Proc. of the 3rd Int. Real-Time Scheduling Open Problems Seminar (RTSOPS 2012), Pisa, Italy, July 10, 2012.

Exploiting online WCET estimates

Sverre Hendseth[1]
Department of Engineering Cybernetics, NTNU, Trondheim Norway
Email: Sverre.Hendseth@itk.ntnu.no

Giorgio Buttazzo
Scuola Superiore Sant'Anna, Pisa, Italy
Email: Giorgio.Buttazzo@sssup.it

***Abstract*—This problem formulation builds on the acknowledgment that, for some systems, offline WCET analysis is too conservative to be useful. For many classes of programs, better bounds on the execution time can be found effectively online by utilizing information like input data or program state. Both the systematic extension of the class of algorithms for which such bounds can be found, and the general techniques for the exploitation of the bounds are largely unexplored problems.**

I. Introduction

Having upper bounds on the execution time for the jobs in a real-time system is critically important in that guaranteeing meeting any deadline is impossible without it. The system is then, classically, built by over-allocating resources so that even in the worst case, all timing requirements are met. However, in many cases this approach is based on very pessimistic assumptions and leads to a large resource waste. Two trends have reinforced this picture in the last decades: First, the embedded systems software has become more complex both in terms of size and complexity, leading to challenges to find tight upper bounds of the execution time. Second, with the increase of battery-operated and wireless systems the overallocation of resources becomes too expensive.

This paper takes the position that offline worst-case execution time analysis is not efficient as a dimensioning factor for building and scheduling real-time systems, and we suggest exploring the better bounds that can be found online when input data and program state are available and can be used in combination with the results from the offline analysis.

The idea of utilizing online job data for improving the scheduler's decisions is not entirely novel: Mok and Chen [1] accept the potential importance for the scheduler to use more information on the execution time of a job than the worst case, but assume this data known. Choi et al. [2], take the opposite approach of letting the application itself control CPU voltage and frequency based on job parameters, but without notifying the scheduler.

In scenario-based design the connection is made between between the data available to the program and the scheduler's decision-making. Here, the possible executions of the application are partitioned, offline, into scenarios, and a prediction algorithm running online in the program's address space provides the scheduler with the information on which scenario is occurring [3].

[1]Work done at Scuola Superiore Sant'Anna, Italy

Audsley et al. [4] uses *Gain Points* in the running code to notify the scheduler of updated WCET values when the outcome of major control flow decisions has been calculated by the program. This can be e.g. just after the test of an if-statement has been performed or when a loop bound has been calculated. The application envisioned here is the execution of optional components to hard real-time programs with online guarantees.

We propose that using both offline and online information for job scheduling deserves a broader and more systematic investigation.

II. Model

A number of jobs τ_i are to be scheduled according to their deadlines D_i in presence of significant dynamicity. Each job is described by their worst-case execution time, here denoted $WCET^{off}$ and the *online WCET*, $WCET^{on}$, which will be calculated at the job's release time R_i. C_i denotes the real computation time which is unknown until the job has finished.

An estimation algorithm is assumed, which measures relevant metrics available online, and converts them to the $WCET^{on}$ by a transformation prepared off-line. The relevant input data of the job is assumed either available at the release time or inexpensive to acquire by the algorithm.

As a job becomes ready, the scheduler immediately runs the corresponding estimation algorithm and then utilizes the gained information in its decision making. The estimation algorithm is assumed to have an execution cost that, while not being insignificant, will be very small compared to that of the job itself.

The existence of effective estimation algorithms is not obvious: In the easiest case, information like the type of the frame to be processed, or the length of the list to be traversed, is directly available at job release time. On the other end of the scale we have programs for which termination itself is an open question, yielding the nonexistence of effective estimation algorithms. Between these extreme cases we have any number of program categories, potentially requiring different techniques for estimation and inviting different bound qualities.

In a real-time setting we know that we never deploy programs that do not terminate in bounded time. This can be taken to indicate that we will not be struggling with the hardest categories of programs.

$Crit$	Job	$WCET^{off}$	$WCET^{on}$	C	R	D
Hard	τ_1	12	9	8	0	16
Soft	τ_2	8	6	5	0	10

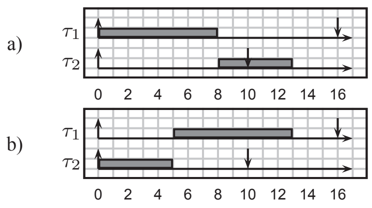

Fig. 1. Scheduling can be improved for this mixed criticality job set if $WCET^{on}$ is used, enabeling the schedule in b). Using only $WCET^{off}$ forces the schedule in a) where τ_2 misses its deadline.

III. Potential utilization areas of online WCET data.

The following points envision some possible application areas for the online WCET:

- Some schedulability or timeliness properties of a system could be refined using the $WCET^{on}$.
- Algorithms for reclaiming the CPU capacity allocated to a job but not used, might be improved by getting better information on the real job load at the job release time. Similarly, strategies for overload handling, jitter reduction, power optimization etc. could potentially be improved.
- Figure 1 shows a simple mixed criticality job set where access to the tighter $WCET^{on}$ improves schedulability.
- In a quality of service perspective, programs might offer more modes of operation. Comparing the online WCET of each mode with the available CPU for the next period might be useful for selecting the mode.
- Algorithms for dynamic deployment of jobs onto processors and for load balancing can be improved using the online execution time estimates.

IV. Principles for online estimation of WCET.

The following points suggest how the estimation challenge can be approached for increasing algorithm complexity:

- The trivial case is when the estimation metric is directly available. This can be the “message size”, “frame type”, “length of the linked list”, etc. The estimation algorithm would just look up the information and translate it into the online WCET.
- A slightly worse class of programs would be when a subset of the calculations must be carried out by the estimation algorithm - like calculating the bounds of the significant loops or the outcomes of the major conditionals. Figure 2 gives an example of a type of program where this would be successful. A framework for automatically generating such estimation algorithms is under development.

```
for (m=1;m<mmax;m+=2) {
  for (i=m;i<=n;i+=istep) {
    j=i+mmax;
    tempr=wr*data[j]-wi*data[j+1];
    tempi=wr*data[j+1]+wi*data[j];
    data[j]=data[i]-tempr;
    data[j+1]=data[i+1]-tempi;
    data[i] += tempr;
    data[i+1] += tempi;
  }
  wr=(wtemp=wr)*wpr-wi*wpi+wr;
  wi=wi*wpr+wtemp*wpi+wi;
}
```

(a)

```
g_estimate += 1;
for(m=1;m < mmax;m+=2){
  g_estimate += 13;
  for(i=m;i <= n;i+=istep){
    g_estimate += 32;
  }
}
```

(b)

Fig. 2. a) shows the two inner loops of the Numerical Recipes FFT implementation. A corresponding algorithm for estimating its the execution time has been automatically generated in b) (here plainly counting the applications of C binary operators). Further optimizations are apparent.

- Even more challenging could be a program which was heavy on control-decisions like a sorting or Huffman decoding program. However, on one hand many of these algorithms are well analyzed already so that suitable bounds may be found in the literature. On the other hand, good and inexpensive metrics might still be found by accepting uncertainties on the lower levels of granularity.

V. Open Problems

The two proposed open problems are:

- The systematic extension of the class of programs for which effective estimation algorithms exists which yields tight execution time bounds by utilizing data available online, like input data or program state.
- The corresponding utilization of these online execution time bounds in making systems with better timeliness properties.

References

[1] A.K. Mok and D. Chen, “A multiframe model for real-time tasks,” *Software Engineering, IEEE Transactions on*, vol. 23, no. 10, pp. 635 –645, oct 1997.
[2] Kihwan Choi, Karthik Dantu, Wei-Chung Cheng, and Massoud Pedram, “Frame-based dynamic voltage and frequency scaling for a mpeg decoder,” in *IN ICCAD 2000*, 2002, pp. 732–737.
[3] S. V. Gheorghita, M. Palkovic, J. Hamers, A. Vandecappelle, S. Mamagkakis, T. Basten, L. Eeckhout, H. Corporaal, F. Catthoor, F. Vandeputte, and K. De Bosschere, “System scenario based design of dynamic embedded systems,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 1, January 2009.
[4] N.C. Audsley, R.I. Davis, and A. Burns, “Mechanisms for enhancing the flexibility and utility of hard real-time systems,” in *Real-Time Systems Symposium, 1994., Proceedings.*, dec 1994, pp. 12 –21.