Fixed-Priority Multiprocessor Scheduling: To Partition or not to Partition

Stefan Voigt

Setting the stage

- Real-time scheduling vs. minimal makespan
  - Scheduling algorithms
  - Schedulability tests
- Periodic tasks vs. aperiodic tasks
- Fixed priority vs. dynamic priority
- Preemptive
- Partitioned vs. non-partitioned (global)
  - Comparison of both methods
  - Hybrid algorithms

Agenda

- Introduction
  - Characteristics of both methods (partitioned + global)
- Comparison of both methods
  - Introduction of some representatives
  - Performance
  - Preemption density
- Conclusion

Introduction

- Decision whether a task set is schedulable is NP-hard
- No method dominates the other
- But:

  Partitioned method
  - Provide performance guarantees
  - Good average case performance
  - Polynomial time (sufficient schedulability test)
- Non-partitioned method
  - Received much less attention
  - No efficient schedulability test exist (pessimistic)
  - No efficient priority-assignment scheme has been found

Partitioned method

- Two parts:
  - Dividing the task set into m groups
  - Scheduling each group locally on one processor
- The problem of scheduling each group of tasks on a processor is known
  - Rate monotonic scheduling (static priorities)
  - Earliest deadline first (dynamic priorities)
- Dividing the tasks into groups is NP-hard

Partitioned algorithm: RM-FFDU

Rate-Monotonic-First-Fit-Decreasing-Utilization

- Sort the task set (non-increasing utilization)
- Start with one processor
- For each task:
  - Try to assign the task τ_i to the processors P, starting with P_1
  - A task τ_i with utilization u_i can be assigned to P_j when:

    $$u_i \leq 2 / \prod_{l=1}^{k_j} \left(u_{j,l} + 1\right) - 1$$ (k_j = number of tasks assigned to P_j)
  - If τ_i cannot be assigned to the existing m processors, m will be increased by one and τ_i will be assigned to P_m

Partitioned algorithm: RM-FFDU

Example

u_1=0.6

u_2=0.5

u_3=0.4

u_4=0.3

u_5=0.2

u_6=0.1

$$u_i \leq 2 / \prod_{l=1}^{k_j} \left(u_{j,l} + 1\right) - 1$$

$u \leq 2/1.6/1.1 - 1 = 0.24$

P$_1$ $u \leq 0.25$

P$_2$ $u \leq 0.33$

P$_3$ $u \leq 0.42$

Partioned algorithm: RM-FFDU

Evaluation

- Worst-case tight bound of 5/3 (comparison with optimal scheduling)
- Worst case:
  - n =15*k ($k \in$N)
  - u_i =0.2 $\forall$ i $\in$[1;n]
  - RM-FFDU: 3 tasks per processor, 15k/3 = 5k processors

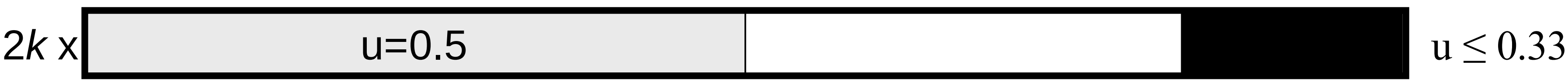
5k x	u=0.2	u=0.2	u=0.2			$u \leq 0.15$

  - Optimal scheduling: 5 tasks per processor, 15k/5 = 3k processors

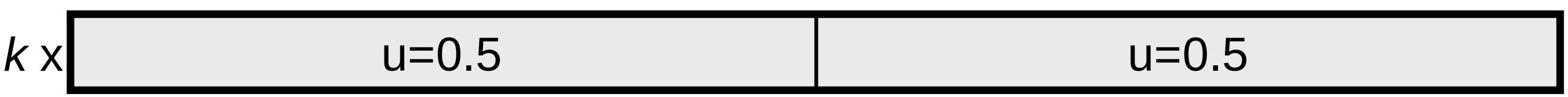
3k x	u=0.2	u=0.2	u=0.2	u=0.2	u=0.2

Partitioned algorithm: RM-FFDU
Evaluation II

- Worst-case tight bound of 2?
- Worst case:
  - $n = 2*k$ ($k \in$ N)
  - $u_i = 0.5 \ \forall \ i \in [1;n]$
  - RM-FFDU: 1 tasks per processor, $2k/1 = 2k$ processors



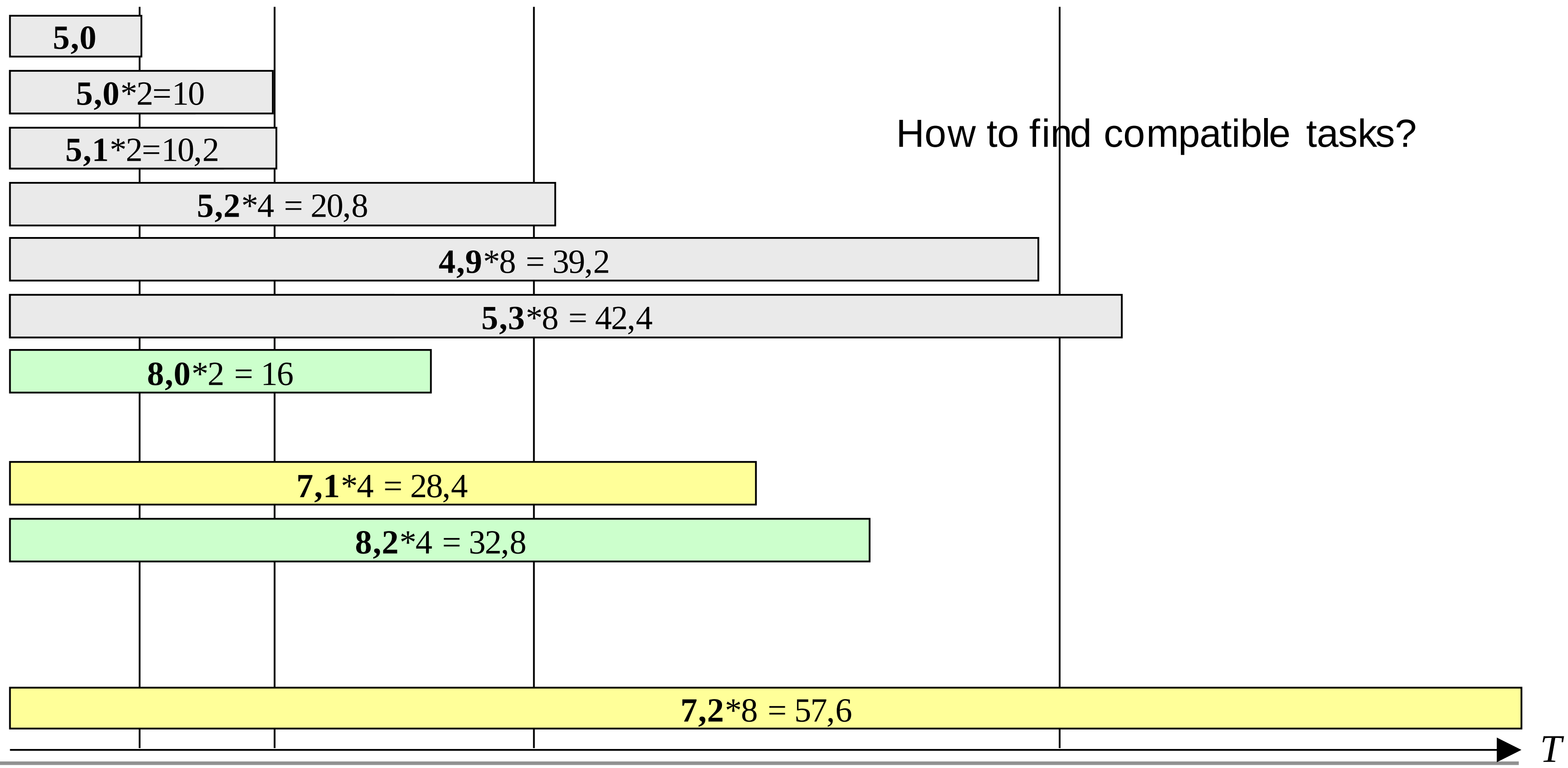
  - Optimal scheduling: 2 tasks per processor, $2k/2 = k$ processors



- $O(nm + n \log n)$

Partitioned algorithm: R-BOUND-MP
Compatible Tasks

- Tasks with same period
- Tasks that have a period closest to a power of two

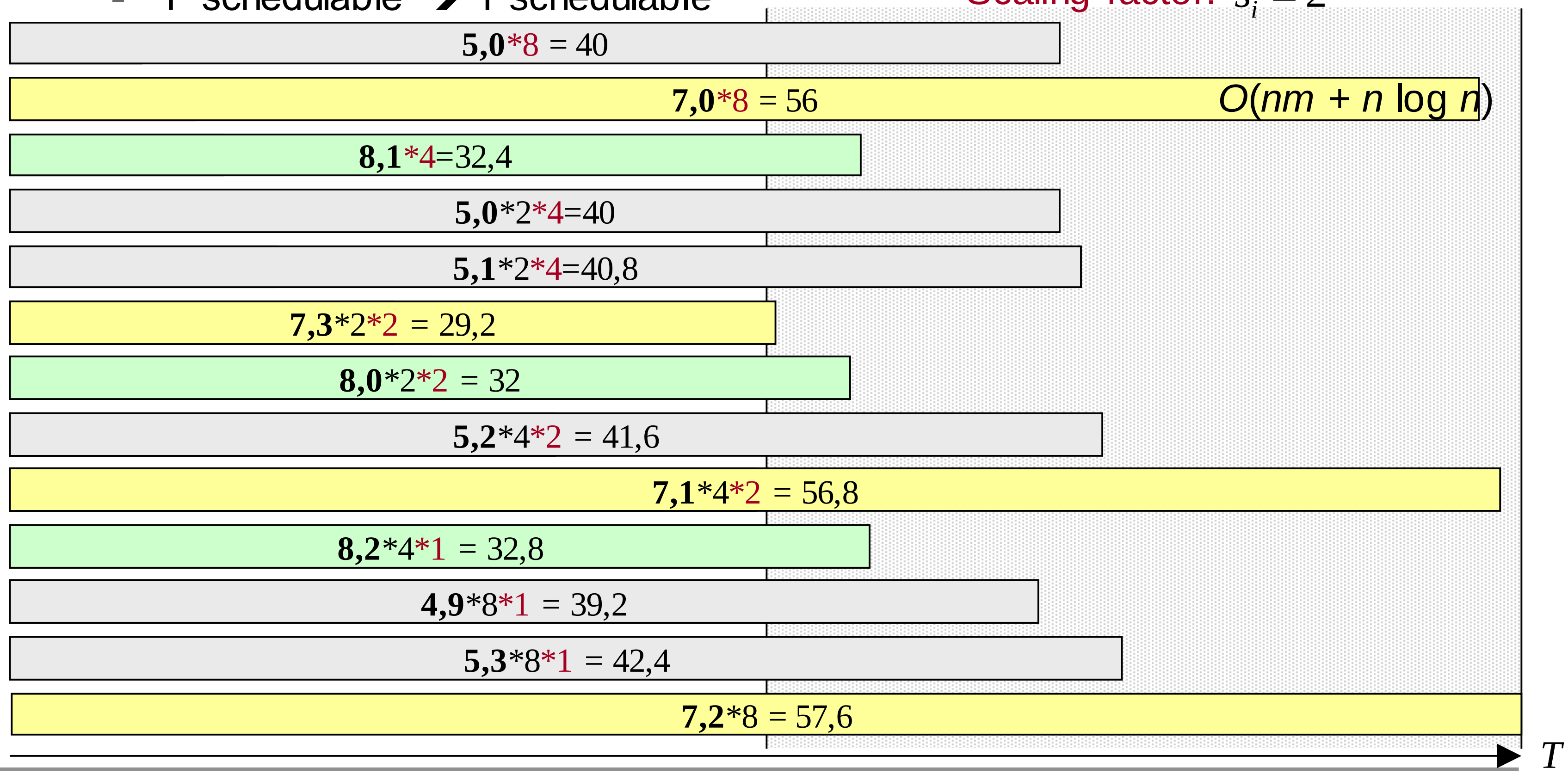


Partitioned algorithm: R-BOUND-MP

Scale Task Set

- $T = \{(C_1,T_1), (C_2,T_2),\ldots, (C_n,T_n)\}$ ➔ $T' = \{(2C_1,2T_1), (C_2,T_2),\ldots, (C_n,T_n)\}$
  - u_i constant
  - T' schedulable ➔ T schedulable

Scaling factor: $s_i = 2^{\left\lfloor \lg_2 T_n / T_i \right\rfloor}$



Non-partitioned method

- Main problem:
  finding a priority assignment that guarantees schedulability as long as the system utilization is below a certain value

Non-partitioned algorithm: RM

- Rate monotonic priority assignment
  - $p_i = 1/T_i$
- Suffers from Dhall‘s effect
  - Non-schedulable Task Set:
    - $T = \tau_1 \ldots \tau_{m+1}$
    - $(T_i = 1, C_i = 2\varepsilon) \; \forall \; i \in [1;m+1]$
    - $(T_{m+1} = 1+\varepsilon, C_i = 1)$
    - τ_{m+1} has lowest priority and will miss it‘s deadline
    - $\lim_{\varepsilon \to 0} U = 1$
    - $\lim_{m \to \infty, \varepsilon \to 0} U_S = 0$
- $O(n \log n)$ (sorting the task set)

Non-partitioned algorithm: RM-US[US-Limit]

- Guarantees that all task sets with $U_S \leq$ US-LIMIT are schedulable
- Tasks divided into two categories:
  - Tasks τ_i for which $U_i \leq$US-LIMIT:
    - $p_i = 1/(1+T_i)$ $p_i \in]0;1[$
  - Tasks τ_i for which $U_i >$US-LIMIT
    - $p_i = 1$
- Optimal US-LIMIT = 0.37482
- $O(n \log n)$ (sorting the task set)

Non-partitioned algorithm: adaptiveTkC

- $p_i = 1/(T_i - k * C_i)$

$$k = \frac{1}{2}\frac{m - 1 + \sqrt{5m^2 - 6m + 1}}{m}$$

- $$U_S = 2\frac{m}{3m - 1 + \sqrt{5m^2 - 6m + 1}}$$
- $\lim_{m\to\infty} U_S > 0.38$
- $O(n \log n)$ (sorting the task set)

Hybrid solution: RM-FFDU + adaptiveTkC

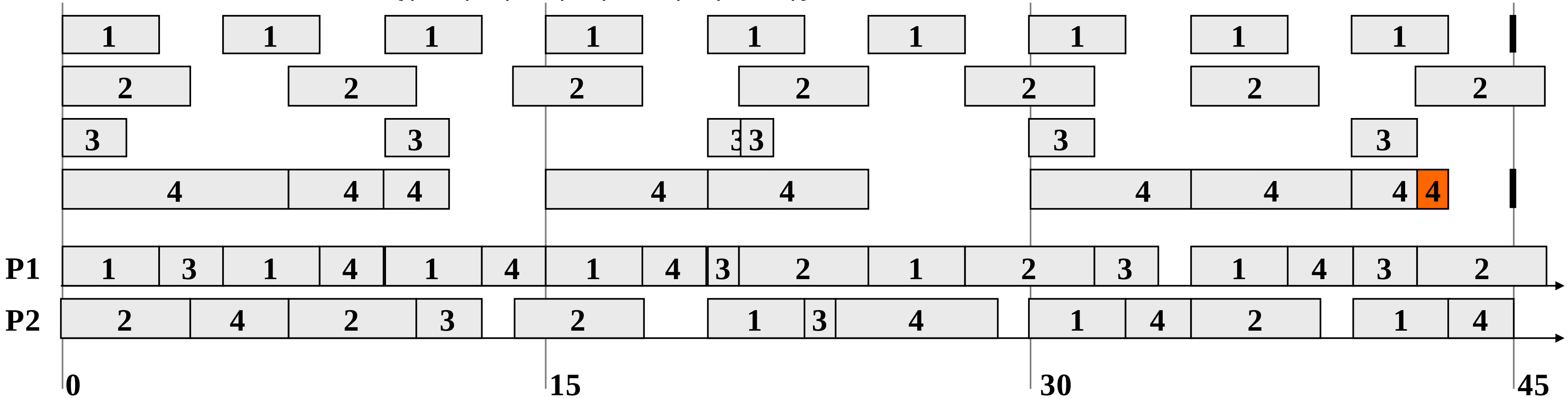
- Partition as many tasks as possible on the given number of processors (RM-FFDU)
- Assign global priorities to the remaining tasks (adaptiveTkC)
- m local queues + 1 global queue
- If the local ready queue of a processor is empty, a task from the global ready queue is executed.

Performance comparison:
Experimental setup

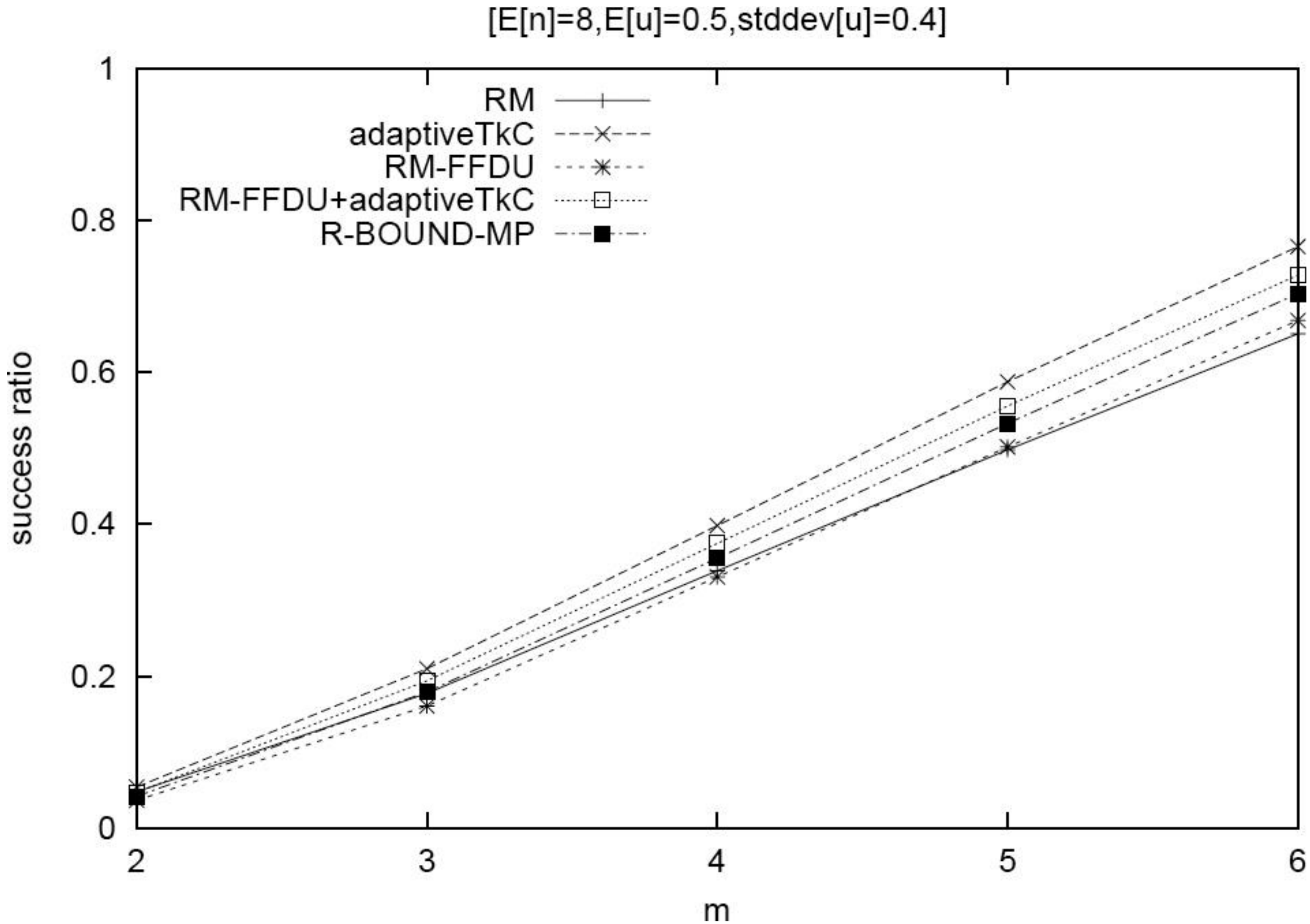
- $m = 4$ processors
- n: uniform distribution
  - $E[n] = 8$, minimum $= 0.5\ E[n]$, maximum $= 1.5\ E[n]$ ($n \in \{4,\ldots,12\}$)
- $T \in \{100,200, \ldots,1500,1600\}$
- u_i: normal distribution
  - $E[u_i] = 0.5$, $stddev[u_i]=0.4$
  - $u_i<0$ or $u_i>1$: generate new u_i
- e_i: computed
  - $e_i = floor(u_i*t_i)$
  - $e=0$: task generated again
- Success ratio
  - Fraction of all generated task sets that are successfully scheduled
  - For each point in a plot: average of 2,000,000 task sets

Performance comparison:
Experimental setup

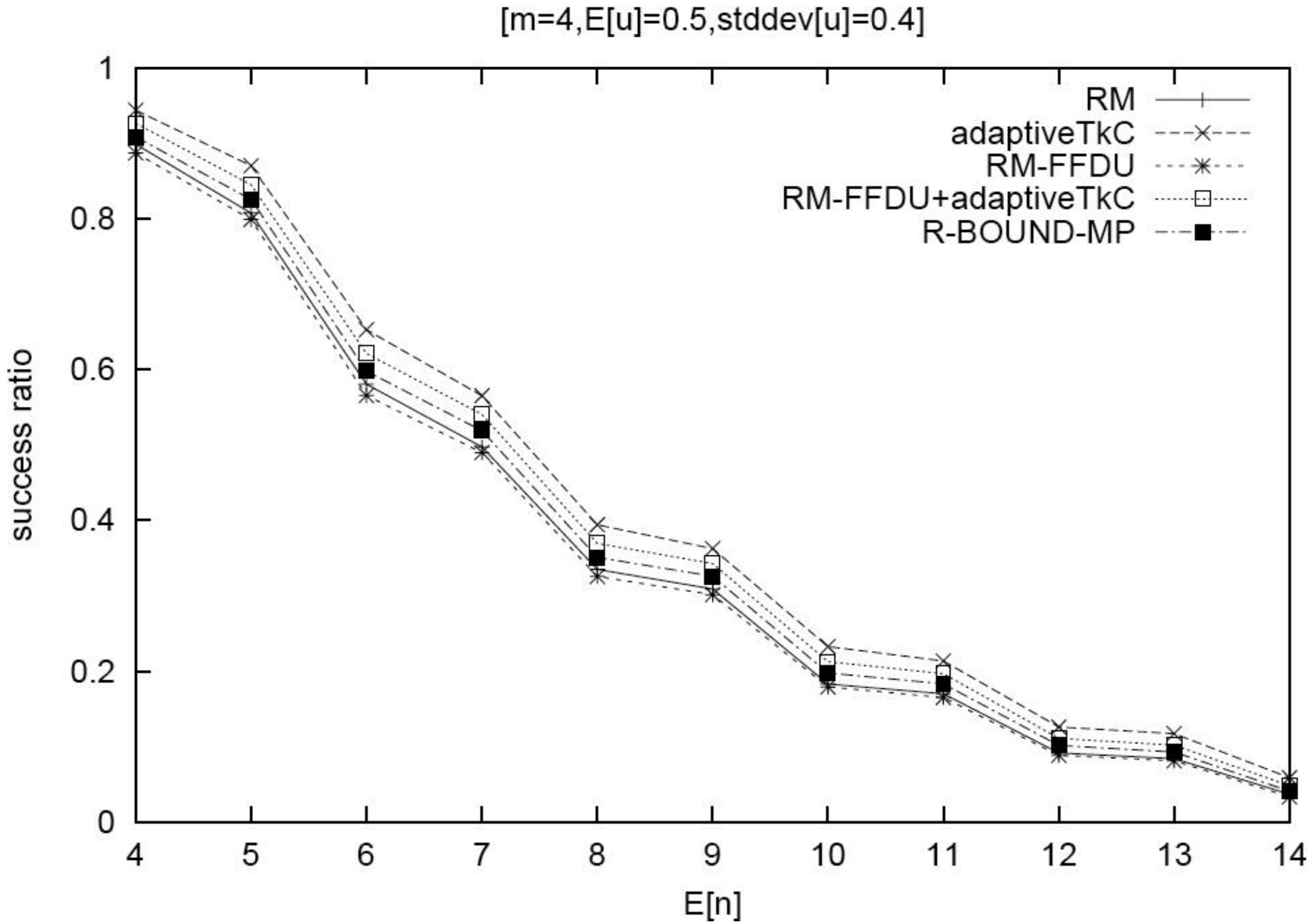
- Task set is schedulable, when:
  - Partitioned:
    - $m_{required} <= m_{given}$
  - Non-partitioned + hybrid:
    - Simulation of a meta period = LCM(T_i)+max(T_i)
    - All task instances completed no later than their deadlines
  - Why meta period?
    - T = {(3,5), (4,7), (2,10), (7,15)}



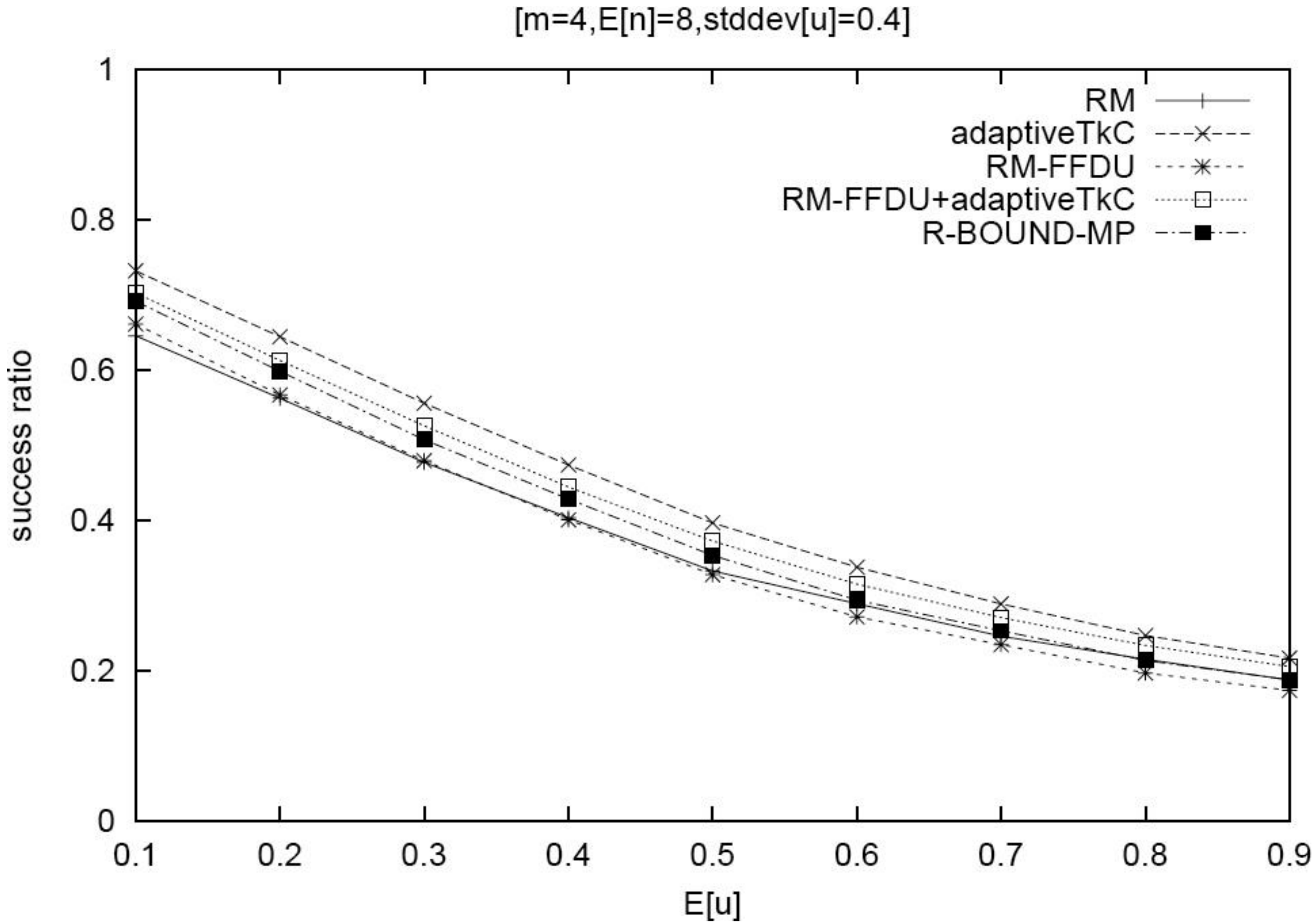
Performance comparison:
Results



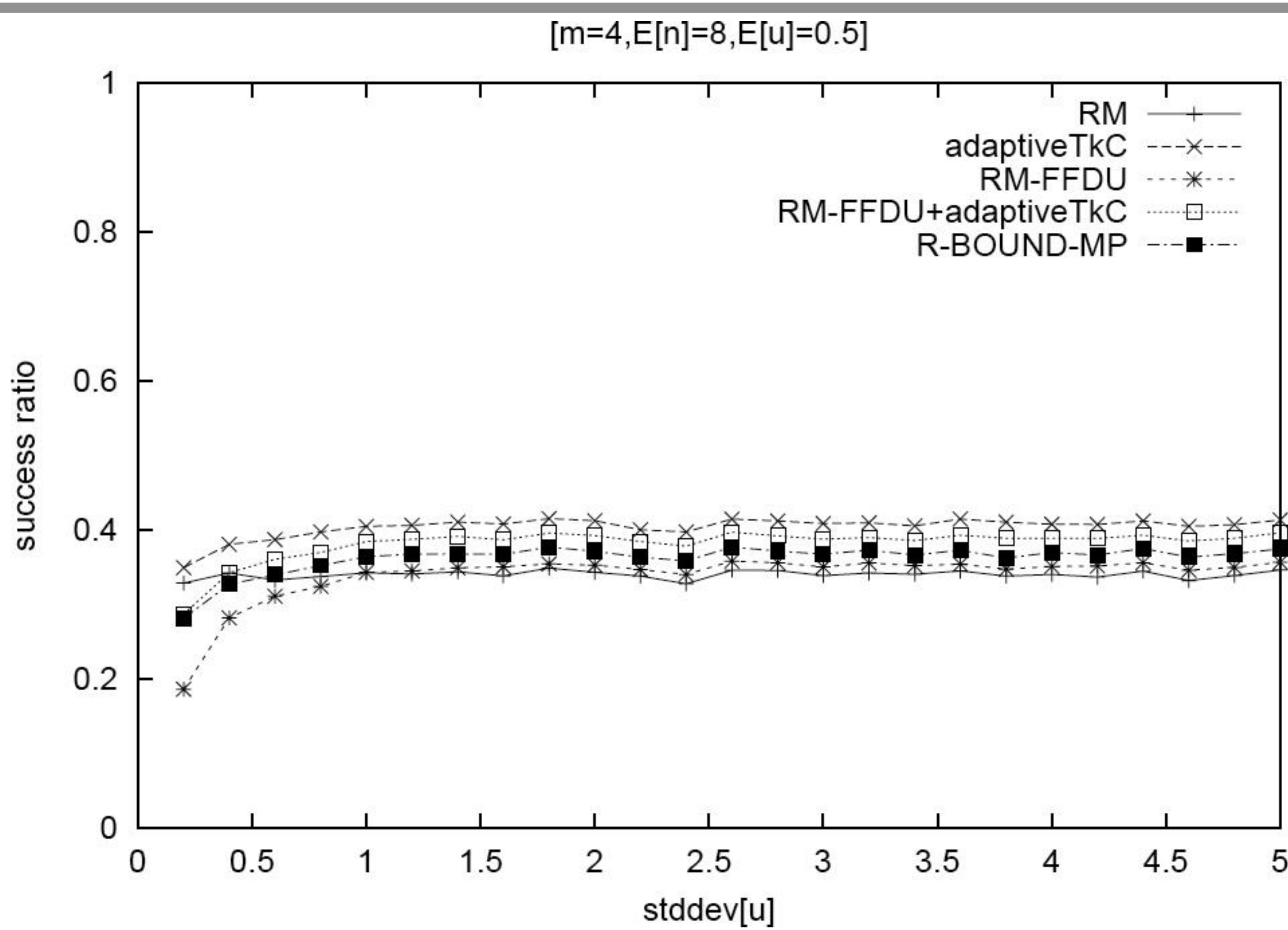
Performance comparison:
Results



Performance comparison:
Results

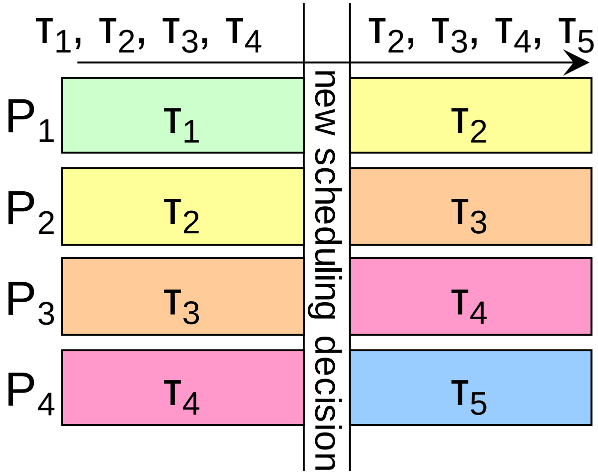


Performance comparison:
Results

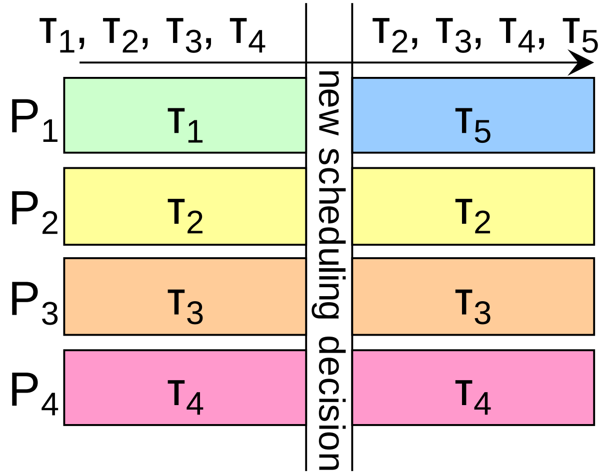


Preemption density comparison

- Architectural impact
  - Costs for preemption and migration are not negligible
  - Preemption aware Dispatcher for adaptiveTkC
    - adaptiveTkCaware

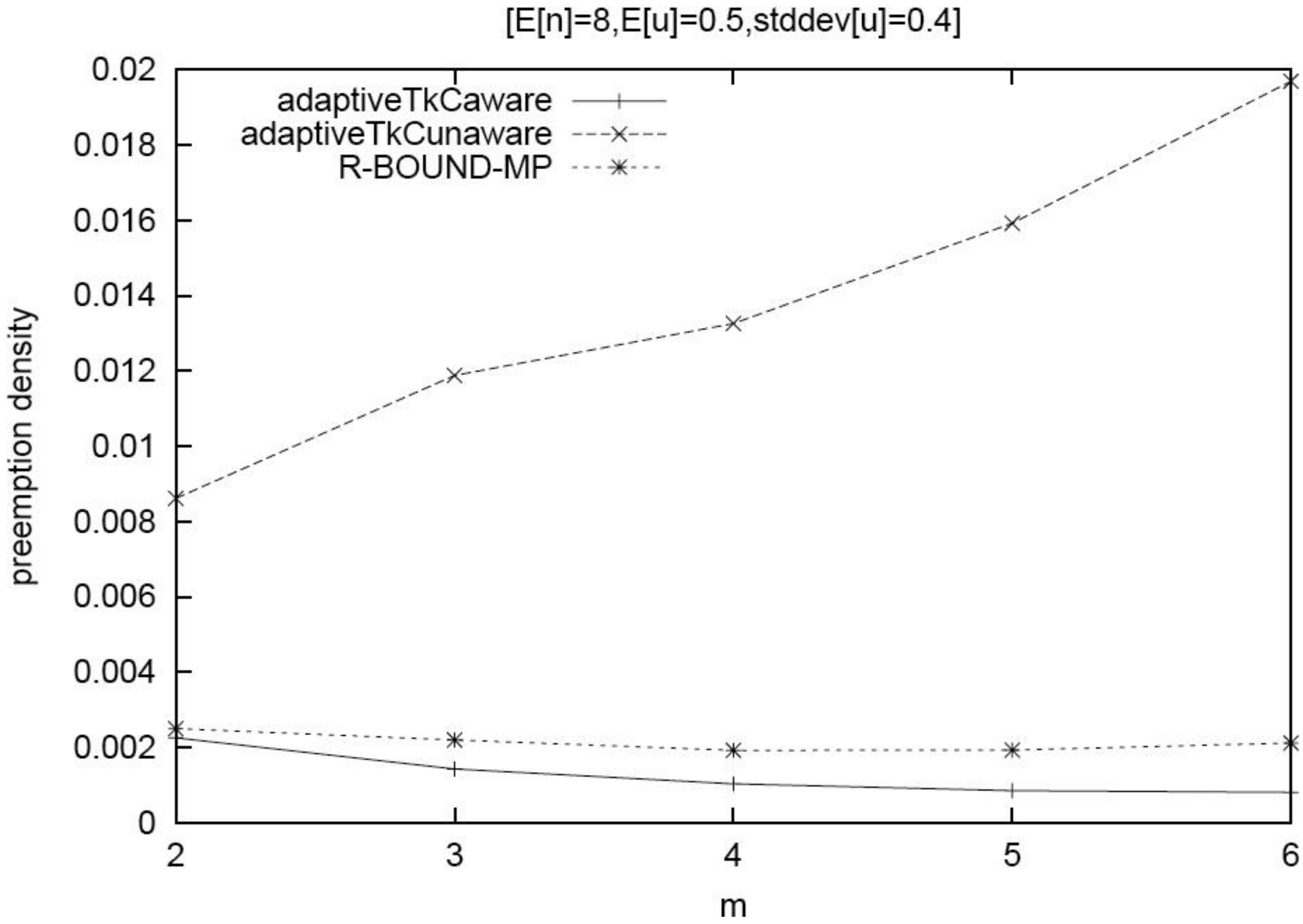


Preemption unaware Dispatcher

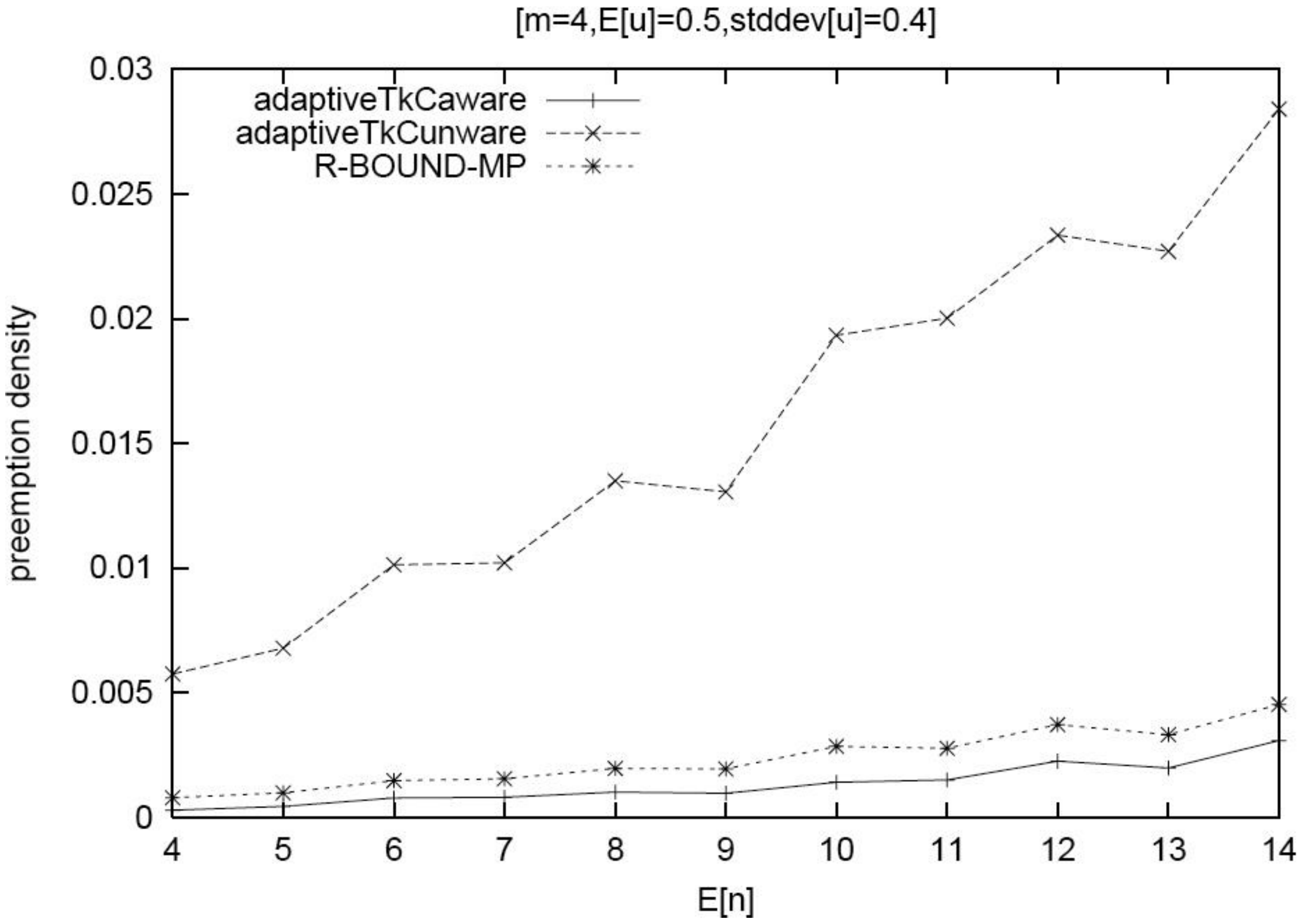


Preemption aware Dispatcher

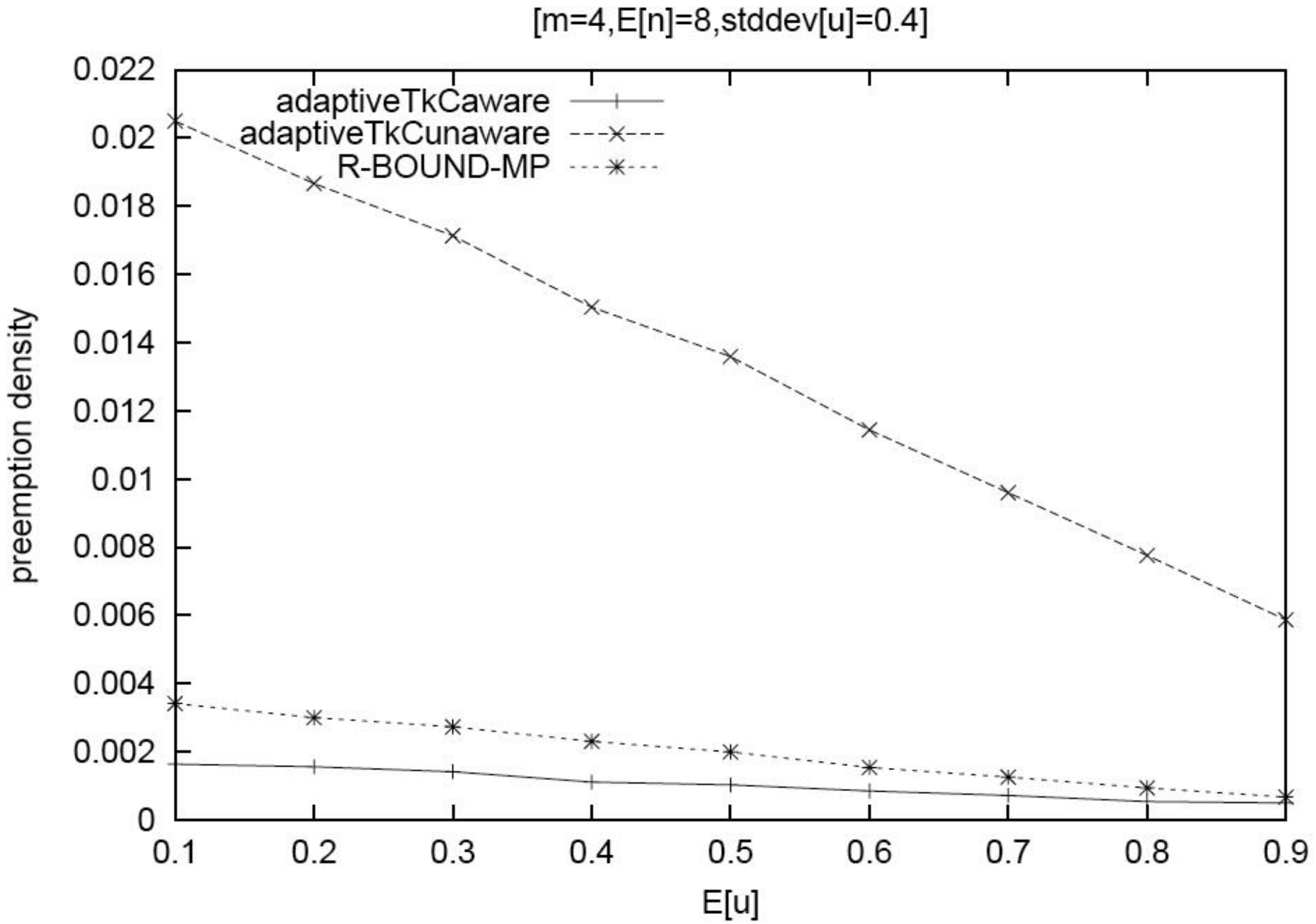
Preemption density comparison:
Results



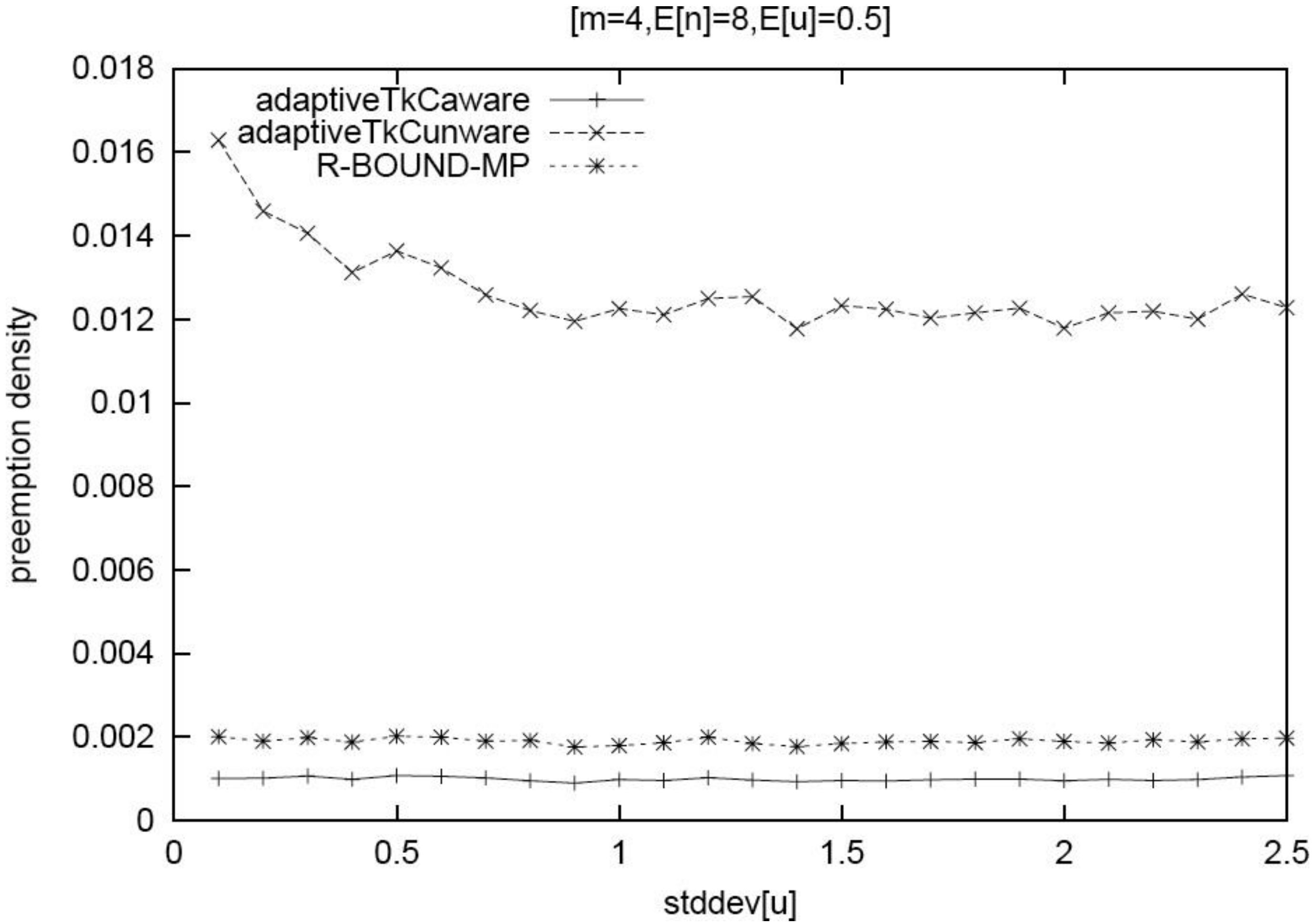
Preemption density comparison:
Results



Preemption density comparison:
Results



Preemption density comparison:
Results



Conclusion

- System utilization:
  - Non-partitioned approach: $U_S<0.38$
  - Partitioned approach: $U_S<0.41$ (RM)
    - Varying execution times can cause low system utilization
- Computational complexity:
  - Non-partitioned approach: $O(n \log n)$
  - Partitioned approach: $O(nm + n \log n)$
- Preemption cost:
  - Non-partitioned approach can reduce preemptions using a preemption aware dispatcher

References

[PartOrNot] B. Andersson and J. Jonsson. **Fixed-priority preemptive multiprocessor scheduling: To partition or not to partition.** In Proc. of the International Conference on Real-Time Computing Systems and Applications, pages 337–346, Cheju Island, Korea, December 12–14, 2000.

[RM] C. L. Liu and J.W. Layland. **Scheduling algorithms for multiprogramming in a hard-real-time environment.** *Journal of the Association for ComputingMachinery*, 20(1):46–61, January 1973.

[RM-US[US-LIMIT]] Lars Lundberg: **Analyzing Fixed-Priority Global Multiprocessor Scheduling.** IEEE Real Time Technology and Applications Symposium 2002: 145-153

[AdTkC] B. Andersson and J. Jonsson. **Some insights on fixed-priority preemptive non-partitioned multiprocessor scheduling.** In *Proc. of the IEEE Real-Time Systems Symposium – Workin- Progress Session*, Orlando, Florida, November 27–30, 2000. Also in TR-00-10, Dept. of Computer Engineering, Chalmers University of Technology.

References

[RM-FFDU] Y. Oh and S. H. Son. **Fixed-priority scheduling of periodic tasks on multiprocessor systems.** Technical Report 95-16, Department of Computer Science, University of Virginia, March 1995.

[R-Bound-MP] S. Lauzac, R. Melhem, and D. Moss´e. **An efficient RMS admission control and its application to multiprocessor scheduling.** In Proc. of the IEEE Int'l Parallel Processing Symposium, pages 511–518, Orlando, Florida, March 1998.

[U bounds] D. Oh and T. P. Baker. **Utilization bounds for n-processor rate monotone scheduling with static processor assignment.** Real-Time Systems, 15(2):183–192, September 1998.

[Comp] S. Lauzac, R. Melhem, and D. Moss´e. **Comparison of global and partitioning schemes for scheduling rate monotonic tasks on a multiprocessor.** In *10th Euromicro Workshop on Real Time Systems*, pages 188–195, Berlin, Germany, June 17–19, 1998.