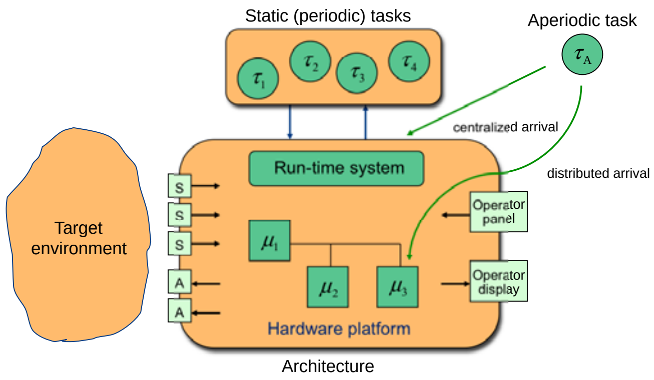
CHALMERS

Handling aperiodic tasks



CHALMERS

Server-less approach

Handling (hard) aperiodic tasks on multiprocessors:

1. Feasibility test:
   - Check whether an aperiodic task can be scheduled on the local processor (distributed arrival) or on any processor (centralized arrival)
2. Task forwarding:
   - In case an aperiodic task cannot be scheduled locally, attempt to forward it to another processor. (distributed arrival only)

CHALMERS

Server-less approach

Feasibility tests for aperiodic tasks:

- Requires a fast on-line algorithm
  - Utilization-based tests are good candidates
- Requires a new concept of processor load
  - Traditional load measures assume periodic tasks
- Requires a new view on task priorities
  - Traditional views, such as static and dynamic, are based on periodic task models
- Requires support for multiprocessors
  - Solutions needed for partitioned and global scheduling

CHALMERS

Feasibility test

Utilization bound for liquid tasks: (Abdelzaher and Lu, 2001)

- Redefined the theory for schedulability analysis by …

  … assuming a liquid task model, where task computation times are in general much smaller than task deadlines ($C_i << D_i$)

  … assuming a time-independent scheduling policy where task priorities do not depend on task arrival times (i.e. not EDF)

  … introducing the concept of synthetic utilization to account for individual task instances that are current at a given time

  … introducing a synthetic utilization bound which, if not exceeded, guarantees that deadlines are met for all current task instances

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Utilization bound for liquid tasks:

- For time-independent scheduling on a uniprocessor …

  … deadline-monotonic scheduling is optimal for liquid aperiodic tasks in the sense that it maximizes the utilization bound

  … the feasibility bound is lower than the bound for rate-monotonic scheduling of periodic tasks (by Liu & Layland)

  … the real utilization of admitted tasks can be much higher than the synthetic utilization bound, often close to 100% ⇒ using the (seemingly low) bound will not underutilize the processor

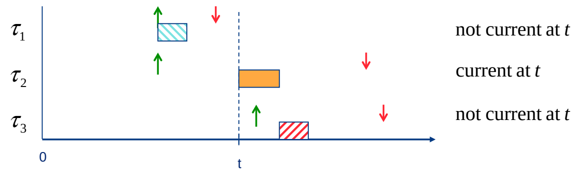
  (note that feasibility tests for periodic tasks do not exhibit such a property)

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Current tasks:

- A task is said to be current at time t if it has arrived at or before t, but its deadline has not yet expired.
- In the example below, task τ_2 is current at time t, while tasks τ_1 and τ_3 are not.



CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Synthetic utilization:

Let $V^{\zeta}(t)$ be the set of current tasks at time t, that is:

$$V^{\zeta}(t) = \{\, \tau_i \mid a_i \leq t \leq a_i + D_i \,\}$$

The synthetic utilization $U^{\zeta}(t)$ is then defined as:

$$U^{\zeta}(t) = \sum_{\tau_i \in V^{\zeta}(t)} \frac{C_i}{D_i}$$

Note that the synthetic utilization is a function of time!

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Processor-utilization analysis for liquid tasks:

- A sufficient condition for DM scheduling of liquid tasks is

$$U^{\zeta}(t) \leq \frac{1}{2} + \frac{1}{2n} \qquad \text{for } n < 3$$

$$U^{\zeta}(t) \leq \frac{1}{1 + \sqrt{\frac{1}{2}\left(1 - \frac{1}{n-1}\right)}} \qquad \text{for } n \geq 3$$

CHALMERS UNIVERSITY OF TECHNOLOGY

Feasibility test

Processor-utilization analysis for liquid tasks:

- A conservative lower bound on the utilization can be derived by letting $n \rightarrow \infty$

$$\lim_{n\to\infty} \frac{1}{1+\sqrt{\frac{1}{2}\left(1-\frac{1}{n-1}\right)}} = \frac{1}{1+\sqrt{\frac{1}{2}}} \approx 0.586$$

This bound is thus lower than the RM bound for periodic tasks

CHALMERS UNIVERSITY OF TECHNOLOGY

Feasibility test

Synthetic utilization (improved definition):

If the processor becomes idle due to a task completing before its deadline, the synthetic utilization can be immediately reset to zero (since tasks that precede the idle time will have no effect on the schedulability of future task arrivals.)

The synthetic utilization $U^{\zeta}(t)$ can therefore be redefined as:

$$U^{\zeta}(t) = \begin{cases} \sum_{\tau_i \in V^{\zeta}(t)} C_i / D_i & \text{if the processor is busy} \\ 0 & \text{if the processor is idle} \end{cases}$$

The average synthetic utilization will thus generally be lower than the average real utilization of admitted tasks

CHALMERS UNIVERSITY OF TECHNOLOGY

Feasibility test

Some follow-up work:

- Multiprocessor scheduling of liquid tasks
  - Time-independent global multiprocessor scheduling
  - Independent liquid aperiodic tasks
- Relaxation of the liquid task assumption
  - Time-independent uniprocessor scheduling
  - Generic aperiodic tasks with shared resources
- Multiprocessor scheduling of generic tasks
  - Priority-driven global multiprocessor scheduling
  - Priority-driven partitioned multiprocessor scheduling
  - Independent generic aperiodic tasks

CHALMERS UNIVERSITY OF TECHNOLOGY

Feasibility test

Multiprocessor scheduling of liquid tasks: (Abdelzaher et al, 2002)

- Time-independent global multiprocessor scheduling
- Deadline-monotonic scheduling is also optimal among time-independent multiprocessor scheduling policies
- Synthetic utilization bound is identical to the uniprocessor case, and is independent of the number of processors
- Synthetic utilization is redefined as:

$$U^{\zeta}(t) = \begin{cases} \frac{1}{m}\sum_{\tau_i \in V^{\zeta}(t)} C_i / D_i & \text{if all processors are busy} \\ 0 & \text{otherwise} \end{cases}$$

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Relaxation of liquid task assumption: (Abdelzaher & Sharma, 2003)

- Time-independent uniprocessor scheduling
- Presents a generalized synthetic utilization bound that is a function of parameters that depend on the scheduling policy used:
  - Preemptable deadline ratio
  - Resource blocking ratio
- For deadline-monotonic scheduling, the synthetic utilization bound reduces to the optimal bound for liquid tasks

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Relaxation of liquid task assumption:

Let $D_{\max}^{k}$ be the longest relative deadline among the tasks with priority equal to or higher than task τ_k.

The preemptable deadline ratio α is then defined as:

$$\alpha = \min_{\forall k} \frac{D_k}{D_{\max}^{k}}$$

Note that $\alpha = 1$ for deadline-monotonic scheduling since tasks with higher priorities have shorter deadlines.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Relaxation of liquid task assumption:

Let B_k be the longest critical region being used by tasks with a priority lower than task τ_k, and that calls critical regions with a ceiling priority equal to or higher than the priority of τ_k (that is, PCP is assumed).

The resource blocking ratio γ is then defined as:

$$\gamma = \max_{\forall k} \frac{B_k}{D_k}$$

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Relaxation of liquid task assumption:

The generalized synthetic utilization bound U_{LB}^{ζ} for time-independent uniprocessor scheduling is then defined as:

$$U_{LB}^{\zeta} = 1 + \alpha - \sqrt{1 + \gamma + \alpha^2}$$

A sufficient condition for time-independent uniprocessor scheduling of aperiodic tasks is thus:

$$U^{\zeta}(t) \leq 1 + \alpha - \sqrt{1 + \gamma + \alpha^2}$$

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Relaxation of liquid task assumption:

For deadline-monotonic scheduling ($\alpha = 1$) of independent tasks ($\gamma = 0$), the utilization bound evaluates to

$$1+\alpha-\sqrt{1+\gamma+\alpha^2} = 2-\sqrt{2} = \frac{2(1+\sqrt{\frac{1}{2}})-\sqrt{2}(1+\sqrt{\frac{1}{2}})}{1+\sqrt{\frac{1}{2}}} = \frac{1}{1+\sqrt{\frac{1}{2}}}$$

The bound for DM thus reduces to that of the liquid task case

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Multiprocessor scheduling of generic tasks: (Andersson et al, 2003)

- Global multiprocessor scheduling with *m* processors:
  - The EDF-US{m/(2m-1)} scheduling policy is extended to handle aperiodic tasks
  - Added difficulty: the number of "heavy" tasks is not necessarily the same at all times (as is the case with periodic tasks) ⇒ the number of processors available for "light" tasks may vary with time
  - It is shown that the aperiodic EDF-US{m/(2m-1)} has a utilization bound of at least m/(2m-1)
  - It is shown that no priority-driven global scheduler can have a utilization bound higher than 0.5 + 0.5/m

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Global multiprocessor scheduling of aperiodic tasks:

- Priority-driven multiprocessor scheduling:
  - For every pair of tasks, one task has higher priority than the other task in the pair, and these relative priority orderings never change (i.e., not pfair scheduling)
- Synthetic utilization is redefined as:

$$U^{\zeta}(t) = \begin{cases} \frac{1}{m}\sum_{\tau_i \in V^{\zeta}(t)} C_i/D_i & \text{if } \exists \text{ a busy processor} \\ 0 & \text{otherwise} \end{cases}$$

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Global multiprocessor scheduling of aperiodic tasks:

- A sufficient condition for global scheduling of aperiodic tasks on *m* processors using EDF-US{m/(2m-1)} is

$$U^{\zeta}(t) \le \frac{m}{2m-1}$$

EDF-US{m/(2m-1)} thus has close-to-optimal performance (recall that maximum achievable bound is 0.5+0.5/m). For an infinite number of processors, EDF-US{m/(2m-1)} is optimal among priority-driven global schedulers for aperiodic tasks.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Multiprocessor scheduling of generic tasks: (Andersson et al, 2003)

- Partitioned multiprocessor scheduling with m processors:
  - An EDF-FF scheduling policy is proposed to handle aperiodic tasks
  - Added difficulty: a task "disappears" as soon as its deadline has expired ⇒ "first-fit" bin-packing must be redefined
  - It is shown that the aperiodic EDF-FF scheduler has a utilization bound of at least 0.31
  - It is also shown that no analysis of capacity bounds for the aperiodic EDF-FF scheduler can achieve a utilization bound higher than 3/7 ≈ 0.428

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Partitioned multiprocessor scheduling of aperiodic tasks:

The synthetic utilization $U_p^\zeta(t)$ on processor p is defined as:

$$U_p^\zeta(t) = \sum_{\tau_i \in V_p^\zeta(t)} \frac{C_i}{D_i}$$

where

$$V_p^\zeta(t) = \{\, \tau_i \,|\, a_i \le t \le a_i + D_i \wedge (\tau_i \text{ is assigned to processor } p) \}$$

The total synthetic utilization is still $U^\zeta(t) = \frac{1}{m}\sum_{\tau_i \in V^\zeta(t)} C_i / D_i$

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Partitioned multiprocessor scheduling of aperiodic tasks:

A processor is said to be occupied at time t if there is at least one task that is both current at time t and that is assigned to the processor. A processor that is not occupied is called empty.

Let $transition_p(t)$ be the latest time $\le t$ such that processor p makes a transition from being empty to being occupied.

If a processor p has never been occupied, then
$transition_p(t) = -\infty$

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Partitioned multiprocessor scheduling of aperiodic tasks:

The EDF-FF algorithm:

When a task τ_i arrives it is assigned to the occupied processor with the earliest $transition_p(a_i)$ for which

$$U_p^\zeta(t) = \sum_{\tau_k \in V_p^\zeta(t) \cup \tau_i} \frac{C_k}{D_k} \le 1$$

If no occupied processor passes the test, the task is assigned to an arbitrary empty processor (if no empty processor exists, EDF-FF declares failure.)

CHALMERS
UNIVERSITY OF TECHNOLOGY

Feasibility test

Partitioned multiprocessor scheduling of aperiodic tasks:

- A sufficient condition for partitioned scheduling of aperiodic tasks on *m* processors using EDF-FF is

$$U^{\zeta}(t) \leq 0.31$$

EDF-FF thus has no tight utilization bound (it is known from earlier work that maximum achievable bound for partitioned multiprocessor scheduling is 0.5). Recall, however, that no analysis of EDF-FF can achieve a utilization bound higher than 3/7.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Server-less approach

Forwarding tasks in a multiprocessor system:

- Immediate drop:
  - If an aperiodic task is not centrally schedulable, it is dropped.
- Focused addressing and bidding:
  - If an aperiodic task is not locally schedulable, it is forwarded to a suitable processor candidate based on statistics and bids.
- Load sharing:
  - If an aperiodic task is not locally schedulable, it is forwarded to a suitable processor candidate based on load thresholds and "buddy" processors.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Task forwarding

Focused addressing and bidding: (Ramamritham *et al*, 1989)

- Each processor maintains a table of currently-guaranteed tasks. It also maintains a table of the surplus computational capacity at every other processor. The surplus capacity is expressed as fractions of a (future) time window of a common size.
- If a processor cannot guarantee an aperiodic task locally, it consults its surplus table and selects the processor (focused processor) that is most likely to successfully schedule the task.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Task forwarding

Focused addressing and bidding:

- Because of possible out-of-date entries in the surplus table, the processor might also send out requests-for-bids to other lightly-loaded processors. These bids are then sent to the focused processor.
- The focused processor determines whether to schedule locally or pass the task on to the highest bidder. Tasks that cannot be guaranteed locally, or through focused addressing and bidding, are rejected.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Task forwarding

Focused addressing and bidding:

- Important overhead factors that must be taken into account during the bidding process:
  - task deadline and task execution time
  - task transfer time (between processors)
  - time taken by the focused processor to make a decision
  - time taken to respond with a bid (schedulability test)
  - surplus time available at bidder

CHALMERS
UNIVERSITY OF TECHNOLOGY

Task forwarding

Load sharing: (Shin & Chang, 1989)

- Each processor has three states of processor loading:
  - Underloaded: the processor is judged to be in a position to accept and execute tasks from other processors.
  - Fully loaded: the processor will neither accept tasks from other processors, nor offload tasks onto other processors.
  - Overloaded: the processor looks for other processors on which to offload some tasks.
- The processor load is derived from the number of task instances awaiting service in the processor's ready queue

CHALMERS
UNIVERSITY OF TECHNOLOGY

Task forwarding

Load sharing:

- To update processor state, the load is compared against a set of load thresholds corresponding to the loading states.
- When a processor makes a transition into and out of the underloaded state, it broadcasts an announcement to its buddy set which is a limited subset of the processors chosen mainly based on the nature of the interconnection network.

CHALMERS
UNIVERSITY OF TECHNOLOGY

Task forwarding

Load sharing:

- Each processor is aware of whether any members in its buddy set are in the underloaded state. An overloaded processor chooses an underloaded member (if any) in its buddy set on which to offload a task.
- Each processor has an ordered list of preferred processors. Careful design of the lists will reduce the risk of “flooding” underloaded processors.