Planificación basada en planes temporales en Ada Ravenscar

XXI Jornada Técnica de AdaSpain (JTAS 2024)

Sergio Sáez, Jorge Real y Alfons Crespo

Universidad Politécnica de Valencia

17 de Mayo de 2024

Índice

Introducción

Planificación en Ravenscar

- Planificación basada en **prioridades fijas**.
- No se puede cambiar la prioridad ... voluntariamente.
  - Objetos protegidos (PO) con herencia inmediata de prioridades.
  - El runtime cambia la prioridad de las tareas al usar PO.
- No hay Control Asíncrono de Tareas (ATC).
  - No se puede suspender a otra tarea de forma asíncrona.
- Soporte multiprocesador.
  - No se puede cambiar de CPU.
  - `Not_A_Specific_CPU = Main_CPU = CPU'First`
- ...

Planificación basada en planes temporales (I)

- Un plan temporal indica **cuándo** se hacen las *cosas* en el sistema:
  - controla el acceso a un recurso, p.ej. procesador.
- Un plan temporal (*major frame* o MAF) es una lista de ventanas/marcos de tiempo (*minor frames*) que se repite.
- El tamaño del plan temporal es fijo.

Predecibilidad Se sabe **cuándo** va a pasar **qué**.

Sobrecarga Baja sobrecarga en tiempo de ejecución (*p.ej. jitter*).

+ decisiones $\rightarrow$ **+** sobrecarga.

Estático No se adapta bien a eventos inesperados. Falta de reactividad.

Complejidad Construir el plan puede ser complejo y ocupar un gran espacio de almacenamiento en tiempo de ejecución

+ flexible $\rightarrow$ **+** complejo.

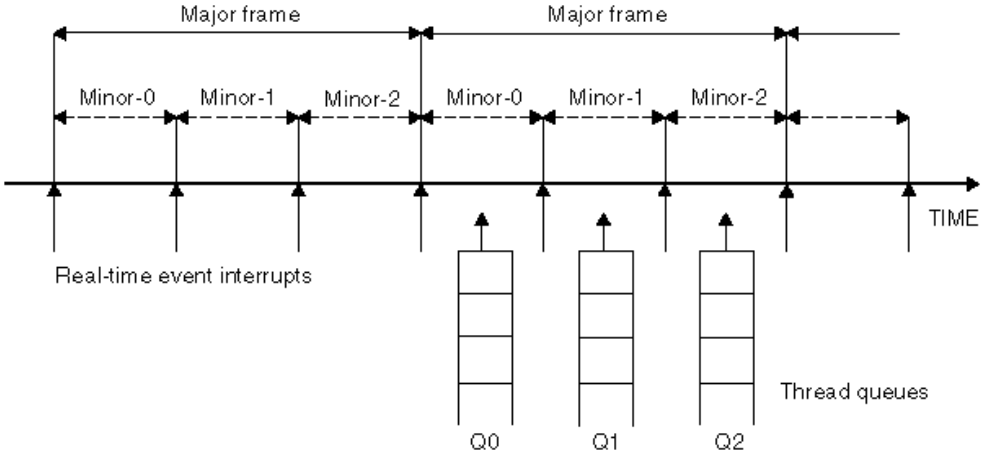
Planificación basada en planes temporales (II)

Se diferencian principalmente:

- Tamaño de las ventanas.
  - Todas iguales.
    ☞ El plan define sólo **qué se ejecuta** en cada ventana.
  - Tamaños diferentes.
    ☞ El plan define el **tamaño** de cada ventana y **qué se ejecuta** en ella.
- Ordenación de las tareas dentro de una ventana.
  - Una **única** tarea. El plan temporal es un plan estático para tareas (*jitter* reducido).
  - Una **lista ordenada**: FIFO, prioridades, ...
  - Un **planificador** de segundo nivel decide el orden de las tareas de esa ventana.
- Las tareas que se ejecutan pueden o no ser expulsadas.
  - **Sin expulsión:** las tareas **deben** caber en una ventana (¿desbordamientos?).
  - **Con expulsión:** más flexible → una tarea se puede suspender y continuar en otra ventana.
    ☞ Complica el acceso a los recursos compartidos.

Planificación basada en planes temporales (III)

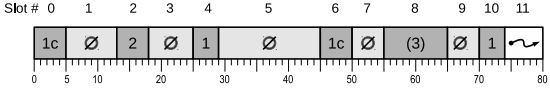
Ejemplo: Plan temporal con ventanas (*minor frames*) de igual tamaño.



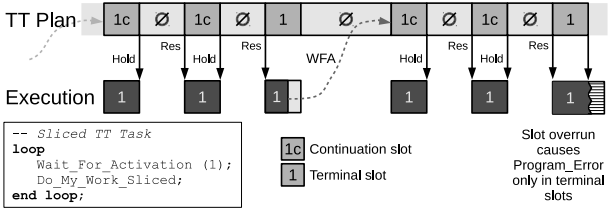
Trabajo previo

Time-Triggered Scheduling en Ravenscar (I)

- Plan estático con ventanas de distinto tamaño.
  - Se ha añadido soporte a ventanas de tamaño variable y cancelables.

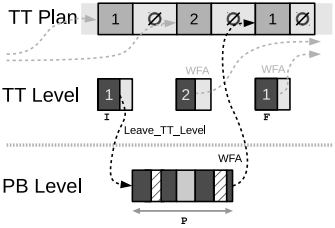


- Cada ventana ejecuta una única tarea o *trabajo*.
- Permite la suspensión de tareas:
  - una tarea se prolonga a lo largo de varias ventanas no consecutivas.



Time-Triggered Scheduling en Ravenscar (II)

- Las ventanas *vacias* y el tiempo no utilizado permite la ejecución de tareas basadas en prioridades fijas.
  - Dos niveles de planificación:
    1. plan estático;
    2. prioridades fijas en el tiempo libre.
  - Añade flexibilidad → una tarea puede comenzar en una ventana y continuar ejecutándose en el nivel de prioridades fijas.




```
-- Initial-Priority_Based-Final
loop
   Wait_For_Activation (1);
   Do_Initial_Part;
   Leave_TT_Level (1);
   Do_Priority_Based_Part;
   Wait_For_Activation (1);
   Do_Final_Part;
end loop;
```

Interference from TT tasks

Interference from PB tasks

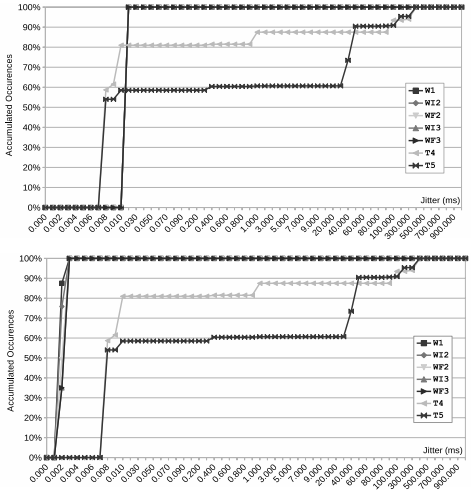
Time-Triggered Scheduling en Ravenscar (III)

Interfaz

```
--  Set new TT plan to start at the end of the next mode change slot
procedure Set_Plan
  (TTP : Time_Triggered_Plan_Access);

--  TT works use this procedure to wait for their next assigned slot
--  The When_Was_Released result informs caller of slot starting time
procedure Wait_For_Activation
  (Work_Id           : TT_Work_Id;
   When_Was_Released : out Ada.Real_Time.Time);

--  TT works use this procedure to inform that the critical part
--  of the current slot has been finished. It tranforms the current
--  slot in a continuation slot
procedure Continue_Sliced;

--  TT works use this procedure to inform the TT scheduler that
--   there is no more work to do at TT priority level
procedure Leave_TT_Level;
```

Time-Triggered Scheduling en Ravenscar (IV)

Resultados



Time-Triggered Scheduling en Ravenscar (V)

Extensiones al Ada RTS

- Control Asíncrono de Tareas (ATC) para uso interno del runtime.
  - Necesario para suspender tareas que siguen su ejecución en una ventana distinta.
  - `Suspend` y `Resume` con soporte para Objetos Protegidos.
    - La suspensión de una tarea queda diferida hasta que sale del último PO.
    - El `Suspend` sigue siendo limitado → Sólo se puede suspender a la tarea en ejecución.
- Modificación de los `Timing_Events` para que devolvieran el reloj utilizado para activarse.
  - El Ada RTS usa un temporizador periódico (STM32F4 → 1 ms).
  - Permite resincronizar/alinear el plan con el temporizador del sistema
    - Sin necesidad de leer el reloj en espacio de usuario y estimar el retardo.
    - Siempre que la granularidad del temporizador sea compatible.

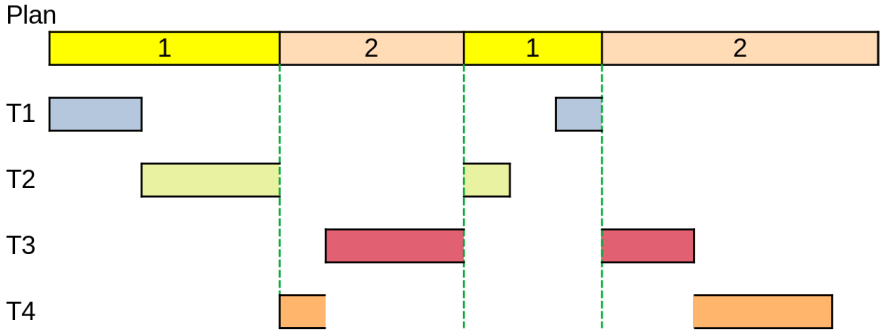
Propuesta

Soporte para particionado en el tiempo (I)

- Un planificador basado en prioridades fijas con expulsiones en cada ventana.
  - Las tareas en distintas ventanas están aisladas temporalmente
    $\rightarrow$ Partición temporal.
  - Las tareas de una ventana se pueden suspender y continuar en la siguiente activación de su ventana.
- El número máximo de *tipos* de ventana es fijo.
  - Definido en la implementación del Ada RTS.
  - En el soporte TTS previo era dependiente de la aplicación (paquete genérico).
- Independiente del tipo de plan utilizado:
  - Ventanas de igual/distinto tamaño o incluso variable.
  - Los *tipos* de ventanas se pueden repetir dentro del plan.
- Independiente del mecanismo utilizado para activar una ventana.
  - Eventos externos al sistema, `Timing_Events`, ...

Soporte para particionado en el tiempo (II)

Ejemplo



- Tareas T1, T2 → ventana 1
- Tareas T3, T4 → ventana 2

Soporte para particionado en el tiempo (III)

- Soporte multiprocesador
  - Cada tipo de ventana está asignada a una CPU fija.
  - Una ventana por omisión para cada CPU
    → una tarea *Idle* por ventana.
- Las tareas no pueden cambiar de CPU, pero pueden cambiar de *tipo* ventana.
  - Entre ventanas de la misma CPU.
  - → Soporte a cambios de modo.
  - → Las tareas inactivas en un modo pueden asignarse a una ventana que no se activa.
- Cada ventana tiene:
  - Una lista de tareas preparadas.
  - Una lista de tareas suspendidas → alarmas (`delay until`).
    - Las tareas se despiertan en su ventana
    - → no interfieren en las otras ventanas.
- `Timing_Events` globales entre ventanas (pero específicos de CPU).
  - Para permitir la implementación del plan temporal.

Soporte para particionado en el tiempo (IV)

Interfaz de alto nivel

```
package XAda.Dispatching.Time_Partitioning is

   subtype Dispatching_Window is System.Dispatching_Windows.Dispatching_Window;

   -- Creates a new dispatching window
   procedure Create_Dispatching_Window(DW : Dispatching_Window; CPU : System.Multiprocessors.CPU);

   -- Changes the active dispatching window in a given CPU.
   --   It only affects the CPU where the dispatching window is assigned to
   procedure Activate_Dispatching_Window(DW : Dispatching_Window);

   -- Assigns a task to a given dispatching window
   --   The CPU of the target dispatching window has to match CPU's task
   procedure Set_Dispatching_Window(DW : Dispatching_Window; T : Task_Id := Current_Task);

end XAda.Dispatching.Time_Partitioning;
```

Cuestiones abiertas

Uso de objetos protegidos

- Restringidos a cada ventana.
  - Etiquetar los PO para detectar violaciones del protocolo.
  - Requiere mecanismos de comunicación entre ventanas/particiones.
    - Puertos: colas y/o polling → la memoria es compartida.
    - Mecanismos *lock-free*.
- Objetidos protegidos *globales* entre ventanas.
  - El cambio de ventana podría quedar diferido → **desbordamientos**.
  - Diferenciar los PO locales de los globales.

Implementación

- Requiere modificaciones (más o menos) profundas del Ada RTS.
  - Complica la portabilidad.

Tareas pendientes

- Implementación del mecanismo de objetos protegidos seleccionado.
- Verificación extensiva ...
- Implementación en una plataforma multiprocesador ...
- Integración en la propuesta TTS, permitiendo un modelo híbrido.
  - Ventanas únicas asignadas a las tareas con restricciones de *jitter*.
  - Las ventanas reservadas para prioridades fijas podrían especificar la ventana de prioridades fijas activa.

Publicaciones relacionadas

J. Real, S. Sáez, and A. Crespo, "Combining Time-Triggered Plans with Priority Scheduled Task Sets," in *Reliable Software Technologies – Ada-Europe 2016* (M. Bertogna, L. M. Pinho, and E. Quiñones, eds.), vol. 9695 of *Lecture Notes in Computer Science*, (Cham), Springer International Publishing, 2016.

J. Real, S. Sáez, and A. Crespo, "Combined Scheduling of Time-Triggered and Priority-Based Task Sets in Ravenscar," in *Reliable Software Technologies – Ada-Europe 2018* (A. Casimiro and P. Ferreira, eds.), vol. 10873 of *Lecture Notes in Computer Science*, (Cham), Springer International Publishing, June 2018.

J. Real, S. Sáez, and A. Crespo, "A hierarchical architecture for time- and event-triggered real-time systems," *Journal of Systems Architecture*, vol. 101, December 2019.

J. Real and S. Sáez, "Time-Triggered Scheduling in Ravenscar (source code, latest version)." DOI: 10.5281/zenodo.1168722, 2020.

Preguntas

Planificación basada en planes temporales en Ada Ravenscar

XXI Jornada Técnica de AdaSpain (JTAS 2024)

Sergio Sáez, Jorge Real y Alfons Crespo

Universidad Politécnica de Valencia

17 de Mayo de 2024