Real-Time Architectures 2003/2004

Resource reservation

Reinder J. Bril

Resource Reservation

- Description
- Example
- Motivation
- Application domains
- Some issues…
- Concluding remark

Description

- Resource reservation is a technique (or *mechanism*) at the level of an RTOS, providing a *virtual* platform.
- Based on four elements:
  - Admission control:
    - Schedulability test.
  - Scheduling:
    - CPU: similar to scheduling of tasks.
  - Accounting:
    - Keep track of what has been “consumed”.
  - Enforcement:
    - Do *not* allow over-consumption !

Example

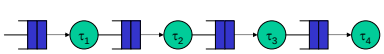
- Resource budget example for CPU:
  - Algorithm Y, rate = 50Hz, budget = 25%
  - for every period T_Y of 20 msec, a budget B_Y of 5 msec CPU time is allocated to Y
  - Hence, a virtual processor with 25% of the cycles.
- Multiple resources (topic of research):
  - Processor(s),
  - Co-processor(s), e.g. VLD;
  - Memory;
  - Bus;
  - Network.

Motivation

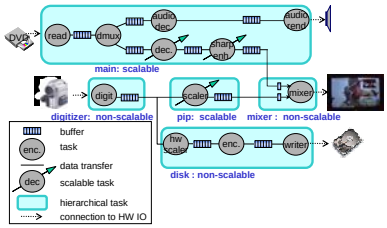
- Priorities typically serve multiple purposes:
- They are *used* to:
  - Guarantee deadlines (FPS and DPS):
    - Given task characteristics and
    - based on particular assumptions (e.g. C_j);
  - Assign relative importance (FPS only):
    - Higher priority task τ_H always takes precedence over a lower priority task τ_L:
      - Over-runs of τ_H hamper τ_L, but
      - over-runs of τ_L do not hamper τ_H.
    - Optionally split a job in sub jobs to remain optimal (RMS); see [Sha et al 86].

Motivation

- Priorities *used* for:
  - Co-operation protocols (FPS only): see [Gonzalez et al 91] and [Groba et al 02];
  - Example: buffer-allocation:
    - $\rho_1 > \rho_2 > \rho_3 > \rho_4$: push-model: pile-up before τ_4;
    - $\rho_1 < \rho_2 < \rho_3 < \rho_4$: pull-model: pile-up before τ_1;
    - $\rho_1 > \rho_2 > \rho_3 < \rho_4$: push/pull-model: pile-up before τ_3.
    - General: pile-up before lowest priority task in stream, allowing a pile-up where the memory needs are minimal.
    - Example: *before* video decoder.



Motivation



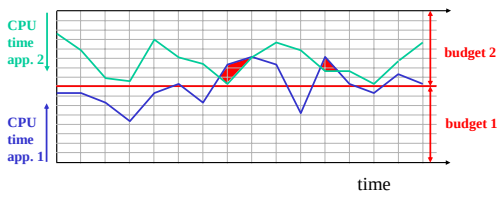
Motivation

- References:
  - [Gonzalez et al 91] M. González Harbour, M.H. Klein, and J.P. Lehoczky, *Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority*, In: Proc. 12th IEEE Real-Time Systems Symposium (RTSS), pp. 116 – 128, 1991.
  - [Groba et al 02] A.M. Groba, A. Alonso, J.A. Rodriques, M. Garcia Valls, *Response time of streaming chains: analysis and results*, In: Proc. 14th IEEE Euromicro Conference on Real-Time Systems, pp. 182 – 189, 2002.
  - [Sha et al 86] L. Sha, J. Lehoczky, and R. Rajkumar, *Solutions to some practical problems in prioritized preemptive scheduling*, In: Proc. 7th IEEE Real-Time Systems Symposium (RTSS), pp. 181 – 191, 1986.

Motivation

- Temporal protection
  - FPS assumes worst-case close to average-case.
  - Worst-case computation time difficult to estimate:
    - Complexity of the code;
    - Non-deterministic platform (optimized for *average-case* rather than *worst-case* situations):
      - Cache memories;
      - Bus;
      - …
  - When jobs need more time, temporal faults occur:
    - Analysis no longer valid;
    - Potential deadline misses:
      - FPS: task itself and *all lower priority* tasks;
      - DPS: task itself and *all other* tasks !
  - A main motivation for resource reservations !

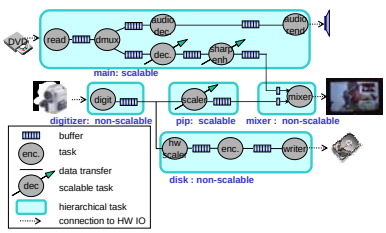
Temporal protection



Motivation

- Independent design, analysis and validation:
  - Complexity of real-time systems
    - Increase of development cost (time and money);
    - Final integration phase currently a bottleneck.
      - Single set of priorities for all tasks in all subsystems.
  - Alternative:
    - Loosely coupled subsystems;
    - Running on a virtual platform, with;
    - Hierarchical scheduling:
      - Virtual platforms;
      - Applications/tasks/threads on a virtual platform
        - » Standard RTOS API;
        - » Local (proprietary) RTOS/scheduling.

Motivation



Motivation

- 5 hierarchical tasks (or “RCEs”):
  - 3 “user” applications: main, pip, disk;
  - 2 “infra-structure” applications: digitizer, mixer.
- Hierarchical scheduling:
  - Hierarchical tasks by, e.g., DPS (high utilization);
  - Tasks of an RCE by, e.g., FPS (current industrial practice, buffer allocation).
- Virtual platform contains:
  - “private” processor;
  - “private” memory (buffers).

Motivation

- Re-use of legacy applications:
  - Integration in new system by means of a virtual platform *with characteristics of old system*.
- Quality of Service (QoS):
  - Trade quality for resources
    - Relaxing timing requirements (e.g. frame-rate fluctuations);
    - Approximate results (e.g. quality reduction of individual frames).
- Hybrid open systems:
  - Real-time next to non-real time applications:
    - Worst-case resource allocation for real-time applications;
    - Average-case for non-real time applications.
  - Traditionally conceived as open systems;
  - Example: PC evolving to multimedia platform.

Application domains

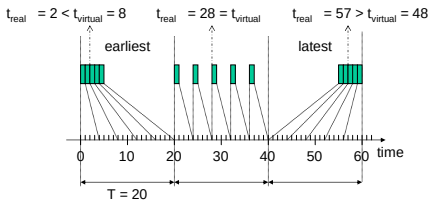
- Aerospace:
  - Hard real-time systems;
  - Need for temporal protection.
- Real-time control systems:
  - Hybrid systems:
    - Critical (hard real-time) applications next to
    - less critical applications (e.g. multimedia).
- Multimedia systems:
  - Media processing: from dedicated HW to SW;
  - High load fluctuations;
  - Consumer devices:
    - Control + media processing → hybrid open systems;
    - Robustness and predictability → temporal protection;
    - Short time-to-market → independent development;
    - Cost-effective → worst-case not affordable → QoS.

Some issues…

- Processing of applications and budgets:
  - Synchronous (3D graphics);
  - Asynchronous (high quality video).
- Granularity:
  - Virtual time versus real time;
  - Cost of context switching and cache flushing.
- Blocking on shared physical resources:
  - Priority inversion problem for budgets ?
  - Accounted to which budget ?

Virtual time and real time

T = 20 msec and B = 5 msec, i.e. 25% of CPU



- *allocation* & *provision* granularity: 1 msec
- *temporal* granularity: max(|t_{real} – $t_{virtual}$|) = 15, acceptable ?
- i.e. 25% of CPU *on average* not of any arbitrary interval

Concluding remark

- Origin:
  - For memory (spatial protection): MMUs, old !
  - for CPU (temporal protection): [Mercer et al 94].
- RTOS-vendors:
  - Integrity: notion of partition;
  - VxWorks: notion of protection domain.
- Results:
  - Some initial academic results,
  - Some practical experience, and
  - Many challenges [Steffens et al 03].

References

- [Mercer et al 94] C.W. Mercer and S. Savage and H. Tokuda, *Processor Capability Reserves: Operating System Support for Multimedia Applications*, In: Proc. International Conference on Multimedia Computing and Systems (ICMCS)", pp. 90-99, May 1994.
- [Rajkumar et al 98] R. Rajkumar and K. Juvva and A. Molano and S. Oikawa, *Resource Kernels: A Resource-Centric Approach to Real-Time and Multimedia Systems*, In: Proc. SPIE Vol. 3310, Conference on Multimedia Computing and Networking, pp. 150-164, January 1998.
- [Steffens et al 03] L. Steffens, G. Fohler, G. Lipari, G. Buttazzo, *Resource Reservation in Real-Time Operating Systems – a joint industrial and academic position*, In: Proc. International Workshop on Advanced Real-Time Operating System Services (ARTOSS), pp. 25 – 30, July 2003.