Sistemas Embarcados – 2015.2
Saulo Ricardo Gonçalves Fernandes da Silva – 20121y6-rc0060

Java e Sistemas de Tempo Real: Vale a pena utilizar?

No desenvolvimento habitual de sistemas de tempo real, os programadores escolhem linguagens como C, C++ e Ada para desenvolver seus projetos. Porém com a atual popularidade da linguagem de programação Java e sua usabilidade, faz com que vários desenvolvedores queiram utilizar para sistemas de tempo real, entretanto vale a pena utilizar Java para STR?

Na versão padrão do Java, não. Java possui vários fatores que causam um indeterminismo temporal que é inaceitável para sistemas de tempo real inflexíveis como sistema de coleta de lixo (Garbage Collection), ausência de um modelo de threads baseado em prioridades (Gerenciamento de Thread), Carregador e compilador de classes que serão explicados a seguir como também a solução:

Sistema de coleta de lixo (Garbage Collection)

Os sistemas de lixo tradicionais podem gerar atrasos grandes nas aplicações de STRs que são potencialmente impossíveis de prevenir. Atrasos de milissegundos não são anormais de ocorrer.

Gerenciamento de Thread

Java padrão não garante nem o escalonamento e nem a prioridade das *threads*. Uma aplicação que não tem como assegurar que uma *thread* de baixa prioridade não será escalonada na frente de uma de maior prioridade.

Carregador e compilador de classes

A JVM somente faz o carregamento de uma classe quando esta é referenciada a primeira vez. Esse carregamento pode demorar um certo tempo (geralmente mais que 10 milissegundos) dependendo de alguns fatores, dentre eles, o tamanho da classe. Se for preciso carregar várias classes, o atraso para fazer isso, pode gerar um atraso bastante significante. É preciso então fazer o projeto para o carregamento antecipado de todas as classes, porém a JVM padrão não permite isso, fazendo com que seja necessário o carregamento manual das classes.

A compilação de classes para código nativo encontra um problema similar ao do carregamento. A maioria das JVM modernas interpretam métodos Java, compilando somente métodos frequentemente executados para código nativo. A não compilação imediata poupa tempo ao iniciar a aplicação, porém é terrível para aplicações com requisitos *hard* de tempo, pois métodos executados inicialmente na fase de interpretação executam mais devagar. Um

atenuante para isso pode ser a criação de um método que utilize o compilador de classes para compilar os métodos ao iniciar a aplicação.

Solução

Para solucionar estes e os demais problemas, em junho de 1998 e culminando em novembro de 2011 com a publicação da especificação de Java para Tempo Real – *Real Time Specification for Java* (RTSJ) e a aprovação em 2002. É também definido de forma clara e precisa, um vocabulário de tempo real, com conceitos antes usados de forma contraditória, e as vezes até mesmo errônea, pela comunidade de usuários, desenvolvedores e revendedores. Nela são feitas algumas ampliações na definição original da linguagem Java.

Conclusão

A linguagem Java antes da publicação (RTSJ) não possui a padronização e muito menos solução para aplicações de sistemas de tempo real, porém com a padronização a maioria das aplicações funcionam bem com a linguagem Java, contudo ainda a problemas como no *AsyncEventHandler* e a dificuldade no desenvolvimento.

Referências

Java e sistemas de Tempo Real: vale a pena usar? Disponível em: https://pc2008evermat.wordpress.com/2008/07/02/java-e-sistemas-de-tempo-real-vale-a-pena-usar/ 29/09/15

Java e Sistemas de Tempo Real: Vale a pena utilizar? Disponível em:

https://determinismotemporal.wordpress.com/2008/06/10/java-e-sistemas-de-tempo-real-vale-a-pena-utilizar/ 29/09/15

Vale a pena usar Java em aplicações de tempo real? Disponível em:

http://sisttemporeal.blogspot.com.br/2008/08/vale-pena-usar-java-em-aplicaes-de.html 29/09/15