Optimizing Preemption-Overhead Accounting in Multiprocessor Real-Time Systems

Bryan C. Ward[1], Abhilash Thekkilakattil[2], and James H. Anderson[1]

[1]University of North Carolina at Chapel Hill
[2]Mälardalen University

Real systems experience runtime overheads, which must be accounted for in schedulability analysis.

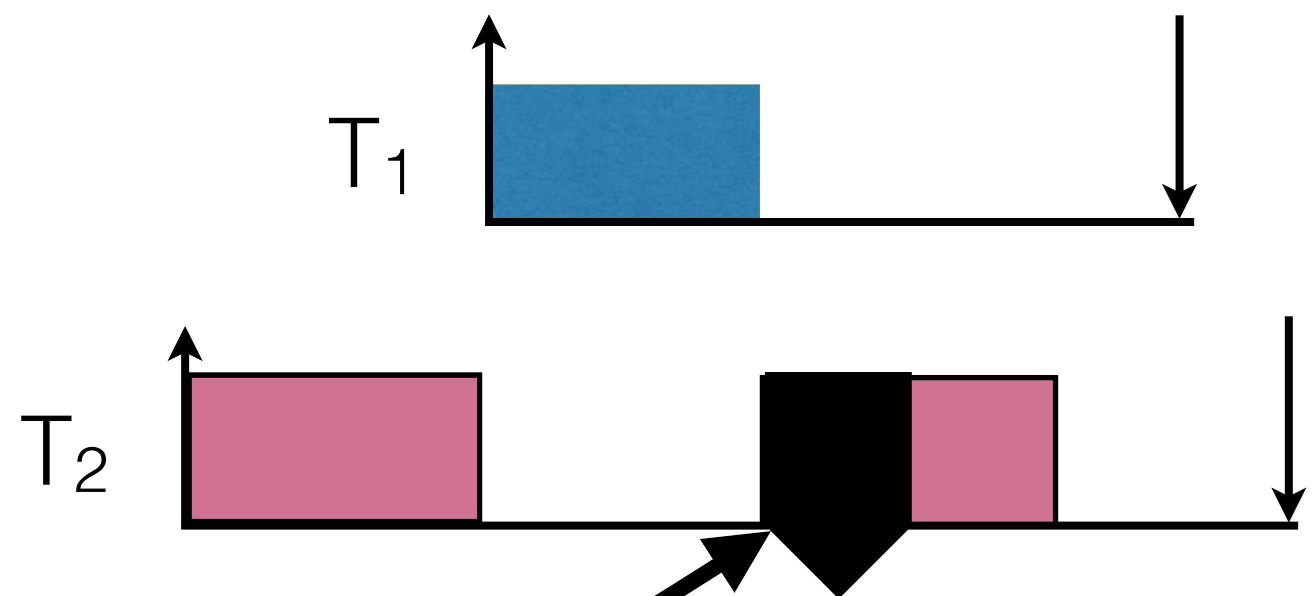
Theory

$$
\begin{aligned}
W_j(t_d) &\leq \mathrm{DBF}(\tau_j, Y_j, y_i - (t_d + C_j - \delta - (Y_j + x_j + C_j))) - \delta \\
&\leq \{\text{By Lemma 1, and by (21)}\} \\
&\quad U_j(y_i - (t_d + C_j - \delta - (Y_j + x_j + C_j))) + C_j\left(1 - \frac{Y_j}{T_j}\right) - \delta \\
&\leq \{\text{Rearranging, using } U_j = C_j/T_j\text{, and because } U_j \leq 1)\} \\
&\quad U_j(y_i - t_d) - U_jC_j + U_j\delta + \frac{C_jY_j}{T_j} + U_jx_j + U_jC_j + C_j - \frac{C_jY_j}{T_j} - U_j\delta \\
&= \{\text{Cancelling}\} \\
&\quad U_j(y_i - t_d) + U_jx_j + C_j
\end{aligned}
$$

Practice

Preemption Overheads

- Loss of cache affinity
- Scheduling
- Context switching
- Pipeline delays
- …



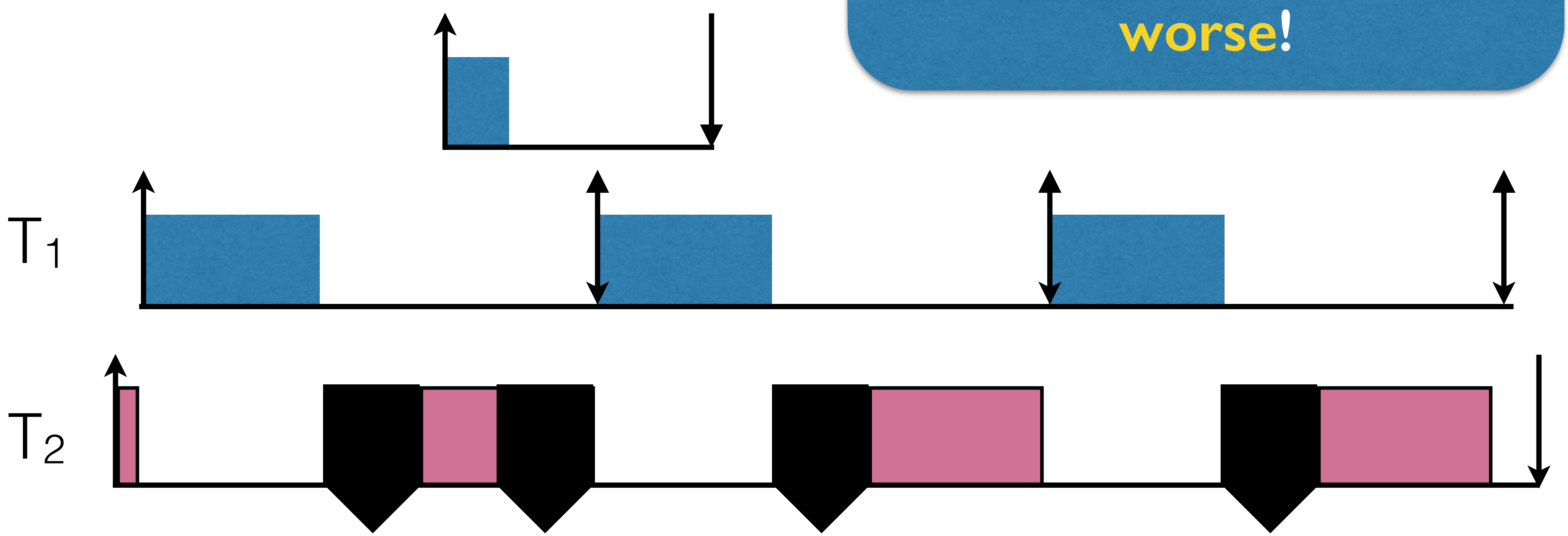
Preemption-Overhead Accounting

- Inflate execution costs to account for overheads.
- Two main approaches:
  - Task centric,
  - Preemption centric.
- Our contribution: a hybrid of these two.

Task Centric

- Each task's execution time is inflated to account for every possible preemption.
- Requires preemption-count bounds.

More tasks make matters worse!

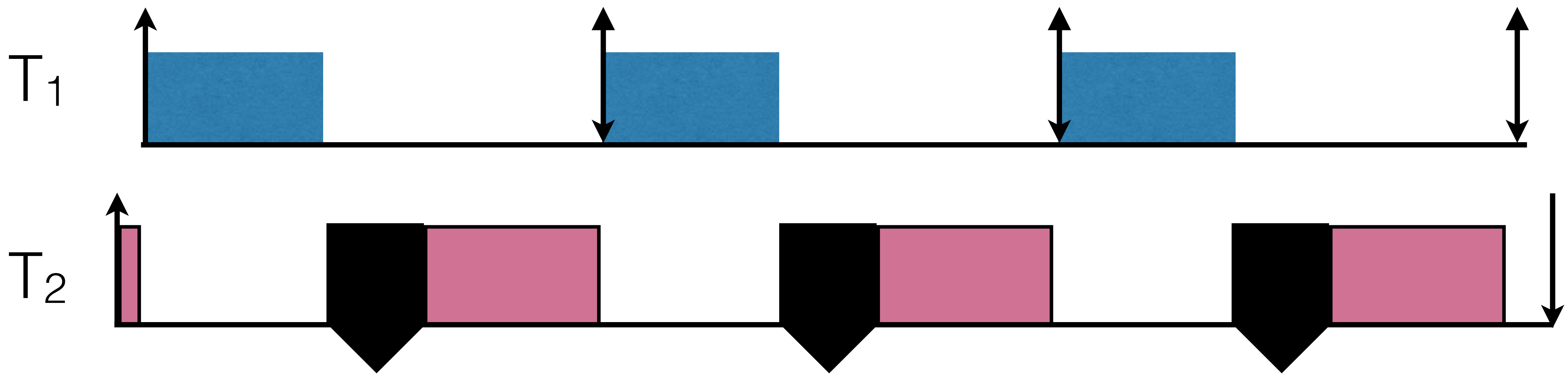


Preemption Centric

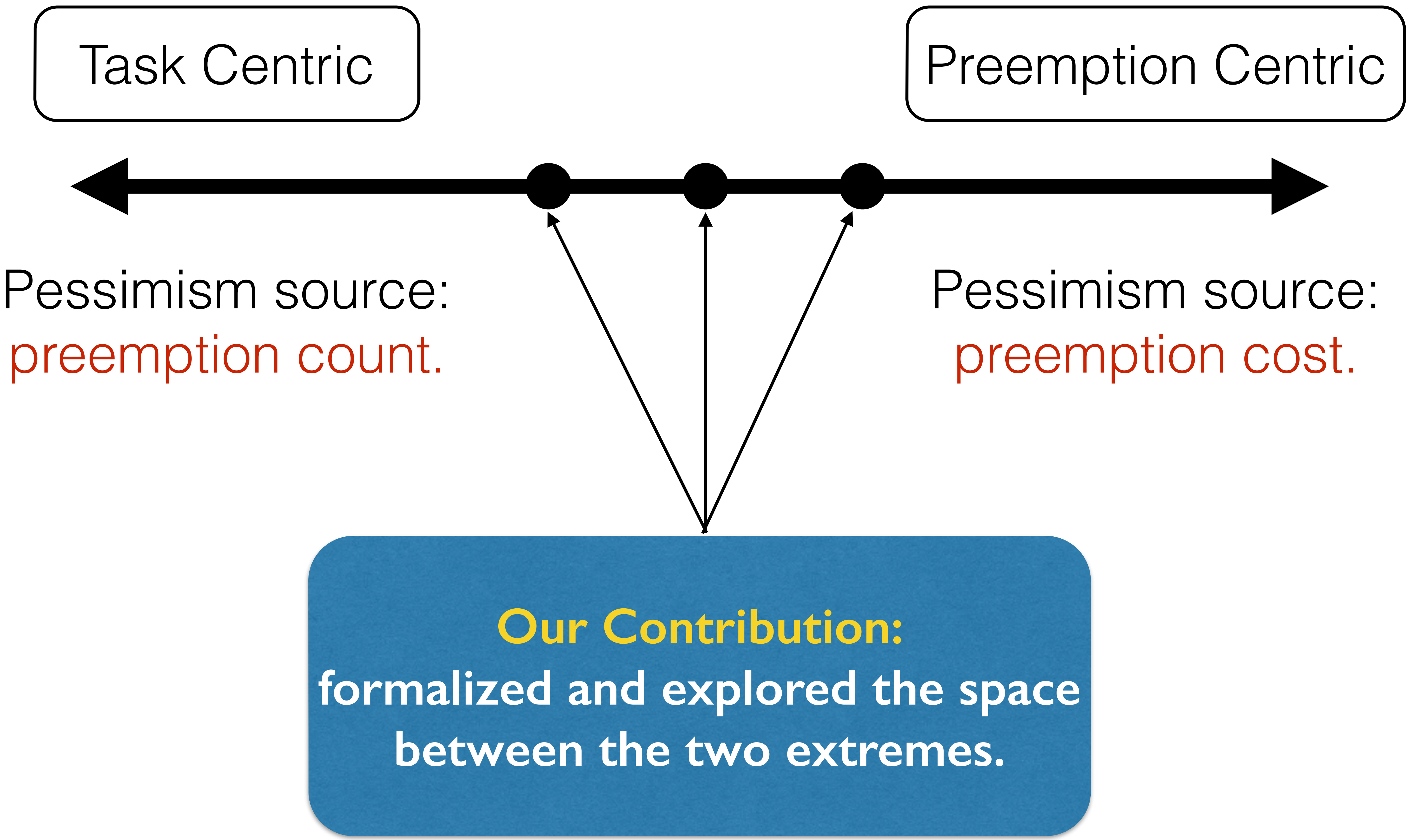
- The relinquishing task is charged the overhead.

The relinquishing task must pay for the largest overhead of any resuming task.

But each task must only pay for one preemption.



Tradeoff


Task Centric
Preemption Centric
Pessimism source:
preemption count.
Pessimism source:
preemption cost.
Our Contribution:
formalized and explored the space
between the two extremes.

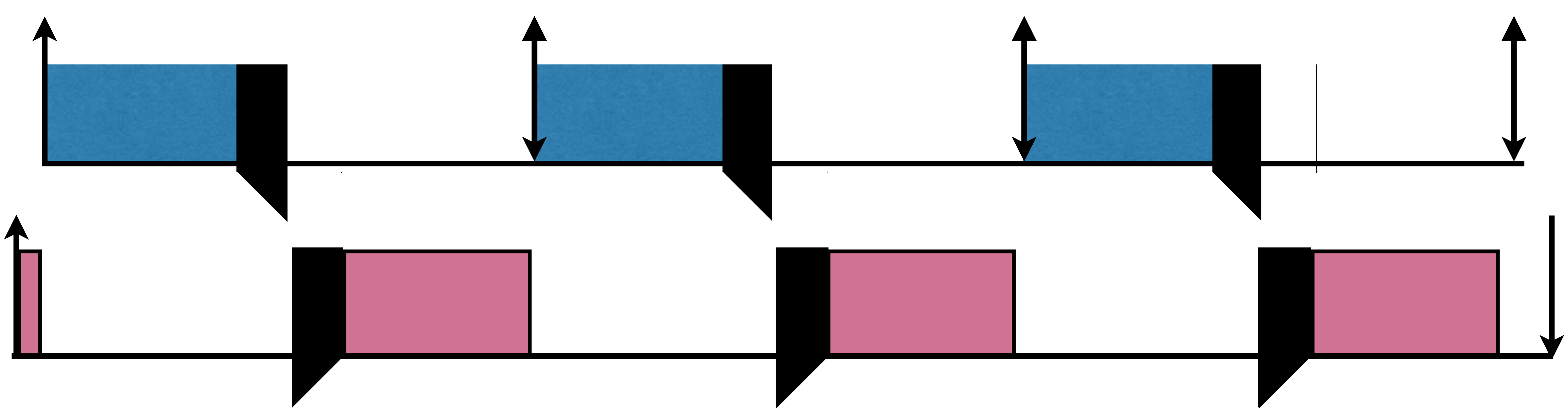
ARPO

- Analytical Redistribution of Preemption Overheads

- Hybrid approach: relinquishing task "pays" some, resuming task "pays" the rest.

T_1

T_2



Details

- Every task pays a global charge, G.
- Each task pays the difference between the actual overhead and the global charge.
    - Large G: preemption centric.
    - G = 0: task centric.
- Applicable to any job-level fixed-priority scheduler, *i.e.*, G-EDF, G-FP.

Linear Program

- Optimization objective: minimize utilization
- Subject to:
  - Inflated cost >= Original + Local Charge + G.
  - G >= 0.
  - Inflated per-task utilization <= 1.

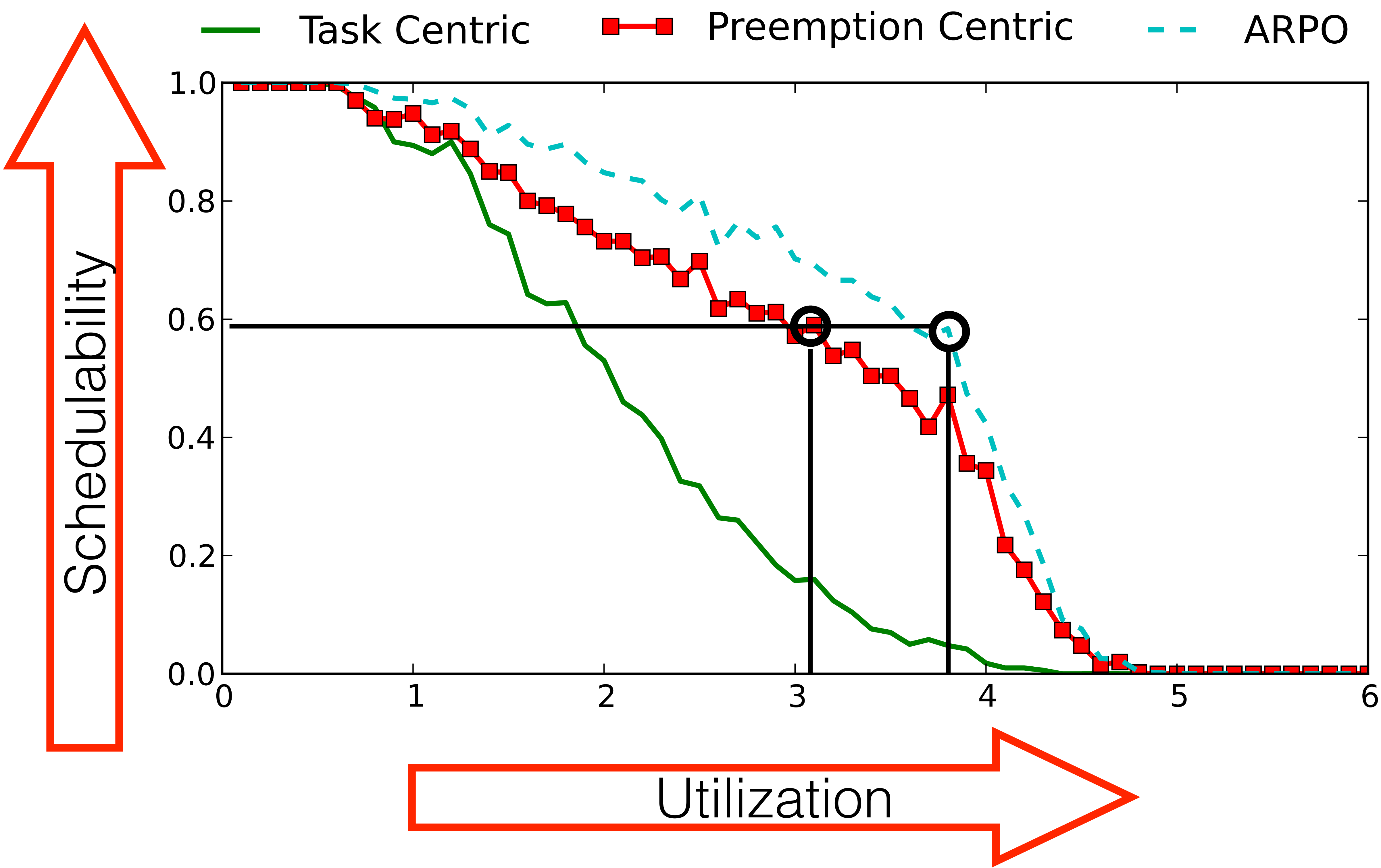
Limited Preemptions

- To avoid preemption overheads, preemptions can be limited to specific preemption points.
- ARPO can also be applied to these schedulers.
- Number of preemption points is the preemption-count bound.

Schedulability Framework

- Periods & utilizations chosen similarly to prior work.
- WSS chosen corresponding to execution cost.
  - 9 different distributions (uniform, constant, bimo).
  - Previous studies considered a single WSS for all tasks.
- Produced over 200 schedulability graphs.

G-EDF Schedulability



Extensions

- SRT - Optimizing for utilization is optimal.
- HRT:
    - Integrate with ILP-based RTA.
    - Added utilization-based schedulability test* as a constraint. (didn't work well).

*J. Goossens, S. Funk, and S. Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. Real-Time Systems, 2003.

Conclusions

- ARPO is a hybrid of task- and preemption-centric overhead accounting.
- Based on linear programming.
- Developed a new schedulability framework with non-constant WSSs.
- Improved schedulability.

Questions?