Software Simulation as a Means to Estimate Feasibility of Real-Time Software Applications

Sergey Baranov and Victor Nikiforov

Abstract. An approach to study the behavior of real-time multi-task applications is described based on a software simulation tool written in the Forth programming language.

Introduction

Software applications for real-time systems (RTS) are usually built as collections of tasks which are sequential programs closed w.r.t. to control flow. During application run its tasks share common system resources: the executive ones (processors and processor cores of multi-core processors) and informational ones – global data arrays, interface registers of peripheral devices, elements of human-machine interface, etc. Access to executive resources is governed by the scheduling mode in use, while for access to informational resources certain access protocols are used.

A key requirement to an RTS software application is guaranteed on-time execution of each of its tasks (called "application feasibility") in all correct situations for the application to run. Application feasibility may be checked either through an analytical estimation of the response time for each application task, or through simulation of the application run with an appropriate software tool. Verifying the correctness of a particular situation is another important task beyond the scope of this presentation.

For an RTS designed to run on a conventional single-core processor exact analytical estimations of their feasibility exist; however, applying these methods to RTS on multi-core processors provides substantially inexact results which are too pessimistic, and no exact analytical estimates are known by now.

In this presentation a software simulation tool is described which ensures a more exact estimation of application feasibility for a RTS application on a multi-core platform under various combinations of scheduling modes and protocols of access to shared resources than the known analytical methods.

1. The Problem

A characteristic feature of an RTS is the requirement for on-time execution, usually expressed as a requirement that for each task τ_i the longevity $r(\tau_i^j)$ of any of its jobs τ_i^j shall not exceed some pre-defined deadline value D_i: $\forall i, j(r(\tau_i^j) \leq D_i)$. With the notion of the task response time $R_i = max\{r(\tau_i^1), r(\tau_i^2), ...\}$ this may be reformulated as $\forall i(R_i \leq D_i)$ with any allowable scenario of system events and is often interpreted as the property of feasibility of the given multi-task application. To check application feasibility, various structural models of its tasks are built and analyzed to provide reliable estimates for the response times of the application tasks, taking into account all impacting factors.

Software simulation is an acknowledged method to check feasibility of real-time multi-task applications. This paper describes an experience of constructing such simulator in Forth with the VFX Forth for Windows [1] as a development platform. A freeware option for the platform is gForth [2]. Forth was selected as the implementations language due to the flexibility it provides for implementing programming solutions. The simulator employs a simple model of a multi-task application under study which may use several scheduling modes with various task priorities for allocation of the processor computational resource and several access protocols to access shared informational resources. The simulator helps to study multi-task application behavior and check whether a given combination of the scheduling mode and access protocol guarantees application feasibility under the given processor performance and system event scenarios. It may also identify the minimal processor performance which still ensures application feasibility under the given conditions.

By now, the nomenclature of scheduling modes and access protocols implemented in the simulator consists of two classical scheduling modes – RM (rate monotonic) and EDF (earliest deadline first) – and three access protocols – NI (no inheritance), BI (basic inheritance), and PI (priority inheritance). However, it may be further extended to simulate systems with other scheduling modes on a multi-processor and/or multi-core platform and other protocols of access to shared informational resources [3].

2. Solution

The computer program RTMT (Real-Time Multi-Tasking) [4] was designed to study various combinations of scheduling modes and protocols access to shared informational resources in real-time multi-task applications for a multi-core platform [4]. A particular interest is how these combinations impact application density [5]. Obtained characteristics determine an optimal selection for application parameter realization which ensure application feasibility under all possible scenarios of their correct interaction with the external environment.

Simulation is based on components of four kinds: *resources, tasks, jobs*, and *events*. Resources and tasks are entities of the application under study; jobs and events are entities created and operated on by the simulator. Resources and tasks are also represented within the simulator with respective entities. The overall simulation process is governed by the structure *Eventlist* which contains events sorted w.r.t. their timestamps, and the structure *Joblist* which contains ready-to-run jobs sorted w.r.t. their current priorities. The overall simulation structure is represented in Figure 1.

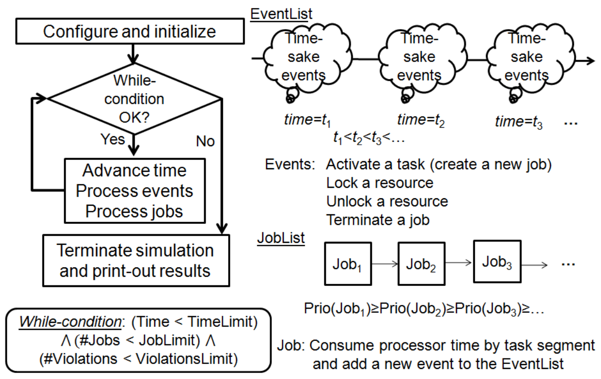
Configure and initialize
While-condition OK?
Yes
No
Advance time
Process events
Process jobs
Terminate simulation and print-out results
While-condition: (Time < TimeLimit) ∧ (#Jobs < JobLimit) ∧ (#Violations < ViolationsLimit)
EventList
Time-sake events
time=t1
time=t2
time=t3 …
t1<t2<t3<…
Events: Activate a task (create a new job)
Lock a resource
Unlock a resource
Terminate a job
JobList
Job1
Job2
Job3
…
Prio(Job1)≥Prio(Job2)≥Prio(Job3)≥…
Job: Consume processor time by task segment and add a new event to the EventList

FIGURE 1. The overall structure of the RTMT simulator

The simulation internal loop considers considers time gaps between two successive groups of same-time events in *Eventlist* one after another, starting from *time=0*. The gap size is determined through the minimum of the upper bound of the time yet to be consumed and the time-stamp of the next same-time event group.

For all current jobs at the gap start their counts of consumed time are increased by the gap size, while the counts of yet-to-be-consumed time are respectively decreased. and if this count reaches zero, then a new event is added to *Eventlist* – to terminate this job at the respective time moment. Then all events from the same-time-event group are processed which may change the contents of *Joblist*. Upon completion of processing all these events, the updated contents of *Jpblist* is considered and RTMT transits to the next same-time event group. Thus successive simulation iterations advance the system time and terminate when the specified time limit is exhausted ($Time < TimeLimit$) or when the overall number of created jobs reaches the specified limit ($\#Jobs < JobLimi$) or the total number of registered response time violations is reached ($\#Violations < ViolationsLimit$), whatever occurs earlier. The described internal loop is embedded into the main loop controlled by the parameter of application hardness H for which its bounds and stride are specified. It turned out that it's more convenient to specifiy the value $1/H$ from which H is recalculated.

Conclusion

The simulator was written in Forth with VFX Forth for Windows, version 4.70, provided to the author at the courtesy of MPE, and is just 985 lines of code under the respective coding standards. It uses only fixed-point arithmetic and works remarkably fast on a PC. To avoid memory overflow, the simulator uses its own simple subsystem for memory allocation and reuse for chained list elements, jobs and events. Further work will be focused on improving the user interface, extending the nomenclature of scheduling modes and access protocols of this simulator, and transition to simulation of multi-core and multiprocessor platforms, as well as running more experiments with models of real-time multi-task applications.

References

[1] VFX Forth for Windows. User manual. Manual revision 4.70, 19 August 2014. MicroProcessor Engineering Limited, Southampton, 2014.

[2] gForth. Free Software Foundation, Inc., (2014), available at `https://www.gnu.org/software/gforth/`.

[3] B. Andersson, S. Baruah, and J. Jonsson, *Static-Priority Scheduling on Multiprocessors.* Proc. of 22nd IEEE Real-Time Systems Symposium. London (2001), pp.193-202.

[4] Baranov S.N. *The Program RTMT for Simulation of Multi-Task Application Run.* Certificate of official registration of a computer program No.201661395, 16 March 2016 (RU), (in Russian).

[5] Baranov, S., Nikiforov, V. *Density of Multi-Task Real-Time Applications.* Proc. Association FRUCT-17. Yaroslavl, (2015), pp.9-15.

Sergey Baranov
Lab.of information and computing systems and software engineering
SPIIRAS
St. Petersburg, Russia
e-mail: `SNBaranov@iias.spb.su`

Victor Nikiforov
Lab.of information and computing systems and software engineering
SPIIRAS
St. Petersburg, Russia
e-mail: `nik@iias.spb.su`