Sistemas de control con distintos niveles de criticidad

A. Crespo, J. Simó, P. Balbastre, S. Peiró, M. Masmano
Instituto de Automática e Informática Industrial (AI2)
Universitat Politècnica de València

***Resumen*—Con el incremento de la capacidad de cómputo de los procesadores empotrados actuales y la necesidad de interconexión de las aplicaciones, se hace necesario disponer de soluciones tecnológicas que permitan desarrollar sistemas mas seguros y robustos. Al mismo tiempo, se requiere que los sistemas de control proporcionen interfaces gráficas para representar la evolución de los sistemas controlados y otras funcionalidades que pueden no ser de tiempo real. La necesidad de tener partes del sistema de control que son más críticas que otras es común a muchos tipos de sistemas de control. Una de las técnicas novedosas para tratar los sistemas de control con multiples niveles de criticidad es la basada en el particionamiento del sistema en aplicaciones aisladas.**

Los sistemas particionados permiten el desarrollo de varias aplicaciones independientes ejecutándose sobre uno o varios procesadores con distintas exigencias de criticidad. A fin de garantizar el nivel de criticidad exigido por cada aplicación, se requiere una capa de virtualización que aísle temporal y espacialmente las aplicaciones independizando su desarrollo y ejecución.

En este artículo se presenta una visión general de los sistemas particionados y se compara con una visión más clásica del desarrollo de sistemas empotrados de tiempo real. Finalmente, se ofrecen las referencias tecnológicas para el desarrollo de estos sistemas.

Palabras clave: Sistemas particionados, hipervisores, sistemas de tiempo real

I. INTRODUCCIÓN

La automatización industrial ha sufrido un proceso de transformación continuo que en los últimos años se ha acelerado debido al gran desarrollo de sistemas empotrados y su aplicación industrial[8]. La integración de los sistemas empotrados como componentes de control ha hecho que la potencia y funcionalidades de los mismos se haya incrementado de forma notable. Hoy en día es fácil encontrarnos con sistemas de control que pueden ser supervisados y gestionados a través de conexiones inalámbricas, sofisticadas interfaces persona-máquina, con dispositivos basados en visión, etc.

Durante estos últimos años, la evolución de los procesadores para sistemas empotrados ha seguido la misma tendencia que los procesadores de propósito general en cuanto a potencia y consumo. En la actualidad, los procesadores para sistemas empotrados son cada vez más potentes y consumen menos energía. Asimismo, las aplicaciones que se pueden desarrollar sobre estos procesadores son cada vez más complejas incluyendo servicios de conexión remota, servicios web, mantenimiento remoto, etc.

Cuando en un mismo sistema empotrado se ejecutan unas aplicaciones que son de tiempo real con un nivel determinado de criticidad y/o seguridad, junto con otras que no son de tiempo real pueden aparecer unos problemas como:

- Fallos no intencionados: Alguna de las aplicaciones puede fallar afectando al funcionamiento de otras. Estos fallos pueden ser debido a la memoria común para las aplicaciones o al uso de algún recurso del sistema.
- Fallos intencionados: Se puede explotar algún fallo de seguridad conocido en algún programa que se está ejecutando (por ejemplo, un servidor http) y tomar el control del sistema o forzar un funcionamiento incorrecto.

En el caso de la memoria, una aplicación puede acceder y modificar direcciones de memoria que son de otras aplicaciones y así alterar su funcionamiento. Esto puede ocurrir por fallos en la programación de estas aplicaciones o de forma deliberada como mecanismo para acceder y modificar el comportamiento del sistema. Esta técnica es una de las usadas por los hackers para coger el control de una máquina determinada o acceder a la información privada de ésta.

En el caso de los recursos, una aplicación puede fallar y coger el recurso de forma exclusiva. Por ejemplo, un fallo de programación puede hacer que se ejecute un bucle infinito y no deja que otras aplicaciones se ejecuten.

Las principales necesidades o requisitos para un sistema empotrado de tiempo real que integre varias aplicaciones son:

- Disponer de espacios de memoria independientes y totalmente aislados de manera que un fallo en una aplicación, no repercuta sobre las otras aplicaciones. Además, una aplicación nunca deberá poder acceder al espacio de otra. El término que define esta característica es aislamiento espacial.
- Independizar las aplicaciones desde el punto de vista del procesador. Una aplicación nunca podrá hacer un uso exclusivo de la CPU, haciendo que otras aplicaciones no se puedan ejecutar. Esta característica se conoce como aislamiento temporal.
- Gestionar los posibles errores del sistema y aplica-

ciones mediante un monitor de salud del sistema (Health Monitor) que permita detectar los errores en los primeros niveles del sistema y tratarlos de forma que el funcionamiento del sistema sea lo más seguro posible.

Para conseguir estos objetivos, la industria aeronáutica definió la arquitectura de sistemas particionados con el fin de incrementar la seguridad y predicibilidad de los sistemas de control. Los sistemas particionados se basan en la propuesta de [19] en la que se definió un nivel de separación de aplicaciones. La propuesta denominada MILS (Multiple Independent Levels of Security and Safety) [19] fue una iniciativa conjunta entre la academia y la industria para el desarrollo e implementación de arquitecturas altamente seguras para los sistemas empotrados de tiempo real. Esta propuesta se concretó en el concepto Integrated Modular Avionics (IMA).

Los sistemas particionados representan el futuro de los sistemas seguros y confiables habiéndose extendido en poco tiempo a otros ámbitos como el espacial [2], el automóvil [3] y otros.

Los fundamentos teóricos adaptados en la arquitectura propuesta fueron usados para definir el estándar ARINC-653 [1] que especifica un entorno operativo para aplicaciones basadas en sistemas particionados.

En este artículo se presenta una solución para la implementación de sistemas empotrados de control que ofrezcan un alto nivel de seguridad e integren aplicaciones con distintos niveles de criticidad, mediante técnicas basadas en el particionado.

En la sección II se presentan los sistemas particionados y las técnicas de virtualización que lo posibilitan. La sección III se detalla la arquitectura software de estos sistemas con respecto a las soluciones mas clásicas. Finalmente se presentan las conclusiones y el trabajo futuro.

II. SISTEMAS PARTICIONADOS

Un sistema particionado está configurado por varias aplicaciones independientes llamadas particiones ejecutándose sobre una capa de virtualización. La siguiente figura muestra la arquitectura de un sistema particionado.

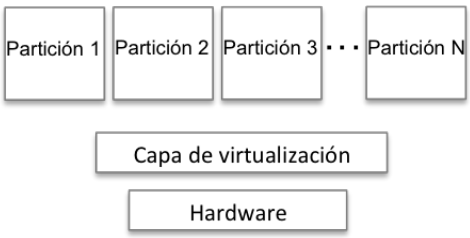

Figura 1. Sistema Particionado.

En la arquitectura se identifican los siguientes niveles:

- Capa de virtualización formada por un hipervisor cuyo cometido es ofrecer una máquina virtual sobre el que se ejecutan las particiones
- Particiones que contienen las aplicaciones que se ejecutan sobre la máquina virtual.

II-A. Hipervisor

El término hipervisor se utiliza para nombrar la capa de software que realiza las funciones de virtualización. El hipervisor, también llamado "Virtual Machine Monitor"(VMM), virtualiza los recursos del sistema y permite que se puedan ejecutar varias particiones con sus correspondientes sistemas operativos.

Las principales funcionalidades que ofrecen los hipervisores son:

- Gestión de particiones: las particiones (o entornos de ejecución) pueden ser instaladas, arrancadas, paradas, sustituidas, etc., dinámicamente.
- Aislamiento temporal: el hipervisor planifica las particiones atendiendo a políticas estáticas (ARINC 653 define una política totalmente estática) o por prioridades con limitación del uso de la CPU por partición.
- Aislamiento espacial: todas las particiones son ejecutadas en su propio espacio de memoria virtual independiente (no comparten la memoria física) y en modo usuario del procesador.
- Virtualización de los principales recursos del sistema: se ofrece una máquina virtual cercana a la máquina nativa con un subconjunto del procesador: CPU, reloj, temporizadores, interrupciones, y gestión memoria.
- Mecanismos de comunicación entre particiones: las particiones pueden comunicarse mediante mecanismos basados en puertos y canales definidos en el estándar ARINC-653.
- Gestión de fallos: ofrece un monitor de salud configurable que permite detectar y tratar los fallos del sistema o de las particiones y tomar acciones para aislar el fallo y evitar su propagación
- Gestión de la seguridad del sistema: junto con el monitor de salud y hardware adecuado, se puede detectar modificaciones en el software en ejecución (ataques a alguna partición) y tomar las acciones adecuadas para aislar el subsistema atacado.
- El sistema se configura de forma estática: para sistemas empotrados de tiempo real, el sistema se define mediante un fichero que detalla los recursos asociados a cada partición. Este fichero incluye, entre otras cosas, la definición de las particiones y los recursos utilizados por cada una de ellas, la plan seguido por el planificador, los puertos de comunicación entre particiones, las acciones a realizar por el monitor de salud del sistema, etc.

En definitiva, los hipervisores no sólo permiten un aislamiento espacial y temporal a la aplicaciones sino que

permiten integrar aplicaciones desarrolladas por distintos suministradores en un único sistema. Este aspecto es muy importante en varios sectores (aeronáutico, espacial, automóvil, etc.), en donde se pueden agrupar en un único procesador distintas aplicaciones que inicialmente estaban destinadas a un procesador específico para esa aplicación.

II-B. Particiones

Una partición es un entorno de ejecución gestionado por un hipervisor que utiliza servicios virtualizados. Cada partición consiste en uno o más procesos concurrentes (gestionados por un sistemas operativo interno a cada partición) que comparte los recursos virtualizados del sistema.

Una partición puede ser:

- Una simple aplicación formada por un código secuencial sin sistema operativo
- Una aplicación formada por varias tareas de tiempo real ejecutándose sobre un sistema operativo de tiempo real
- Una aplicación formada por varios procesos ejecutándose sobre un sistema operativo de propósito general.

Las particiones deben ser para-virtualizadas [10] para ser ejecutadas sobre un hipervisor.

Las particiones pueden ser definidas como de *sistema* o de *usuario* según tengan permiso para usar un subconjunto de servicios especiales definidos por el hipervisor. Estos servicios están relacionados con el acceso o cambio del estado de otras particiones (paro, arranque, inicialización, recarga, etc.). También estos servicios permiten acceder a los errores generados por otras particiones.

La comunicación entre particiones se realiza mediante el paso de mensajes utilizando los mecanismos básicos definidos en el estandard ARINC-653: *sampling* y *queuing ports*.

III. SISTEMAS TRADICIONALES VS PARTICIONADOS

Un sistema de control tradicional esta formado por un sistema operativo de tiempo real que gestiona un conjunto de tareas que implementan los distintos bucles de control, la interfaz de usuario, los servicios de comunicación con el exterior, etc. La siguiente figura muestra este esquema general.

Los principales problemas en el diseño de este tipo de solución se pueden resumir en los siguientes puntos:

- En el diseño coexisten actividades o tareas que tienen distinto nivel de criticidad y de restricciones temporales. En el ejemplo anterior, las tareas de control pueden tener un nivel de criticidad y restricciones temporales alto. Sin embargo, otras tareas, como la interfaz con el usuario, pueden no ser críticas.
- Los servicios requeridos, desde el punto de vista del sistema operativo, por cada tarea pueden ser muy diferentes. Siguiendo con el ejemplo, las necesidades de las tareas de control son las básicas que ofrece un sistema operativo (creación y planificación de tareas, comunicación y sincronización, acceso a los depósitos de entrada y salida, etc.), mientras que otras tareas pueden requerir otros servicios (gráficos, librerías específicas, acceso a dispositivos de comunicación externa, servidores de red, etc.).
- El uso de un único sistema operativo para todas las tareas requiere que sea de tiempo real y que además ofrezca un gran número de servicios típicos de los sistemas operativos de propósito general que hace aumentar la complejidad de éste.
- Fallos en la ejecución de cualquier tarea, pueden afectar al funcionamiento global del sistema.
- Si se requiere un cierto nivel de certificación del sistema, la certificación de un sistema operativo de tiempo real tiene una complejidad proporcional al número de líneas del operativo. Mientras que un operativo de tiempo real básico puede tener alrededor de 5KLOCs, un sistema operativo con servicios extendidos puede multiplicar por diez el número de líneas.
- La ejecución bajo un único sistema operativo de tareas de distinto nivel de criticidad fuerza a que todas las tareas tengan que certificarse con el máximo nivel de criticidad aunque algunas de ellas no sean críticas.

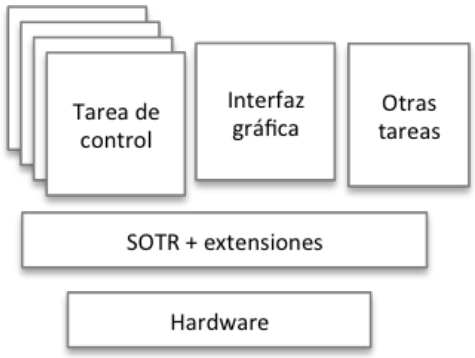

Figura 2. Arquitectura software de un sistema de control.

III-A. Sistemas distribuidos

Algunos de los problemas anteriormente citados has sido abordados de una forma arquitectónica mediante los sistemas distribuidos. La solución frente a la necesidad de aislar los aspectos críticos de los no críticos con el fin de reducir la complejidad de las aplicaciones y aumentar la robustez y la capacidad de certificación se basa en particionar el sistema en distintas máquinas en el que cada una de ellas realiza unas funciones determinadas.

La figura 3 esquematiza esta arquitectura distribuida. En ella se identifica un hardware específico para tareas de tiempo real críticas con su sistema operativo de tiempo real

(SOTR), y otro hardware para las tareas que no son críticas soportadas por un sistema operativo de propósito general (SOPG).

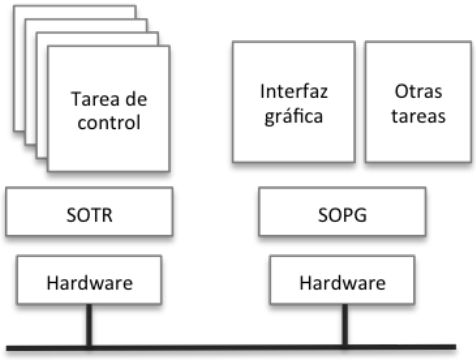

Figura 3. Arquitectura distribuida de un sistema de control.

Esta distribución permite aumentar la capacidad de cómputo global e independiza las aplicaciones. La necesidad de comunicación a través de redes predecibles requiere que los sistemas operativos den soporte a los protocolos de comunicación correspondientes. El análisis de la planificabilidad de este tipo de sistemas se realiza mediante técnicas ampliamente conocidas [20], [12]

III-B. Sistemas particionados

El aumento de cómputo de los procesadores empotrados y la necesidad de integrar aplicaciones con distintos niveles de criticidad y seguridad ha facilitado que los sistemas particionados hayan tenido un amplio desarrollo en estos últimos años. Este desarrollo ha sido muy importante en sectores como el espacial [17], [14] que ha establecido una hoja de ruta para su desarrollo en los próximos años.

En la visión de sistemas particionados, la arquitectura distribuida federada es una arquitectura integrada en la que las aplicaciones son independientes.

La figura 4 muestra un esquema de esta arquitectura integrada.

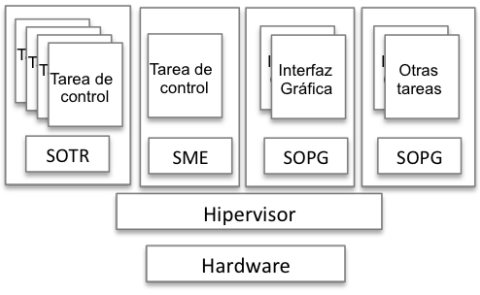

Figura 4. Arquitectura integrada de un sistema de control.

En la arquitectura esquematizada, las tareas críticas de control se pueden ubicar en una partición con un sistema operativo de tiempo real que ofrezca los servicios para ejecutar las tareas. Asimismo, se podrían separar dichas tareas de control en particiones separadas en las que el soporte de ejecución fuese mínimo (SME). En cualquier caso, pueden coexistir ambas alternativas en distintas particiones. De la misma manera, otras particiones pueden albergar las aplicaciones menos críticas (la interfaz gráfica por ejemplo) sobre sistemas operativos de propósito general.

Las principales ventajas de esta solución se detallan a continuación:

- Las particiones son totalmente independientes. Un fallo en una de ellas está aislado en esa partición y no repercute a las otras. El hipervisor es el elemento encargado del aislamiento espacial y temporal de las particiones.
- Los servicios de los sistemas operativos de las particiones se pueden reducir para cubrir las necesidades estrictas de las aplicaciones minimizando el código del operativo y favoreciendo la certificabilidad de la partición.
- Al ser las particiones independientes, la validación y certificación de éstas es totalmente independiente. Esto permite la coexistencia de particiones certificadas (las más críticas) junto con otras no certificadas.
- Cada partición puede tener un nivel de criticidad distinto favoreciendo las arquitecturas para la criticidad mixta (”mixed criticality systems”)
- La modificación o sustitución de una partición no afecta a las otras particiones.
- Una partición puede actuar como supervisora del funcionamiento del resto de particiones y detectar fallos en éstas realizando las acciones oportunas para restituir el sistema sin alterar el funcionamiento de las otras. La gestión de las particiones permite a una determinada partición el poder restituir el código de una partición, pararla, reinicializarla, etc.
- La comunicación entre las particiones se puede realizar mediante los mecanismos que ofrece el hipervisor: canales de comunicación.

Los principales inconvenientes de esta arquitectura se pueden delinear en los siguientes puntos:

- La integración del sistema es mas compleja. El sistema requiere un hipervisor y sistemas operativos adaptados.
- La planificación del sistema pasa a ser de dos niveles. Por un lado el hipervisor planifica las particiones mediante una política de planificación cíclica [7], [4]. Cuando una partición está en ejecución, las tareas internas se planificación de acuerdo a la política de planificación del sistema operativo de la partición.
- El análisis de la planificabilidad del sistema requiere técnicas [9], [5] y herramientas [6] específicas.

IV. Soluciones tecnológicas

Actualmente existen pocas soluciones comerciales que permitan la construcción de este tipo de arquitecturas. En el ámbito del código abierto, nuestro grupo ha desarrollado el hipervisor XtratuM [13], [15] que está siendo utilizado por la ESA para el desarrollo de la futura generación de satélites.

Para el desarrollo de las particiones se pueden usar distintos sistemas operativos:

- PartiKle [18]: sistema operativo de tiempo real que ofrece una interfaz POSIX-PSE51.
- XAL (XtratuM Abstraction Layer): sistema mínimo de ejecución para el desarrollo de particiones basadas en C.
- ORK+ [11]: soporte de ejecución para aplicaciones en Ada basado en el perfil Ravenscar.
- Lithos [16]: sistemas operativos de tiempo real basado en el estándar ARINC-653.
- RTEMS: sistema operativo de tiempo real.
- Linux: sistema operativo de propósito general para el desarrollo de aplicaciones diversas.

IV-A. XtratuM

XtratuM [13], [15] es un hipervisor que ha sido diseñado para sistemas empotrados de tiempo real usando técnicas de para-virtualización [10]. Esta técnica permite reducir la sobrecarga de la capa de virtualización y alcanzar niveles de prestaciones para garantizar las restricciones temporales de las aplicaciones.

Básicamente, un hipervisor suministra máquinas virtuales aisladas. Cada máquina virtual puede ejecutar una aplicación junto con su sistema operativo que configura una partición. En la nomenclatura de sistemas particionados, esto se también se nombra como un entorno de ejecución. El hipervisor es el encargado de suministrar los mecanismo para la comunicación entre las particiones.

Las principales características del hipervisor XtratuM son:

- Aislamiento temporal: Las particiones se almacenan en regiones de memoria independientes no permitiéndose el acceso a regiones de otras particiones.
- Aislamiento temporal: El hipervisor ejecuta las particiones en intervalos de tiempo especificados en un plan de ejecución. Cualquier partición no puede ejecutarse fuera de sus intervalos.
- Predecibilidad: El hipervisor ofrece una serie de servicios que son predecibles.
- Seguridad: La información relativa a una partición (código, datos y pila) está protegida respecto al acceso de otras particiones y accesos externos. Seguridad implica la definición de las políticas de gestión de los recursos y los mecanismos que las implementan.
- Gestión de recursos: Los recursos del sistema (CPU, memoria, E/S, interrupciones, elementos de comunicación, etc.), se definen de una manera estática en la configuración del sistema. Esta especificación de la asignación de recursos constituye la base para que el hipervisor permita el acceso o deniegue a las particiones el uso de los recursos.
- Gestión de fallos: Un aspecto fundamental en sistemas críticos es la detección, manejo y aislamiento de los fallos para que no se propaguen a otros elementos del sistema.

V. Conclusiones

En este artículo se ha pretendido presentar la arquitectura de sistemas particionados como una alternativa tecnológica de futuro para el diseño de aplicaciones de control que integren varias tareas con distintas necesidades en cuanto al nivel de seguridad, criticidad y certificabilidad.

Los sistemas particionados constituyen actualmente una forma eficiente para el desarrollo de sistemas seguros y están siendo adoptados en distintos campos de aplicación que requieren aplicaciones altamente seguras y fiables.

Nuestro grupo de investigación ha desarrollado en los últimos años la tecnología basada en la virtualización concretándose en el hipervisor XtratuM que ha tenido un reconocimiento muy importante en el sector espacial del cual es una referencia.

El trabajo futuro se centra en la mejora de las prestaciones de este tipo de sistemas, la adaptación de otros sistemas operativos y el desarrollo de herramientas para facilitar el diseño y desarrollo de las aplicaciones.

Agradecimientos

El trabajo presentado ha sido desarrollado en el marco de los proyectos HIPARTES y COBAMI financiados por el Ministerio de Ciencia e Innovación (TIN2011-28567-C03-03, DPI2011-28507-C02-02) y HISEM (Prometeo 2009/02) financiado por la Generalitat Valenciana.

Referencias

[1] *Avionics Application Software Standard Interface (ARINC-653).*, March 2006 2006. Airlines Electronic Eng. Committee.

[2] P. Arberet and J. Miro. IMA for space : status and considerations. In *ERTS 2008. Embedded Real-Time Software.*, Jannuary. Toulouse. France 2008.

[3] AUTOSAR. Automotive open system arquitecture. www.autosar.org.

[4] T. P. Baker and A. Shaw. The cyclic executive model and ada. In *Real-Time Systems Symposium, 1988., Proceedings.*, pages 120–129, 1988.

[5] P. Balbastre, I. Ripoll, and A. Crespo. Exact response time analysis of hierarchical fixed-priority scheduling. In *Proceedings of 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, August 2009.