Mixed-Criticality Scheduling with I/O

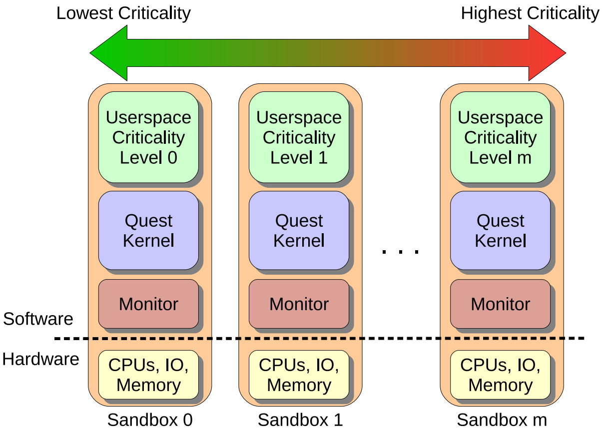
Eric Missimer, Katherine Missimer, Richard West

Boston University
Computer Science

Introduction

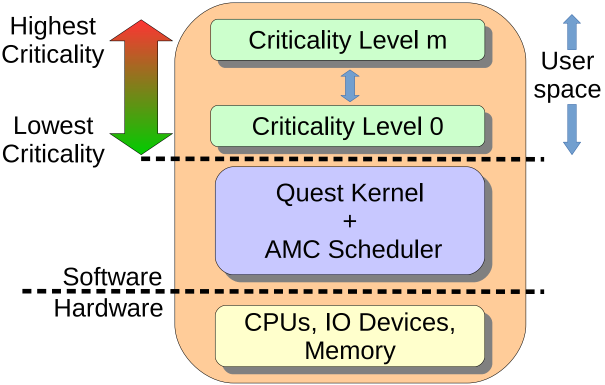
- Previously developed Quest-V separation kernel for mixed-criticality systems
- Mixed-criticality now applied to single Quest kernel
- Mixed-criticality scheduling in presence of I/O
- Work builds on Quest's hierarchical VCPU scheduling framework

Quest-V Mixed-Criticality Support



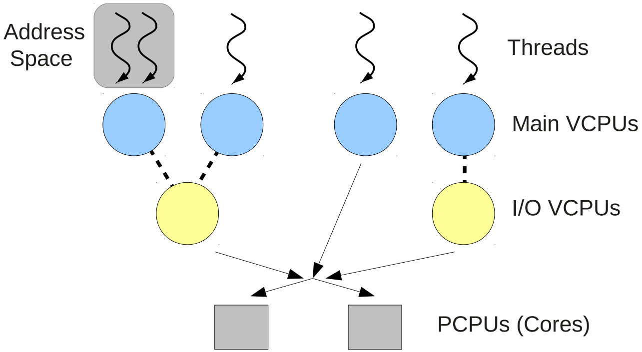
Quest-V – each sandbox has a single criticality level

New Contribution: Quest Mixed-Criticality Support



Quest or a single Quest-V sandbox – multiple criticality levels

Quest Virtual CPUs (VCPUs)



Scheduling hierarchy: threads→VCPUs→PCPUs

Quest VCPUs

- Two classes
  - **Main** $\rightarrow$ for conventional tasks
  - **I/O** $\rightarrow$ for I/O event threads (e.g., ISRs)
- Scheduling policies
  - **Main** $\rightarrow$ Sporadic Server (SS)
  - **I/O** $\rightarrow$ Priority Inheritance Bandwidth-preserving Server (PIBS)

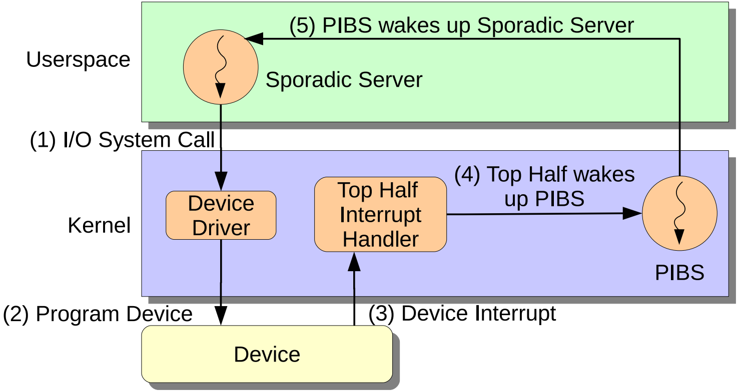
SS Scheduling

- Each SS VCPU has pair (C,T)
- Guarantee budget (C) every period (T) when runnable
- Rate-monotonic scheduling theory applies

PIBS Scheduling

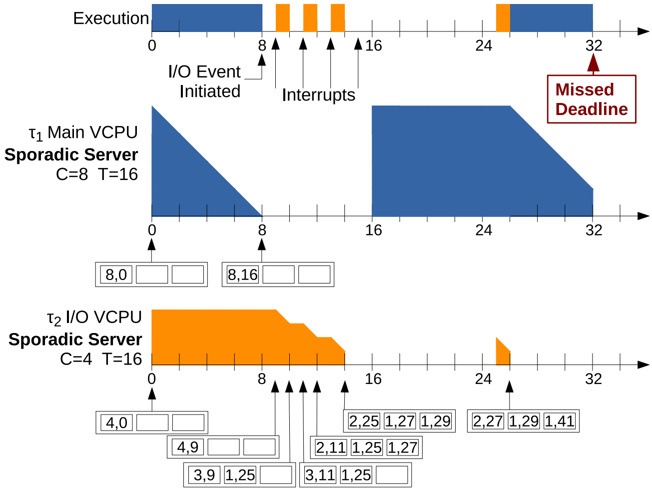
- Each I/O VCPU has utilization factor U_{IO}
- I/O VCPU inherits period & priority from Main VCPU that caused I/O request
- $T_{IO} = T_{Main}$
- $C_{IO} = U_{IO} \times T_{Main}$
- I/O VCPU eligible to execute at $t_e = t + C_{actual} / U_{IO}$
- t =start of latest execution ($>=$ previous eligibility time)

Scheduling Framework



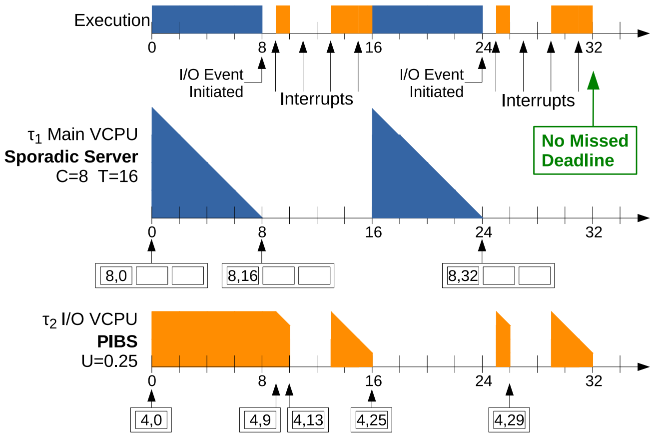
Task & interrupt (CPU & I/O) scheduling in Quest

Sporadic Server Only



Sporadic Server only example – replenishment list fills up, more scheduling events

Sporadic Server + PIBS

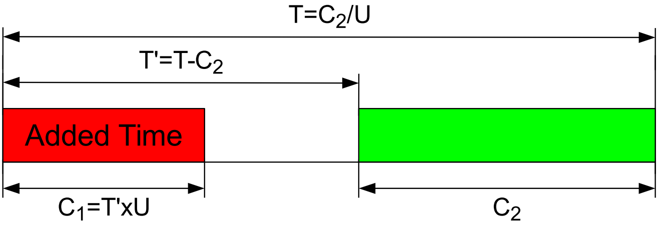


Sporadic Server + PIBS – no delayed replenishment due to fragmentation, less scheduling events

SS+PIBS Utilization Bound

- System of n Main and m I/O VPUs, and one PCPU
- $\sum_{i=0}^{n-1} \frac{C_i}{T_i} + \sum_{j=0}^{m-1} (2 - U_j) \cdot U_j \leq n \cdot (\sqrt[n]{2} - 1)$
- C_i & T_i are the budget capacity and period of Main VCPU V_i
- U_j is the utilization factor of I/O VCPU V_j

PIBS Max Utilization in Main VCPU Period, T



$$
\begin{aligned}
\frac{C_1 + C_2}{T} &= \frac{(T' \times U) + C_2}{T} \\
&= \frac{(T - C_2) \times U + C_2}{T} \\
&= \frac{(C_2/U - C_2) \times U + C_2}{C_2/U} \\
&= (2 - U)\, U
\end{aligned}
$$

U – Utilization of I/O VCPU running PIBS

T – Period of Main VCPU

Adaptive Mixed-Criticality (AMC)

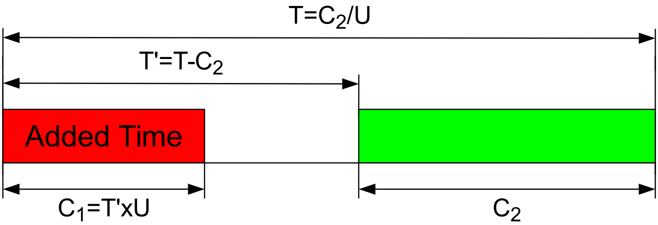
- AMC tasks defined by:
  - Period (T_i), Deadline (D_i)
  - Vector of Capacities $(\overrightarrow{C_i})$ for each criticality level
- System operates in one of a set of criticality levels
  - e.g., Criticality Level $L \in \{\mathtt{LO}, \mathtt{HI}\}$
  - HI-criticality task, $C_i(\mathtt{HI}) \geq C_i(\mathtt{LO})$
  - LO-criticality task, $C_i(\mathtt{HI})$ is undefined

Adaptive Mixed-Criticality (AMC)

- System starts in LO-criticality mode
- If a task uses its entire capacity and does not signal job completion, the system switches into HI-criticality mode
- Only HI-criticality tasks run in HI-criticality mode, using their $C_i(\texttt{HI})$ capacity
- Allows extra capacity to finish HI-criticality jobs before their deadlines

I/O-Adaptive Mixed-Criticality (IO-AMC)

- Extended AMC to include PIBS for I/O requests
- Response time (schedulability) analysis considers
  - when tasks are in HI-criticality mode
  - when tasks are in LO-criticality mode
  - when tasks switch from LO- to HI-criticality mode
  - added interference by PIBS



$$C_1 = (T - C_2) \cdot U = (T - UT) \cdot U$$

I/O-Adaptive Mixed-Criticality (IO-AMC)

- We saw added execution time by a PIBS task is

$$C_1 = (T - C_2) \cdot U = (T - UT) \cdot U$$

- Added interference I_k^q by PIBS τ_k assigned to SS τ_q...

$$\begin{aligned} I_k^q(t) &= (T_q - T_q U_k)\, U_k + \left\lceil \frac{t}{T_q} \right\rceil T_q U_k \\ &= (1 - U_k)\, T_q U_k + \left\lceil \frac{t}{T_q} \right\rceil T_q U_k \\ &= \left(1 + \left\lceil \frac{t}{T_q} \right\rceil - U_k\right) T_q U_k \end{aligned}$$

- IO-AMC response time bound adds I_k^q for each PIBS τ_k to system of Sporadic Servers

I/O-Adaptive Mixed-Criticality (IO-AMC)

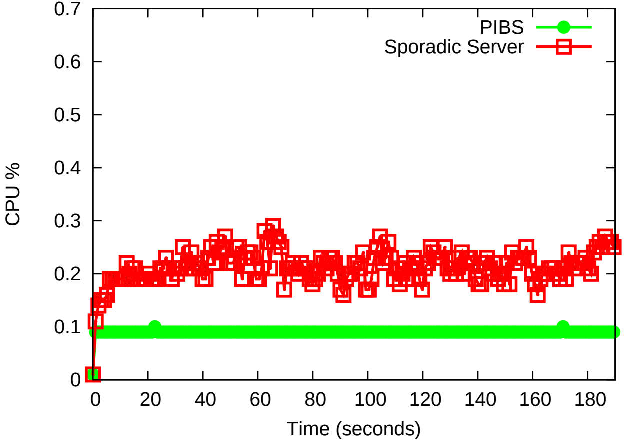
- For each PIBS τ_k, have a vector of utilizations $\overrightarrow{U_k(L)}$ for each criticality level L
- If τ_k is a $\texttt{HI}$-crit PIBS then $U_k(\texttt{HI}) \geq U_k(\texttt{LO})$
- If τ_k is a $\texttt{LO}$-crit PIBS then $U_k(\texttt{LO}) > U_k(\texttt{HI})$
- Then $I_k^q(t, L)$ is the interference by PIBS τ_k assigned to SS τ_q at time t in criticality level L

Quest Experiments

Task	$C(\mathtt{LO})$ or $U(\mathtt{LO})$	$C(\mathtt{HI})$ or $U(\mathtt{HI})$	T
Camera Task ($\mathtt{HI}$-criticality)	$23ms$	$40ms$	$100ms$
CPU Task ($\mathtt{LO}$-criticality)	$10ms$	$1ms$	$100ms$
Bottom Half (PIBS)	$U(\mathtt{LO}) = 1\%$	$U(\mathtt{HI}) = 2\%$	$100ms$
Bottom Half (SS)	$1ms$	$2ms$	$100ms$

Task Set Parameters – Bottom Half handles Camera interrupts

Scheduling Overhead - One USB Camera

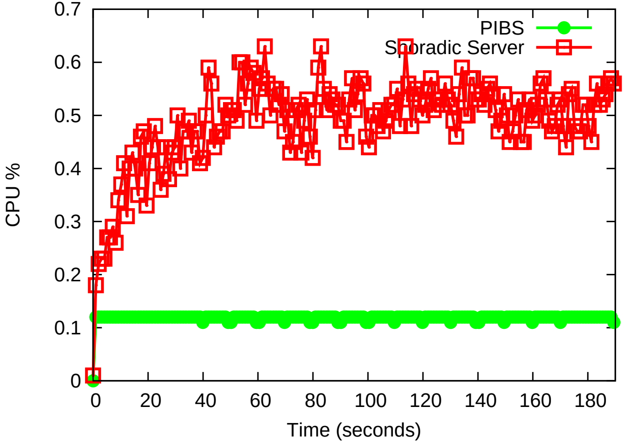


Quest on Intel Core i3-2100 @ 3.1KHz

Camera – $U(\texttt{LO}) = 1\%$ $(1\texttt{ms}/100\texttt{ms})$, $U(\texttt{HI}) = 2\%$ $(2\texttt{ms}/100\texttt{ms})$

Scheduling overhead of SS-only scheme $>$ SS+PIBS

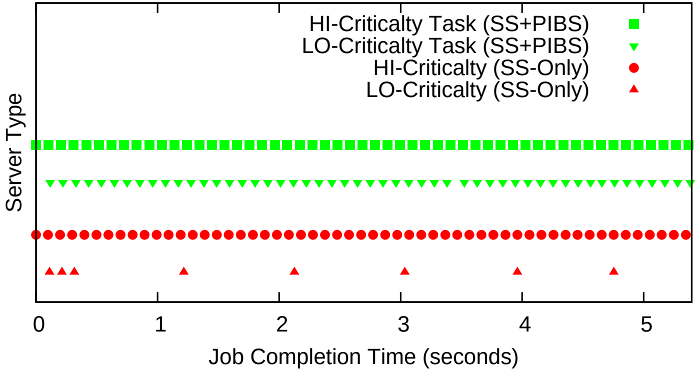
Scheduling Overhead - Two USB Cameras



Camera 1 – $U(\mathtt{LO}) = 1\%$ $(1\mathtt{ms}/100\mathtt{ms})$, $U(\mathtt{HI}) = 2\%$ $(2\mathtt{ms}/100\mathtt{ms})$

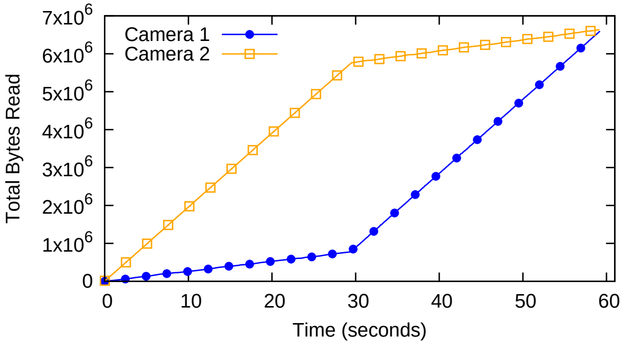
Camera 2 – $U(\mathtt{LO}) = 2\%$ $(2\mathtt{ms}/100\mathtt{ms})$, $U(\mathtt{HI}) = 1\%$ $(1\mathtt{ms}/100\mathtt{ms})$

Extra Interference Forcing a Mode Change



SS-only server for camera interrupts causes task on Main VCPU to deplete budget before job completion.
Mode change with SS-Only causes LO-crit jobs to finish later.

Different Criticality Devices

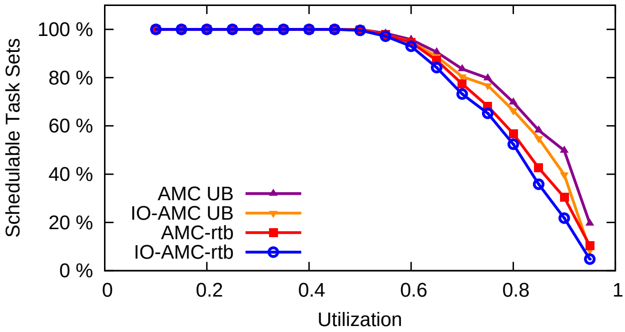


- Can assign criticality levels to devices to control bandwidth
- Mode change at 30 seconds
- Camera 1 – $U(\texttt{LO}) = 0.1\%$, $U(\texttt{HI}) = 1\%$
- Camera 2 – $U(\texttt{LO}) = 1\%$, $U(\texttt{HI}) = 0.1\%$

Conclusions and Future Work

- Added IO-AMC support to Quest RTOS
- Simulations (& analysis) show SS for both tasks & interrupt handlers is theoretically better than SS+PIBS
- Expts show PIBS for interrupt handling incurs lower practical costs
- Ability to assign criticality levels to devices
- Future work to consider more complex scenarios where blocking I/O delays impact task execution

Scheduling Results



500 random task sets per utilization [see paper].
Theoretical performance of IO-AMC-rtb slightly worse than AMC-rtb. This is due to the extra interference PIBS can cause compared to an equivalent Sporadic Server.