

# REAL-TIME JAVA: AN EXPERIENCE REPORT

**Marek Prochazka**

*SciSys  
Clothier Road, Bristol,  
BS4 5SS, United Kingdom  
Tel +44 (0) 117 971 7251  
Fax +44 (0) 117 971 1125  
[marek.prochazka@scisys.co.uk](mailto:marek.prochazka@scisys.co.uk)*

**Jan Vitek**

*Purdue University  
250 N. University Street, West Lafayette (IN),  
47907-2066, USA  
Tel +1 (0) 765 494 6531  
Fax +1 (0) 765 494 0739  
[jv@cs.purdue.edu](mailto:jv@cs.purdue.edu)*

## **Abstract**

The Real-Time Specification for Java was released in 2000 and has continued to generate sustained research and development efforts, having a number of implementations being provided by both software vendors and universities, and most importantly being used in several avionics- or space-related projects. This paper documents our experience with efforts made to improve the performance and predictability of Real-Time Java implementations, its recent successful use in avionics software and its use to advance research in real-time computing. The experience suggests that Real-Time Java has recently achieved significant developments and becomes a viable technology to be used for future avionics and space systems.

## **1 Introduction**

The Real-Time Specification for Java (RTSJ) [2] provides real-time systems programmers with a modern, type-safe programming environment. Its features such as memory safety, checked exceptions, and a rigorously defined memory model, make Java a good programming language for developing mission critical applications. The RTSJ has been evaluated for use in avionics and space systems by Boeing ([1], [3]) and JPL [7]. It has two commercial implementations ([9], [10]), as well as a number of open source implementations (e.g. [5], [8]) and variants.

In the rest of the paper we summarise the authors' experience with the RTSJ in three domains:

- benchmarking of various RTSJ implementations,
- implementing and testing avionics software in the RTSJ,
- conducting real-time systems research with the RTSJ.

## **2 Benchmarking Real-Time Java**

A significant effort has been spent around the world to provide quality RTSJ benchmarking to improve the performance and predictability of existing RTSJ implementations and give valuable feedback to both the authors of the specification and

its implementations designers. One of the benchmarking projects was RTBench, conducted as a part of the DARPA-funded Ovm project [8] and led by the authors of this paper.

RTBench [6] is a benchmarking framework for benchmarking programs written in Real-Time Java. RTJBench extends the standard JUnit framework for unit testing of Java applications with tools for real-time environment configuration, simple data processing and configurable graphical presentation services.

RTJBench has been used for daily automatic regression benchmarking of Ovm. Two different RTJBench benchmarking suites were created, one for latency micro-benchmarking and another for benchmarking Ovm performance using the SPEC JVM98 benchmarks. The same suite was used for benchmarking on three different platforms (Timesys Linux RTOS, Mac OSX, and Embedded Linux BSP on an embedded board). Based on one execution of the regression benchmarking suite, a number of valuable graphs was daily generated automatically during the development of Ovm. Examples of generated graphs are in Figure 1 and Figure 2.<sup>1</sup>

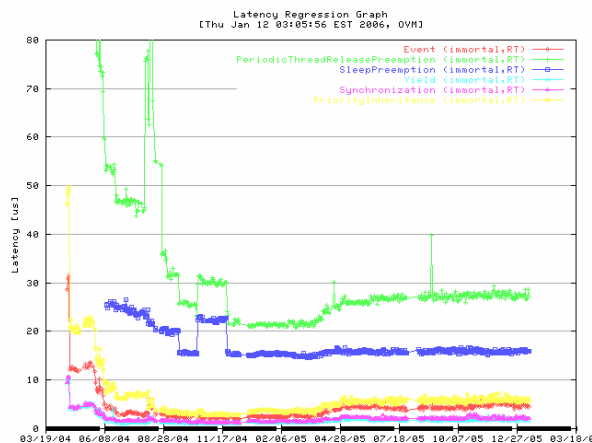


Figure 1: Latency regression graph

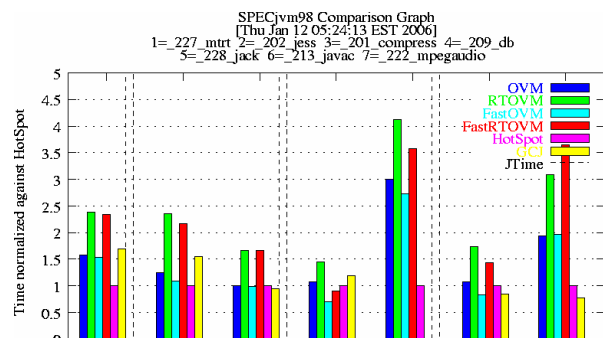


Figure 2: Comparison of various RTSJ VMs using the SPEC JVM98 benchmark

As a part of the benchmarking support, a performance comparison of the RTSJ and C++ was performed using the Linear Parameter Varying (LPV) flight control algorithm with heavy use of floating point operations. Figure 3 shows that the performance of the Java version of the algorithm running on the RTSJ Virtual Machine used (Ovm) is about 40% of the C++ version performance. A commercial Java VM (HotSpot) is about 80% of the C++ performance.

<sup>1</sup> All results are available at <http://ovmj.org/bench>.

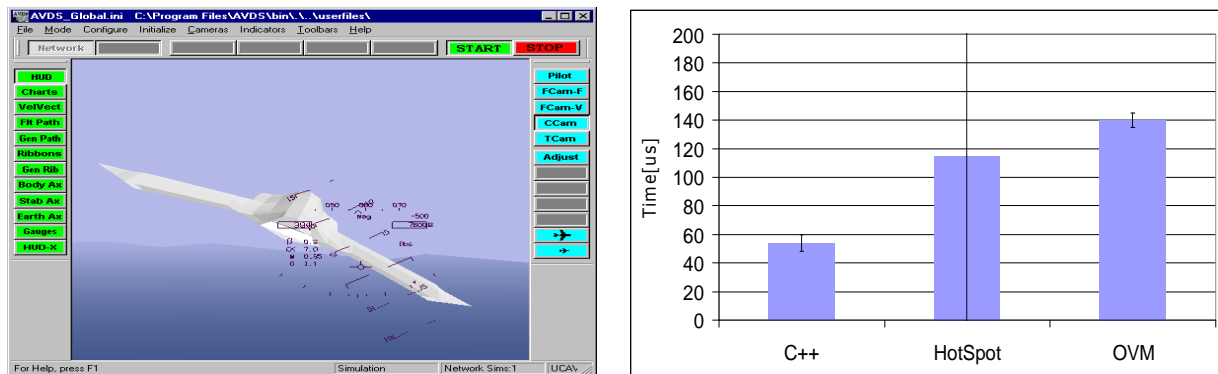


Figure 3: LPV flight control algorithm benchmark by C++, HotSpot and Ovm

The conclusions of the benchmarking efforts presented above are as follows:

- A significant effort has been invested to have good benchmarking data about RTSJ implementations and to reason about their performance and predictability.
- RTSJ implementations can exhibit low latency and jitter values (less than 20 microseconds).
- RTSJ implementations can exhibit performance comparable with C++.

### 3 Real-Time Java in Avionics Applications

Real-Time Java was recently used for autonomous navigation of an unmanned aerial vehicle (UAV) [3]. In 2005, Ovm passed flight qualification and was used in the DARPA PCES programme for its final Capstone Demonstration at the White Sands Missile Range. The programme integrated several projects into a live demonstration of their combined functionality, using both real and simulated components. As a part of the demonstration, Boeing and Purdue University demonstrated autonomous navigation capabilities on an UAV known as the ScanEagle (Figure 4). The ScanEagle is a low-cost, long-endurance UAV developed by Boeing and the Insitu Group. This UAV is four-foot long, has a 10-foot wingspan, and can remain in the air for more than 15 hours. The primary operational use ScanEagle is to provide intelligence, surveillance and reconnaissance data. The ScanEagle software was developed using the Boeing Open Experiment Platform (OEP) and associated development tool set. The OEP provides a number of different run-time product scenarios which illustrate various combinations of component interaction patterns found in actual Bold Stroke avionics systems.

The ScanEagle using the Ovm was designated as the Reconnaissance UAV. Its main function was surveillance of real-time targets during the demo mission. The flight product scenario was responsible for providing autonomous auto-routing and health monitoring by (1) communicating with the flight controls card, (2) computing navigational cues for the flight controls based on threats and no fly zone data from the

ground station, and (3) computing performance monitoring information to be transmitted to the ground station for real-time observation of jitter and priority processing.



Figure 4: ScanEagle

This milestone marked the first flight using the RTSJ on an UAV and received the Java 2005 Duke's Choice Award for innovation in Java technology.

#### 4 Preemptible Atomic Regions for Real-Time Java

In [4], a new concurrency control abstraction for real-time systems called *preemptible atomic regions* (PARs) and its implementation in a RTSJ-compliant Java Virtual Machine were presented. In traditional synchronisation protocols avoiding unbounded priority inversion, the blocking time of a high-priority task is bounded by the worst case execution time (WCET) of the low-priority task needed to execute a critical section protected by a shared lock. PARs provide a transactional mechanism which avoids blocking of high-priority transactions and instead aborts conflicting low-priority ones. Each abort effectively means restoring original values of data modified and hence the worst case blocking time is proportional to the number of write operations in the critical section. This is, in general, significantly easier to predict and calculate than the WCET of critical sections. The evaluation of the PARs implementation in the Ovm running on a uniprocessor suggests that programs that use PARs, depending on their semantics<sup>2</sup>, can run faster and experience less jitter than those that use locks.

#### 5 Conclusions

This paper summarises recent advances of the Real-Time Java technology in terms of performance, predictability, the ability to be used in avionics software and to address nontrivial real-time computing problems. The conclusion of the authors is that Real-

---

<sup>2</sup> The avionics software described in Section 3 was amongst programs used for PARs testing in Ovm.

Time Java has already achieved significant accomplishments and becomes a viable alternative for the development of avionics and space systems in the future.

### **Acknowledgements**

The authors thank Jason Baker, Filip Pizlo, Antonio Cunei, Chapman Flack, Jeremy Manson and Wenchang Liu from Purdue University, USA, as well as David Holmes and Andrey Madan, who worked in the Ovm project or helped with the RTJBench framework. Thanks to Ed Pla from Boeing for providing the LPV flight control algorithm benchmark.

### **References**

- [1] David Sharp, "Real-time distributed object computing: Ready for mission-critical embedded system applications", in Proceeding of the 3<sup>rd</sup> International Symposium on Distributed-Objects and Applications (DOA'01), 2001.
- [2] Greg Bollella, James Gosling, Benjamin Brosgol, Peter Dibble, Steve Furr, and Mark Turnbull, "The Real-Time Specification for Java", Addison-Wesley, 2000.
- [3] Jason Baker, Antonio Cunei, Chapman Flack, Filip Pizlo, Marek Prochazka, Jan Vitek, "Real-Time Java in Avionics Applications", accepted for presentation at the 12<sup>th</sup> IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '06), San Jose, California, USA, April 4 - April 7, 2006.
- [4] Jeremy Manson, Jason Baker, Toni Cunei, Suresh Jagannathan, Marek Prochazka, Bin Xin, Jan Vitek, "Preemptible Atomic Regions for Real-time Java", in Proceedings of the 26<sup>th</sup> IEEE International Real-Time Systems Symposium (RTSS '05), Miami, Florida, USA, December 2005.
- [5] jRate, <http://jrate.sourceforge.net>.
- [6] Marek Prochazka, Andrey Madan, Jan Vitek, Wenchang Liu, "RTJBench: A Real-Time Java Benchmarking Framework", Component And Middleware Performance Workshop, the 19<sup>th</sup> Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004), Vancouver, British Columbia, Canada, October 2004.
- [7] NASA/JPL and Sun, Project Golden Gate, <http://research.sun.com/projects/goldengate>.
- [8] Ovm, <http://ovmj.org>.
- [9] Sun Microsystems, Project Mackinac, <http://research.sun.com/projects/mackinac>.
- [10] Timesys Inc., jTime, <http://www.timesys.com>.