Dr. Kaneff Engineering Consultants

EUROS*vm*

Enhanced Universal Realtime Operating System

EUROS Virtual Machine

Issue 02/2000

EUROS*vm* - The Key to Java

EUROS*vm* is an implementation of the *Java Virtual Machine Specification V1.2*. It has been designed for real-time and embedded systems and offers unparalleled support for these target domain. Among the extraordinary features of EUROS*vm* are

- Hard real-time execution
- Minimal footprint
- ROMizable code
- Native code support
- Dynamic Linking
- Portability
- Fast execution
- Powerful Tools

Hard Real-Time Execution

The EUROS*vm* provides hard real-time guarantees for all features of the languages. This includes dynamic memory management, which is performed by the garbage collector. All threads executed by the EUROS*vm* are real-time threads, there is no need to distinguish real-time from non-real-time threads. Any higher priority thread is guaranteed to be able to pre-empt lower priority threads within a fixed worst-case delay. No language restrictions for real-time code: Since all code is real-time code, even real-time tasks can use the full Java language, i.e., allocate objects, call library functions, etc. No special care is needed, short worst-case execution delays can be given for any code.

Minimal footprint

The Virtual Machine itself occupies just about 120kB of memory (depending on the target platform). The biggest part of the memory required to store a Java applications is typically the space needed for the application's class files. Several measures are taken to minimize the memory needed for Java classes:

- Compaction: Classes are represented in an efficient and compact format to reduce the overall size of the application.
- Smart Linking: The EUROS*vm* analyses the Java application to detect and remove any code and data that cannot be accessed at run-time.

The compaction and smart linking process can be controlled by the user to ensure optimal performance. Compaction typically reduces the size of applications by over 50%, while smart linking allows for much higher gains of well over 90% even for non-trivial applications!

ROMizable code

The EUROS*vm* allows class files to be linked with the Virtual Machine code into a standalone executable. This allows romization since all files required by a Java application are packed into this executable that

can be loaded into flash-memory or burned into ROM. There is no need for file-systems support on the target platform, all data required for execution is contained in the romized application.

Native code support

The EUROS*vm* implements the *Java Native Interface V1.2*. This allows for direct embedding of existing native code into your Java applications, or to encode hardware-accesses and performance-critical code sections in C or machine code routines. The usage of the Java Native Interface provides execution security even with the presence of native code, while binary compatibility with other Java implementations is ensured.

Unlike other implementations, EUROS*vm* provides exact garbage collection even with the presence of native code. Real-time guarantees for the Java codes are not affected by the presence of native code.

Dynamic Linking

One of the most outstanding features of Java is the ability to dynamically load code in the form of classes during execution, be it from local files or from a remote server. The EUROS*vm* support this dynamic class loading enabling the full power of Java for your applications. Any softwarecomponent can be loaded dynamically, allowing on-the-fly reconfiguration, hot swapping of code, dynamic additions of new features, applet execution, ...

Fast Execution

The Interpreter performs several selected optimizations to ensure optimal performance of the Java code. Current implementations of Java use just-in-time compilation technologies that are not applicable in real-time systems: The initial delay for compilation is breaking all real-time constraints.

The Compilation Technology attacks the performance in a new way:

First, methods and classes can selectively be compiled as a part of the Romization process. C-code is used as target code, allowing easy porting to different target platforms.

Second, classes that are loaded dynamically can be compiled at class-load-time. Since class-loading is not a real-time operation, the compilation does not break real-time constraints as a Just-In-Time compiler-does. Nevertheless, fully optimised compiler code can be generated even for any code that is dynamically loaded at run-time.

The Compiler is tightly integrated with the memory management system, allowing for highest performance and reliable real-time behaviour. No conservative reference finding code is required, allowing fully exact and predictable garbage collection.

Powerful Tools

The EUROS*vm* comes with a set of tools that support the development for real-time and embedded systems:

- Romizer: A tool for creating a single executable image out of the Virtual Machine and a set of Java classes. This image can be loaded into flash-memory or ROM, avoiding the need for a file-system in the target platform.
- Memory Analyser: For most efficient memory usage, this tool finds the amount of memory that is actually used by an application.
- Profiler: The profiler can be used to determine an application's hot spots, so that they can be optimised further. The information obtained from the profiler can also be used to guide the compiler, so that compilation is restricted to the most important methods, reducing the memory overhead of compiled code.

Dr. Kaneff Engineering Consultants
Neutorgraben 17
D-90419 Nürnberg, Germany
Tel. +49-911-33 84 33
Fax +49-911-33 86 06
e-Mail: info@kaneff.de
Web: http://www.kaneff.de