Errata and comments to textbook

G. C. Buttazzo, "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications", Second Edition

Springer, 2005 (first reprint 2010).

Marco Cesati — Tue, 08 Mar 2011 15:10:58 +0100

The marker "BEC" indicates that the entry is taken from Buttazzo's Errata Corrige, available at `http://retis.sssup.it/~giorgio/errata-HRT2.pdf`

- Page 13, first two paragraphs

  Cycle stealing is a very old DMA mechanism dating back to the ISA bus. In modern computer architectures much more efficient techniques like *bus mastering* are employed. It is still true, however, that transactions on the bus can suffer from impredictable delays caused by concurrent transactions issued by other devices.

- Page 13, last paragraph

  "Today, however, since memory has an access time almost comparable to that of cache, the main motivation for having a cache is not only to speed up process execution but also to reduce conflicts to other devices."

  I think that this sentence is conceptually wrong, because it suggests that access times of static RAM (caches) and dynamic RAM (memory) are similar or comparable. While it is true that dynamic RAM has become faster, CPU speed (and correspondingly static RAM speed) increased much more, thus the so-called *memory wall* effect is nowadays worse than before. Experiments performed on modern computer architectures show that the caches improve the running time of a generic application of a factor between 40 and 100. (To get an idea, consider that an almost istantaneous 1-second program might require without caches more than one minute to complete!)

Furthermore, the idea that caches reduce conflicts between hardware devices is opinable. I would say that it is rather true the opposite: several devices that want to access the memory subsystem at once may induce consistency and coherency problems that must be addressed with specific arbitration protocols at the cache level.

- Page 14, first paragraph

  "Furthermore, when performing write operations into memory, the use of the cache is even more expensive in terms of access time, because any modification made on the cache must be copied to the memory in order to maintain data consistency."

  This sentence is misleading. Normally, write operations from cache to memory are not directly associated with program instructions that "write in memory". In fact, the usual consequence of a "write" instruction is to change a value inside a cache line: there is no immediate need to update the memory before executing the next instruction of the program.

  Actually, write operations in dynamic memory occur whenever a "dirty" cache line must be evicted in order to make place for some other data. This can happen when executing both "read" and "write" program instructions.

- Page 22, line 12 from bottom

  The definition of step function given in the text is correct. Perhaps an easier definition is: "a function whose domain can be decomposed in semi-closed intervals in which the function assumes a constant integer value between 0 and n". That is, $\sigma : \mathbf{R}^+ \to \{0, 1, \ldots n\}$ and

  $$\forall t_0 \in \mathbf{R}^+,\ \exists\, t_1, t_2 \in \mathbf{R}^+ \text{ s.t. } t_1 \leq t_0 < t_2 \text{ and } \sigma([t_1, t_2)) = \{\sigma(t_0)\}\,.$$

- Page 22, line 6 from bottom

  "Each interval $[t_i, t_{i+1})$" should be "Each maximal interval $[t_i, t_{i+1})$" (that is, each interval characterized by the particular constant value that $\sigma(t)$ assumes in each of its points).

- BEC Page 23, Figure 2.2

  The figure should not have *vertical lines*, and the value $\sigma(t)$ of the schedule function in a context switch time t is equal to $\sigma(t + \varepsilon)$. A similar problem occurs in Figures 2.3, 2.15(d), 4.14, 4.17, 5.3, 5.4, 5.5, 5.6, 5.7, 5.11, 5.12, 5.13, 5.17, 5.18, 5.19, 5.21, 5.22, 6.1, 6.2, 6.3, 6.4, 6.14, 7.6, 7.7, 7.8, 7.12, 7.19, 8.5, 8.6, and 8.10.

- BEC Page 25, line 8 from bottom

  "an aperiodic job by" should be "an aperiodic task by".

- Page 25, lines 5 and 4 from bottom

  "a periodic task can be completely characterized by its computation time C_i and its relative deadline D_i": of course, it is characterized also by its period T_i, when $T_i \neq D_i$.

- Page 28, 5 from bottom

  "a *resource* is any software structure": it could conceivably be a hardware mechanism too, such as a range of memory cells or a shared printer.

- Page 29, lines 5 and 6 from top

  "A piece of code executed under mutual exclusion constraints is called a critical section." Observe that it is possible to use the same locking mechanism (e.g., a specific semaphore) to protect two different code regions. Each of them is a critical section by itself, and furthermore the two sections are mutually exclusive with respect to each other.

- Page 31, Figure 2.11 and corresponding comments in text

  It should be observed that an OS might have mechanisms to reduce the overhead of synchronization primitives, which may violate the simple transitions rules shown in the figure. For example, the OS might keep track of the highest-priority task blocked on a semaphore and perform a direct transition from the WAITING state to the RUN state of that process as soon as the semaphore is freed.

- Page 33, lines 3 from top

  Scheduling decisions can also be taken when a sleeping task awakes, that is, when a task becomes ready to execute again because the resource needed to continue its execution becomes available.

- BEC Page 34, Figure 2.12

  The arrow labeled "preemption" points to the wrong state. It should go from "RUN" to "READY".

- Page 35, Figure 2.13

  "$t\ _0$" should be "t_0".

- BEC Page 41, line 3 from bottom

  "between task J_4 and tasks J_5 and J_6" should be "between task J_4 and tasks J_7 and J_8".

- BEC Page 43, first paragraph under "ANOMALIES UNDER..."

  The (relative) priorities of the tasks are lacking. Similar to the example started at page 39, tasks are assumed to be sorted by decreasing priorities.

- Page 47, line 8 from top

  "$d_a \leq d_b$" should be "$d_a < d_b$".

- Page 63, line 1 from top

  The partition of the tasks in Γ respects the ordering of the task induced by their start times.

- Page 73, lines 15 and 16 from top

  A better description for "response time" is: time interval between the release time and the finish time of the instance.

- Page 73, lines 11 and 10 from bottom

  "the interval between the critical instant and the response time of the corresponding request of the task" should be "the interval between the critical instant and the finish time of the corresponding instance of the task".

- BEC Page 73, line 9 from bottom

  "Relative Release Jitter" should be "Relative Start Time Jitter".

- BEC Page 73, line 6 from bottom

  "Absolute Release Jitter" should be "Absolute Start Time Jitter".

- BEC Pages 98–100

  The example on pages 98–100 contains a number of errors.

  - The iteration steps to determine R_4 on page 100 do not conform to the description of the calculation of R_i on page 98. This problem finds its cause in a partial update from the first toward the second edition, where the calculation has been updated, but the iterative steps to determine R_4 has remained unaltered. Please note that the final result found for R_4 is correct.
  - The value for $I_4(3)$ is 8 (rather than 7, as given in the book).
  - In Figure 4.14, the term f_4^k is not defined and is meant to be equal to $I_4(k) + C_4$. The figure is meant to show I_4 as a function of t, i.e., $I_4(t) = \sum_{j<i} \lceil t/T_j \rceil$. Since this is a function, vertical lines should not be there. Finally, for $t = 0$, it should be $I_4(0) = 0$ and $I_4(t) = 4$ for $t \in (0, 4]$.

- BEC Page 115, Section 5.3.1

  It is worth observing that the schedulability analysis of fixed priority servers, although carried out with the Liu and Layland method, can also be performed using the Hyperbolic Bound approach [1] or the Response Time Analysis [2]. See the related papers for more details.

[1] E. Bini, G. Buttazzo, and G. Buttazzo, "Rate Monotonic Analysis: The Hyperbolic Bound", IEEE Transactions on Computers, Vol. 52, No. 7, pp. 933–942, July 2003.

[2] G. Bernat and A. Burns, "New results on fixed priority aperiodic servers", Proceedings of the 20th Real-Time Systems Symposium, pp. 68–78, December 1999.

- BEC Page 116, line 8 from bottom

  When computing the response time of an aperiodic job under the Polling Server, there is a problem for situations where $C_a = k\,C_s$, where $k \in \mathbf{N}^+$. In that case, $F_a = k$ and $R_a = 0$. The formula for F_a should therefore be: $F_a = \lceil C_a/C_s \rceil - 1$. For the special case where $C_a = k\,C_s$, R_a will then become equal to C_s, leading to the right result.

- BEC Page 376, Exercise 3.4

  Solution to Exercise 3.4 contains two errors:

  - Based on the heuristic function $H = a + C + D$, the schedule found is $\{J_2, J_4, J_3, J_1\}$.
  - As illustrated in Figure 12.2, there is only one feasible schedule, i.e., $\{J_3, J_2, J_4, J_1\}$. This schedule can be found by using the heuristic function $H = a + d$, and not $H = a + D$.

- BEC Page 388, Exercise 6.6

  "$d_1^{(2)} = f_1^{(1)} = 4$" should be "$d_1^{(2)} = f_1^{(1)} = 5$"

  "$d_1^{(3)} = f_1^{(2)} = 3$" should be "$d_1^{(3)} = f_1^{(2)} = 4$"

- BEC Page 391, Exercise 7.5

  Solution to Exercise 7.5 contains an error: D_3 cannot potentially block τ_1, because τ_1 does not use resource D. This does not change the numerical result of the exercise.