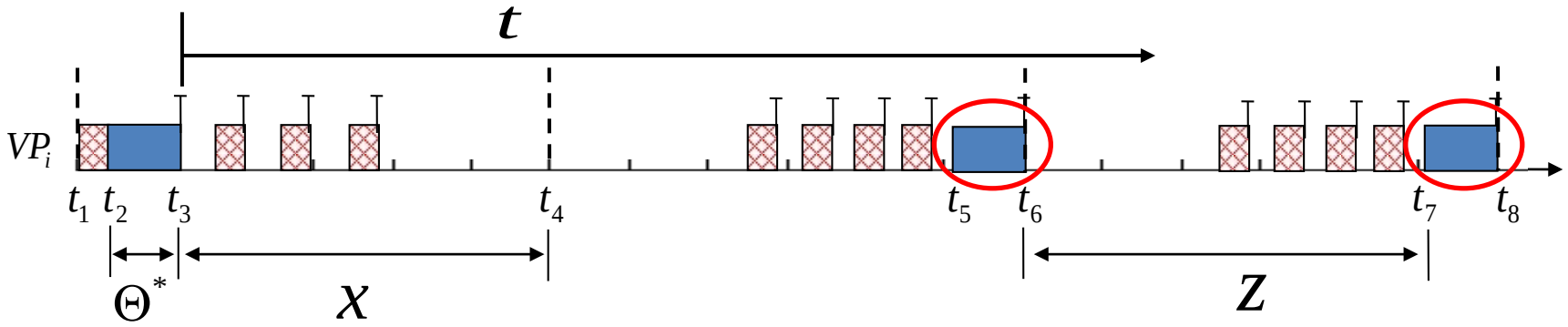
Proof is in the paper.

Effective Resource Supply of Partial VCPU

$$SBF_{VP_i}^{stop}(t) = \begin{cases} y\Theta^* + \max\{0, t - x - y\Pi - z\} & if\ \Theta \neq 0 \\ 0 & if\ \Theta = 0 \end{cases}$$

where VP_i belongs to interface $\mu = \langle \Pi, \Theta, m \rangle$

$$\Theta^* = \max\{0, \Theta - N_{VP_i}^{stop} \Delta^{crpmd}\},\ x = \Pi - \Delta^{crpmd} - \Theta^*\ \ y = \left\lfloor \frac{t - x}{\Pi} \right\rfloor \text{ and } z = \Pi - \Theta^*$$



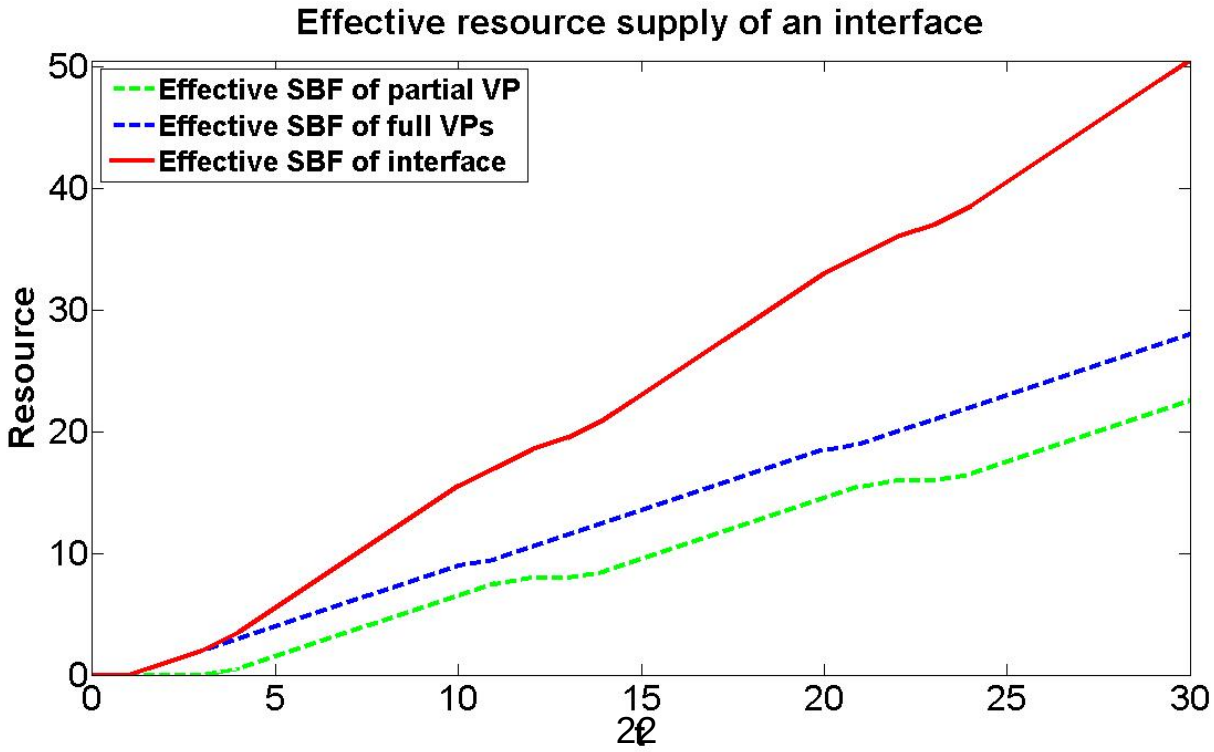
Worst-case effective resource supply of the partial VCPU VP_i

Effective SBF of The Interface

Effective SBF of the partial VCPU

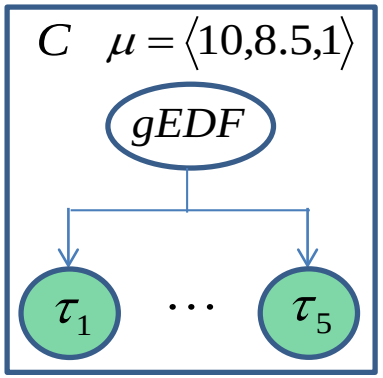
Effective SBF of m full VCPUs

Effective SBF of the interface

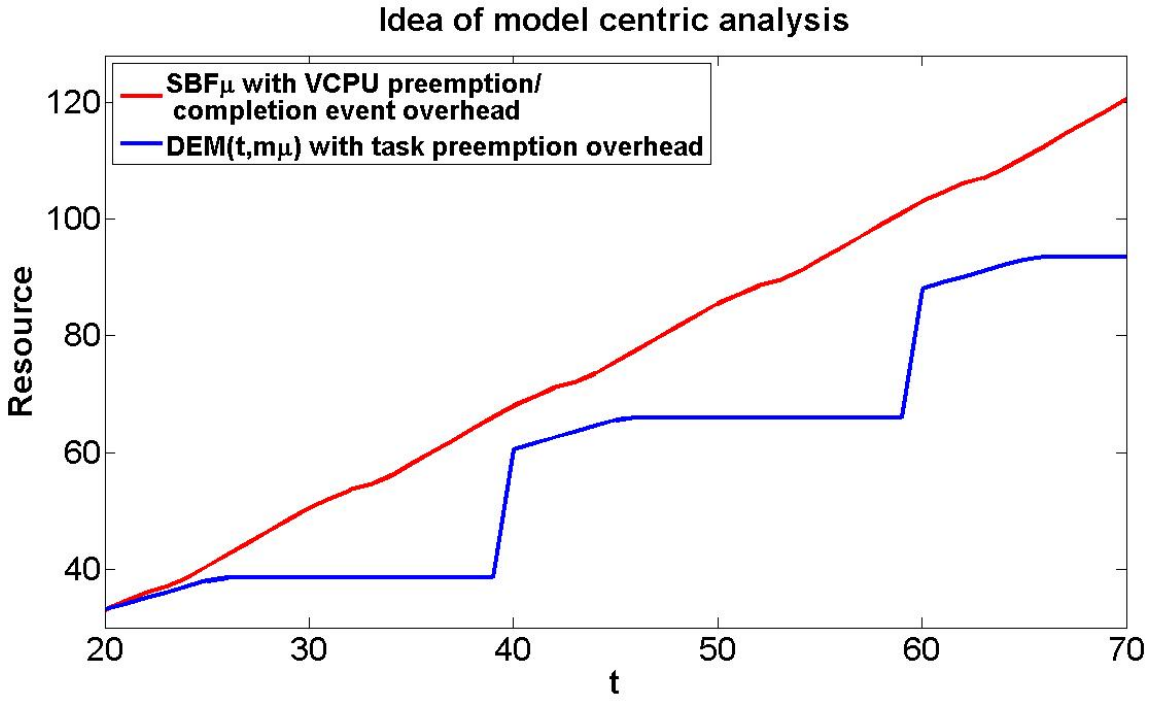


Model-Centric Analysis

- Step 1: Consider task-preemption event overhead
- Step 2: Consider VCPU-preemption/completion event overhead
- Step 3: Check ***if***
  - effective resource supply >= resource demand

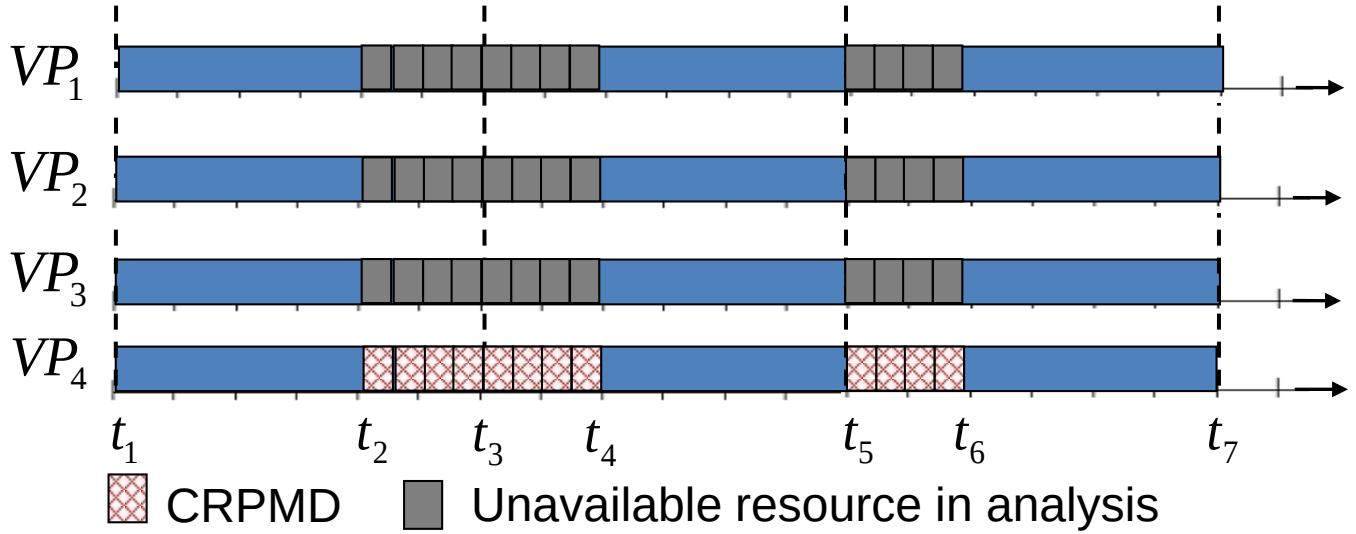


$\tau_1, \cdots, \tau_5 = (20,5)$



Pessimistic When Number of Full VCPUS Is Large

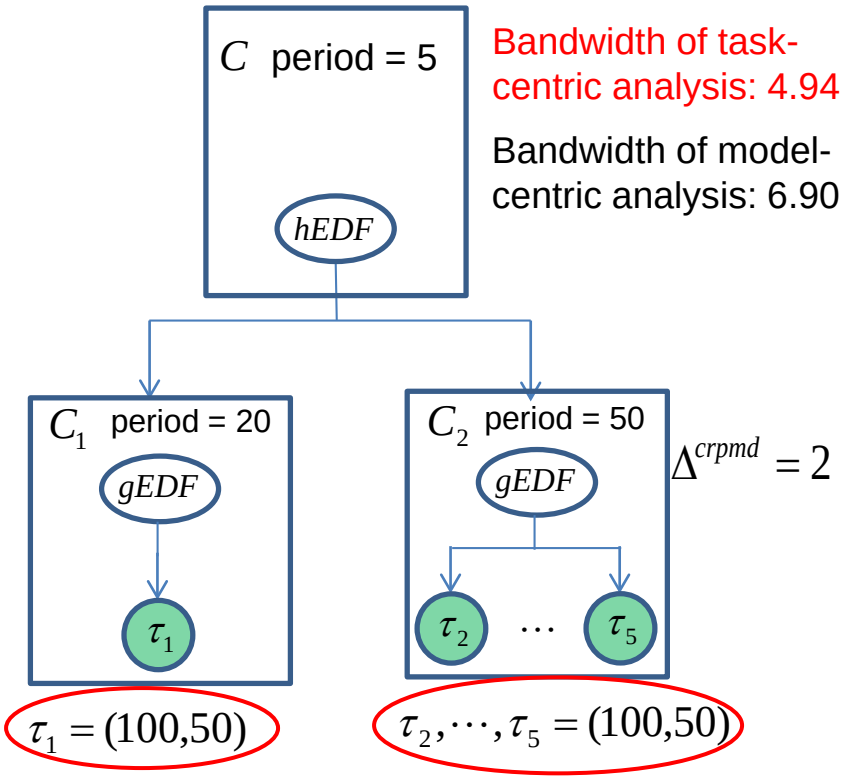
- Only one full VCPU is affected per VCPU-preemption/ completion event in practice
- But all full VCPUs marked unavailable at a VCPU-preemption/ completion event when we compute the effective SBF of m full VCPUs



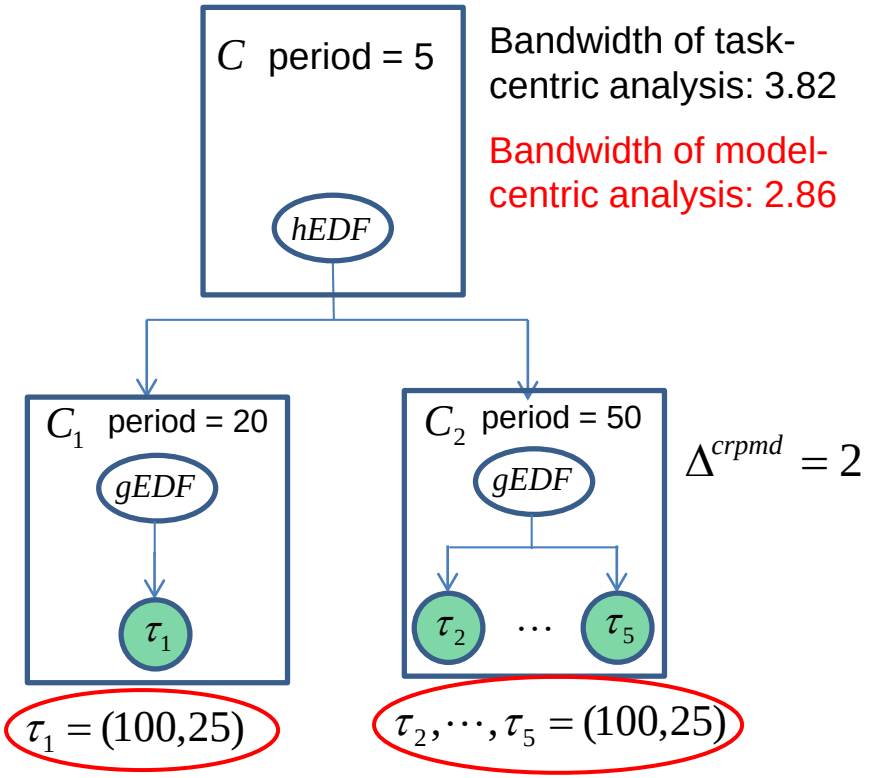
Task-Centric vs. Model-Centric

- Neither of these two analysis dominates the other

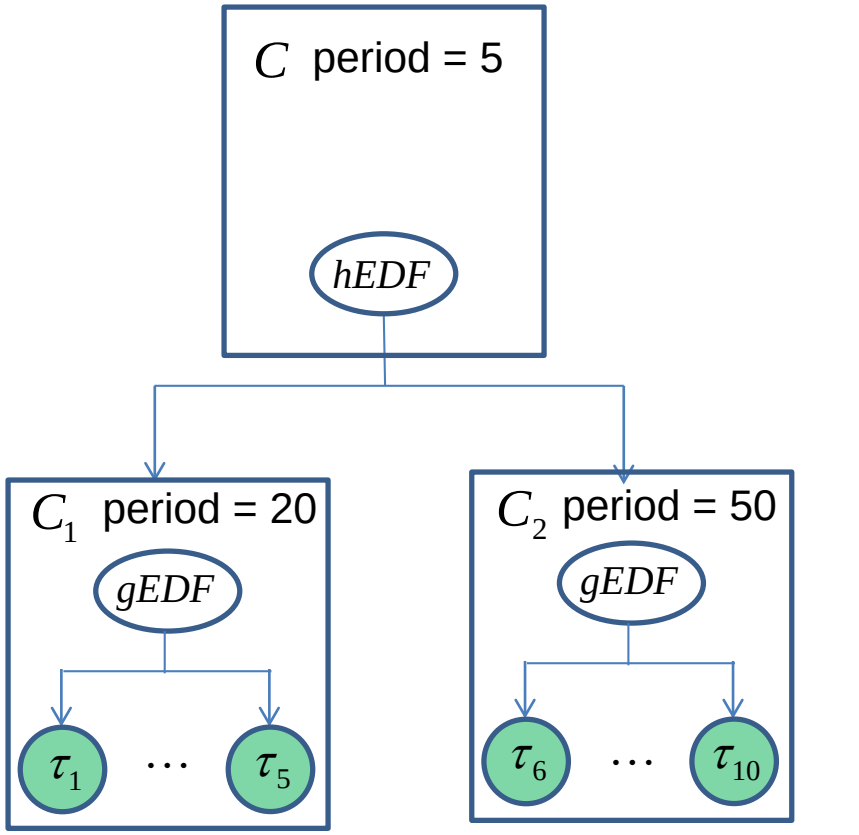
Task-centric is better



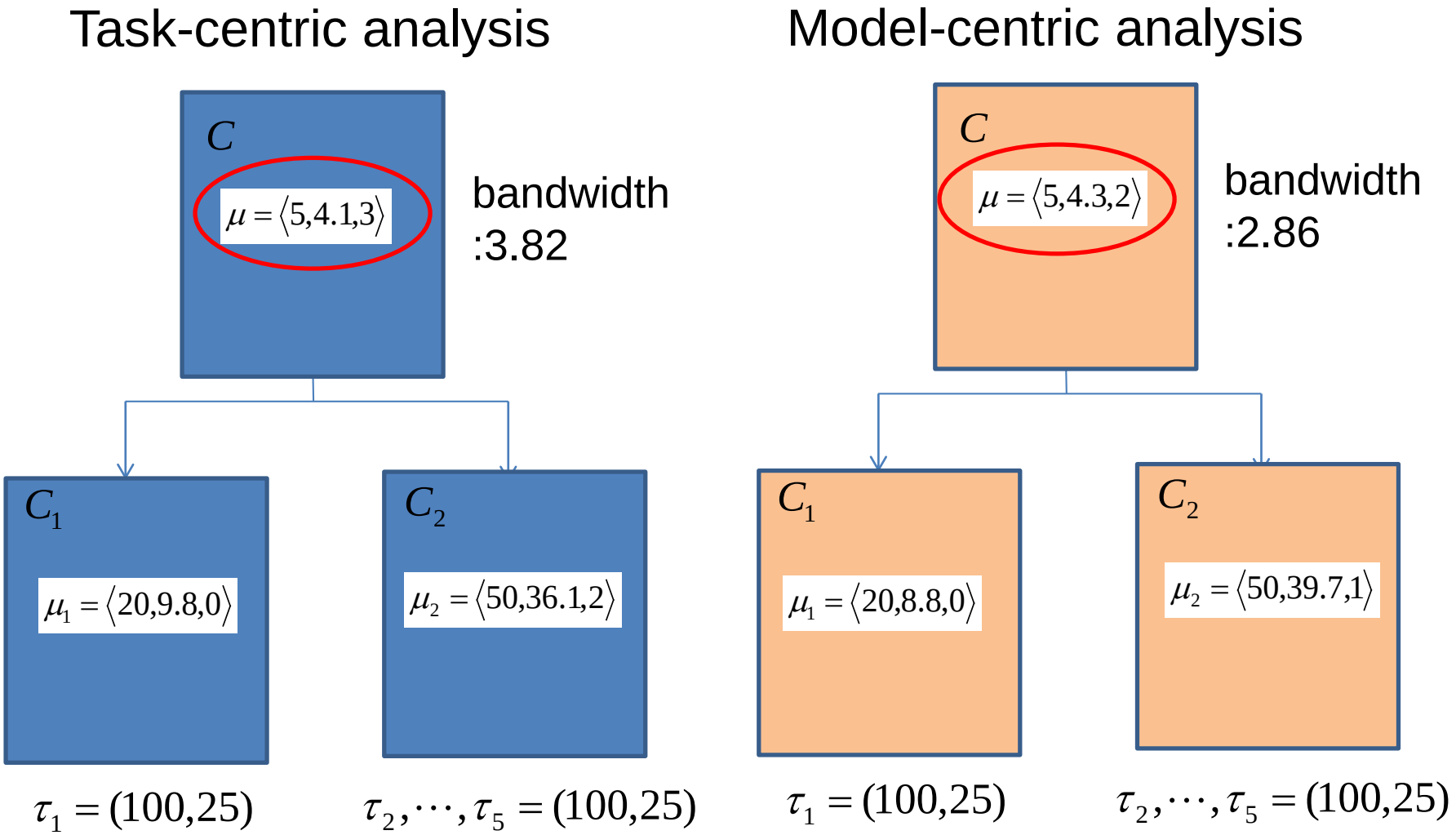
Model-centric is better



Hybrid Cache-Aware Analysis



Hybrid Cache-Aware Analysis



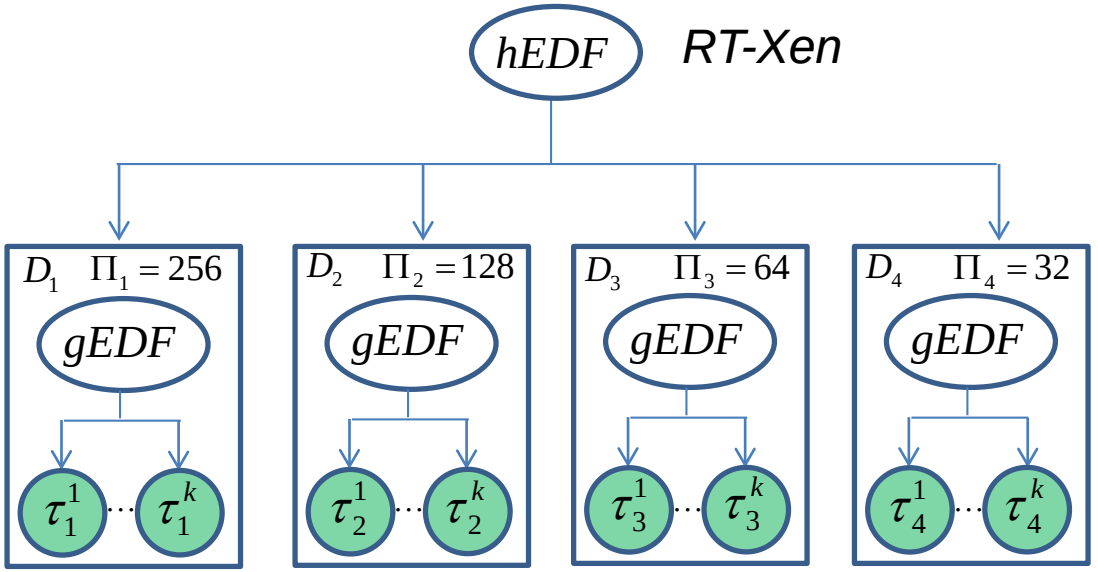
Outline

- Introduction
- Events that cause cache overhead
- Cache-aware compositional analysis
- **Evaluation**

Experimental Setup

Dell Optiplex-980 quad-core workstation
(3 cores for guest VMs, 1 core for VM0)

Hardware



LITMUS

measured

$\Delta^{crpmd} = 1.9$ ms

WSS=256KB

Task set: utilization 1.8;
Task utilization distribution: uniformly in [0.001,0.1]

Cache Overhead Is Not Negligible

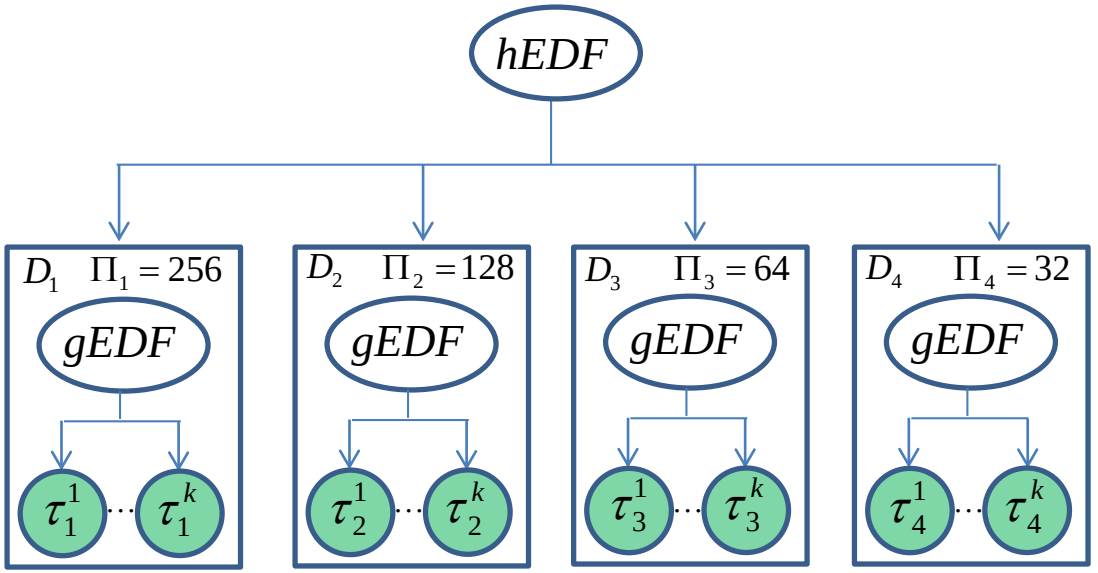
Unsafe—taskset claimed schedulabled by overhead-free analysis is not schedulable in practice

	MPR		DMPR	
	Theory	RT-Xen	Theory	RT-Xen
Schedulable	Yes	No	Yes	No

Safe—same taskset is claimed NOT schedulable by cache-aware analysis

	Cache-aware Hybrid		Cache-aware Task-centric	
	Theory	RT-Xen	Theory	RT-Xen
Schedulable	No	No	No	No

Simulation Setup

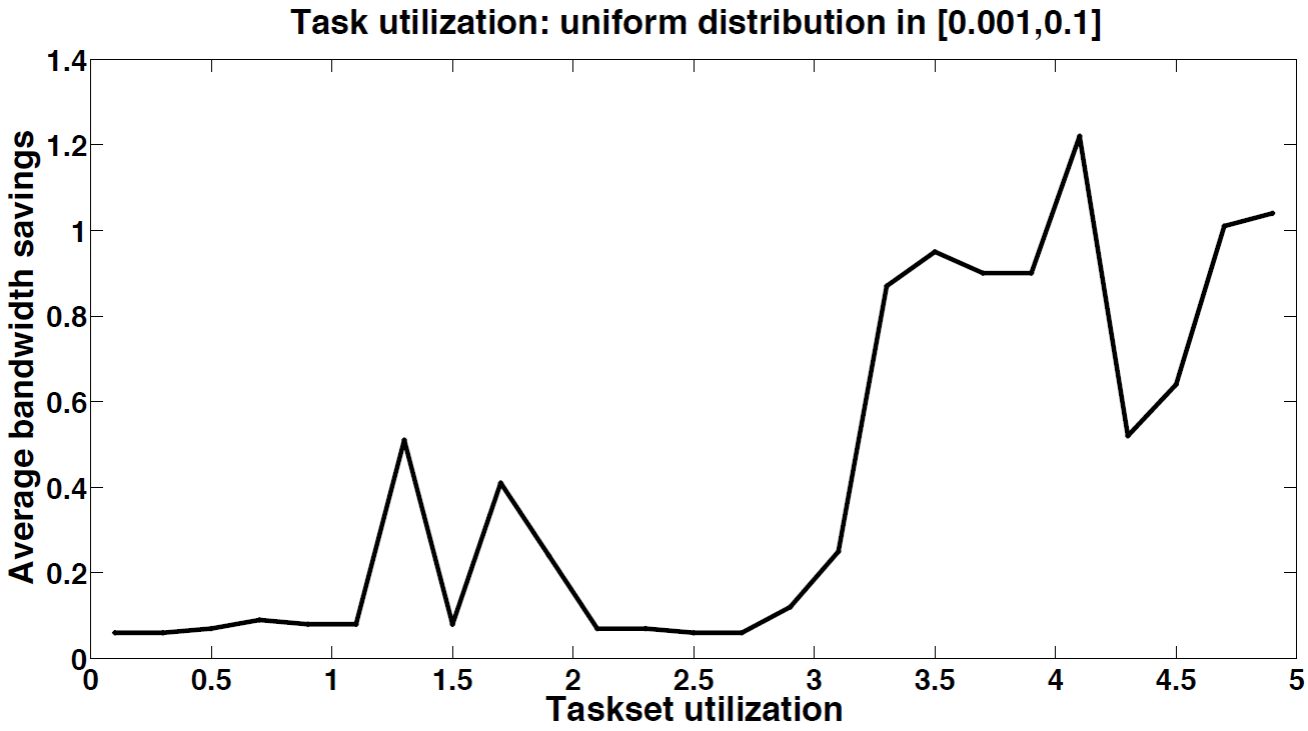


$$\Delta^{crpmd} = 0.9 \text{ ms}$$

Task's period	uniformly in [350ms, 850ms]	
Task's utilization	uniform	uniformly in [0.001,0.1]
	light bimodal	8/9 in [0.1,0.4] and 1/9 in [0.5,0.9]
	medium bimodal	6/9 in [0.1,0.4] and 3/9 in [0.5,0.9]
	heavy bimodal	4/9 in [0.1,0.4] and 5/9 in [0.5,0.9]

Hybrid Analysis Saves Bandwidth

Hybrid approach saves bandwidth for 64% of the tasksets

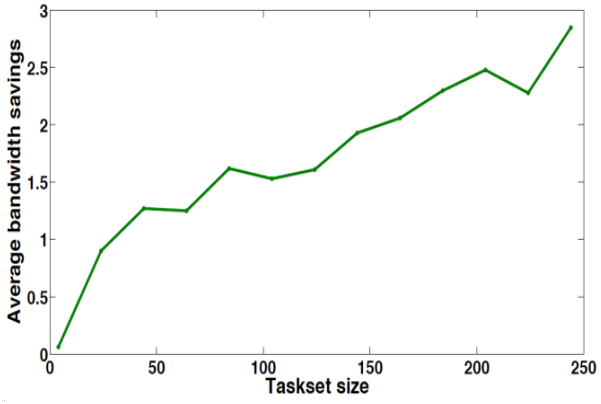


$$\frac{\Delta^{crpmd}}{\text{Average wcet}} = 0.003$$

Hybrid analysis saves bandwidth over task-centric analysis per taskset utilization

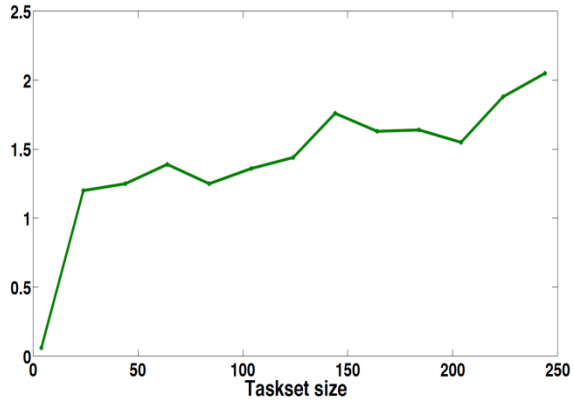
Hybrid Analysis Saves Bandwidth

Hybrid analysis still saves bandwidth over task-centric analysis when the distribution of tasks' utilization changes



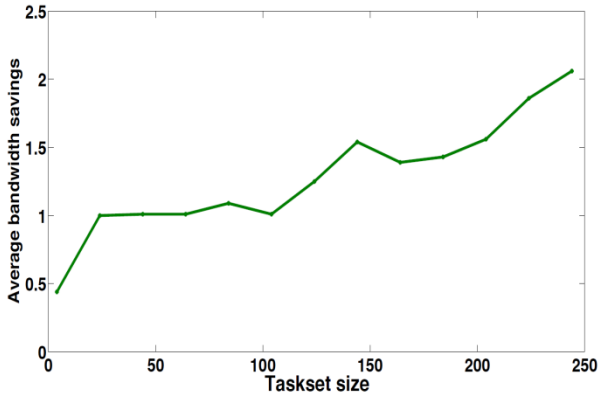
$$\frac{\Delta^{crpmd}}{\text{Average wcet}} = 0.0005$$

a) bimodal-light distribution



$$\frac{\Delta^{crpmd}}{\text{Average wcet}} = 0.0004$$

b) bimodal-medium distribution



$$\frac{\Delta^{crpmd}}{\text{Average wcet}} = 0.0003$$

c) bimodal-heavy distribution

Related Work

- Overhead-free compositional analysis
  - S. Baruah and N. Fisher. Component-based design in multiprocessor real-time systems. In ICESS, 2009.
  - A. Easwaran, I. Shin, and I. Lee. Optimal virtual cluster-based multiprocessor scheduling. Real-Time Systems, 43(1):25–59, 2009.
  - H. Leontyev and J. H. Anderson. A hierarchical multiprocessor bandwidth reservation scheme with timing guarantees. In ECRTS, 2008.
  - G. Lipari and E. Bini. A framework for hierarchical scheduling on multiprocessors: From application requirements to run-time allocation. In RTSS, 2010.
  - E. Bini, M. Bertogna, and S. Baruah. Virtual multiprocessor platforms: Specification and use. In RTSS, 2009.
- Overhead-aware analysis on non-virtualization environment
  - B. B. Brandenburg. Scheduling and Locking in Multiprocessor Real-Time Operating Systems. PhD thesis, The University of North Carolina at Chapel Hill, 2011.
- Methods of getting the cache overhead value
  - A. Bastoni, B. B. Brandenburg, and J. H. Anderson. Cache-Related Preemption and Migration Delays: Empirical Approximation and Impact on Schedulability. In OSPERT, 2010.
  - S. Altmeyer, R. I. Davis, and C. Maiza. Improved cache related preemption delay aware response time analysis for fixed priority preemptive systems. Real-Time Systems, 2012.

Conclusion

- Contribution
  - Propose DMPR resource model
  - Introduce overhead-free compositional analysis under DMPR
  - Quantify events that cause cache overhead
  - Propose cache-aware compositional analysis
- Future work
  - Extend our method to multi-level cache hierarchy with shared cache
  - Explore cache management methods to reduce the cache overhead