Sistemas de tiempo real, una aproximación desde Java

Luis Gerardo Velásquez Vásquez - B06789

Resumen:

Con el creciente uso de computadores en sistemas automáticos, orientados a servicio de usuarios se han venido creando y estudiando distintas arquitecturas para los sistemas de tiempo real. Al manejar tareas en tiempo real se debe tener en cuenta que estas deben de ser servidas lo más pronto posible antes de su tiempo de caducación. Es decir, se pretende reducir la latencia u optimizar tiempo de respuesta de dicho sistema. Por otro lado tenemos el factor de la gran diversidad de fabricantes de dispositivos por lo que producir estos sistemas conlleva temas como la portabilidad, fácilidad para desarrollo entre otros. Java como lenguaje y herramienta de desarrollo provee gran potencial para desarrollar programas que puedan utlizarse en sistemas de tiempo real.

1-Introducción:

Un servicio es la acción ofrecida por parte de un sistema o usuario hacia otro de acuerdo a ciertos lineamientos establecidos por acuerdo mutuo. [1] La distancia entre estos dos puede variar, pueden estar integrados en el mismo computador, a nivel de LAN o WAN. Los sistemas de tiempo real (RTS en inglés, son sistemas que dependen no solamente de la correctitud lógica de un servicio, sino también del instante físico (tiempo) en el que los resultados son producidos. [1] Se pueden distinguir o clasificar en dos tipos de sistemas: *soft real-time systems* (sistemas de tiempo real suaves) y *hard real-time systems* (sistemas de tiempo real duros). [2] Los del primer tipo no aseguran que el servicio va a ser finalizado antes de su plazo establecido, pero que si se les dará prioridad sobre otros; los del segundo tipo si aseguran que se servirá antes de su plazo establecido. Otro concepto importante es la latencia, esta es el tiempo que se tarda entre el evento o solicitud de un servicio hasta que este retorna su resultado o respuesta.

2-Sistemas de tiempo real suaves:

En general los sistemas desarrollados en la actualidad están ejecutandose en varios hilos. Los sistemas de tiempo real suaves comúnmente son aquellos que se desarrollan en software de nivel más alto. Los sistemas embebidos son los que abarcan principalmente los sistemas de tiempo real duros. Las aplicaciones de alto nivel se encargan de tareas como interactividad, usabilidad, configuraciones y visualización. Cabe destacar que estas aplicaciones de alto nivel siempre tienen como base un sistema operativo de tiempo real. [2] Acá entra un concepto importante que es la calidad de servicio calidad de servicio, QoS (Quality-of-service). Las tareas se pueden clasificar en sensitivas como los juegos de video en línea, videollamadas, monitoreo de seguridad entre otros. Estas tareas son sumamente suceptibles a que un fallo en cualquiera de sus nodos de servicio. Un fallo reduce el QoS, por lo que se busca una optimización del sistema global [3]

3-Java como lenguaje de programación:

Java es un lenguaje de programación orientado a objetos que fue introducido por Sun Microsystems en el año 1995. Desde su lanzamiento ha generado mucha atracción en la industria debido a sus neutralidad y portabilidad. Alrededor del año 1997 se notó que se deberían emplear algunas estrategias para lograr resolver problemas en tiempo real. [4] La edición estándar que se lanzó presentaba ya algunos problemas con respecto al desarrollo de aplicaciones. Algunos problemas eran:

- -No provee soporte para hilos y prioridades.
- -Su sistema de recolección de memoria genera tiempos de espera no controlados por el programador en general.
- -No proveía implementación para la sincronización y cuanto tiempo un hilo podía bloquearse.

4-Sistemas de tiempo real en JAVA:

A pesar de que Java tiene la gran facilidad y ventaja que provee la portabilidad mediante la máquina virtual, presenta la desventaja de no proveer un sistemas duros. Java provee algunas facilidades para poder generar código que presente características de tiempo real, pero solo en sistemas de tiempo real suaves. Generalmente Java se utiliza en sistemas de tiempo real suaves y distribuidos. La máquina virtual de JAVA (JVM) es excelente para crear aplicaciones multiplataforma, especialmente en la actualidad donde hay varios fabricantes de dispositivos electrónicos. Generalmente la industria telemática es donde se desarrollan la mayoría de sistemas en tiempo real con Java. [5]

En cuanto a la implementación de Java para sistemas de tiempo real existe una serie de bibliotecas o extensiones llamadas Real-Time specifications for Java(RTSJ). Su lanzamiento oficial se dio en el año 2000.Una característica peculiar e importante de Java es como se maneja la memoria. Java provee un garbage collecter para liberar memoria que no se está utilizando en el heap. El garbage collecter libera los objetos que no están siendo referenciados. RTSJ funciona de manera similar, tiene que estar liberando memoria y chequeando constantemente porque comúnmente las interrupciones que deberá capturar el sistema de tiempo real no se sabe con antelación cuantos habrán ni cuanto espacio ocuparan. [6] Existen tres paradigmas o implementaciones sobre como esta recolección se deberá hacer:

- JamaicaVM: acá cada hilo que captura un evento del exterior deberá declarar su propio garbage collecter y calcular sus propio espacio en memoria para chequear cuando existe algun objeto desreferenciado.
- IBM WebSphere RT: se crea un hilo que recogerá la basura y este tendrá mayor prioridad que los demás, incluso los hilos creados para lidiar con los eventos exteriores.
- Sun Java RTS: el estándar de Sun establece que el garbage collecter se corre en background, es decir con menor prioridad que los hilos para evitar una latencia mayor.

Sin embargo en la actualidad existe una gran gama de versiones de la RTSJ (ver Tabla 1)

Implementación	Desarrollada por	Tipo
RTSJ-RI	Timesys	Referencia inicial
Mackinac	Sun Mycrosystems/Oracle	Comercial
JRockit	Oracle	Comercial
IBM Websphere	IBM	Comercial
Real-Time JRE	Apogee	Comercial
Aonix PERC	Atego	Comercial
Jamaica	Aicas	Comercial
aJ200	aJile	Comercial
J-Rate	Universidad de California	Open Source
OVM	Universidad de Purdue	Open Source
LJRT	Universidad de Lund	Open Source
JOP	Universidad de Viena	Open Source
Fiji	Universidad de Purdue	Open Source

Tabla 1: Algunas variaciones de RTSJ (tomado de [6], modificada por Luis Velásquez)

RTSJ provee siete áreas [6] para mejorar los sistemas de tiempo real:

i. Planificación y despacho: Java mapea cada sistema operativo y núcleo y de acuerdo a las políticas de este se acomoda para un mayor desempeño ante eventos inesperados.
ii. Manejo de memoria dinámico: aún se investiga sobre como se debe manejar la memoria dinámica, sin embargo está claro que el colector de basura es una de sus grandes características a mejorar, también el manejo de entrada/salida
iii. Sincronización y compartimiento de recursos: presenta algunos módulos basados en el *problema de la inversión.* También provee herramientas para trabajar con el model fork/join en su reciente versión JDK 7
iv. Manejo de eventos asincrónico: presenta una clase denominada *Asynchronous Event handler* (Aeh) la cual se encarga de manejar los peores escenarios al darse las interrupciones.
v. Transferencia de control asincrónico: provee mecanismos seguros para abandonar los procesos de menor prioridad; se implementa con la clase Thread
vi. Terminación de eventos asincrónicos: no hay que olvidar que al terminar hilos para manejar interrupciones estos deben ceder su espacio en memoria.
vii. Acceso a memoria física: no solamente se debe administrar y calcular los tiempos de entrada/salida, sino que también se debe lidiar con la creciente tecnología multicore.

5-Conclusion:

Los sistemas de tiempo real, sus servicios e implementación están en la tecnología diaria. Es importante reconocer que la automatización es una gran técnica para brindar mejor calidad de procesos y Java es una gran herramienta que permite que exista portabilidad y fácil desarrollo de aplicaciones para este tipo de sistemas.

Referencias:

[1] M. Turner, D. Budgen, y P. l. Brereton, “Turning Software into a Service,” Computer, vol. 36, no. 10, Páginas. 38-44, 2003.

[2] A. Silberschatz, P. Baer yG. Gagne, “Operating Systems Concepts”, Páginas 283-289, 2013

[3] Lee, Jinkyu y Shin, Insik y Easwaran, Arvind. “Online Robust Optimization Framework for QoS Guarantees in Distributed Soft Real-time Systems” en *Proceedings of the Tenth ACM International Conference on Embedded Software.* Páginas 89-98, 2010

[4] Higuera-Toledano, M. Teresa. “About 15 Years of Real-time Java” en *Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems*. Páginas: 34-43, 2012

[5] A. Bianconi, Maria-Paola y Francois, Nicolas y Cortese, Giovanni and Yu, Erik. “Flexible Java Real-time Profile for Business-critical Systems” en *Proceedings of the 4th International Workshop on Java Technologies for Real-time and Embedded Systems.* Páginas 135-143, 2006

[6] Richardson, T. and Wellings, A. J. y Dianes, J. A. y Diaz, M. “Towards Memory Management for Service-oriented Real-time Systems”. Páginas 128-137, 2010