

High-performance, Space-efficient, Automated Object Locking

Laurent Daynes and Grzegorz Czajkowski

(17th IEEE International Conference on Data Engineering, Heidelberg, Germany)

Introduction by Laurent Daynes and Grzegorz Czajkowski

The technology described in the paper published in the 17th IEEE International Conference on Data Engineering in April 2001 was developed in the context of the Forest project. The two authors were then working on supplementing a virtual machine that implemented Orthogonal Persistence for the Java™ platform (OPJ) with the ability to safely execute simultaneously multiple applications. The approach chosen was dictated by the requirement to enable safe sharing and direct access to objects by multiple applications. Guaranteeing isolation between applications while allowing them to safely share data requires mechanisms akin to those used in database systems. To this end, the OPJ virtual machine was augmented with very light-weight transactional mechanisms that automatically enforces "serializability", thus giving each application the illusion of running stand-alone while actually sharing, potentially concurrently, data with other applications. The principles for this approach were actually elaborated with the very first design of OPJ [1]. The main obstacle to realizing this transaction-based approach to application isolation was the absence of compelling technology to support automated concurrency with performance acceptable for the runtime of a modern programming language [2]. The paper describes in detail a patented solution that address this problem.

Although the work on efficient automated object locking has enjoyed academic success (the paper received the Best Paper Award), the development of a first prototype of transaction-based protection in the context of the Java platform has exposed many problems that made clear that much more research was needed before the approach reached maturity. The pressing need for scalable execution of many applications has resulted in departing from safe sharing and focusing on the simpler case of strict isolation between applications. Under the lead of G. Czajkowski, the two authors are now pursuing this direction to design scalable multi-tasking virtual machines (MVM) for the Java programming language.

The work on MVM grew out of an initial prototype based on bytecode editing that was developed as a proof of concept to lay out the basic principle of a new approach to scalable application isolation [3]. Since then, a more ambitious prototype based on the Java HotSpot™ Virtual Machine has been developed [4]. At the same time, solutions to isolate a JVM™ from unwanted interactions with, and errors from user-defined native methods have also been developed [5]. Both efforts have played a key role in the proposal of the JSR 121, <http://jcp.org/jsr/detail/121.jsp>.

"High-Performance, Space-efficient, Automated Object Locking." © 2001 Sun Microsystems, Inc. and IEEE. Reprinted, with permission, from Seventeenth IEEE International Conference on Data Engineering, Heidelberg, Germany, 2-6 April 2001.

PUBLICATIONS:

1. M. Atkinson, M. Jordan, L. Daynes and S. Spence, "Design Issues for Persistent Java: a type-safe, object-oriented, orthogonally persistent system", Proceedings of the 7th International Workshop on Persistent Objects Systems, Cape May, NJ, May 1996.
2. L. Daynes, "Implementation of Automated Fine-Granularity Locking in a Persistent Programming Language", *Software – Practice and Experience*, 30:1–37, 2000.
3. G. Czajkowski, "Application Isolation in the Java Virtual Machine", Proceedings of ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, Minneapolis, MN, October 2000.
4. G. Czajkowski, L. Daynes, "Multi-tasking without compromise: a Virtual Machine Approach", Proceedings of Object-Oriented Programming, Systems, Languages and Applications, October 2001, Tampa, FL.
5. G. Czajkowski, L. Daynes, M. Wolczko, "Automated and Portable Native Code Isolation", Proceedings of the 12th IEEE International Symposium on Software Reliability Engineering, Hong-Kong, November 2001.