

News for developers of
critical software applications

Editorial

Dear Readers,

Even though this was the hottest summer in Germany for decades, the aicas team did its best to provide you with enhanced development tools.

JamaicaVM's realtime capable garbage collector has kept aicas' technology at the forefront of Java solutions for the past several years. The usability and memory demand improvements in the new 2.2 version will help to maintain this lead.

Particularly, the new Eclipse plug-in for Jamaica has received general approval from our customers. I would like to thank all of you who send us constructive feedback. This has enabled us to provide a higher quality integration of our JamaicaVM into the graphical development tool Eclipse.

We hope this new edition of the aicas newsletter provides you with many interesting articles.

Yours,

Andy Walter.

JamaicaVM 2.2

On October 13th, the JamaicaVM 2.2 is released.

The new version of JamaicaVM, aicas realtime Java implementation, provides a number of additions to the support for the standard Java library as well as optimizations of the virtual machine (VM) itself and the associated JamaicaVM tools.

The most important new feature added to the standard library support is Remote Method Invocation (RMI). RMI enables calling methods on a remote host and forms the basic mechanism for developing distributed systems in Java.

The standard library used in JamaicaVM has been merged with the current GNU Classpath code. In the extended packages, support for *javax.comm* was added for the target operating systems Linux, VxWorks, and NetOS. This API enables the use of serial interfaces from Java code.

The JamaicaVM excels in high performance and low memory demand. Thus it is particularly well suited for embedded systems. In

the new version, the byte-code interpreter has been optimized and the time necessary for loading classes has been decreased. With further optimizations of the internal structure, the memory demand of the VM itself could be reduced significantly, typically around 50%.

Besides the VM, a number of tools are included in the distribution which enable the user to configure an application ideally for each target system, e.g., available memory, processor. The Profiler tool, which analyses an application's runtime behavior, has been extended to monitor memory demand, number of threads used, method invocations, and classes accessed via reflection. This information automates the building and tuning of stand-alone applications.

Now JamaicaVM supports Windows better. For more information, see the article "New JamaicaVM for Windows" on page 4. (rs)

News

Japanese Market Addressed

aicas expands its area of activity and offers its products to the far-east markets. aicas's web site has been extended with a Japanese area under www.aicas.com/japan. From Japan, aicas can be reached via our free phone number 0066.3.384.9014.

Update-Site for Eclipse Plug-In

The JamaicaVM plugin for the Eclipse IDE (www.eclipse.org) is now available via an update-site. This enables straight-

forward downloading of the plug-in from within Eclipse itself. New releases of the plug-in will be available via this means. For more information on the required configuration for Eclipse see www.aicas.com/eclipse.html.

Distributor in Benelux

Mind NV Belgium has become an aicas partner for the distribution and service of JamaicaVM realtime Java Technology. Mind provides consulting, tool chains, development, train-

ing, and support for Embedded Linux and eCos to system development companies.

Partnership for Space Systems

Astrium EADS has also become an aicas partner for the development and promotion of Java solutions for embedded space applications. Both companies join forces to advance the technology developed in the ESA project AERO and the resulting AERO-VM, a customized version JamaicaVM for the space domain.

The Realtime Specification for Java in Practice

This article gives an introduction into the area of using Java Technology in real-time systems. It presents the possibilities offered by the Realtime Specification for Java and the advantages of the realtime implementation JamaicaVM.

The enormous success of Java technology is due to the many advantages, such as higher productivity and safety, the language brings to the developer. Even critical applications, as in automotive or aerospace control, can profit from these advantages. However, there are also drawbacks to traditional Java implementations that prevent their use in critical systems. The most obvious of these drawbacks is the lack of realtime behavior.

The biggest problem: garbage collection

One of the biggest advantages of Java technology, the safe memory management through automatic garbage collection, is also the biggest problem for the use in realtime systems. However, the garbage collector is the basis for the safety mechanisms and is consequently compulsory.

The garbage collector in a classic Java system can stop all application threads at an unpredictable time and for an unpredictable duration. This leads to pauses as

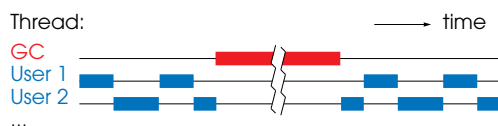


Figure 1: Application threads are stopped by the garbage collector.

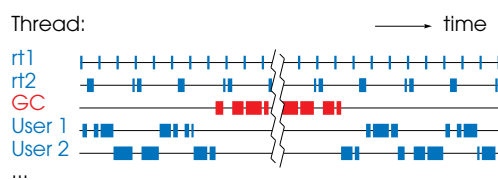


Figure 2: RealTimeThreads in the RTSJ are not affected by garbage collector activity.

illustrated in Figure 1. These pauses make the prediction of timing behavior of the application impossible. To permit realtime programming, one therefore needs a means to avoid these pauses—a deterministic incremental garbage collector that does not cause unpredictable pauses of

the application—needs to be employed.

Realtime programming with the RTSJ

The aim of the Realtime Specification for Java (RTSJ) is to extend the Java language definition and the Java standard libraries to support realtime threads, i.e., threads whose execution conforms to certain timing constraints. Nevertheless, the specification is compatible with different Java environments and backwards compatible with existing, non realtime Java applications.

The most important improvements of the RTSJ affect the following seven areas:

- thread scheduling,
- memory management,
- synchronization,
- asynchronous events,
- asynchronous flow of control,
- thread termination, and
- physical memory access.

With this, the RTSJ also covers areas that are not directly related to realtime applications. However, these areas are of great importance to many embedded realtime applications such as direct access to physical memory (e.g., memory mapped I/O) or asynchronous mechanisms.

Thread Scheduling To enable the development of realtime software in an environment with a garbage collector that stops the execution of application threads in an unpredictable way, new thread classes *RealtimeThread* and *NoHeap-RealtimeThread* are defined. These thread types are unaffected or at least less heavily affected by garbage collection activity. Also, at least 28 new priority levels, logically higher than the priority of the garbage collector, are available for these threads. Figure 2 illustrates the new realtime thread classes that can interrupt garbage collector activity.

Memory Management For realtime threads not to be affected by garbage collector activity, these threads need to use memory areas that are not under the control of the garbage collector. New memory classes, *ImmortalMemory* and *ScopedMemory*, provide these memory areas. One important consequence of the use of special memory areas is, of course, that the advantages of dynamic memory management is not fully available to realtime threads.

Synchronization In realtime systems with threads of different priority levels, priority inversion situations must be avoided. Priority inversion occurs when a thread of high priority is blocked by waiting for a monitor that is owned by a thread of a lower priority. The RTSJ provides the alternatives priority inheritance and the priority ceiling protocol to avoid priority inversion.

The RTSJ offers powerful features that enable the development of realtime applications. Figure 3 shows an example how the RTSJ can be used in practice. In this example, a periodic thread is created. This thread becomes active every 20ms and creates a short output onto the standard console. A *RealtimeThread* is used to implement this task. The priority and the length of the period of this periodic thread need to be provided. A call to *waitForNextPeriod()* causes the thread to wait after the completion of one activation for the start of the next period. An introduction to the RTSJ with numerous further examples is given in the book by Peter Dibble (4).

The RTSJ provides a solution for realtime programming, but it also brings new difficulties to the developer. The most important consequence is that applications have to be split strictly into two parts: a realtime and a non-realtime part. The communication between these parts is heavily restricted: realtime threads can-

```

/* Periodic thread in Java: */
import javax.realtime.*;
public class HelloRT {
    public static void main(String [] args) {
        /* priority for new thread: min+10 */
        int pri = PriorityScheduler
            .instance()
            .getMinPriority() + 10;
        PriorityParameters prp =
            new PriorityParameters(pri);
        /* period: 20ms */
        RelativeTime period =
            new RelativeTime(20 /* ms */,
                0 /* ns */);
        /* release parameters for periodic thread: */
        PeriodicParameters perp =
            new PeriodicParameters
                (null,period,null,null,null,null);
        /* create periodic thread: */
        RealtimeThread rt= new RealtimeThread(prp,
            perp) {
            public void run() {
                int n=1;
                while (waitForNextPeriod() && (n<100)) {
                    System.out.println("Hello "+n);
                    n++;
                }
            }
        };
        /* start periodic thread: */
        rt.start();
    }
}

```

Figure 3: Example programme that starts a periodic task using a RealtimeThread.

not perform memory operations such as the allocation of objects on the normally heap which is under the control of the garbage collector. Synchronization between realtime and non-realtime

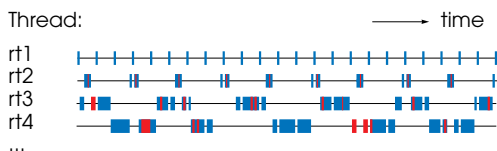


Figure 4: Realtime Garbage Collection: All Threads are realtime threads.

threads is heavily restricted since it could otherwise cause realtime threads to be blocked by the garbage collector.

Realtime Garbage Collection

In a system that supports realtime garbage collection, this strict separation into realtime and non-realtime threads is not necessary. The strict splitting of an application is consequently not required. Threads are activated only depending on their priorities. Figure 4 illustrates such a system. The JamaicaVM is a Java implementation that provides realtime behaviour for all threads.

The realtime garbage collector performs its work predictably

within the application threads. It is activated when memory is allocated. The work done on an allocation must be pre-emptible, such that more urgent threads can become active.

The implementation of a realtime garbage collector has to solve a number of technical challenges. Garbage collector activity must be performed in very small single increments of work. In JamaicaVM, one increment consists of garbage collecting only 32 bytes of memory. On every allocation, the allocating thread 'pays' for the memory by performing a small number of these increments. The number of increments can be analyzed, such that this is possible even in realtime code.

The RTSJ provides a powerful extension to the Java specification. Its full power, however, is achieved only by the combination with a realtime garbage collector that helps to overcome its restrictions.

Summary

Even though a number of technical challenges had to be solved, the Realtime Specification for Java now provides a standard that enables the development of time critical software using the Java language. Consequently, this successful language becomes usable for embedded and realtime systems.

The advantages of using the Java language are even higher, when realtime features are not a mere 'add-on' to a classic Java implementation, but the implementation is inherently realtime capable and consequently does not suffer from the restrictions of the RTSJ. Such an implementation provides the advantages, that made Java so successful, to the developer of realtime systems.

Outlook

The next issue of the *aicas news* will present

Dynamic Class Loading

Using the dynamic features of Java in embedded systems properly.

(fs)

- (1) Gregory Bollella (Editor): "The Real-Time Specification for Java", Addison-Wesley, 2000. The current version is available online at www.rtfj.org.
- (2) The Real-Time for Java Expert Group (RTJEG), www.rtfj.org
- (3) The Java Community Process, www.jcp.org
- (4) Peter C. Dibble: Real-Time Java Platform Programming, Prentice Hall, 2002

Trainings & Presentations

Realtime Java Education at the research center FZI

The Authorized Java Center (AJC) at the FZI Karlsruhe has expanded its training to include realtime and embedded Java as a response to customer requirements and project experience. A new course (REJ017) covers RTSJ based on JamaicaVM and the use of UML and UML's specific extensions for realtime applications. Part of this course is a workshop that illuminates the subject through a practical example covering the complete development chain from requirements analysis through design to implementation and testing. Contact: judt@fzi.de, wwwswt.fzi.de

Java in Wireless and Embedded Environments

6-10 October 2003,
9-13 February 2004.

A 5-day training course is being offered in cooperation with RP-Cube and Microconsult covering Java editions, configurations, and profiles; requirements for embedded platforms; Java environments and Java applications; wireless and embedded scenarios; realtime OS, VM, and tools; Java technology and realtime; performance and code size.

Contact: info@aicas.com

Java compression saves Time & Money

jaccelerator breaks down flash memory requirements and transmission delays

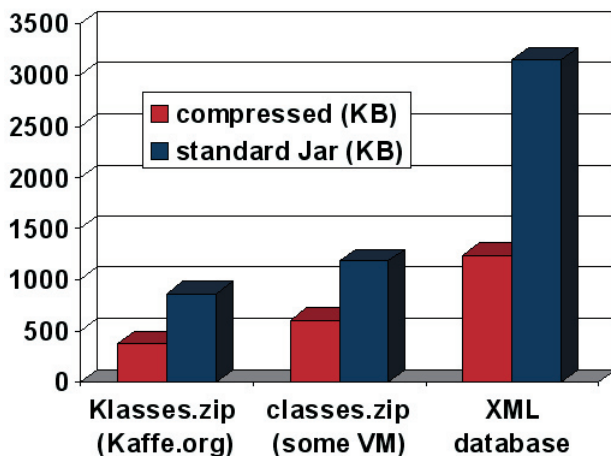
jaccelerator shrinks Java applications to 1/2 to 1/3 of the original JAR size. Typical areas of application are wherever Java applications require expensive flash memory or are transmitted via costly and/or slow wireless channels: Wireless phones and PDAs,

automotive infotainment, interactive TV-MHP etc.

Depending on the application, compression is part of the development chain, acts on-the-fly in an application or source server, or runs 'in-system' embedded in the target system.

and runs with J2ME CLC and CLDC configurations as well as with full J2SE, and, of course, with JamaicaVM. Optionally, a special edition of JamaicaVM with integrated jaccelerator support is available from aicas.

Rolf Matzner, RP-Cube / (aw)



Jaccelerator compression rates

Decompression is handled by the VM's extended class loader and takes place at class loading time. An optionally available image extension compresses the PNG files often found as resources embedded in JAR archives.

jaccelerator is RT-OS independent

New JamaicaVM for Windows

No additional Tools required

JamaicaVM version 2.2 from aicas GmbH, has improved its support for Windows as a development platform. Unlike earlier versions, no additional tools need to be installed on the Windows computer.

This version now supports code generation for Windows console applications as well as the VxWorks/PowerPC, VxWorks/x86 platforms from Wind River and NetOS/Net+ARM from NetSilicon. For these targets, support for native threads, filesystems operations, network functions, and serial communication are readily available.

The JamaicaVM plug-in for Eclipse is also available for windows from the aicas web site. Eclipse makes development of Java applications for embedded systems on Windows significantly more convenient than using the Jamaica Builder directly. (tr)

Release 4.0 of RCE and BDE

The Revision Control Engine
The Byte Differencing Engine

aicas GmbH continues to support the needs of RCE and BDE customers with new releases in October.

RCE provides high quality revision control with both a Visual interface (VRCE) and an API for integrating revision control in industrial products.

The new version provides a faster streaming interface. Now input and output data streams can be passed directly into RCE instead of over files or file descriptors. Thus performance for RCE in network applications can be significantly improved.

A revamped GUI for VRCE significantly improves its usability.

BDE, a library for extremely compact delta compression, will also sport the next streaming interface making it ideal for handheld applications. (jjh)

Events

GSOFT, Noordwijk

14-16 October. The second Galileo Software Workshop organized by the European Space Agency ESA will take place at the ESTEC space center in the Netherlands.

SPS/IPC/Drives, Nuremberg

25-27 November. SPS/IPC/DRIVES is the exhibition for automation technology. It covers all components down to the system level and offers herewith integrated automation solutions. Exhibitors and visitors profit from the advantage that visiting numerous exhibitions on components becomes superfluous. aicas will present their real-time technology together with Euros Embedded in Hall 7A, booth 622.

Embedded World, Nuremberg

17-19 February. The most important European embedded systems show. Visit us and Euros Embedded in Hall 11, booth 222.

Contact

Editors:

Dr. James J. Hunt (jjh), Dr. Torssten Rupp (tr), Roman Schnider (rs), Dr. Fridtjof Siebert (fs), Andy Walter (aw)

aicas GmbH

Hoepfner Burg
Haid-und-Neu-Str. 18
76131 Karlsruhe
Germany

phone +49.721.663.968-0
fax +49.721.663.968-99
email info@aicas.com
web www.aicas.com