Application Specific Performance Indicators for Quantitative Evaluation of the Timing Behavior for Embedded Real-Time Systems

Frank König
Dave Boers
Carl von Ossietzky University Oldenburg
Unterm Berg 34, D-26676 Harkebrügge
mail@frank-koenig.eu / djb@qne.de

Frank Slomka
Department of Embedded Systems/Real-Time Systems
Faculty of Engineering Science and Computer Sciences
Ulm University, 89069 Ulm
frank.slomka@uni-ulm.de

Ulrich Margull
1 mal 1 Software GmbH
Maxstraße 31, D-90762 Fürth
ulrich.margull@1mal1.com

Michael Niemetz
Gerhard Wirrer
Continental Automotive GmbH
P.O. Box 100943, D-93009 Regensburg
michael.niemetz@continental-corporation.com

Abstract

In the design and development of embedded real-time systems the aspect of timing behavior plays a central role. Especially, the evaluation of different scheduling approaches, algorithms and configurations is one of the elementary preconditions for creating not only reliable but also efficient systems - a key for success in industrial mass production. This is becoming even more important as multi-core systems are more and more penetrating the world of embedded systems together with the large (and growing) variety of scheduling policies available for such systems. In this work simple mathematical concepts are used to define performance indicators allowing to quantify the benefit of different solutions of the scheduling challenge for a given application. As a sample application some aspects of analyzing the dynamic behavior of an combustion engine management system for the automotive domain are shown. However, the described approach is flexible in order to support the specific optimization needs arising from the timing requirements defined by the application domain and can be used with simulation data as well as target system measurements.

1 Introduction

For real-time systems the selection of the scheduling algorithm is an important decision, as it is the key to keeping the required system timing as well as to building a robust and efficient system. During the last decades research has developed a large number of powerful scheduling algorithms and policies together with theoretical proofs showing their optimality and/or performance (see for example [4]). Lately, this area of research has gained new momentum due to the fact that multi-core controllers are more and more entering the world of embedded real-time systems. Theoretical proofs are essential when developing new scheduling algorithms, however, they are less helpful for the challenge of selecting a scheduling algorithm in practice, as typically some of the constraints of the proofs are violated in real systems. In addition, an industrial embedded system has a broad range of very different timing requirements. This is even more the case when developing product lines, where requirements of individual systems are merged in order to create a generic product platform, with different, even contradicting optimization targets.

The automotive business, for example, has typically (e.g. in the power train area) a very high pressure for optimized resource usage, due to the very high volumes of hardware that need to be produced.

Despite this, the state of the art in terms of scheduling in the automotive power train area is still dominated by OSEK/VDX — an operating system standard featuring a fixed priority scheduling which seems quite simple compared to the concepts developed by researchers and used in other areas since years. Therefore there have been already attempts to combine the reliability of existing solutions with the better real-time performance of newer concepts (e.g. see [2]). Clearly, for improving the situation an evaluation of different scheduling concepts with respect of the specific optimization preferences of the application at hand is re-

978-1-4244-3781-8 DATE 09 / Nizza

quired.

Besides hardware-in-the-loop (HIL) and real environment tests, a promising approach to determine the benefits for a given application that can be expected from scheduling algorithms is to set up a simulation model of the system and to evaluate different candidates.

When evaluating the timing behavior of a system, either with simulation, hardware-in-the-loop or in the real environment, the following questions arise:

- Comparison of two scheduler algorithms or configurations for a system: Which one of the available schedulers is "better" than the other?
- Characterization of the quality of scheduler activity in case of system changes: This is useful to evaluate a scheduler under changing constraints, e.g. increasing CPU load or changing operation states.
- Measure how closely a simulated system reflects the real system: Is the simulation result comparable to the real system behavior?
- Production of product health indicators: Calculation of indicators suitable for monitoring the quality of an application during development or life time.

This work shows several concepts that have been used in an automotive power train environment to judge the benefits of changes in the scheduling. The work consists of two parts: part one gives a non-exhaustive list of disciplines — or performance indicators — for each individual task. In the second part it is shown how the data which have been obtained typically on a per-task basis can be condensed to a linear scale by using concepts well known in mathematics in order to enable easy comparison of different system configurations. Finally, as an example we show an application of the described methods for comparing the effect of two different priority assignments in a priority based system.

2 Relevant Performance Indicators

The first step when evaluating the different scheduling concepts for a given real-time system is to define what are the critical properties in the system that need to be controlled and optimized. Consequently, performance indicators need to be developed expressing those properties, so that the required data can be obtained e.g. from some kind of simulation or a real target system.

In this paper, we define a *performance indicator* to be a (positive) number that expresses a relevant aspect of the timing behavior of the system. The indicator can be related to one task (e.g. response time) or the complete task set / system (e.g. overall idle time). To ease the handling, we require that for each performance indicator a larger value expresses a worse performance than a smaller value. It is useful to normalize the indicators so that they are comparable between different tasks or even between different systems.

The indicators given in the following sections should be considered as examples being applicable for soft and hard real-time applications. Due to the huge variety of requirements to such systems, adjustment to the domain of the given application will be often inevitable in order to reach the desired quality of results.

2.1 Response Time

The response time r of a task T is defined as the difference between finishing time f and activation time a

$$r = f - a \tag{1}$$

For a given deadline d, the response time r is related to the lateness via $l = r - d$.

The response time r of a task T varies for each instance of the task. When looking at a running system for some time interval, one gets a response time vector $\mathbf{r} = (r_1, r_2, \ldots r_M)$. This can be condensed into a single value by taking the maximum response time $r_{\max} = \max(\mathbf{r})$, which represents the worst-case time behavior of the task. However, in real systems the worst-case value is too pessimistic. In addition, the maximum value of a real system is not a very robust indicator, since different runs typically produce varying maximum values.

Additionally, we also use the $r_{\mathrm{t97.5}}$ value, which is the smallest value larger than 97.5% of the result values.

Another important indicator is the average response time $r_{\mathrm{av}} = \frac{1}{N}\sum_j r_j$, or the median response time r_{med}.

2.2 Jitter

Jitter here is defined as the variation of periodic calculations. Let t_i be the points in time when a certain periodic calculation is performed, then the jitter is defined as $\mathbf{j} = (t_1 - t_0, t_2 - t_1, \ldots, t_M - t_{M-1})$. For many embedded systems the task jitter is very important. For example, the quality of a control loop may depend on the jitter of the calculation.

Typically, the jitter is condensed into $j_{\max} = \max(|\mathbf{j}|)$.

In our evaluation we use the task start jitter (start-to-start or s2s) and the task end jitter (end-to-end or e2e), namely $j_{x2x,\mathrm{av}}$ and $j_{x2x,\max}$, where $x2x$ is either $s2s$ or $e2e$.

2.3 Interruptions / Context Switches

Another performance indicator I is the number of times a task is interrupted by another task. This is relevant since

each context switch generates an overhead in the operating system, thus reducing available computing time for the applications. Again, the average number of interruptions I_{av} as well as the maximum number of interruptions $I_{\max}$ are of interest.

2.4 Normalization of Performance Indicators

In order to have comparable task indicators, it is useful to normalize them respectively. All indicators based on response time (namely r_{av}, $r_{\max}$, $r_{\mathrm{t97.5}}$) are normalized by the respective task deadline, giving the *relative* response time. A response time of 1 then means that the task finishes exactly at the deadline (i.e. the lateness is zero).

Jitter of periodic tasks ($j_{\max}$, j_{av}) is best normalized by the period, which yields the *relative* jitter. A jitter $j_{s2s,\max} = 1$ means that the delay between two successive starts of the task may be as large as the period. If necessary, all other performance indicators can be normalized as required.

In the following sections, only normalized performance indicators are used for response time and jitter.

3 Evaluation of Complete Task Sets

When evaluating a system, the resulting performance indicator is valid only for one task, e.g. the maximum response time of this task. In order to evaluate the timing behavior of the complete system, the results of all tasks must be taken into account. This section concentrates on introducing methods for condensing the data of a complete set of tasks into a scalar value and defines mechanisms for comparing different result sets.

3.1 Definitions

Let the task set

$$\mathbf{T} = (T_1, T_2, \ldots, T_N) \tag{2}$$

be a vector of N tasks T_i in a system, where N is known and typically fixed for a embedded real-time system. When the system is activated, all tasks are executed according to a given scenario (either in some simulation environment, a HIL test bench or the real system environment), while the selected indicators (e.g. those defined in the previous section) are recorded for each task T_i individually. This delivers a result vector for each performance indicator π

$$\mathbf{x}^{\pi} = (x_1^{\pi}, x_2^{\pi}, \ldots, x_N^{\pi}) \tag{3}$$

with π being one of the selected performance indicators, e.g. absolute or average response time, number of interruptions, etc.

3.2 Norm Definitions

In order to define an absolute performance measure, we use different norms on result sets, namely

- the euclidian norm defined by

$$||\mathbf{x}||_e = \sqrt{x_1^2 + x_2^2 + \ldots + x_N^2} \tag{4}$$

- the 1-norm (taxicab or manhattan norm) defined by

$$||\mathbf{x}||_1 = \frac{1}{N} \sum_i |x_i| \tag{5}$$

- the maximum norm

$$||\mathbf{x}||_\infty = \max |x_i| \tag{6}$$

Note that (5) is normalized to the number of tasks in the system, which allows comparison of different systems.

Additionally, the norms can be extended by weighting each task in the norm, leading to a weighted norm, e.g.

$$||\mathbf{x}||_{1,weighted} = \frac{1}{N \cdot W_{sum}} \sum_i |\gamma_i x_i| \tag{7}$$

where γ_i is the weight of the i-th task and W_{sum} is a suitable normalization factor. The weight could express the importance of a task, or the criticality of the task for the whole system. For example, a safetly critical task might be weighted higher than a non-safety relevant task.

3.3 Metric Definition

Based on the above norms, one can easily define corresponding metrics

$$D(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}|| \tag{8}$$

e.g. D_1 and $D_{\max}$, which will be used in the following sections to calculate the distance between two result sets.

3.4 Definition of Tendency

In order to answer the question "which of the two schedulers is better", one could use a suitable norm and say that result set $\mathbf{x}$ is better than result set $\mathbf{y}$ if and only if $||\mathbf{x}|| < ||\mathbf{y}||$. However, this only gives a true / false statement ("is better" or "is worse"). In order to have a more robust indicator for "betterness" we introduced the tendency

$$T'(\mathbf{x}, \mathbf{y}) = \frac{\sum\limits_{x_i - y_i > 0} |x_i - y_i|}{\sum\limits_i |x_i - y_i|} \tag{9}$$

which is scaled to $[-1, 1]$ using

$$T(\mathbf{x}, \mathbf{y}) = 2T'(\mathbf{x}, \mathbf{y}) - 1 \tag{10}$$

An different tendency can be derived by just counting the tasks for which x_i is larger than y_i

$$\tilde{T}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_i \Theta(x_i - y_i) \tag{11}$$

where $\Theta(x)$ denotes the Heaviside function

$$\Theta(x) := \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{12}$$

3.5 Feasibility Test using the Maximum Norm

A schedule is said to be *feasible* if all tasks always complete within their deadlines (see [1]), which is equivalent to saying that a schedule is *feasible* if and only if for all possible result sets $\mathbf{x}$

$$F(\mathbf{x}) = ||\mathbf{x}^{\mathrm{maxRT}}||_\infty < 1 \tag{13}$$

Using the above indicator and norm, one can not only express the classical feasibility test (true / false), but also a *degree of feasibility*.

For example, this could be useful when dealing with validation of timing behavior in product development. One might determine F for each release cycle, e.g. $F = 0.9$ for one product release, and $F = 0.95$ for the next one. The second value clearly indicates a heavier system load and might reduce confidence, thus triggering additional activities in the development process.

3.6 Visualization of Result Sets

For easier comprehension and comparison several useful visualizations are possible. In Figure 1 a polar comparison plot is shown combining the values of the metric (length of the "needle") and the tendency (angle). Here, two different results sets $X = \mathbf{x}^{\pi_1}, \mathbf{x}^{\pi_2}, \ldots$ and $Y = \mathbf{y}^{\pi_1}, \mathbf{y}^{\pi_2}, \ldots$ that are acquired by performing two measurements, e.g. two simulations with different schedulers, are compared with respect to the performance index π by calculation of an angle ϕ^π using the tendency

$$\phi^\pi(\mathbf{x}^\pi, \mathbf{y}^\pi) = \arcsin(T(\mathbf{x}^\pi, \mathbf{y}^\pi)) \tag{14}$$

and a length r^π using the metric

$$r^\pi(\mathbf{x}^\pi, \mathbf{y}^\pi) = ||\mathbf{x}^\pi - \mathbf{y}^\pi|| \tag{15}$$

Doing this for a set of performance indices $\{\pi_1, \pi_2, \ldots\}$ yields a set of (ϕ_i, r_i) vectors, which can be visualized in a polar coordinate system. If the vector points to the left, i.e. $\phi_i < 0$ respective $T(\mathbf{x}, \mathbf{y}) < 0$, then system X is better with respect to performance index π_i, otherwise Y is better. If the vector does not point to left or right, i.e. $\phi_i \approx 0$ respective $T(\mathbf{x}, \mathbf{y}) \approx 0$, then both systems are rather equal. The length of the vector r_i defines the difference with respect to π_i. Small length means small difference, while bigger length means a larger difference; however, this also depends strongly on the chosen normalization of the performance indicator. One strong advantage of this representation is that it is possible to visualize two systems with respect to many performance indicators in one diagram. If all π_i vectors point to the same direction, a clear tendency is show. If, on the other hand, some vectors point to the left, while other point to the right, no tendency is displayed. In the example in Figure 1 two performance indicators are shown.

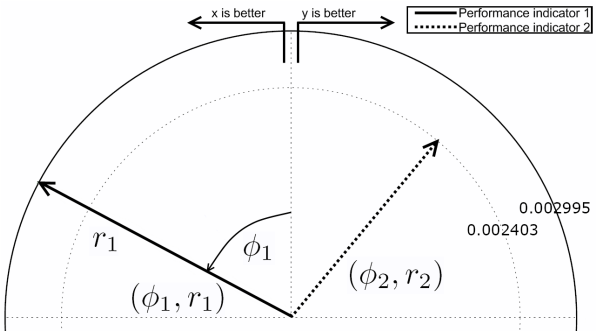

Figure 1. Example for polar comparison of two result sets X and Y using π_1 and π_2

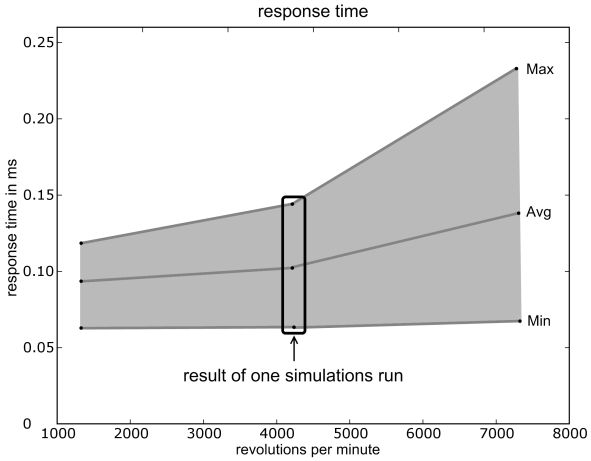

Figure 2. Example for displaying results (namely $||\boldsymbol{r}_{\mathbf{min}}||_1$, $||\boldsymbol{r}_{\mathbf{av}}||_1$ and $||\boldsymbol{r}_{\mathbf{max}}||_1$) of experiments with a continuously varied parameter

If a sequence of results with a continuously varied parameter (e.g. system load, controller clock frequency) needs to be visualized, a different kind of diagram can be used like shown in Figure 2. In this example the engine speed is raised (increasing the calculation repetition rate for some calculations) thus increasing the load of the system. There are three sample points displayed. At each sample point the three performance indicators minimum, average and maximum response time are determined. All three values are displayed in the figure, and a band around the mean value ranging from the minimum to the maximum is drawn. (The linear interpolation between the sample points is just performed for optical reasons.) This diagram can be extended to different experimental setups using different colors for each setup (however, this requires a color print-out).

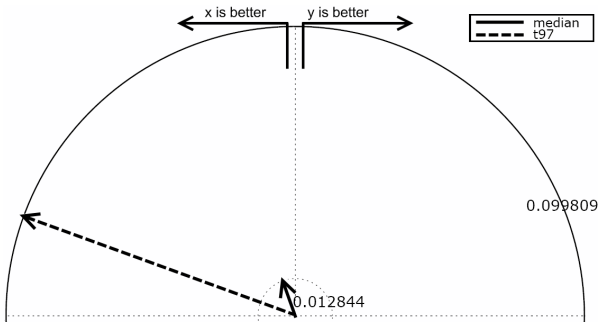

Figure 3. Polar comparison of the response time between deadline monotonic (left) vs. artificially bad adjusted priority pattern (right)

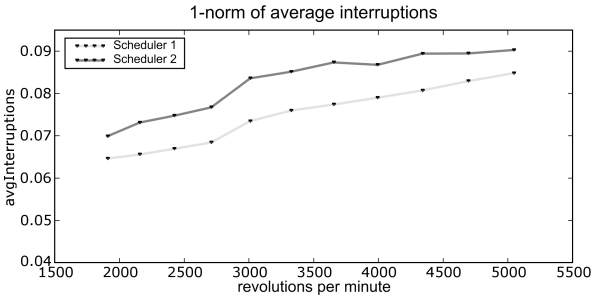

Figure 4. Number of context switches $||I_{\mathbf{av}}(\mathbf{rpm})||_1$ for two different schedulers

4 Example: Comparing Two Priority Sets

In order to demonstrate the application of the concepts described in the previous section a sample use case has been selected. The underlying system is a multi task single core automotive application with a scheduling based on priorities that are fixed at design time. The challenge now is to compare the effects of two different sets of priority assignments on the performance of the system.

4.1 Simulation method

In the following, all result data has been obtained using a simulation approach. In order to make the simulation results closely resemble the real application, runtime measurements are performed by executing the software on the target system. Then, using the acquired timing data we construct a simulation model which is then executed by the simulation tool chronSim which allows easily to change scheduling parameters or other aspects of the system (e.g. task priorities). Each simulation run delivers a trace that contains all the information about the timing behavior of the tasks in the simulation. From the trace the performance indicators described in Section 2 can be extracted. Finally, the resulting data can be be treated with the concepts of section 3.1 in order to get the desired comparison on system level.

For more details on how the model is set up and how the simulation can be performed, please see [5] and the web page of the tool vendor [3].

4.2 Results of simulation

As a first example application of the methodology we use two simulation runs of the same task set consisting of 20 tasks. The tasks have very different execution times and recurrence definitions. The two simulation runs differ in the used set of priorities assigned to the tasks. In the first run a priority pattern based on deadline monotonic scheduling is used and in the second run the priorities are distributed according to a three level priority scheme based on a functional grouping mainly neglecting the deadline or arrival rate of the calculations. The expectation with this very basic setup is, of course, that the performance indicators clearly show that the deadline monotonic approach is better suited to keep the deadlines (which is, after all, a major goal in a real-time system).

Figure 3 shows the result for two performance indicators, the median and the t97.5 value of the relative response time. In the shown polar plot both ”needles” are pointing to the left side, indicating as expected that the deadline monotonic priority assignment shows a better response time behavior. Please note that especially the t97.5 value, being more sensitive to the maximum values than the median, shows a strong difference.

The second example shows the number of interruptions for two different scheduler concepts using the same task set as described above. In order to check the system under dif-

ferent operation conditions the engine rotation speed was varied and sample points for 11 values have been taken. In Figure 4 the results are shown, indicating clearly that usage of scheduler 2 results in a higher number of context switches regardless of the operating state of the system.

Finally, Figure 5 shows the results for another two scheduler setups regarding the response time behavior. In this experiment 23 sample points have been taken at different operating conditions. Obviously, the scheduler 1 shows an remarkable performance improvement regarding the average and maximum response time.

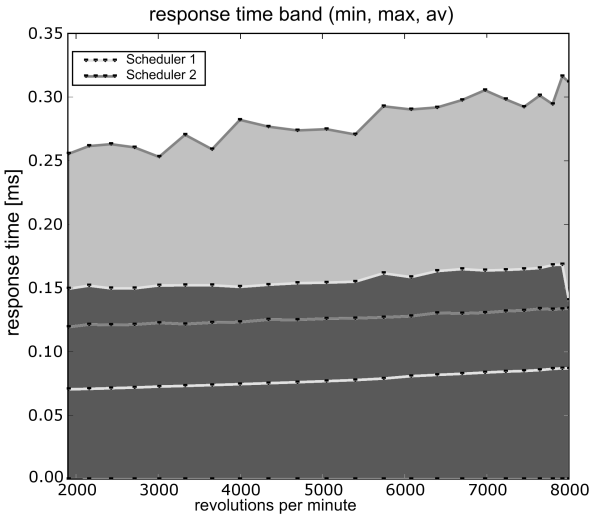

Figure 5. Response time bands ($||r_{\mathbf{min}}||_1$, $||r_{\mathbf{av}}||_1$ and $||r_{\mathbf{max}}||_1$) for two different schedulers

5. Summary

State of the art scheduling analysis approaches deliver a true / false statement on the schedulability of a given task set. Despite this, designers of systems in reality need more fine grained information about their system. For this often timing measurements (e.g. using instrumented software or special measurement hardware) and simulation techniques are used. In this work it was shown how the data obtained with such techniques can be condensed into key performance indicators for the investigated system allowing the comparison of different concepts in a very detailed way. Given the fact that the indicators can be tuned to the specific needs of the investigated application this is an valuable tool to design or improve the application architecture by selecting the most efficient scheduling algorithm and setting up the optimal parameters for it.

The presented approach offers a powerful and expandable framework for visualization and evaluation of the systems timing behavior. The main challenge hereby lies in the selection of relevant performance indicators and norms / metrics, which may differ for different systems (or classes of systems). However, once the key performance indicators are chosen, evaluation and control of nearly all aspects of the systems timing behavior is possible, in the system design phase as well as in validation of the system.

6. Acknowledgements

We thank Denis Claraz and Pierre Fourty (Continental Automotive SAS, Toulouse, France) for numerous discussions having inspired the work on the results presented in this paper.

References

[1] G. C. Buttazzo. *Hard Real-Time Computing Systems*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.

[2] C. Diederichs, U. Margull, F. Slomka, and G. Wirrer. An application-based edf scheduler for osek/vdx. In *Date 2008*, 2008.

[3] INCHRON GmbH. chronSim. http://www.inchron.de/.

[4] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association of Computing Machinery*, 20(1), January 1973.

[5] U. Margull, P. Fourty, and G. Wirrer. Simulation methods for evaluation of resource critical real-time systems. To be published.

[6] R. Münzenberger, M. Dörfel, U. Margull, and G. Wirrer. Entwurf echtzeitfähiger Steuergerätesoftware in FlexRay-Netzwerken. In *KFZ-Entwicklerform Design&Elektronik*, 2007.