Implementation Experience with Ada 2005
Session Report

Chair: Alan Burns
Rapporteur: Andy Wellings

Session Goals

The goals of this session were to

- Discuss implementation experience with the new real-time features
- Review the support provided by the new real-time features
- Review features proposed but omitted from Ada 2005

Implementing the New Real-time Features

Mario Aldea Rivas first gave an overview of the approach their paper had taken on implementing the new Ada 2005 real-time services in MaRTE OS and GNAT. The most important of these new services are:

- timing events
- execution-time clocks and timers
- dynamic priorities for protected objects
- immediate priority changes
- group execution-time budgets
- new scheduling and task dispatching mechanisms

Of these, the first five had already been implemented and would be released by AdaCore in the near future. The rest would be done during the summer of 2007.

For timing events, Mario indicated that it was not possible to implement timing event straight from the clock interrupt handler as there was no mechanisms provided by POSIX to do so. He indicated that there were essentially two approaches: one where run-time threads are introduced for each timing event, the other where the OS is changed. He said they had implemented both approaches and that by changing the OS, there was a significant performance gain.

For execution-time clocks and handlers, Mario reported that the implementation was much simpler as both the POSIX and Ada 2005 standards took a similar stance. In particular:

- Neither of the standards define which task/thread is charged with the overheads of interrupt handlers and run-time services on behalf of the system
- Both standards state that the execution time is set to zero at the creation of the task /thread
- Ada 2005 says the time spent during task activation must be charged to the task execution time clock - this happens in GNAT since activation is executed by the thread used to implement the Ada task.

As a consequence, no modifications to the compiler or to the run-time system have been necessary. Mario reported that execution time accounting introduces a small overhead to context switch time (less than 5

Execution-timers had been built on top of the timers and had caused no significant implementation problems. Group execution time accounting, however, required significant modifications to the OS as POSIX did not support thread groups. The facility added an extra 9

Juan Zamorano gave a presentation on their implementation of the same facilities in the Open Ravenscar Kernel on a bare board Leon (based on the SPARC V8 architecture). Juan indicated that the scarce hardware support for timers on that board meant that significant software support was required. This had added a 50

Following the two presentations it was noted that the Workshop was not aware of other projects implementing 2005 real-time facilities.

Discussions on the New Features

The main discussions following the presentations focused on the overheads and inaccuracies of the CPU accounting model.

The following issues were raised:

1. Context switch time – Mario reported that there was no leakage of CPU time during context switches.
2. System and Application interrupts (e.g. clocks) – whilst Ada allowed interrupt handling to be charged

to the executing tasks there was concerns that this was a significant inaccuracy.

3. Timing events code – it was again noted that the code executed by timing event handlers was application level code and therefore was not fixed. This would again be charged inappropriately to the running task. However, it was also pointed out that as the code was a protected procedure, the time was at least bounded per handler.

4. Proxy model of Protected Objects – concern was expressed that the proxy model of implementation for protected objects could result in a significant inaccuracy as one task could execute a significant amount of code on behalf of another.

There was a long discussion of whether the CPU accounting model was useable given the inherent inaccuracies. Various points were noted:

- The facilities could be used with a measurement-based approach. Execution time could be measured during system testing and this figure used at run-time. However, this approach is fragile. Any small change to the application code would mean that the system-level timing measurements would have to be redone.

- For hard real-time systems, it was noted that there had to be an associated analysis model. The worst-case overheads could then be added to the execution time of each task. However, this approach could be very pessimistic as each task would be charged the worst case overhead.

- It was also pointed out that the greatest error was on the value of the worst-case execution time itself and that adding a small error was at the noise level.

Another point raised was that the impact of handling low priority interrupts on high priority task could be significant.

The workshop concluded that there is a need to investigate the overheads and the extra cost of trying to do better accounting. Also the overhead of a better model of prioritized interrupt handling should be investigated.

Application-level Scheduling

Michael Gonzalez Harbour gave an overview of the current status of application-defined scheduling work that had been reported at the last workshop. Although this had failed to get in to the standard, an implementation had been produced and would be released as an extension to GNAT. The hope was that people would use the facilities and that it might eventually become a de facto standard. The workshop reaffirmed its support for the need of such a facility in Ada.

Ravenscar

This session of the Workshop concluded with a discussion of the continuing experience with the Ravenscar profile. Juan Antonio de la Puente raised the issue of execution timers and group budgets. Although Ravenscar allows execution-time clocks, it prohibits timers and group budgets. He proposed that we should allow one timer per task. The motivation is to make sure a task does not consume more than its budget.

Whilst there was some support for this proposal, concern was expressed on how a Ravenscar program would respond to a timer expiring. There are not asynchronous interaction mechanisms in Ravenscar. Juan Antonio indicated that this was similar to the way task termination was handled. If a task terminated in Ravenscar (which it should not), the event is brought to the attention of the program and then it is implementation-defined what mechanisms the programmer can use.

It was pointed out that a monitor task could always read the execution times of other tasks and discover the overrun. However, there would clearly be a delay in doing this. There was no consensus position reached.

The Workshop felt that adding Group budgets opened up a new profile. This ought to be considered perhaps in a context where there are more than one Ravenscar applications (in effect, partitions) running on the same run-time.

Summary

The following summarised the positions taken by the Workshop during this session:

1. There is a need to investigate the overheads and the extra cost of trying to do better accounting and of the overheads of doing a better model of prioritized interrupt handling.

2. There is continued support for application-defined scheduling.

3. There is no consensus on adding CPU Timers into Ravenscar (i.e. it is an open issue that needs further investigation).

4. Group budgets and the coexistence of multiple Ravenscar applications on a single processing node needs further investigation.