Scheduling and Shaping of Complex Task Activations for Mixed-Criticality Systems

*Biao Hu, †**Kai Huang**

*College of Information Science and Technology
Beijing University of Chemical Technology

†School of Data and Computer Science
Sun Yat-Sen University

Content

Content

Background

Nowadays...

Due to reducing the SWaP (space, weight and power), embedded systems are evolving into mixed-criticality systems (MCS). A mixed criticality system is one that has two or more distinct levels.

- Can be safety critical, mission critical and low-critical.

Background

Nowadays...

Due to reducing the SWaP (space, weight and power), embedded systems are evolving into mixed-criticality systems (MCS). A mixed criticality system is one that has two or more distinct levels.

- Can be safety critical, mission critical and low-critical.
- Up to 5 levels defined in DO-178B standards.
  - A-level:catastrophic; B-level: hazardous; C-level: Major; D-level: Minor; E-level: No effect;

Background

Nowadays...

Due to reducing the SWaP (space, weight and power), embedded systems are evolving into mixed-criticality systems (MCS). A mixed criticality system is one that has two or more distinct levels.

- Can be safety critical, mission critical and low-critical.
- Up to 5 levels defined in DO-178B standards.
    - A-level:catastrophic; B-level: hazardous; C-level: Major; D-level: Minor; E-level: No effect;

Main issue in MCS is that

Background

Nowadays...

Due to reducing the SWaP (space, weight and power), embedded systems are evolving into mixed-criticality systems (MCS). A mixed criticality system is one that has two or more distinct levels.

- Can be safety critical, mission critical and low-critical.
- Up to 5 levels defined in DO-178B standards.
  - A-level:catastrophic; B-level: hazardous; C-level: Major; D-level: Minor; E-level: No effect;

Main issue in MCS is that

- How to reconcile the conflicting requirements of tasks with different criticality.

Problem of Often Assumed Task Model

Sporadic Task τ_i

- T_i: period (minimum arrival interval)

Problem of Often Assumed Task Model

Sporadic Task τ_i

- T_i: period (minimum arrival interval)
- D_i: relative deadline ($D_i \leq T_i$)

Problem of Often Assumed Task Model

Sporadic Task τ_i

- T_i: period (minimum arrival interval)
- D_i: relative deadline ($D_i \leq T_i$)
- L_i: criticality level (dual criticality: LO, HI).

Problem of Often Assumed Task Model

Sporadic Task τ_i

- T_i: period (minimum arrival interval)
- D_i: relative deadline ($D_i \leq T_i$)
- L_i: criticality level (dual criticality: LO, HI).
- $C_i(L_i)$: worst-case execution time depending on criticality
  - LO-critical task: $C_i(LO)$
  - HI-critical task: $C_i(HI) \geq C_i(LO)$.

Problem of Often Assumed Task Model

Sporadic Task τ_i

- T_i: period (minimum arrival interval)
- D_i: relative deadline ($D_i \leq T_i$)
- L_i: criticality level (dual criticality: LO, HI).
- $C_i(L_i)$: worst-case execution time depending on criticality
  - LO-critical task: $C_i(LO)$
  - HI-critical task: $C_i(HI) \geq C_i(LO)$.

Model limitation

- Cannot handle blocking, jitter, burst activations and arbitrary deadline.

Problem of Often Assumed Task Model

Sporadic Task τ_i

- T_i: period (minimum arrival interval)
- D_i: relative deadline ($D_i \leq T_i$)
- L_i: criticality level (dual criticality: LO, HI).
- $C_i(L_i)$: worst-case execution time depending on criticality
  - LO-critical task: $C_i(LO)$
  - HI-critical task: $C_i(HI) \geq C_i(LO)$.

Model limitation

- Cannot handle blocking, jitter, burst activations and arbitrary deadline.
- Pessimistic assumption.
  - Transform a periodic with jitter task to a task with shorter period.

Content

Task Activation Bound: Arrival Curve

Arrival Curve: Denote $R[s,t)$ as the number of events that arrive on an event stream in the time interval $[s,t)$. Then, $\overline{\alpha}^u$ and $\overline{\alpha}^l$ represents the upper and lower bound on the number of event in any interval $t-s$, that is,

$$\overline{\alpha}^l(t-s) \leq R[s,t) \leq \overline{\alpha}^u(t-s), \forall t \geq s \geq 0,$$

with $\overline{\alpha}^l(\Delta) \geq 0$, $\overline{\alpha}^u(\Delta) \geq 0$ for $\forall \Delta \in \mathbb{R}^{\geq 0}$.

- Generalizes conventional event stream models, such as sporadic, periodic, periodic with jitter, and arbitrary event streams.

Task Activation Bound: Arrival Curve

Arrival Curve: Denote $R[s,t)$ as the number of events that arrive on an event stream in the time interval $[s,t)$. Then, $\overline{\alpha}^u$ and $\overline{\alpha}^l$ represents the upper and lower bound on the number of event in any interval $t-s$, that is,

$$\overline{\alpha}^l(t-s) \leq R[s,t) \leq \overline{\alpha}^u(t-s), \forall t \geq s \geq 0,$$

with $\overline{\alpha}^l(\Delta) \geq 0$, $\overline{\alpha}^u(\Delta) \geq 0$ for $\forall \Delta \in \mathbb{R}^{\geq 0}$.

- Generalizes conventional event stream models, such as sporadic, periodic, periodic with jitter, and arbitrary event streams.
- For instance, for the arbitrary events modeled with the period p, the jitter j, and the minimum inter arrival distance d between successive two events, its upper arrival curve is

$$\overline{\alpha}^u(\Delta) = \min\{\lceil\frac{\Delta + j}{p}\rceil, \lceil\frac{\Delta}{d}\rceil\}.$$

Task Activation Bound: Minimum Distance Function

A similar bound to the upper arrival curve

Minimum Distance Function: The minimum distance function $\delta(q)$ is a pseudo super-additive function, which returns a lower bound on the time interval between the first and the last event of any sequence of $q+1$ event occurrences.

- The minimum distance function is an inverse description of upper arrival curve. For example, $\delta(k) = \Delta_k$ denotes that, the first and the last event of any sequence of $k+1$ events is at least Δ_k time units apart, i.e., $\overline{\alpha}(\delta(k)) = k+1$.

Two Ideas

- **Schedulability Test of Complex Task Activations** for Mixed-Criticality Systems
- **Shaping Task Activation Events** to Improve the QoS of Low-critical Tasks.

Two Ideas

- **Schedulability Test of Complex Task Activations** for Mixed-Criticality Systems ⟵
- **Shaping Task Activation Events** to Improve the QoS of Low-critical Tasks.

Content

Problem Formulation

Arbitrarily Activated Task τ_i

- α_i^u or $\delta_i(q)$: arrival curve or minimum distance function
- D_i: relative deadline
- L_i: criticality level (dual criticality: LO, HI).
- $C_i(L_i)$: worst-case execution time depending on criticality
  - LO-critical task: $C_i(LO)$
  - HI-critical task: $C_i(HI) \geq C_i(LO)$.

Problem Formulation

Arbitrarily Activated Task τ_i

- α_i^u or $\delta_i(q)$: arrival curve or minimum distance function
- D_i: relative deadline
- L_i: criticality level (dual criticality: LO, HI).
- $C_i(L_i)$: worst-case execution time depending on criticality
  - LO-critical task: $C_i(LO)$
  - HI-critical task: $C_i(HI) \geq C_i(LO)$.

Problem Definition

- Given a dual-criticality task set, is it possible to schedule this task set by fixed-priority.

Motivation Example

A task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ as follows

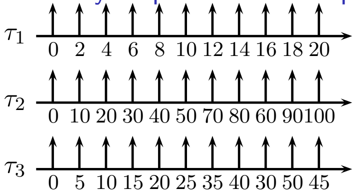
τ_i	L_i	$C_i(LO)$	$C_i(HI)$	D_i	$\overline{\alpha}_i^u(p, j, d)$
τ_1	LO	3	-	7	(10, 30, 2)
τ_2	HI	5	10	35	(30, 50, 10)
τ_3	HI	20	40	300	(100,220,5)

Motivation Example

A task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ as follows

τ_i	L_i	$C_i(LO)$	$C_i(HI)$	D_i	$\overline{\alpha}_i^u(p, j, d)$
τ_1	*LO*	3	-	7	(10, 30, 2)
τ_2	*HI*	5	10	35	(30, 50, 10)
τ_3	*HI*	20	40	300	(100,220,5)

The as-early-as-possible arrival pattern under the task model assumption



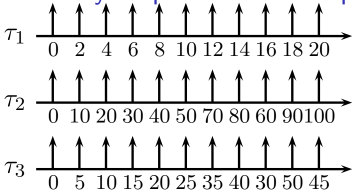
Sporadic arrival pattern

Motivation Example

A task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ as follows

τ_i	L_i	$C_i(LO)$	$C_i(HI)$	D_i	$\overline{\alpha}_i^u(p, j, d)$
τ_1	LO	3	-	7	(10, 30, 2)
τ_2	HI	5	10	35	(30, 50, 10)
τ_3	HI	20	40	300	(100,220,5)

The as-early-as-possible arrival pattern under the task model assumption



Sporadic arrival pattern

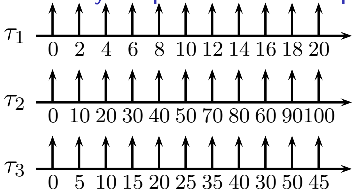
impossible to be scheduled

Motivation Example

A task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ as follows

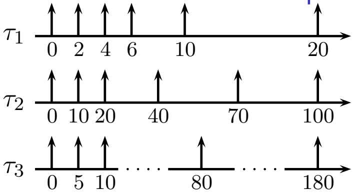
τ_i	L_i	$C_i(LO)$	$C_i(HI)$	D_i	$\overline{\alpha}_i^u(p, j, d)$
τ_1	LO	3	-	7	(10, 30, 2)
τ_2	HI	5	10	35	(30, 50, 10)
τ_3	HI	20	40	300	(100,220,5)

The as-early-as-possible arrival pattern under the task model assumption



Sporadic arrival pattern

impossible to be scheduled



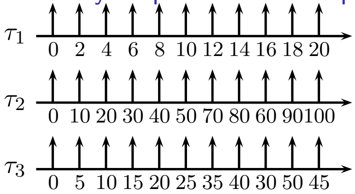
Arrival-curve arrival pattern

Motivation Example

A task set $\tau = \{\tau_1, \tau_2, \tau_3\}$ as follows

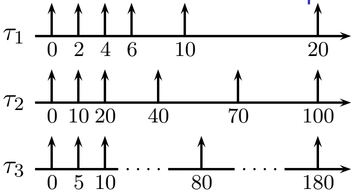
τ_i	L_i	$C_i(LO)$	$C_i(HI)$	D_i	$\overline{\alpha}_i^u(p, j, d)$
τ_1	LO	3	-	7	(10, 30, 2)
τ_2	HI	5	10	35	(30, 50, 10)
τ_3	HI	20	40	300	(100,220,5)

The as-early-as-possible arrival pattern under the task model assumption



Sporadic arrival pattern

impossible to be scheduled



Arrival-curve arrival pattern

may be possible to be scheduled

Schedule Conditions

Demand Bound Function

The demand-bound function $\mathrm{dbf}(\tau_i, \Delta)$ gives an upper bound on the maximum possible execution demand of the task τ_i in any time interval of length Δ, where demand is calculated as the total amount of required execution time of events with their whole scheduling windows within the time interval.

Schedule Conditions

Demand Bound Function

The demand-bound function $\mathrm{dbf}(\tau_i, \Delta)$ gives an upper bound on the maximum possible execution demand of the task τ_i in any time interval of length Δ, where demand is calculated as the total amount of required execution time of events with their whole scheduling windows within the time interval.

Schedulable Conditions

$$\text{Condition 1} : \forall \Delta \geq 0 : \quad \sum_{\tau_i \in \tau} \mathrm{dbf}_{\mathrm{LO}}(\tau_{\mathrm{i}}, \triangle) \leq \Delta,$$

$$\text{Condition 2} : \forall \Delta \geq 0 : \sum_{\tau_i \in HI(\tau)} \mathrm{dbf}_{\mathrm{HI}}(\tau_{\mathrm{i}}, \triangle) \leq \Delta.$$

where Δ represents the supply of a dedicated unit-speed uniprocessor.

DBF in LO and HI modes

Demand-Bound Function in LO mode

If the system is in LO mode, every task behaves as a normal task with parameters $(\alpha_i(\Delta) \text{ or } \delta_i(q), C_i(LO), D_i(LO))$. According to the framework of real-time calculus, a tight demand bound function of a task τ_i is that

$$\mathrm{dbf}_{\mathrm{LO}}(\tau_i, \Delta) = \alpha_i(\Delta - D_i(LO)) \cdot C_i(LO). \tag{2}$$

DBF in LO and HI modes

Demand-Bound Function in LO mode

If the system is in LO mode, every task behaves as a normal task with parameters ($\alpha_i(\Delta)$ or $\delta_i(q)$, $C_i(LO)$, $D_i(LO)$). According to the framework of real-time calculus, a tight demand bound function of a task τ_i is that

$$\mathrm{dbf}_{\mathrm{LO}}(\tau_i, \Delta) = \alpha_i(\Delta - D_i(LO)) \cdot C_i(LO). \quad (2)$$

Demand-Bound Function in HI mode

The demand-bound function of HI mode is thus concluded as follows:

$$\mathrm{dbf}_{\mathrm{HI}}(\tau_i, \Delta) = (k+1) \cdot C_i(HI) - [C_i(LO) - (\Delta - \delta_i'(k))]_0,$$

where

$$\delta_i'(k) = \begin{cases} k \cdot C_i(LO), & k \leq h \\ h \cdot C_i(LO) + \delta_i(k) - \delta_i(h), & k > h, \end{cases} \quad (3)$$

where $\delta_i(h+1) - \delta_i(h) > C_i(LO)$, $k \in \mathbb{N}^+$.

Two Ideas

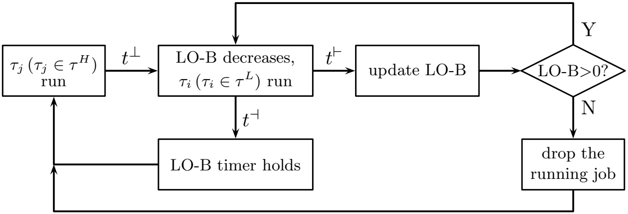
- **Schedulability Test of Complex Task Activations** for Mixed-Criticality Systems
- **Shaping Task Activation Events** to Improve the QoS of Low-critical Tasks.

Two Ideas

- **Schedulability Test of Complex Task Activations** for Mixed-Criticality Systems
- **Shaping Task Activation Events** to Improve the QoS of Low-critical Tasks. $\longleftarrow$

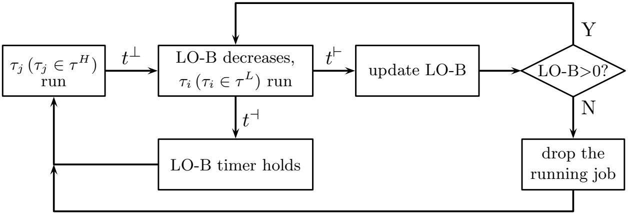
Content

Shaping Workflow



Steps:

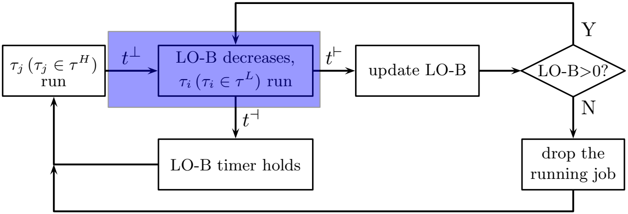
Shaping Workflow



Steps:

1. First of all, we set up a LO-B timer that constrains how much a LO-critical task can run in HI mode.

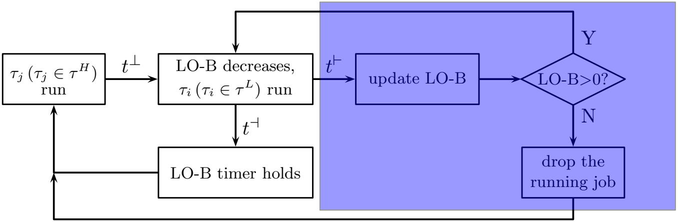
Shaping Workflow



Steps:

1. First of all, we set up a LO-B timer that constrains how much a LO-critical task can run in HI mode.
2. Suppose at a time $t^{\perp}$, the system starts to run a LO-critical task; meanwhile the LO-B timer starts to decrease.

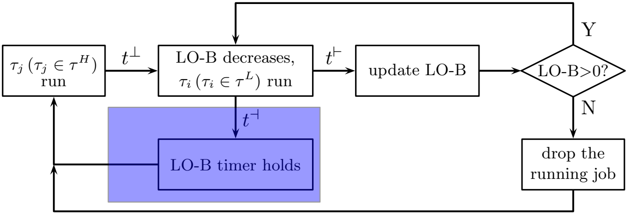
Shaping Workflow



Steps:

1. First of all, we set up a LO-B timer that constrains how much a LO-critical task can run in HI mode.
2. Suppose at a time $t^{\perp}$, the system starts to run a LO-critical task; meanwhile the LO-B timer starts to decrease.
3. In a case that this task does not finish till the LO-B timer times out (time instant $t^{\vdash}$), the system will update LO-B. The new updated LO-B will either allow this task to run further if the new LO-B is greater than zero, or drop it otherwise.

Shaping Workflow



Steps:

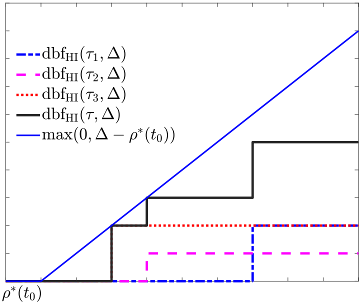
4. In another case that this task finishes before LO-B timer times out (time instant $t^{\dashv}$), LO- timer will hold its current value at the task finishing time and be used for shaping future LO-critical tasks.

Updating LO-B

Based on the task procrastination technique, the LO-B is computed as follows

$$\rho^*(t) = \max\left\{\rho : [\Delta - \rho]_0 \geq \sum_{\tau_i \in \tau^H} \mathrm{dbf}_{\mathrm{HI}}(\tau_i, \Delta, t),\ \forall \Delta \geq 0\right\}, \quad (4)$$

where LO-B is set to $\rho^*(t)$.



Content

Evaluations Setup

Testing approaches

1. AC-sched: Our proposed approaches that cover the schedulability test towards sporadic and arbitrary activation tasks.
2. Greedy: The greedy tuning approach that is also based on the demand-bound function.
3. AMC-max: The test via the response-time calculation for fixed-priority scheduling.
4. EDF-VD: The approach that is also based on the virtual deadlines. However, EDF-VD scales down the deadlines at the same margin for all HI-critical tasks.
5. AC-Shaping: The shaping approach that we proposed for improving the QoS to LO-critical tasks.

Parameters

Generating Tasks

The task is set as *pjd* pattern because *pjd* pattern can represent the burst and jitter. There are four parameters that will be studied, i.e., $(\mathcal{P}, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$, whose meanings are listed in the following.

1. In generating a task set, the probability of a random task being a HI-critical task is $\mathcal{P}$.

Parameters

Generating Tasks

The task is set as *pjd* pattern because *pjd* pattern can represent the burst and jitter. There are four parameters that will be studied, i.e., $(\mathcal{P}, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$, whose meanings are listed in the following.

1. In generating a task set, the probability of a random task being a HI-critical task is $\mathcal{P}$.
2. The jitter j_i is set as $\mathcal{X} \cdot p_i$, where p_i and j_i are the parameters p and j. Besides, $\mathcal{X} \in [0.5, 4.5]$.

Parameters

Generating Tasks

The task is set as *pjd* pattern because *pjd* pattern can represent the burst and jitter. There are four parameters that will be studied, i.e., $(\mathcal{P}, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$, whose meanings are listed in the following.

1. In generating a task set, the probability of a random task being a HI-critical task is $\mathcal{P}$.
2. The jitter j_i is set as $\mathcal{X} \cdot p_i$, where p_i and j_i are the parameters p and j. Besides, $\mathcal{X} \in [0.5, 4.5]$.
3. The minimum inter distance d_i is set as $\mathcal{Y} \cdot p_i$, where d_i is the parameter d. Besides, $\mathcal{Y} \in [0.1, 1]$.

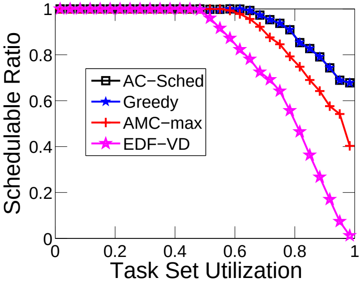
Parameters

Generating Tasks

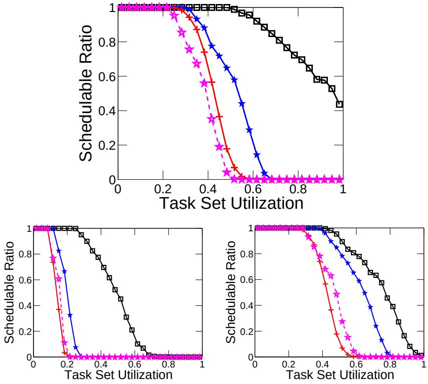
The task is set as *pjd* pattern because *pjd* pattern can represent the burst and jitter. There are four parameters that will be studied, i.e., $(\mathcal{P}, \mathcal{X}, \mathcal{Y}, \mathcal{Z})$, whose meanings are listed in the following.

1. In generating a task set, the probability of a random task being a HI-critical task is $\mathcal{P}$.
2. The jitter j_i is set as $\mathcal{X} \cdot p_i$, where p_i and j_i are the parameters p and j. Besides, $\mathcal{X} \in [0.5, 4.5]$.
3. The minimum inter distance d_i is set as $\mathcal{Y} \cdot p_i$, where d_i is the parameter d. Besides, $\mathcal{Y} \in [0.1, 1]$.
4. The relative deadline is set as that $D_i(LO) = D_i(HI) = \mathcal{Z} \cdot p_i$, where $\mathcal{Z} \in [0.5, 5]$.

Schedulability Test Results

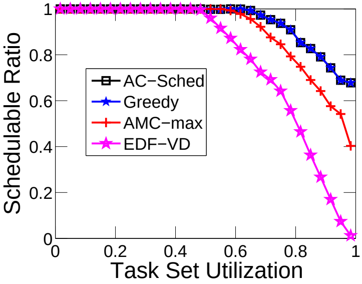


Sporadic tasks

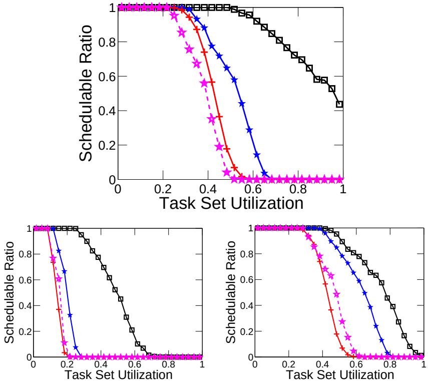


Irregularly activated tasks

Schedulability Test Results



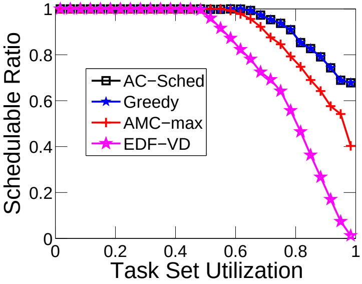
Sporadic tasks



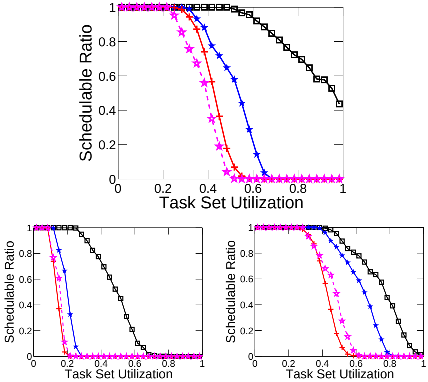
Irregularly activated tasks

- AC-sched has the same performance as Greedy on scheduling sporadic tasks.

Schedulability Test Results



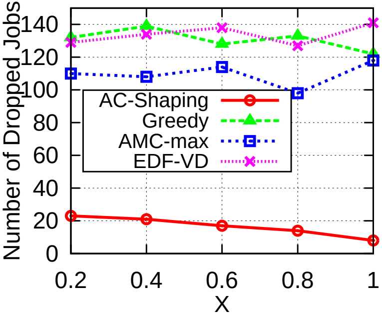
Sporadic tasks



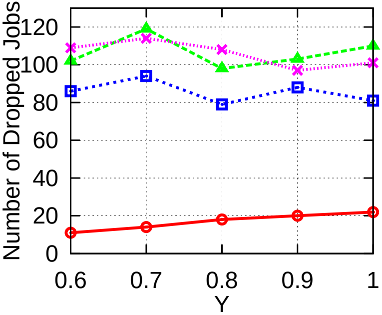
Irregularly activated tasks

- AC-sched has the same performance as Greedy on scheduling sporadic tasks.
- AC-shed performs much better for irregularly activated tasks than other approaches.

Shaping Results



Varying $\mathcal{X}$

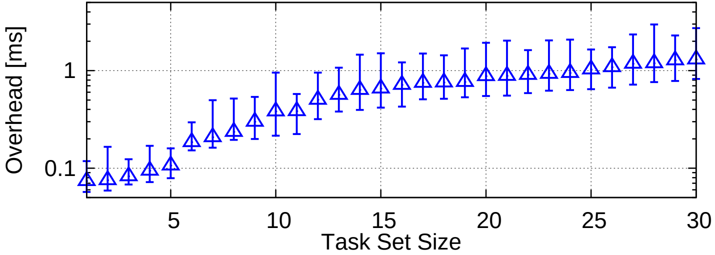


Varying $\mathcal{Y}$

- AC-Shaping reduces the dropped jobs to less than one fourth of the other scheduling approaches.

Shaping Results

Computation Overhead of Shaping



- As shown on a logarithmic scale in the above figure, with the increase of task set size, the computation expense increases. But even for a task set with 30 tasks, the average computation expense is only a little more than 1 ms.

Thank you!
Q&A