IBM Research Report

The Jikes RVM Project:
Building an Open Source Research Community

B. Alpern[1], S. Augart[2], S. M. Blackburn[3], M. Butrico[4], A. Cocchi[5], P. Cheng[1], J. Dolby[1], S. Fink[1], D. Grove[1], M. Hind[1], K. S. McKinley[6], M. Mergen[4], J. E. B. Moss[7], T. Ngo[8], V. Sarkar[1], M. Trapp[9]

[1]IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598

[2]16 Brooks Street, Winchester, MA 01890

[3]Australian National University, Canberra, ACT 0200, Australia

[4]IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598

[5]Lehman College, The City University of New York, 250 Bedford Park Blvd. West, Bronx, NY 10468

[6]The University of Texas at Austin, 1 University Station #C0500, Austin, TX 78712

[7]University of Massachusetts, 140 Governor's Drive, Amherst, MA 01003-9264

[8]IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120

[9]100world AG, Vordere Cramergasse 11, D-90478, Nürnberg, Germany

IBM

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

The Jikes RVM Project:
Building an Open Source Research Community

B. Alpern   S. Augart   S.M. Blackburn   M. Butrico   A. Cocchi   P. Cheng
J. Dolby   S. Fink   D. Grove   M. Hind   K.S. McKinley   M. Mergen
J.E.B. Moss   T. Ngo   V. Sarkar   M. Trapp

December 9, 2004

Abstract

This article describes the evolution of the Jikes RVM project from IBM internal research infrastructure, called Jalapeño, into an open source project. After summarizing the original goals of the project, we discuss the motivation for releasing it as an open source project, and the activities performed to ensure the success of the project. Throughout, we highlight the unique challenges of developing and maintaining an open source project designed specifically to support a research community.

1   Introduction

On October 15, 2001, IBM™ Research launched the Jikes™ RVM (Research Virtual Machine) open source project. Jikes RVM provides novel virtual machine software infrastructure, suitable for research on modern programming language design and implementation techniques. Over the past three years, the project has grown and made a significant impact on the programming language research community.

This article describes the evolution of the Jikes RVM project from an IBM internal research infrastructure, called Jalapeño, into a full-fledged open source project. The story provides an instructive case study on how a small systems research project can grow into a shared project used by hundreds of researchers. The article discusses a variety of challenges that arose in this process, including

- technical enhancements and software engineering practices needed to ensure the project's success,

- dealing with intellectual property, corporate process, and licensing issues,

- promoting the system in the research community, and

- developing a community to maintain and enhance the system in the future.

The Jikes RVM project develops, as its primary focus, a *white box* software platform designed as a testbed to prototype new technologies. In contrast, most open source projects develop software products for use by the general public instead of use as a research infrastructure. We focus on the implications of this distinction throughout the paper.

The remainder of this paper proceeds as follows. Section 2 begins with a general discussion of issues pertaining to a research-oriented open source project. Section 3 discusses the project's childhood, as an IBM internal project. Sections 4 and 5 review an adolescence during which the software was made available to a small set of universities under a restricted license. Sections 6 and 7 discuss the preparation for, emergence,

1

and first few years of Jikes RVM as a full-fledged, mature open source project. Section 8 presents some metrics that document the system's impact, and Section 9 concludes with some lessons learned from this case study.

2   Open Source for Research

The majority of open source projects develop software that is primarily used as a *black box*. Community members adopt the software as a tool to perform some function, but most users often do not care about internal design and implementation of the software. Examples of such software tools include most of the best-known open source projects, including operating systems [1, 2], word processors [3, 4], integrated development environments [5], compilers [6], and graphical desktop environments [7, 8].

In contrast, the Jikes RVM project develops a piece of software that is intended to be used as a *white box* that enables research. The overwhelming majority of Jikes RVM users modify the system in a nontrivial way to produce interesting research results.

This research focus has major implications on the character of the open source project. In particular,

- Most of our community are professors and graduate students, using the system to advance an individual research agenda.

  This community has different motivations than most open source software contributors; they are primarily driven by the desire to produce publications and technical results, and not primarily by a desire to produce production-quality software. However, it is desirable for the research infrastructure to be as close to production-quality as possible to strengthen the credibility of their research.

  A related consequence is that our user community has less motivation to add functionality. We believe that many open source developers are initially motivated to contribute to a (black box) project by a desire to resolve some irritating deficiency with the system. For example, perhaps a user wants to use their favorite digital camera with an open source photo editor; this could motivate the user to write and contribute a device driver. In contrast, our user community generally seems content with the functionality the system provides, which is sufficient for producing high-quality research results on common benchmark programs. If particular functionality is missing or broken, most users can work around the problem and still achieve their individual goals.

- Open source licenses obligations, including source code availability, do not apply to research results.

  Many open source projects effectively enforce community sharing with their licenses, such as the GPL and CPL. These licenses, in differing ways, require that recipients who create and distribute derivative works of open source software must make source code to their derivative works available to their recipients. However, these licenses do not require source code to be made available solely when a paper is published about a system because no distribution of the derived system is involved. Furthermore, many academics tend to keep research infrastructure private as a competitive advantage. In any case, the majority of researchers using Jikes RVM choose not to make their code available. We present an argument against this approach in Section 9.

- The research community will accept less polished software than the general public.

  Jikes RVM provides a critical piece of infrastructure to many academic groups that is crucial to their project's success. Because of its perceived importance, the community will generally tolerate a great deal of pain in learning and installing the system in exchange for a system that is easier to change and modify. Although Jikes RVM is fairly robust and well-tested compared to most research projects, the initial user experience is, for various reasons, less pleasant than that using commercial "black-box" software such as a product JVM.

- Our user community requires extensive documentation on the inner workings of the system, not just the exposed command-line or application programming interfaces.

  Documentation and explanations of the inner workings of the system are crucial, and a key part of the project's deliverables. Furthermore, changes in internal structures can cause disruption among the user community, and must be managed carefully.

3 Motivation and History of the Jalapeño Project (1997-2000)

Sun™ Microsystems introduced the Java™ programming language in May 1995. The Java programming language offered some significant technical advantages over previous commercial programming languages, including a portable program representation and some safety guarantees. To support these features, the language runs on a *virtual machine*, a software execution engine that provides a managed runtime environment for executing Java programs.

The language quickly gained popularity and was widely endorsed by many industry players, including IBM. To improve performance, the industry, in general, and IBM, in particular, began significant investments into virtual machine technology.

In November 1997, a small group of researchers at IBM's T.J. Watson Research Center began the Jalapeño project, whose goal was to develop internal research infrastructure for VM technologies. The virtual machine, known as Jalapeño, was targeted as a flexible, extensible testbed to research, prototype, and evaluate VM implementation techniques. In contrast to other IBM JVMs, Jalapeño did not serve any immediate product deliverables, and so was unencumbered by production-strength process and quality requirements.

Many papers document various technical aspects of the system [9, 10, 11, 12, 13]. Some technical highlights include

- the virtual machine is implemented mostly in the Java programming language,

- an m-to-n quasi-preemptive thread system targeting server applications,

- an aggressive optimizing compiler,

- a state-of-the-art adaptive optimization system, and

- a family of high performance garbage collectors.

The initial implementation ran on PowerPC™ processors running the AIX operating system.

Early on, Jalapeño researchers decided to develop the system in a "clean-room" manner, i.e., the developers would not have access to any non-IBM source code for the JVM and libraries. The Jalapeño VM was written entirely from scratch, and used IBM-developed libraries from the OTI lab. This decision maximized flexibility for future evolution of the project, by avoiding potential intellectual property and copyright issues.

During 1998, the project team grew into two separate research groups: the "runtime" group and the "optimizing compiler" group. The runtime group focused on core VM functionality, such as threading, the baseline compiler, the garbage collectors, JNI, and the library interface. The optimizing compiler group focused on the optimizing compiler and adaptive optimization system. At its peak the project included approximately 15 full-time employees, plus a large number of academic visitors and PhD students.

Most early Jalapeño researchers joined the project because it seemed interesting, or seemed to be a useful vehicle to pursue a particular research idea. In some ways, the Jalapeño team resembled an open source community in that volunteerism was the main driver behind work assignments and milestones. The initial focus of the project was to simply improve the state-of-the-art in virtual machine technology. The main products of the early Jalapeño project were research papers. Much of the initial research was performed on private versions of the system, which were never merged back into the main repository.

When the project started, the researchers did not foresee that this system would be widely used, let alone open sourced. This influenced the initial development practices and methodologies.

Initially, the team shared source code with a centralized `rcs` [14] repository and some home-grown scripts that provided higher-level functionality. The project consciously adopted a *prototype-first* development methodology. Much code was written in a "throw-away" style under pressure of impending paper deadlines, with all intentions of complete rewrites at the earliest opportunity. Some of this "throw-away" code persists in the system to this day.

Like many "research" projects, the software engineering practices applied to the early Jalapeño code base were haphazard, when compared to a mature product-quality development process. Source code formatting practices were nonstandard, and the volume and quality of comments varied from acceptable to nonexistent, depending on the inclination of individual coders. Initially, there were no project-wide code reviews, no official documentation, no formal bug tracking system, no unit tests, no quality assurance process, and no regular regression testing. Most Jalapeño team members recognized the value of these processes, and some subteams followed some of these processes by agreeing on small sets of test cases, and using CMVC for bug tracking. However, these processes were only introduced on a project-wide basis after the pain of omission became intolerable.

Between 1997 and 1999, the number of implementers, i.e., committers to the code base, increased from a small handful to nearly 20. Functional regressions became increasingly unbearable, until in the summer of 1999, the project finally instituted automated nightly regression testing. This procedure consisted of running several benchmarks and tests on several configurations of the system, varying policies for garbage collection, optimization, and assertion checking. Performance regressions were also tracked using the same system. Results were archived and sent to a project mailing list. As the project evolved, the position of *night sanity guru* was created. This position, rotated among the team on a weekly basis, was responsible for diagnosing and summarizing the regression results for the rest of the team. Combined with `rcs` check-in logs, project members were able to reconstruct when and why a configuration of the system was first broken. Over time, the frequency of nightly regressions gradually decreased, as the system matured and the social taboo against breaking the overnight tests took root.

4   The University Releases

During the course of the Jalapeño project, its members published results in various research forums. This publicity raised awareness of the system in the academic community, and some university professors inquired about the system's availability as research infrastructure for their own projects; the first inquiry came from researchers at the University of Massachusetts, Amherst.

Although the original project goals did not include distribution to the academic community, the project team members were excited about the potential for other researchers using the system. Thus, in early 2000 the team agreed to pursue making the system available to universities under a source license agreement. (Although a "binary-only" version of the system was made available to one university around this time it was not a useful solution to most researchers because of its black box nature.) This decision spawned two nontrivial parallel activity tracks: bureaucratic and technical.

The bureaucratic tasks included securing approval from upper management to release the code, deciding on an appropriate legal agreement and licensing structure, documenting origins of all code to be released, and resolving issues regarding IBM intellectual property embodied in the code. Although these tasks consumed some time, the team leaders resolved these issues relatively quickly with appropriate support from the contract and legal departments.

The first technical activity was to port the system to Linux™/PowerPC, the platform that the initial university users would use. This was performed by two collaborators at the University of Massachusetts,

Amherst and the Australian National University, working as part-time IBM employees, and was completed in the first half of 2000.

The other major technical issue for the university release concerned developing a process to release tagged versions of the code base to our university partners. Previously, all code had lived in a `rcs` repository and the current system was simply the head of the `rcs` tree. Making releases available to universities required dealing with the problem of development continuing on the main trunk, while maintaining branches to identify bugs in tagged releases and merging changes back to the main trunk when appropriate. The extant home-grown `rcs` scripts did not support these activities well.

To address these issues, we committed to migrate to a `cvs` [15] source code repository, which better supports branching and merging. Many team members had not previously used `cvs`, so the main issue was education of the team, a task handled by a few team members with relevant expertise. Using `rcs`, the project had relied on pessimistic concurrency control of source code using `rcs` locks. There was some concern that the `cvs` optimistic concurrency control would lead to excessive merge conflicts; in retrospect, this was not a serious problem, and in fact, occasional merge conflicts were a small price to pay in exchange for eliminating e-mail and telephone negotiations over `rcs` locks. The other discussions concerned the discontinuity of the "history" in the source code system; team members were forced to consult a (frozen) copy of the old repository to access old file histories.

The actual migration was performed over a few days by a small set of team members during an enforced code freeze. During the migration, the source code directory structure was reorganized.

Other important technical activities included improving the documentation of the system. A small set of team members wrote an initial user's guide, which was particularly important because of this system's complexity and non-standard build process. Some team members endeavored to introduce Javadoc™ tags for all methods, and to specify and conform to uniform coding conventions. These last two activities were particularly painful to perform on a large code base. We employed tools to help with these activities and made significant improvements in large swathes of the system; however, coverage varied considerably across subsystems.

The university release also motivated a long-overdue general code cleanup. In particular, the code base had been successful in serving as a research vehicle for several projects at Watson [16, 17, 18, 19] which never graduated to "first-class" support in the Jalapeño implementation. To enable sharing at Watson, these projects had incorporated their code into the main `rcs` tree. Since development on these projects was not actively tested, the corresponding code was removed before making the university release.

The first university release was made on January 23, 2001 to the following universities: University of Massachusetts, Amherst; University of Colorado; Kent University; Purdue University; University of Wisconsin; and Rutgers University. During the subsequent ten months, several other universities expressed interest in the system. Each university had to sign a licensing/nondisclosure agreement, which often entailed several months of correspondence between IBM and university lawyers. By Oct 15, 2001, 16 universities had completed the license, 6 others were in process, and 10 other universities had inquired about obtaining a license.

5 University Activities

The growing enthusiasm of university researchers quickly led to the first contributions to Jalapeño from outside IBM. These contributions tended to fit into two broad categories: extending Jalapeño's reach through ports to other platforms, and extending its utility as a research platform.

The desire to port Jalapeño has been driven by both the practical concern of utilizing commonly available platforms and the research motivation of exploring novel platforms. It was the first of these that motivated the initial port to Linux/PowerPC and more recently a port to MacOS X by a developer at the University of

Massachusetts. This also drove the port to Linux/IA32 by the team at IBM, which is described in more detail in Section 6. The desire for a 64-bit research platform encouraged researchers at a number of universities to pursue a port to the 64-bit PowerPC, resulting in a functional 64-bit port for both Linux and AIX. Among other things this port is the basis for research on memory management in large address spaces.

In the spirit of a community project, the universities took a major role in maintaining and testing a growing set of ports. The initial port to Linux/PowerPC was maintained and tested at the University of Massachusetts each time a university release was made. This distribution of effort was originally simply because of the lack of a Linux/PowerPC platform within the IBM team, but subsequently became a way of dealing with the expanding set of platforms. Over time, testing moved to a system of distributed nightly regression tests, with different sites taking responsibility for certain platforms and results being sent to the `jikesrvm-regression` mailing list. This was an important step in the evolution to an open source project as responsibility was distributed, and developers were more immediately aware of the impact of their changes on a diverse user base.

The other major source of contributions was in the area of extending the utility of Jalapeño as a research platform. Researchers at the University of Massachusetts were particularly interested in using Jalapeño as a vehicle for memory management research. The performance of the optimizing compiler, the robustness of the system, and Jalapeño's growing credibility within the programming languages community were all important. Although the system came with a number of high performance, highly scalable garbage collectors [13], their monolithic implementation made them unattractive as the basis for memory management research.

In October 2000, researchers at the University of Massachusetts set about developing GCTk, a Garbage Collection Toolkit. Because this work was commenced prior to (but in anticipation of) the university release of Jalapeño, the implementation was performed by a University of Massachusetts researcher working as an IBM employee. GCTk was developed from scratch as a flexible toolkit for research on garbage collection and was a plug-in replacement for the collectors that shipped with Jalapeño. From the outset, the toolkit was intended to be open and once Jalapeño was licensed and available, GCTk was used by a number of other universities.

The project provided insight into some of the more subtle and complex issues associated with user contributions that would arise later as Jalapeño was open sourced. For example, in the absence of write access to the IBM repository there was a constant problem of keeping GCTk up to date with respect to the IBM repository and vice versa.

Once GCTk was functional it became the basis for projects at other institutions. These generated a series of publications, including one of the first Jalapeño publications based entirely on research done outside of IBM. This began a pattern of Jalapeño-based research at universities that continues (discussed further in Section 8) and is increasingly productive.

GCTk had a number of shortcomings, including a lack of support for noncopying collectors and free list allocators. An effort to address these concerns eventually led to a collaboration between IBM and two universities to develop a comprehensive replacement for GCTk and the collectors that originally shipped with Jalapeño. The result was MMTk, the Memory Management Toolkit [20, 21], which is discussed in more detail in Section 7.1.

6   Preparing for Open Source

The university releases enjoyed considerable success (16 university licenses and 16 more requests in just ten months), but the costs of sharing the code grew to a considerable expense. One reason was the complexity of the process required to obtain the system. Under the university license, a specialized version of the license agreement was created for each university, which needed to be approved by the university's contract office. This typically took one to two months.

Another significant issue was the limitations of the university agreement. Several professors expressed interest in using the system to teach courses. With most software, professors ask their students to download the software from a specified web address. However, this was not possible with the university agreement because the professor was responsible for controlling access to the software.

It became obvious that a true open source release would eliminate these problems, and make the system much more accessible to a larger audience. This was reinforced by one of the recommendations from an internal study conducted by the IBM Academy of Technology on future directions for virtual machine technology at IBM. Additionally, team members began to advocate open sourcing the system to promote research in virtual machine technologies. Specifically, by providing common high-performing research infrastructure that would allow comparisons among research projects, we would help to ensure credible baseline results, and enable researchers to focus on innovation rather than on building infrastructure.

For these reasons, the project began to investigate the ramifications of an open source release in early 2001. Many questions were raised, such as

- Would there be objections by IBM's product divisions to an open source release?

- Would IBM commit the necessary resources needed for supporting an open source release?

- Would IBM's internal open source steering committee approve of the release?

- How would the libraries be handled?

- What kind of infrastructure was available for supporting open source projects?

In addition to these strategic and process issues, the project also faced a large technical limitation that threatened open source viability. At that time the system ran on AIX/PowerPC and, in a more limited form, on Linux/PowerPC. Although our university licensees acquired PowerPC-based systems, clearly the Linux/IA32 platform would be much more attractive to a larger audience. To make open source a reality, we would have to commit to porting the system to Linux/IA32, which required a substantial investment of time and effort. With little difficulty, the team reached consensus on a decision to make this investment, and began porting the system in early 2001.

Although the system is mostly written in Java, and thus, a good portion of it is platform independent, the system does utilize two dynamic compilers (baseline and optimizing) that produce native code. Thus, both compilers had to be re-targeted to the IA32 architecture. The baseline compiler would provide functionality, and the optimizing compiler would provide performance. By August 2001, the baseline compiler was mostly complete, resulting in a functioning system on Linux/IA32. Over the following five months, work on the optimizing compiler resulted in a doubling of performance and reached 95% of the performance of the IBM's production system (IBM DK 1.3.0) [22]. Further details on this activity, including additional performance information, can be found in [22].

While a large subset of the team ported the system to the Linux/IA32 platform, the project managers investigated the strategic and process issues raised above.

The first two issues, product division approval and commitment of resources, are often major obstacles to open source donations from corporations. By early 2001, IBM had invested over three years worth of full-time activity of a team of up to fifteen members in the project (an R&D expense of over $10 million). Would many business executives find it wise to give away the result of this investment? Furthermore, we required not only permission to donate a past investment, but also a significant future investment to maintain, steer, and promote the open source project.

Fortunately, IBM had developed a history of contributing to the open source community, starting with the open source release of the Jikes Java source-code-to-bytecode compiler [23], and including several other projects, many of which are described in articles in this issue. This environment made it easier to consider

further contributing to the open source community. Further, IBM already had two high-quality commercial systems, the IBM DK for Java [24] and the J9 VM [25], so Jalapeño was not a vital proprietary asset, even with its differentiating technologies. IBM Research management saw the greater benefits of impacting the research community and supported the additional resources required.

Although the project secured support from all necessary management chains, several obstacles remained. One important issue was the release of the libraries. Recall that the university releases, made under an agreement of confidentiality with each university, included libraries from OTI. Because it was not possible to open source these libraries, we needed approval to release them under a separate non-open-source license. Working with the IBM legal team, we eventually developed a click-through license that allowed their use in non-commercial settings. Although this was not a perfect solution, it did serve our primary audience, the research community, who focused on research involving the VM and not the libraries.

To ensure that new open source projects are successful, IBM provides significant guidance and requirements. Each open source project must present a plan for maintenance and the evolution of the project. Many successful research projects create open source versions of their code base, but do not contribute the resources required to foster an open source community and maintain the system. Requiring the project to address these issues before the open source release helped increase the chance for success. Thus, in addition to resources required to produce the code base (1997–2000) and the commitment required to open source the code base (2001), there was an additional resource commitment required to maintain the code base for a significant period of time. Once again, research management had the foresight to support this activity, despite the fact that the "product" would not generate any revenue.

With the approvals in place, the focus now turned to the pragmatics of transferring the system to an open source project. This required another source code repository migration to the IBM's publicly available developerWorks `cvs` site (`www.ibm.com/developerworks`), creating mailing lists for the projects, creating a project web page, and other tools. We migrated all appropriate defects from our internal repository to the developerWorks bug tracking database.

We decided to announce the release of that system at the OOPSLA conference on October 15, 2001. Working with the IBM communications department, we prepared an official press release announcement. We also held a Birds-of-a-Feather session at the conference to announce the release.

Although things mostly went smoothly, one significant bump lay late in the road. Approximately three weeks before the release date we learned that we should not release a system named "Jalapeño" as another company had claimed trademark rights in the term. After a flurry of meetings, we decided to use the name *Jikes Research Virtual Machine* or *Jikes RVM* for short. IBM already had several open source systems using the Jikes name, the most popular of which was the Jikes Java source-to-bytecode compiler, so we leveraged this association. However, we did realize the potential confusion that might arise between the two systems. Although both were developed by the Software Technology Department at IBM Research, their developers and code were independent.

We also needed to be clear in our documentation that the system we were releasing was not a "JVM". Although researchers often use this term to mean a system that executes some Java programs, it is in fact a trademark of Sun Microsystems that can only be used by systems that pass the official Java Test Compatibility Kit (TCK). Since we had not run these tests, we could not use this term.

Thus, on October 15, 2001, Jikes RVM came into existence. The initial open source release was called 2.0 to distinguish it from the university releases, which were numbered 1.x.

7   Evolving the Project

As mentioned in previous sections, considerable effort was made in transforming the Jalapeño internal research infrastructure into the Jikes RVM open source project. Although the technology in the system was

already a research success, as measured by the number and quality of publications using the system, this did little to ensure that the project would be an open source success. The next three sections describe the strategic decisions and efforts that were made to enable Jikes RVM to be an open source success. These sections address the steps taken to 1) enhance the technology, 2) advance the project, and 3) promote the project.

7.1 Enhancing the System

In the three years following the open source release of Oct 15, 2001, thirteen subsequent releases of the system were created. These releases included significant performance improvements on Linux/IA32, new functionality, and bug fixes. We highlight below some of the major enhancements to the system. Interested readers can find more details in the release notes for each release at the Jikes RVM web site [26].

VM: pluggable object model [27]; separate name spaces for classloaders; elimination of classloading lock; support for reading soot-style annotations [28]; support for accessing hardware performance monitors on AIX/PowerPC and Linux/IA32

GC: memory management toolkit [20] (details below); adaptive heap sizing; support for soft, weak, and phantom references

Compilers: many improvements to the optimizing compiler, including a new implementation of linear scan register allocator [29]; restructuring of the baseline compiler to increase platform-independent code, make it re-entrant, and collect dynamic branch profiles to be used by subsequent recompilations

System-wide: introduction of packages; migration to the GNU Classpath libraries (details below); a bytecode verifier; introduction of logical build configurations; on-stack replacement [30]; increased platform support: Linux/PowerPC, Mac OSX, 64-bit PowerPC on both AIX and Linux.

As mentioned earlier, although Jikes RVM often had competitive performance with production JVMs, it could not run arbitrary Java programs due to unimplemented features in the VM and the libraries. However, it was able to run the benchmarks that were of interest to the research community, such as SPECjvm98™ [31] and SPECjbb2000™ [32]. The open source release of the Eclipse IDE [5], a large program written in Java, provided a significant test for any Java VM. There were several reasons why getting Jikes RVM to run Eclipse was attractive

- It would provide the research community with a much larger benchmark to use for their research experiments.

- Because of its interactive nature, in contrast to the SPEC™ benchmarks, it would test a VM's general responsiveness and, in particular, its startup performance.

- Although Eclipse was open source, some open source Linux distributors would not distribute Eclipse because it required a VM to run, and there were no open source VMs that could run Eclipse.

- It might help promote the use of Eclipse among the research community.

Thus, in spring 2002, the project decided that it would be desirable for Jikes RVM to run Eclipse. To accomplish this task, an Extreme Blue project was initiated in the summer of 2002 in Cambridge, MA. Extreme Blue is an IBM internship program where a team of four interns work in an intense environment on a common focused project. The team is composed of three technical students and one MBA student. The program is extremely competitive. Technical and business mentors are assigned to the project, who

9

work closely with the team. The Eclipse on Jikes RVM project ran from June through August of 2002. The project mostly consisted of trying various Eclipse functionality on top of Jikes RVM, diagnosing failures, and fixing them in either the VM or the libraries. More concretely, the technical work was done by the Extreme Blue team in the summer of 2002 and by Jalapeno team members in the months both before and after that summer. There were four major categories of work.

1. There were several large pieces of core VM functionality that were needed to run Eclipse and essentially did not exist. The largest of these was class loaders: before the work on Eclipse, Jikes RVM used one class loader and the classloader APIs were minimally supported. Before, during, and after the summer, an implementation of class loaders was done, with invasive changes to the core structures. Missing pieces of JNI functionality were also added.

2. The standard libraries that Jalapeno and then Jikes RVM used until then were really just core system classes, which were woefully insufficient. Rather than write large chunks of library code, the Extreme Blue team started using portions of the libraries from the GNU Classpath project. For instance, large pieces of GNU Classpath library code were used to support the various protocols for the web-based help system integrated into Eclipse. The GNU Classpath code was also used to replace portions of existing libraries, notably portions of the I/O libraries and security code.

3. The Extreme Blue team also wrote code that was needed to integrate Jikes RVM and Eclipse, The largest piece of this work was code to allow Eclipse to use Jikes RVM as a subprocess to run Java programs. This required writing a specialized Eclipse plugin, which was essentially an adaptor that allowed Eclipse to understand how to invoke Jikes RVM.

4. The number of actual bugs in Jikes RVM was smaller than we expected, but was substantial nonetheless. A good portion of the Extreme Blue team's time was spent tracking down and fixing problems in Jikes RVM. The largest single source of such bugs was the interaction of the baseline compiler and the GC system; these bugs were especially painful because they manifested themselves as obscure crashes due to corrupted memory.

In the end, the project was a huge success; Jikes RVM was able to run Eclipse.

However, to achieve the goals mentioned above, the modifications made by the team, performed on an internal branch of the system, needed to be merged back into the main `cvs` source tree. This was done over the last part of 2002. However, there still remained one significant technical hurdle. As mentioned in Section 6, Jikes RVM was distributed with non-open-source libraries. Although this might not be a problem for research uses, it posed a significant issue for open source Linux distributors.

Fortunately, there was an existing open source project, GNU Classpath, whose goal was to develop a complete set of Java libraries [33]. Shortly after the open source release of Jikes RVM, suggestions were posted on the Jikes RVM mailing list to switch to these libraries. An open source developer, John Leuner, started the effort to make this transition and contributed his work to the project. Using the experiences from this work, the Extreme Blue team used some of the non-core Classpath libraries to run Eclipse. In the 2.2.1 release of April, 2003 the project completely switched to the Classpath libraries; for the first time, both the system and libraries were completely open source.

Coincidentally, around the same time, Classpath developers were interested in demonstrating that their libraries were functional enough to run Eclipse, and encouraged the several VMs that use their libraries to take up this challenge. Jikes RVM was one of a few systems that were able to run Eclipse.

The memory management subsystem was also the focus of major enhancements. In the first half of 2002 a major overhaul of the Jalapeño collectors was commenced at IBM. Simultaneously, a new successor to GCTk was being planned at the Australian National University and the University of Texas. When these two groups discovered their mutual goal, they held a meeting at PLDI in June 2002, and decided to collaborate. The

result was an ambitious project to build a new toolkit for memory management from scratch. Ideally, the new toolkit would perform as well as, or better than, the highly-tuned existing collectors, and be at least as flexible as GCTk, while accommodating a broader range of memory management algorithms. Within three months, JMTk (the Java Memory management Toolkit) was functional. As portability to other runtimes and other languages became a focus, JMTk was renamed MMTk (the Memory Management Toolkit).

The project was very successful, meeting both its performance and flexibility objectives. It is now widely used within the memory management community and has been ported to other non-Java runtime systems. MMTk represents both a success in the area of software engineering [20] and research, where it became the platform for the first comprehensive apples-to-apples study of key memory management algorithms [21], as well as the basis for new algorithms [34]. Above all, MMTk is a strong example of the value of collaboration between industry and academia in developing research infrastructure.

7.2 Evolving the Project

A key ingredient of a successful open source project is evolving participation from the initial developers to a self-sustaining community of volunteers that is not necessarily dependent on the initial developers.

When Jikes RVM was initially released, the project depended heavily on support from the IBM developers. However, even before releasing the system, plans were made regarding how the project could evolve to ensure its long-term success, even if IBM employees were no longer active participants. This strategy encourages users of the system to become experts in one or more areas of the system, which then puts them in a position to help others, and potentially to maintain the system in the future.

The mechanics of achieving this were put in place over time. For example, nine months after the initial open source release a process was developed for accepting user contributions to the code base and documentation. This process, developed with the legal department, is simple enough to encourage contributions, but still satisfies the legal department's concerns about contribution originality. (Details are on the Jikes RVM web site.) Perhaps the most significant user contribution has been the port of the system to 64-bit PowerPC. This port was coordinated and assisted by several IBM core team members, but the bulk of the work was performed by university researchers at Ghent, University of Massachusetts, and University of New Mexico. In addition to the 64-bit port, in the 2+ years since we have been accepting contributions, we have received 35 nontrivial contributions from 26 individuals. Just under half of these were enhancements to MMTk, the memory management component of Jikes RVM. Other contributions included ports to MAC OS/X, a partial implementation of bytecode verification, and a number of other smaller enhancements.

In June 2003, we formalized the structure of the project by forming a steering committee and a core team. The steering committee is responsible for the strategic direction and success of the project. It is expected to ensure the project's welfare and guide its overall direction.

The core team is responsible for virtually all of the day-to-day technical decisions associated with the project. Core team members have write access to the source code repository. They decide what new code is added to the system and process contributions from the user community. Active participation on the mailing lists is a responsibility of all core team members, and is critical to the success of the project. Becoming a core team member is a privilege that is earned by contributing to the project.

As expected, the initial core team was a subset of the initial IBM implementors. However, over time, it has evolved to include several people from outside of IBM. As of November 2004, there are 11 members on the core team, 6 of whom are IBM employees. The core team will continue to evolve as new volunteers present themselves, and current members move on to other activities.

There are several important ingredients to being accepted as an open source project by open source developers. First, one must release the code under a license approved by the Open Source Initiative [35] (OSI). Jikes RVM uses the CPL license, which is approved by the OSI. The GNU Classpath libraries use a variant of the GPL, which is also OSI-approved. Second, the project must be run in an open manner, i.e.,

discussion about the project's evolution should be open to all potential participants and publicly archived. Jikes RVM strives to achieve this by using the `jikesrvm-core` mailing list for these discussions, refraining as much as possible from "hallway" discussions at IBM.

Some open source software developers — often followers of the Free Software Foundation's philosophy — are ideologically committed to Free Software. For this community, there is a third requirement: that all software tools used to build and run the system should also be Free Software. (This is the same community that requires an open source VM in order to run Eclipse.) Thus, despite Jikes RVM's huge success with the university researchers, Free Software developers have been reluctant to use or contribute to the system because of its reliance on proprietary software in building its boot image file. To understand this reliance, we need to give an overview of the Jikes RVM build process. Further details are provided in [10].

As mentioned earlier, Jikes RVM is mostly written in the Java programming language. It *runs* on itself without the need for a second virtual machine. To start its execution it reads a *boot image* file that contains a frozen instance of an initial VM. The program that creates this image is also written in the Java programming language. The program basically takes all of the core classes in Jikes RVM and compiles them to native code and then writes them to the boot image file. Because this program compiles a large number of classes, it is a useful stress test for any Java VM. As of mid-2004, no open source VM was robust enough to run this boot image writer program. Thus, Jikes RVM required the use of a non-open-source JVM for creating its boot image.

An effort has begun to create the boot image with an open source VM. Ultimately, we expect Jikes RVM to be that VM, but because of complications involving classloading, this is not as straightforward as it might seem. Recall that the VM will be running a Java program that is trying to load and natively compile another version of itself. Thus, we are initially trying to create the boot image with another open source VM. As of version 2.3.2 (April, 2004), the Kaffe virtual machine [36] was able to build a version of the Jikes RVM boot image that utilizes the baseline compiler. Although the time to build the image was about six times longer than using a proprietary JVM, it does satisfy the wish of those who want to only use open source tools when developing Jikes RVM. Because of this accomplishment, a Debian™ package containing Jikes RVM is now available and we are working on getting it into the Debian distribution. Work also continues on being able to self-build with Jikes RVM.

7.3   Promoting the System

Creating a successful open source project requires much more than useful technology. Complex technology, like a virtual machine, requires not only promotion of the availability of the system, but also a considerable amount of education on how the system works. From the first day the system was released as open source, the project has placed promotion of the system as an important goal. This has taken a number of forms.

Birds-of-a-Feather Sessions: We held these sessions at the premier conferences for programming languages, when the university release was available (PLDI'01), and when the open source release was available (OOPSLA'01 and PLDI'02). The sessions were organized as information sessions, where a project member would take about 20 minutes to explain what the system was (at a high level) and then existing users would discuss how they are using the system. Attendance varied from 20–100 people.

Tutorials: To help provide a more detailed explanation of the system we conducted a number of technical tutorials about the system. In September 2001, we presented an 8-hour tutorial on the complete system at the PACT conference in Barcelona. In June and October 2002, we presented tutorials on the details of the optimizing compiler and adaptive optimization system at PLDI (Berlin) and OOPSLA (Seattle), respectively. In October 2004, we presented a tutorial on the MMTk garbage collection toolkit at ISMM in Vancouver.
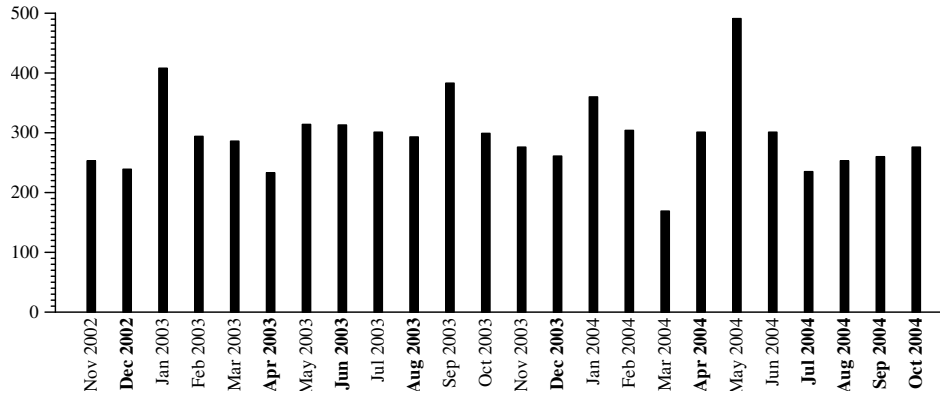
Figure 1: Downloads/month. The web site was down for two weeks in March 2004.

Online Information: The Jikes RVM home page provides a large amount of information for the prospective user. It includes the slides from all tutorials, a list of publications that use the system (96 as of November 2004), a list of courses that have been taught using the system (20 as of November 2004), including links to the courses with lots of helpful information, and a list of current users of the system. There are also details about the code base, such as the user's guide, a browseable CVS repository, the complete Jikes RVM API, mailing list archives, and the bug/feature database.

Mailing Lists: There are four mailing lists associated with the project geared towards 1) general announcements, 2) the results of nightly regression tests, 3) general questions, and 4) more detailed (core team) questions. From the beginning, the mailing lists have received a significant amount of questions. As expected, in the early days almost all questions were answered by the initial IBM implementors of the system. However, as outside expertise with the system has grown over time, more and more questions are being answered by the user community.

8   Measuring Impact

This section presents some quantitative information that measures activity related to the Jikes RVM project. Because no one metric sufficiently quantifies success, we explore several metrics.

Jikes RVM can be obtained in two ways: direct access from the `cvs` repository, or downloaded as a release, which is a packaged snapshot of the system. Figure 1 presents the number of downloaded snapshots per month since November 2002. (We do not have download data for the first year of the project.) The drop in downloads in March 2004 resulted from the project web site being down for two weeks. The figure reports approximately 300 downloads per month, with significant variations month-to-month. On the x-axis, months when a new release was made appear in bold font, such as Dec 2002. Not surprisingly, the figure shows an apparent trend, where the month after a release has a large number of downloads, which then becomes smaller until a new release is issued (for example, consider January–April 2004, and May–July 2004.)

Although this metric seems straightforward, it is not clear how appropriate it is for a research user community, which performs their research on a locally modified version of the system. For such users, downloading a new release and then porting their modifications is not always desirable, despite the improvements in functionality and robustness. Thus, some users may get great benefit from the system for several years, but download it only once. Furthermore, other users frequently get copies of the system using `cvs` directly,
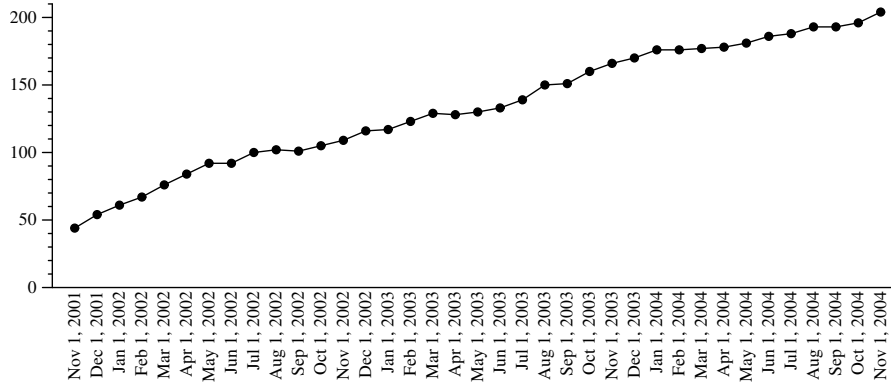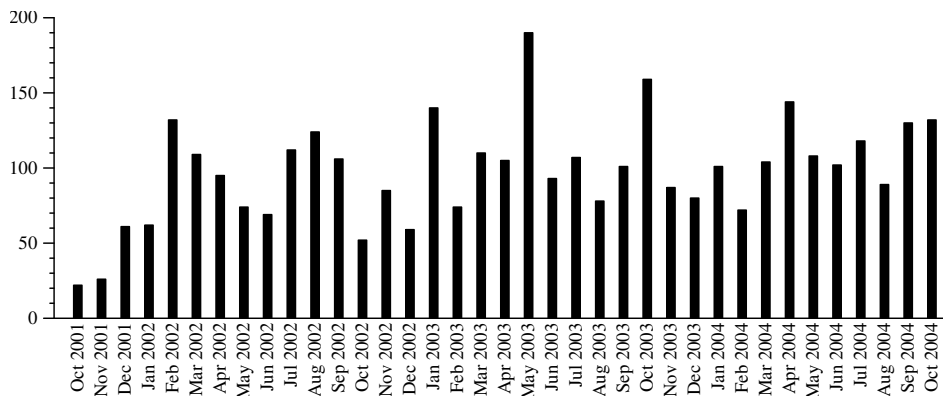
Figure 2: Mailing List Subscribers

Figure 3: Mailing List Traffic

and thus, they are not counted in the download metric.

Another manner to measure user activity is to monitor the main mailing list, `jikesrvm-researchers`. Figure 2 presents the number of people subscribed to this list on the first day of each month since the project's inception. The figure illustrates a steady increase of interest in the project's activities, about 50/year. The number of IBMers subscribed to this list has varied from 17–19 people over the duration displayed. It is reasonable to assume that someone who subscribes to the list is interested in discussions about the system, and thus it is a reasonable metric of a project's success. However, as all mailing list discussions are archived on the project web page, one does not need to subscribe to benefit from the information contained on the list, so although this metric is useful, it is not perfect.

Another interesting metric is the amount of traffic that is received on the main mailing list. Figure 3 presents this value on a monthly basis since the project's inception. The figure illustrates a steady flow of mail activity, averaging about 100 messages per month. Typical traffic is usually a user posting a question about the system, which is then answered by someone with familiarity in the area being queried. In the early days of the project, almost all answers came from the IBM developers. However, over time this has changed so that the majority of questions are now answered by the larger community.

The primary goal of the Jikes RVM project is to enable users to advance the state-of-the-art in virtual machine technologies. Thus, the most result-oriented metric of the success of the project is how well the users
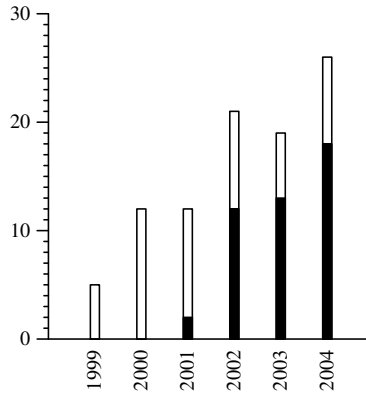
14

Figure 4: Publications Using Jikes RVM. The white segment counts papers co-authored by at least one person from IBM. The black segment counts publications without any IBM authors.
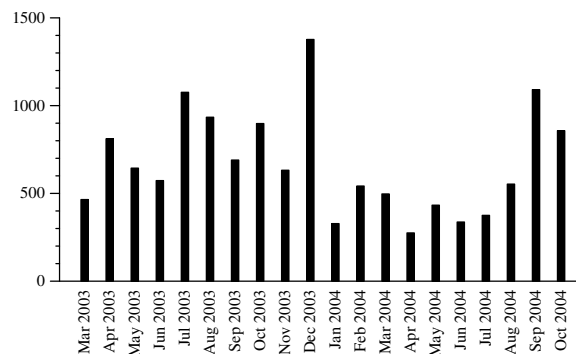

Figure 5: `cvs` commits

of the system are succeeding in this effort. Clearly, this depends as much on the quality of the user community as that of the system. Figure 4 presents the number of publications that used Jikes RVM/Jalapeño per year starting in 1999, the year of the first Jalapeño publication. It includes publications by Jikes RVM developers, other researchers at IBM, and the user community. To help better understand how the system has impacted researchers outside of its initial developers and other IBMers, the figure partitions publications into two categories: publications with at least one author affiliated with IBM (white bars), and publications with no IBM authors (black bars). This is not a perfect partition. Some "white bar" papers are written predominantly by non-IBMers. Thus, the black bars are a conservative underestimate of the publications by non-IBM-related researchers. The figure illustrates that, as the system has become available outside of IBM, more researchers have successfully published their work. Since the project's inception, 96 papers (that we are aware of) have been published using the system. A complete list of publications, including links to the papers in most cases, is available at the project web site.

In addition to publications, the system has been used to teach at least 20 courses at 12 different universities. Furthermore, researchers at at least 66 universities have used the system for their research.

Finally, Figure 5 presents the number of `cvs` commits to the system since March 2003, when this data became available. Commits are made by core team members and are predominately bug fixes and new feature additions to the code base. A small minority of commits are improvements to the supporting infrastructure,

15

such as the documentation, building tools, and regression testing. This metric is a good indication of the activity of the project over time.

As mentioned in Section 6, an expected benefit to IBM of the making the system available as an open source project was an improved awareness of IBM's technology, leading to improved PhD recruiting. The project has seen evidence of this happening. In particular, the existence of the project has attracted dozens of graduate students interested in internships, and resulted in at least four new PhD hires over the past three years.

9   Conclusions and Lessons Learned

This paper has highlighted the evolution of the Jikes RVM project from internal research infrastructure, to university releases, to successful open source project. We conclude with an attempt to summarize some lessons learned during this experience.

- **Success with the open source research community demands much more than just source code.** The research community uses a system as a white box, and will have a voracious appetite for information on the system's inner workings. To provide this information, project maintainers must provide extensive documentation, tutorials, and access to "experts" via a mailing list.

- **Open source can serve as an invaluable catalyst for productive research collaboration.** The successful development of MMTk provides a shining example of cooperative development among researchers on different continents, with different backgrounds, spanning industry and academia. MMTk has resulted in numerous high-quality research results, dozens of publications, and an immensely stimulating collaboration. Furthermore, it has advanced the science of memory management research by providing high-quality infrastructure by which researchers the world over can reproduce and improve on published results. Open source sharing makes all this possible.

- **Flexibility vs. maintainability presents difficult tradeoffs.** Software designed for research use should be flexible enough, and provide a rich enough feature set, to support a wide variety of implementation options. However, software used for production should be easily maintained, and thus benefits from a minimal feature set to reduce testing requirements. An open source research project constantly faces tension between these forces, and must carefully manage a system's evolution to strike a fair balance.

- **When starting a systems software research project, assume the infrastructure will be adopted by a large community, and manage the software development accordingly.** Assume from the beginning that the project will have a significant impact and raise significant interest in the infrastructure. If the project will not have a significant impact, find another project.

Finally, we close this section with a commentary on the practice of systems software research. For science to advance, researchers must be able to reproduce past results, and ultimately improve on them. In systems software research, results often depend on myriad implementation details that cannot be conveyed in a research paper. Although many systems research papers include information on algorithms, benchmarks, and experimental methodology, this information ultimately fails to facilitate reproducible results.

We hoped, and still hope, that widely adopted open source research infrastructure such as Jikes RVM will change the game, allowing researchers to publish source code used in studies, and allowing other researchers to build on the results. However, in our experience, the vast majority of researchers choose not to make their implementations publicly available. We suspect that the main issue is that the research community, as represented by program committees and tenure committees, does not explicitly value producing open

software. So, most researchers do not see strong incentive to devote time and energy into publishing software for research.

We hope that, as more open source research infrastructure is built, the community mindset will gradually change, until open software for reproducible systems research becomes the rule rather than the exception. We believe that such a sea change would dramatically advance the discipline of systems software as a science.

10   Acknowledgements

Sun, **Java**, **Javadoc**, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Debian is a registered trademark of Software in the Public Interest, Inc., a non-profit corporation.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

AIX, **PowerPC**, **IBM**, and **Jikes** are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

SPECjvm, **SPECjbb**, and **SPEC** are registered trademarks of The Standard Performance Evaluation Corporation (SPEC), a non-profit corporation.

References

[1] The Linux Kernel Archives. http://www.kernel.org.

[2] The FreeBSD Project. http://www.freebsd.org.

[3] OpenOffice.org: Home Page. http://www.openoffice.org.

[4] LaTeX project: LaTeX - A document preparation system. http://www.latex-project.org.

[5] Eclipse Foundation. Eclipse. http://www.eclipse.org.

[6] GCC Home Page - GNU Project - Free Software Foundation (FSF). http://gcc.gnu.org.

[7] KDE Home Page - Conquer your Desktop! http://www.kde.org.

[8] GNOME: The Free Software Desktop Project. http://www.gnome.org.

[9] B. Alpern, C. R. Attanasio, J. J. Barton, M. G. Burke, P. Cheng, J.-D. Choi, A. Cocchi, S. J. Fink, D. Grove, M. Hind, S. F. Hummel, D. Lieber, V. Litvinov, M. F. Mergen, T. Ngo, J. R. Russell, V. Sarkar, M. J. Serrano, J. C. Shepherd, S. E. Smith, V. C. Sreedhar, H. Srinivasan, and J. Whaley. The Jalapeño virtual machine. *IBM Systems Journal*, 39(1):211–238, February 2000.

[10] Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Barton, Susan Flynn Hummel, Janice C. Sheperd, and Mark Mergen. Implementing Jalapeño in Java. *ACM SIGPLAN Notices*, 34(10):314–324, October 1999. Published as part of the proceedings of OOPSLA'99.

[11] Michael G. Burke, Jong-Deok Choi, Stephen Fink, David Grove, Michael Hind, Vivek Sarkar, Mauricio J. Serrano, V. C. Sreedhar, Harini Srinivasan, and John Whaley. The Jalapeño dynamic optimizing compiler for Java. In *ACM 1999 Java Grande Conference*, pages 129–141, June 1999.

[12] Matthew Arnold, Stephen Fink, David Grove, Michael Hind, and Peter F. Sweeney. Adaptive optimization in the Jalapeño JVM. *ACM SIGPLAN Notices*, 35(10):47–65, October 2000. Proceedings of the 2000 ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA'00).

[13] Clement R. Attanasio, David F. Bacon, Anthony Cocchi, and Stephen Smith. A comparative evaluation of parallel garbage collector implementations. In *14th International Workshop on Languages and Compilers for Parallel Computing*, pages 177–192, August 2001.

[14] Walter F. Tichy. RCS: A system for version control. *Software—Practice and Experience*, 15(7):637–654, July 1985.

[15] Concurrent Versions System (CVS). http://www.cvshome.org.

[16] J. Choi and H. Srinivasan. Deterministic replay of Java multithreaded applications. In *SIGMETRICS Symposium on Parallel and Distributed Tools*, pages 48–59, August 1998.

[17] Mauricio Serrano, Rajesh Bordawekar, Sam Midkiff, and Manish Gupta. Quicksilver: a quasi-static compiler for Java. *ACM SIGPLAN Notices*, 35(10):66–82, October 2000. Proceedings of the 2000 ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA'00).

[18] Vugranam C. Sreedhar, Michael Burke, and Jong-Deok Choi. A framework for interprocedural optimization in the presence of dynamic class loading. *ACM SIGPLAN Notices*, 35(5):196–207, May 2000. Published as part of the proceedings of PLDI'00.

[19] Bowen Alpern, Anthony Cocchi, and David Grove. Dynamic type checking in Jalapeño. In *Usenix Java Virtual Machine Research and Technology Symposium (JVM'01)*, pages 41–52, April 2001.

[20] Stephen Blackburn, Perry Cheng, and Kathryn McKinley. Oil and water? High performance garbage collection in Java with JMTk. In *Twenty-sixth International Conference on Software Engineering*, pages 137–146, May 2004.

[21] Stephen Blackburn, Perry Cheng, and Kathryn McKinley. Myths and reality: The performance impact of garbage collection. In *ACM SIGMETRICS — Performance 2004, Joint International Conference on Measurement and Modeling of Computer Systems*, pages 25–36, June 2004.

[22] Bowen Alpern, Maria Butrico, Anthony Cocchi, Julian Dolby, Stephen Fink, David Grove, and Ton Ngo. Experiences porting the Jikes RVM to Linux/IA32. In *Usenix Java Virtual Machine Research and Technology Symposium (JVM'02)*, pages 51–64, August 2002.

[23] Jikes. Jikes project. http://oss.software.ibm.com/developerworks/opensource/jikes.

[24] T. Suganuma, T. Ogasawara, M. Takeuchi, T. Yasue, M. Kawahito, K. Ishizaki, H. Komatsu, and T. Nakatani. Overview of the IBM Java Just-in-Time Compiler. *IBM Systems Journal*, 39(1):175–193, February 2000.

[25] Nikola Grcevski, Allan Kilstra, Kevin Stoodley, Mark Stoodley, and Vijay Sundaresan. Java just-in-time compiler and virtual machine improvements for server and middleware applications. In *Usenix 3rd Virtual Machine Research and Technology Symposium (VM'04)*, May 2004.

[26] Jikes Research Virtual Machine (RVM). http://www.ibm.com/developerworks/oss/jikesrvm.

[27] David F. Bacon, Stephen J. Fink, and David Grove. Space- and time-efficient implementations of the Java object model. In *16th European Conference on Object-Oriented Programming*, June 2002.

[28] Patrice Pominville, Feng Qian, Raja Vallée-Rai, Laurie Hendren, and Clark Verbrugge. A framework for optimizing Java using attributes. In *Compiler Construction, 10th International Conference (CC 2001)*, pages 334–554, November 2001.

[29] Massimiliano Poletto and Vivek Sarkar. Linear scan register allocation. *ACM Transactions on Programming Languages and Systems*, 21(5):895–913, September 1999.

[30] Stephen J. Fink and Feng Qian. Design, implementation and evaluation of adaptive recompilation with on-stack replacement. In *International Symposium on Code Generation and Optimization*, pages 241–252. IEEE Computer Society, 2003.

[31] Standard Performance Evaluation Corporation. SPECjvm98 Benchmarks. http://www.spec.org/jvm98.

[32] Standard Performance Evaluation Corporation. SPECjbb2000 Java Business Benchmark. http://www.spec.org/jbb2000.

[33] GNU. Gnu classpath, essential libraries for Java. http://www.gnu.org/software/classpath.

[34] Stephen M. Blackburn and Kathryn S. McKinley. Ulterior reference counting: fast garbage collection without a long wait. *ACM SIGPLAN Notices*, 38(11):344–358, November 2003.

[35] Open Source Initiative. http://www.opensource.org.

[36] Kaffe.org. http://www.kaffe.org.

Bowen Alpern *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (alpernb@us.ibm.com).* Bowen received a B.S. from the University of Michigan in 1974 and a Ph.D. from Cornell University in 1986. In that year, he became a research staff member at the IBM Thomas J. Watson Research Center. His research interests include virtual machine implementation and application virtualization.

Steven Augart *16 Brooks Street, Winchester, Massachusetts, 01890. (saugart@yahoo.com).* Steven Augart is a freelance software engineer. He first did free software work for the GNU project in 1983 and hasn't looked back since. He received a Bachelor's degree from Harvard in 1989 and an M.S. from the University of California (Irvine) in 1992. After a varied career including seven years at the USC Information Sciences Institute (Networking division), he became involved in the Jikes RVM project in 2003 and is a member of the core team. His interests are in operating systems, realtime programming (especially networking protocols) and free software.

Stephen M. Blackburn *Department of Computer Science, Australian National University, Canberra, ACT 0200, Australia (Steve.Blackburn@anu.edu.au).* Stephen M. Blackburn is a Research Fellow at the Department of Computer Science at the Australian National University. He received his B.Sc. and B.E. degrees from the Univeristy of New South Wales in 1990 and 1992, and his Ph.D. from the Australian National University in 1997. During a post-doc at the University of Massachusetts, he became involved in the Jikes RVM project, and is a member of the core team and steering committee. His interests include memory performance, memory management and architecture.

Maria Butrico *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (butrico@us.ibm.com).* Maria Butrico joined IBM in 1984 and is currently a Senior Software Engineer. She received a B.S. degree in Computer Science from Pace University in 1984 and an M.S, degree in Computer Science from Columbia University in 1990. Her interest include operating systems, database systems, distributed systems, and middleware. She received an Outstanding Technical Achievement Award for her work on the Virtual Shared Disk.

Anthony Cocchi *Department of Mathematics and Computer Science, Lehman College, The City University of New York, 250 Bedford Park Boulevard West, Bronx, New York 10468 (acocchi@lehman.cuny.edu).* Mr. Cocchi is a Distinguished Lecturer at Lehman College (a division of the City University of New York). He has a Bachelor and Masters degree in Electrical Engineering from Pratt Institute (Brooklyn New York) and a Master degree in Computer Science from New York Polytechnic Institute. On the Jikes RVM project, he worked on the garbage collector, the runtime, the basic compilers, and did performance measurement and analysis.

Perry Cheng *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (perryche@us.ibm.com).* Dr. Cheng is a Research Staff Member in the Software Technology Department at the IBM T.J. Watson Research Center. He received his B.S. degree from Rice University in 1994, and his M.S. and Ph.D. degrees from Carnegie Mellon University in 1998 and 2001, respectively. He subsequently joined IBM, where he is a member of the Jikes RVM core team and helped develop its garbage collector framework. His research interests include program language design and implementation, virtual machines, and runtime systems. He is a member of ACM.

Julian Dolby *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (dolby@us.ibm.com).* Julian Dolby is a Research Staff Member in the Software Technology Department at the IBM T.J. Watson Research Center. He attended the University of Wisconsin-Madison and the University of Illinois at Urbana-Champaign. He has been a member of the team that developed Jikes RVM, and was a mentor on the Extreme Blue project in 2002. He now works primarily on advanced program analysis techniques, focusing on demanding analysis applications for programming tools.

Stephen Fink *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (sjfink@us.ibm.com).* Dr. Fink is a Research Staff Member in the Software Technology Department at the IBM T.J. Watson Research Center. He received the B.S. degree from Duke University in 1992 and the M.S and Ph.D. degrees from the University of California, San Diego in 1994 and 1998, respectively. He subsequently joined IBM, where he was a member of the team that produced the Jikes Research Virtual Machine, and is currently investigating the application of static program analysis to Enterprise Java Beans. His research interests include programming language implementation techniques, program analysis, and parallel and scientific computation. He is a member of ACM.

David Grove *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (groved@us.ibm.com).* Dr. Grove is a Research Staff Member in the Software Technology Department at the IBM T.J. Watson Research Center. He received the B.S. degree from Yale College in 1992, and the M.S. and Ph.D. degrees from the University of Washington in 1994 and 1998, respectively. He subsequently joined IBM, where he is a member of the Jikes RVM core team and helped develop its adaptive optimization system, optimizing compiler, and runtime system. His research interests include program language design and implementation, virtual machines, and adaptive optimization. He is a member of ACM.

Michael Hind *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (hindm@us.ibm.com).* Dr. Hind is a Research Staff Member and manager in the Software Technology Department at the IBM T.J. Watson Research Center. He received the B.A. degree from the State University of New York at New Paltz in 1985 and the M.S. and Ph.D. degrees from New York University in 1991. From 1992-1998, Michael was a professor of computer science at the State University of New York at New Paltz. In 1998, Michael joined IBM to work on the Jalapeño project, particularily the adaptive optimization system and the optimizing compiler. In 2000, he became the manager of the Dynamic Optimization Group at IBM Research. His research interests include program analysis, adaptive optimization, and memory latency issues. He is a member of ACM.

Kathryn S. McKinley *Department of Computer Sciences, The University of Texas at Austin, 1 University Station #C0500, Austin, Texas 78712 (mckinley@cs.utexas.edu).* Kathryn S. McKinley is an Associate Professor at the University of Texas at Austin. She received her PhD in 1992, Masters in 1990, and Bachelors in 1985, all from Rice University. Her research interests include compilers, memory management, and architecture. She has received two IBM Faculty Awards (2003, 2004), an NSF CAREER Award (1996-2000), IEEE Senior member, a Texas Institute for Computation and Applied Mathematics (TICAM) Visitor Fellowship (1999), and a Chateaubriand Scholarship for the Exact Sciences, Engineering, and Medicine (1992). She is the program chair for the ACM conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) 2004, and the International Conference on Parallel Architectures and Compilation Techniques (PACT) 2005.

Mark Mergen *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (mergen@us.ibm.com).* Dr. Mergen managed the group responsible for the nonoptimizing compilers, virtual machine, and garbage collection in Jikes RVM, and previously managed the research effort leading to the High-Performance Compiler for Java (HPCJ) product. He is currently working in the K42 open-source, scalable, customizable OS kernel project, and previously managed the research prototype leading to the first 64-bit AIX product release. He has also worked on PowerPC architecture and virtual memory software. He has a B.S. degree in mathematics and an M.D. degree from the University of Wisconsin at Madison.

J. Eliot B. Moss *Department of Computer Science, University of Massachusetts, 140 Governor's Drive, Amherst, MA 01003-9264 (moss@cs.umass.edu).* Dr. Moss is an Associate Professor and has been a member of the Computer Science faculty at the University of Massachusetts, Amherst since 1985. His work focuses on the efficient implementation of modern programming language and runtime systems on modern architectures. He participated in the designs of the Clu and Argus programming language while a graduate student, and contributed the nested transaction model in his dissertation. Since then he has explored persistent object store and persistent programming language implementation, and more recently helped advance the state of the art in automatic storage management (garbage collection). He also pursues interests in hardware support for language features and in application of machine learning techniques to systems problems, especially in compiler optimization. His interactions with IBM Research led to the first academic source license for Jikes RVM (Jalapeno at the time). Dr. Moss received a Ph.D. in 1981, an M.S. in 1978, and a B.S. in 1975, all in Computer Science from the Massachusetts Institute of Technology.

Ton Ngo *IBM Almaden Research Division, 650 Harry Road, San Jose, California 95120 (ton@us.ibm.com).* Dr. Ngo received his Ph.D. and M.S. in Computer Science from the University of Washington, M.S. in Electrical Engineering from the Florida Institute of Technology and B.S. in Electrical Engineering from the Georgia Institute of Technology. He joined the IBM System Product Division in 1982, then the IBM T. J. Watson Research Center in 1987 and finally the Almaden Research Center in 2001. In the Jikes RVM project, he developed the debugger, the Java Native Interface and contributed to the Record/Replay tool. Currently he is with the WebFountain project, working on a cluster of several hundreds Linux machine for web-scale data mining.

Vivek Sarkar *IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (vsarkar@us.ibm.com).* Dr. Sarkar is a Research Staff Member and Senior Manager of Programming Technologies in the Software Technology Department at the IBM T.J. Watson Research Center, where he oversees research projects in the areas of programming models, programming tools, code optimization and virtualized runtime systems. He received the B.Tech. degree from the Indian Institute of Technology — Kanpur in 1981, the M.S. degree from the University of Wisconsin — Madison in 1982, and the Ph.D. degree from Stanford University in 1987. During 1998–2000, he was manager of the Dynamic Optimization group, which was responsible for the dynamic optimizing compiler and adaptive optimization system in Jalapeo. After becoming Senior Manager in 2000, he led the team that evolved the Jalapeño research project into the Jikes RVM open source release in 2001. Dr. Sarkar's primary research interests are in optimizing and parallelizing compilers, static and dynamic program analysis, and parallel programming models. He has been a member of the IBM Academy of Technology since 1995.

Martin Trapp *100world AG, Vordere Cramergasse 11, D-90478 Nürnberg, Germany (martin.trapp@100world.de).* Dr. Trapp is a Chief Developer at 100world AG working in the Middleware Services group. He received his diploma and doctoral degrees from the University of Karlsruhe in 1999. At Karlsruhe, he worked on the translation and optimization of object oriented programs. From 2000-2002, Martin was a member of the Dynamic Optimization Group in the Software Technology Department at the IBM T.J. Watson Research Center. His main interests are reliability of software systems and improvements in software engineering.