

aicas news

News for Software Developers of Critical Applications

Autumn 2005

↓ Editorial

Dear Reader,

I am pleased to present to you the latest newsletter from aicas. Another busy summer has passed since our last issue. Several preferred customers already have applications running on our new Swing. This functionality will be available to all in the 2.8 release of the JamaicaVM. We also have a new Safety Critical Java Product in store for you. But no aicas news would be complete without a deeper look at technology for realtime Java.

Our main article is a contribution on distributing realtime Java applications using CORBA from Object Interface Systems. They introduce Realtime CORBA and what is necessary for its use. Of course, their ORBexpress RT is available for the JamaicaVM.

I hope this issue meets your expectation for informative articles on new technologies. We welcome your feedback. We are particularly interested in hearing of technical issues of interest to you.

Sincerely,

Dr. James J. Hunt
CEO

JamaicaVM Swings!

The JamaicaVM, version 2.8, features AWT and Swing graphics

In time for the fall harvest, the aicas' team is pleased to announce the newest release of the JamaicaVM. Version 2.8 considerably expands the application domain of the JamaicaVM. Highlights include graphics support, performance, and improved usability.

The major feature of this new release is the inclusion of a core subset of AWT and Swing. This core enables sophisticated graphic applications to be included in embedded devices. The graphical capabilities are currently available for VxWorks, OS-9, and Unix. Graphics for Windows CE is in progress and scheduled for release by the end of 2005. Thanks to its hard realtime garbage collector, JamaicaVM 2.8 enables the development of modern embedded applications that not only require a friendly and rich user interface, but also provide reliable and predictable behavior.

According to our partner NeoMore, the JamaicaVM

from aicas addresses all issues currently raised today in using Java in critical realtime applications. The high quality and innovative nature of JamaicaVM was the basis for their decision to become a reseller of the JamaicaVM.

The JamaicaVM is available for evaluation free of charge. Release 2.8 can be downloaded from the aicas web site at www.aicas.com.



↓ News

New Sales Partners

The aicas sales and support network is expanding with new partners in Switzerland and France.

NetModule has recently signed a marketing agreement with aicas to be a reseller for the JamaicaVM in **Switzerland**. NetModule is a technology company that provides Embedded Java Solutions. Their professional services range from electronic design to complete so-

lution, comprising operating system, Java Virtual Machine, and Java application. Their products include hardware platforms such as graphic terminals optimized for Java, the software framework JPC (Java for Process Control), and the JOPC-Bridge (Java interface to Microsoft's OPC).

Founded in 2003 by development tools experts, **NeoMore** brings to customers in **France** solutions for developing embedded and real-

time applications including software development tools, realtime operating systems, communication stacks, middleware, and hardware. The NeoMore team has 15 years of experience in embedded systems.

NetModule and NeoMore are valuable additions to the aicas worldwide sales and support network. More information about NetModule, NeoMore and our other partners can be found on our web page: www.aicas.com/resellers.html

Realtime CORBA for Realtime Java

A contribution from Victor Giddings and Joseph Jacob of Objective Interface Systems, Inc.

Introduction

The Real-Time Specification for Java (RTSJ), especially with deterministic garbage collection, provides a standard platform for realtime programming in Java, bringing Java's "Write Once, Run Everywhere" approach to realtime developers. An RTSJ implementation can deliver realtime systems requirements, such as the ability to schedule tasks, predict the performance of those tasks, and perhaps most importantly, predict how the system will react to an overload situation or resource failure. However, realtime systems are often composed of networked processors that also must provide realtime guarantees for communication. Realtime CORBA is the first standard middleware for realtime communication among processing nodes in a realtime system.

Realtime CORBA enables developers to provide end-to-end predictability throughout their system, while simultaneously providing location transparency in development, testing and deployment. Unlike the standard Common Object Request Broker Architecture (CORBA) and Java's Remote Method Invocation (RMI), Realtime CORBA offers additional features for managing the propagation of priorities between nodes. Priority propagation is necessary to ensure consistent task execution order over all processing nodes. The realtime CORBA specification adds this and other features to standard CORBA, including priority banded connections, thread pools, and mutexes.

Priority Propagation requires the priority of a thread making an invocation to be transmitted from the client to the server so the server may process the invocation at a consistent priority. This enables scheduling analysis to ensure that an application which meets realtime deadlines on one processor can be extended to a distributed system. Traditional realtime scheduling analysis recognizes the processor as the main shared resource that must be correctly scheduled in order to meet realtime deadlines. In a distributed system, the communications media and infrastructure is also a shared resource. In order

to allow this resource to be partitioned among processing at different priority levels, the Priority Banded Connections feature specifies the establishment of multiple transport connections and their assignment to different priority levels.

Threadpools offers greater control over the number and priority of threads used to process requests on a server. None of these features are effective unless the Object Request Broker (ORB) implementation is developed using realtime priorities. To keep implementation options open, no particular scheduling policy is required. However, a Mutex is provided by the ORB to enable applications to be built with a consistent policy.

ORBexpress RT for Java lets developers use CORBA to enhance deterministic performance of realtime Java applications in distributed environments. It is an implementation of the realtime CORBA Specification adopted by the Object Management Group. The interfaces and mechanisms provided by ORBexpress facilitate a predictable combination of ORB and application. The application manages the resources by using the ORBexpress interfaces and the ORBexpress mechanisms coordinate the activities that comprise the application.

ORBexpress RT is available in C++, Java, and Ada. Although each ORB shares a common architecture, they are not language bindings: each ORB is written entirely in its native language. This gives OIS the unique opportunity to test ORBexpress RT for Java and compare it with ORBexpress RT for other languages. As each ORB had a common architecture, it would be interesting to test performance and predictability across languages.

Attaining Realtime Behavior

An ORB can only be as good as the underlying system on which it runs. For example, if an ORB is run on an operating system that does not respect priorities, then the ORB can not have perfect priority-respecting behavior. A major test for any realtime Java Virtual Machine (JVM) is the predictability of its behavior in an application.

Over the years, Objective Interface has developed a test that ships with every license of ORBexpress. The test, called *rttaskdemo*, exercises the priority-respecting behavior of the operating system, JVM, ORB, stack, etc. This test, along with other internal tests, provides a benchmark for realtime Java performance. These tests show that realtime CORBA (such as ORBexpress) plus a realtime JVM (such as the JamaicaVM) provides realtime developers with a deterministic platform for

- highly portable realtime systems,
- predictable, realtime communication,
- a higher level of abstraction above the operating system, and
- interoperability with other RTOS and language environments.

One significant issue faced by realtime developers is Priority Inversion. When threads share resources, as is often the case, unexpected things can happen. A realtime JVM employs priority-based preemptive scheduling to manage these threads. The RT JVM assigns each thread a unique priority level. The scheduler ensures that, of those threads that are ready to run, the one with the highest priority is always the thread that actually runs first. To meet this goal, the scheduler can preempt a lower-priority thread, even if it is in the middle of its execution process.

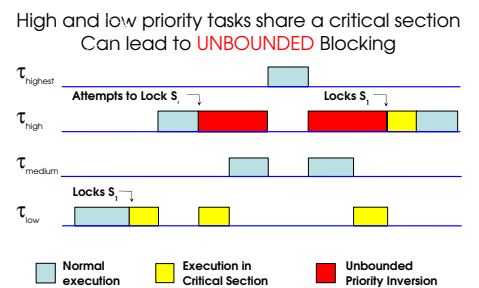


Figure 1: Example of priority inversion

Because threads share resources, events outside the scheduler's control can prevent the highest priority ready thread from running when it should. If this happens, a critical deadline could be missed, causing the system to fail as shown in Figure 1. Priority inversion is the term for a scenario in which the highest-priority ready thread fails to run

when it should. In essence, a low priority thread holds a shared resource that is required by the high priority thread, causing the high priority thread to be blocked until the low priority thread is finished with that resource. This "inverts" the relative priorities of the two threads.

The key to system predictability is bounding priority inversions. Bounded priority inversions can be accounted for in scheduling analysis. Unbounded priority inversions cannot be so bound.

Figure 2 shows an example of how a realtime ORB, by itself, will not provide robust predictable behavior without an underlying realtime foundation. This test, using *rttask-demo*, was run using ORBexpress RT for C++ on Windows XP. Each "dot" on the graph (the thick "lines" are actually a series of dots) represents the completion time of a thread performing a number of remote CORBA calls. These threads are started "simultaneously." Completion time is graphed versus priority. A perfect result on the test would be the highest level priority finishing before the next highest level priority has any resources allocated to it, with the lowest level priority (priority 0) not running at all until all of the other processes have finished.

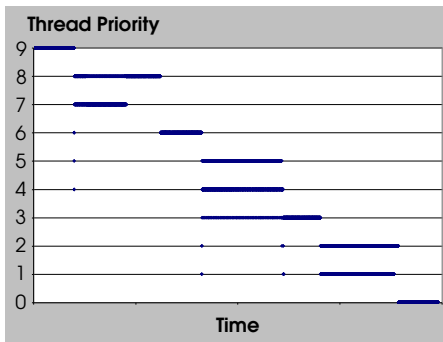


Figure 2: Realtime Corba for C++ on Windows XP

A realtime implementation of CORBA alone is also not sufficient to provide realtime guarantees. Figure 3 illustrates the inability of standard Java to ensure properly sequenced execution. The example shows ORBexpress RT running on a standard JVM. There are various flaws in this performance. Tasks run out of sequence, so that predictability is not achieved. The lack of realtime garbage collection results in the system behaving in unpredictable ways. This is not necessarily a problem for systems that do not have realtime requirements, but for

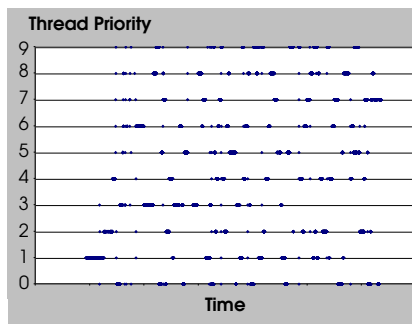


Figure 3: Realtime CORBA for Java with a standard JVM and an RTOS

a realtime developer, it renders the system useless.

ORBexpress RT and the JamaicaVM

Figure 4 shows the priority behavior of Realtime CORBA on a Realtime JVM and a realtime operating system. In this example, all tasks run in priority order. This behavior can be obtained both with *NoHeapRealtimeThreads* and with normal *RealtimeThreads* and a realtime garbage collector.

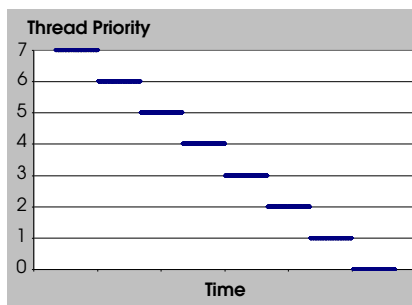


Figure 4: Realtime CORBA for Java with an RTSJ JVM and an RTOS

The one drawback of RTSJ with a standard garbage collector is that Java developers must worry about memory in a way similar that of C and C++ developers, who must concern themselves with managing memory explicitly. Only *NoHeapRealtimeThreads* threads can be used in such an implementation. This restricts the usage of memory in realtime tasks to the new memory regions defined in the RTSJ: *ScopedMemory* and *ImmortalMemory*. When a Runnable object is in one of these memory areas, all "new" operations allocate memory from that area, not from the heap. After the last Runnable object exits, memory in a *ScopedMemory* is "finalized" and deallocated; memory allocated in *ImmortalMemory* lives until the application ends.

RTSJ also presents significant challenges in the implementation of an ORB. These new memory features are tied to "threads of execu-

↓ Events

We would like to welcome you at the following events:

Embedded Technology 2005, Yokohama
16 - 18 November 2005

SPS 2005, Nuremberg
22 - 24 November 2005

Embedded World 2006, Nuremberg
14 - 16 February 2006

RTS Embedded Systems 2006, Paris
04 - 06 April 2006

Hannover Messe Industrie 2006, Hannover
24 - 28 April 2006

tion", not to objects. The ORB is implemented as a library; it is entered and left regularly in the course of execution. This makes maintaining a consistent internal state for the ORB difficult. Once the application has entered *ScopedMemory*, the ORB cannot save any information from the user or from an invocation without copying it into an ORB-controlled scope; otherwise it may cause scoping violations.

Java developers can avoid this requirement for memory management through a well implemented realtime garbage collector. This makes the JamaicaVM, with its very strong realtime garbage collection capabilities, ideally suited for developers wanting take full advantage of Realtime CORBA. Best of all, the Java ORB is quite fast, beating the performance of the Ada product and many open sourced C++ ORBs.

Summary

Realtime JVMs can provide hard, realtime, predictable behavior. This predictability can be extended to distributed systems with realtime CORBA. Using a realtime garbage collector is critical for avoiding the memory management requirements of RTSJ. The performance is competitive as well. These advances are already available using ORBexpress RT with the JamaicaVM.

Outlook

The next issue of the aicas news will present resource usage analysis for Java programs.

The GNU Classpath Project

aicas collaborates with other Java Technology providers

Introduction

A Java Virtual Machine would not be complete without a library of the standard classes. The Jamaica VM has its own core classes but also uses class from the open source project GNU classpath. GNU classpath is part of the GNU project, whose goal is to build a complete open source computation environment. Classpath aims to provide a fully compatible implementation of all Java standard class libraries. A major strength of Classpath is that one can combine any part with other libraries. This is how the internationalization of the CLDR project in 274 languages was possible.

It supports 274 languages as opposed to only 134 as currently available in Java from Sun Microsystems.

History

The Classpath project was founded in 1998 to create a free Java Standard Edition class library for a virtual machine named Japhar. At first, no thought was lent to VM portability. The initial team consisted of only five programmers.

Over time, the goal was extended to support other virtual machines as well. In 2000, the classpath from GCJ was integrated and more programmers joined. Three years later the project Kaffe showed interest and traded patches with the developers of GNU classpath. At about the same time, Classpath was integrated into the Jamaica VM as well.

Previously, the JamaicaVM class library had its own implementation of all its standard classes. Though some highly VM specific classes and support for realtime is still done in house, the vast majority of classes now come from GNU classpath.

The aicas team has become a significant contributor to the Classpath project. There is a regular exchange of updates between aicas and the classpath team. Within classpath, aicas programmers concentrate mainly on code size, platform independence, and graphics packages such as AWT and Swing.

The Players behind Classpath

The GNU classpath project is run by the Free Software Foundation. Currently, about twenty developers are active in the project. Approximately half of them are from the commercial bodies Red Hat and aicas, the other half is comprised of members of free software projects.

Legalities

GNU classpath is distributed under an open source licence from the Free Software Foundation. This licence is an extended form of the GNU Public Licence (GPL), which allows the incorporation of the Classpath libraries in commercial products. Other programs may use GNU classpath without having to release application source code.

Current Status

All classes from the core of J2SE 1.2 have been completed. Full compatibility with J2SE 1.4 is in progress. A large part of the core class libraries are already implemented and conform to the specification. The core of AWT is stable, but the widgets are not complete. The Swing implementation is based on AWT and many of the basic widgets work. Swing is already used by many projects and is advancing quickly to completion.

Future Development

A major part of the current development is concentrated on finishing the graphics toolkits AWT and Swing. Red Hat and aicas are the major contributors in this area. Work also continues on attaining full 1.4 compatibility. After that, the new features of Java 5 will be implemented.

The JamaicaVM and Classpath

Classes from the Classpath project have been incorporated in the Jamaica VM since version 2.0 released in 2002. The result is that the JamaicaVM has a full 1.2 implementation and is largely compatible with JDK 1.4. Due to the special requirements of realtime systems, Jamaica uses a completely separate implementation of *java.lang* as well as a series of modifications in *java.io*, *java.net*, and *java.util*. The aicas team not only uses and contributes to the Classpath Project, but also works on its development. Before Classpath code is released with the JamaicaVM, it is extensively tested in house ensuring the integrity of aicas' JVM products. Whereas Classpath focuses on desktop systems, aicas concentrates on the needs of embedded and realtime systems.

↓ **New Product**

Safety Critical Java

The aicas team is proud to announce a new product for safety critical systems. The first implementation of the Safety-Critical Java profile defined by the HJA project will be available from aicas. A certifiable version of the JamaicaVM will be available for using Java technology in safety-critical applications requiring certification such as DO-178B up to the highest criticality level.

The Safety Critical JamaicaVM will provide a limited subset of the Realtime Specification for Java and standard classes without difficult to certify features such as garbage collection.

A set of tools for static correctness analysis for safety critical Java applications will be provided. These tools include resource analysis (heap and stack usage), correctness of the region-based memory management provided by the SC-Java, Worst-Case Execution Time analysis, and absence of Java runtime errors such as null pointer uses or class cast exceptions.

Early prereleases of safety critical version will be available on selected platforms by the end of this year with a full release with JamaicaVM 3.0 in the spring.

↓ **Contact**

aicas GmbH
 Haid-und-Neu-Straße 18
 D-76131 Karlsruhe
 Germany

tel. +49 721 663 968-0
 fax +49 721 663 968-99
 email info@aicas.com
 web www.aicas.com