The Jikes Research Virtual Machine 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

This paper describes the evolution of the Jikes™ Research Virtual Machine project from an IBM internal research project, 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.

On October 15, 2001, IBM Research launched the Jikes* RVM (Research Virtual Machine) open-source project. Jikes RVM provides a 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 paper describes the evolution of Jikes RVM from an IBM internal research project, 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 paper discusses a variety of challenges that arose in this process, including the 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 primary focus of the Jikes RVM project was the development of a software platform designed to be a research testbed for the prototyping of new technologies. In contrast, most open-source projects develop software products for use by the general public. We focus on the implications of this distinction throughout the paper.

The remainder of this paper is as follows. The next section begins with a general discussion of issues

pertaining to a research-oriented open-source project. The section "Motivation and history of the Jalapeño project" discusses the Jalapeño project's origin as an IBM internal project. The sections "The university releases" and "University activities" review an intermediate phase, during which the software was made available to a small number of universities under a restricted license. "Preparing for open source" and "Project evolution" discuss the preparation for, emergence, and first few years of Jikes RVM as a full-fledged, mature open-source project. "Measuring impact" presents some metrics that document the system's impact, and the paper concludes with some lessons learned from this case study.

■ The majority of Jikes RVM users modify the system to produce interesting research results ■

OPEN SOURCE FOR RESEARCH

The majority of open-source projects develop software that is used primarily as a "black box," that is, a system used solely for its functionality, whose internal design is not of interest. Community members adopt the software as a tool to perform some function, and most users often do not care about the internal design and implementation of the software. Examples of such software tools include most of the best known open-source projects, including operating system,[1,2] word processors,[3,4] integrated development environments,[5] compilers,[6] and graphical desktop environments.[7,8] In contrast, the Jikes RVM project developed software that is intended to be used as a "white box," a system whose internal design is of interest and 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 as follows:

1. Most of our community members are professors and graduate students, who use 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, not 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 the 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 project by a desire to resolve some irritating deficiency with the system. For example, a user may want to use a 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 provided by the system, 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.

2. Open-source license requirements, including source-code availability, do not apply to research results. Many open-source projects effectively enforce community sharing by means of their licenses, such as the GPL (General Public License) and CPL (Common Public License). These licenses, in differing ways, require that recipients who create and distribute derivative works of open-source software must make the source code of their derivative works available to their recipients. However, these licenses do not require source code to be made available 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. The majority of researchers using the Jikes RVM choose not to make their code available. We present an argument against this approach in the section on conclusions and lessons learned.

3. The research community tends to accept less polished software than the general public. The Jikes RVM provides for many academic groups a critical element of infrastructure that is crucial to their project's success. Because of its perceived importance, the community will generally tolerate a great deal of difficulty in learning and

installing the system in exchange for a system that is easier to change and modify. Although the Jikes RVM is fairly robust and well-tested compared to the software of most research projects, the initial user experience is, for various reasons, less pleasant than that encountered when using commercial proprietary software such as a product Java** virtual machine.

4. Our user community requires extensive documentation on the inner workings of the system, not just the exposed command line or application programming interfaces (APIs). Documentation and explanations of the inner workings of the system are crucial and constitute a key part of the project's products. Changes in internal structures can cause disruption in the user community and must be managed carefully.

MOTIVATION AND HISTORY OF THE JALAPEÑO PROJECT

Sun Microsystems introduced the Java** programming language in May 1995. It 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* (VM), that is, 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 making significant investments in virtual machine technology.

In November 1997, a small group of researchers at the IBM Thomas J. Watson Research Center began the Jalapeño project, whose goal was to develop an internal research infrastructure for VM technologies. The virtual machine, known as Jalapeño, was targeted as a flexible, extensible testbed for researching, prototyping, and evaluating VM implementation techniques. In contrast to other IBM Java virtual machines, Jalapeño was not part of any product, and so was unencumbered by the full preproduction process and quality requirements. Many papers document various technical aspects of the system.[9–13] Some technical highlights of Jalapeño include the implementation of the VM mostly in the Java programming language, an m-to-n quasi-preemptive thread system[14] 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, such that the developers would not have access to any non-IBM source code for the Java virtual machine and libraries. The Jalapeño virtual machine was written entirely with new code and used libraries developed by IBM at the OTI (Object Technology International) laboratory. This decision maximized flexibility for the 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 (Java Native Interface), 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 fulltime employees, plus a large number of academic visitors and Ph.D. candidates.

Most early Jalapeño researchers joined the project because it seemed an interesting or useful vehicle for pursuing 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 that 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 distributed as open-source software, and this influenced the initial development practices and methodologies.

Initially, the team shared source code with a centralized RCS[15] (Revision Control System) repository and some homegrown scripts that provided higher-level functionality. The project consciously adopted a "prototype first" development method-

ology. Much code was written in a throwaway style under pressure of impending paper deadlines, with the intention of complete revision at the earliest opportunity. Some of this throwaway 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

■ The system is mostly written in Java, and thus a good portion of it is platform independent ■

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, little 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 teams followed some of these processes by agreeing on small sets of test cases and using CMVC (Configuration Management Version Control) for bug tracking. However, these processes were only introduced on a project-wide basis after the problems caused by their omission became intolerable.

Between 1997 and 1999, the number of implementors, that is, committers to the code base, increased from a small handful to nearly 20. Regressions in function due to code modifications became increasingly problematic, and in the summer of 1999, the project 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. Regression in performance was 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 persons on the team on a weekly basis and 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 causing failure of the overnight tests took root.

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 a research infrastructure for their own projects. The first inquiry came from researchers at the University of Massachusetts at 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 to use 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 the source code was not available.) This decision necessitated significant bureaucratic and technical efforts.

The bureaucratic tasks included securing approval from upper management to release the code, deciding on an appropriate legal agreement and licensing structure, documenting the 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 Jalapeño system to the Linux**/PowerPC platform, the platform that the initial university users would use. This was performed by two collaborators at the University of Massachusetts at 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 for releasing tagged versions of the code base to our university partners. Previously, all code had resided in an RCS repository, and the current system consisted of the head of the RCS tree. Making releases available to universities required dealing with the problem of development continuing on the main trunk of this tree, while maintaining branches to identify bugs in tagged releases and merging changes back to the main trunk when appropriate. The extant homegrown RCS scripts did not support these activities well.

To address these issues, the project committed to migrate to a CVS[16] (Concurrent Versions System) source-code repository, which better supports branching and merging. Many team members had not previously used CVS, so their education was handled by a few team members with relevant expertise. Using RCS, the project had relied on pessimistic concurrency control of source code (i.e., a concurrency-control strategy that explicitly forbids concurrent writes) using RCS locks. In contrast, optimistic concurrency control assumes there will not be contention and requires remediation should contention arise. 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. Another issue concerned the discontinuity of the "history" in the source-code system; team members were forced to consult a frozen copy of the old repository in order 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 nonstandard 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 difficult to perform on a large code base. We employed tools to help with these activities and made significant improvements in large segments 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 the Watson Research Center,[17–20] which had nevertheless not 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 the university release.

The first university release was made on January 23, 2001 to the following universities: the University of Massachusetts at Amherst, the University of Colorado, Kent University, Purdue University, the 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. 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.

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 the Linux/PowerPC platform and more recently, a port to the Mac OS** X operating system by a developer at the University of Massachusetts. This also drove the port to the Linux/IA32 system by the team at IBM, which is described in more detail in the section "Preparing for open source." The desire for a 64-bit research platform encouraged researchers at a number of universities to pursue a port to the 64-bit PowerPC platform, 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 the Linux/PowerPC platform was maintained and tested at the University of Massachusetts each time a university release was made. This distribution of effort was originally due to 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-language 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 became open-source software. 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.

After 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 the section "Measuring impact") 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,[21,22] which is discussed in more detail in the section "Enhancing the system."

PREPARING FOR OPEN SOURCE

The university releases enjoyed considerable success (leading to 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.

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 recommenda-

tions 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 making the system open source to promote research in virtual-machine technologies. Specifically, by providing a 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, including:

- 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 its viability as open-source software. At that time, the system ran on the AIX/PowerPC platform, and, in a more limited form, on the Linux/PowerPC platform. Although our university licensees acquired PowerPC-based systems, it was clear that 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 the Linux/IA32 platform, 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 the Java programming language, 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 retargeted for the IA32 architecture. The baseline compiler would provide functionality, and the optimizing compiler would provide performance. By August 2001, the baseline compiler was almost complete, resulting in a functioning system on the Linux/IA32 platform. Over the following five months, work on the optimizing compiler resulted in a doubling of performance and reached 95 percent of the performance of the IBM production system (IBM Developer Kit [DK] Version 1.3.0).[23] Further details on this activity, including additional performance information, can be found in Reference 23.

■ The Memory Management Toolkit project was very successful, meeting both its performance and flexibility objectives ■

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 activity from corporations. By early 2001, IBM had invested over three years worth of fulltime activity of a team of up to 15 members in the project (a research and development 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-byte-code compiler,[24] and including several other projects, many of which are described in papers in this issue. This environment made it easier to consider further contributions to the open-source community. Additionally, IBM already had two high-quality commercial systems, the IBM DK for Java[25] and the J9 VM,[26] 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 commitment of 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. The university releases, made under an agreement of confidentiality with each university, included libraries from OTI. Because it was not possible to make these libraries open source, 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 the use of the libraries in noncommercial 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 the maintenance and 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. By requiring the project to address these issues before the open-source release, its chances for success were increased. Thus, in addition to resources required to produce the code base (from 1997 to 2000) and the commitment required to make the code base open source (in 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 pragmatic details of transferring the system to an open-source project. This required another source-code repository migration to IBM's publicly available developerWorks CVS site (www.ibm.com/developerworks), creating mailing lists for the projects, and 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 (Object-oriented Programming Languages and Applications) 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"[27] session at the conference to announce the release. Although things mostly went smoothly, one significant obstacle arose late in the process. 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-byte-code compiler, so we leveraged this association. However, we did realize the potential confusion that might arise between source-to-bytecode Jikes RVM and the Jikes compiler. 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). Because 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 Release 2.0 to distinguish it from the university releases, which were numbered 1.0, 1.1, and so forth.

PROJECT EVOLUTION

As mentioned in previous sections, it required considerable effort to transform 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 relating to the system, this did little to ensure that the project would be an open-source success. The following three sections describe the strategic decisions and efforts that contributed to making the Jikes RVM an open-source success. These sections address the steps taken to (1) enhance the technology, (2) advance the project, and (3) promote the project.

Enhancing the system

In the three years following the Jikes open-source release of October 15, 2001, thirteen subsequent releases of the system were created. These releases included significant performance improvements for the Linux/IA32 platform, new functionality, and bug fixes. Enhancements were made to the VM (including a pluggable object model), garbage collection (including the memory-management toolkit) and compilers (including a new implementation of the linear scan register allocator).[28] System-wide, enhancements included the migration to the GNU Classpath libraries, a bytecode verifier, on-stack replacement, and increased platform support for the Linux/PowerPC, Mac OS X, and 64-bit PowerPC on both AIX and Linux. Interested readers can find more details in the release notes for each release at the Jikes RVM Web site.[24]

As mentioned earlier, although Jikes RVM often displayed performance which was competitive with production Java virtual machines, 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**[29] and SPECjbb2000**.[30] The open-source release of the Eclipse IDE,[5] a large program written in the Java programming language, provided a significant test for any Java virtual machine. There were several reasons why getting Jikes RVM to run Eclipse was attractive:

1. It provided the research community with a much larger benchmark to use for their research experiments.

2. Because of its interactive nature, in contrast to the SPEC (Standard Performance Evaluation Corporation) benchmarks, it would test a VM's general responsiveness and, in particular, its startup performance. Although Eclipse was open-source software, some open-source Linux distributors would not distribute Eclipse because it required a VM to run, and there were no open-source VMs before Jikes that could run Eclipse.

3. In addition, the ability to run Eclipse on Jikes RVM might help promote the use of Eclipse among the research community.

Thus, in spring 2002, it was 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, Massachusetts. Extreme Blue is an IBM internship program in which 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 M.B.A student. The program is extremely competitive. Technical and business mentors are assigned to the project, and 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 Jikes RVM team members in the months both before and after that summer. There were four major categories of work:

■ An effort has begun to create the boot image with an open-source VM ■

1. Several large pieces of core VM functionality were needed to run Eclipse that essentially did not exist. The largest of these related to class loader functionality: before the work on Eclipse, the Jikes RVM used one class loader, and the class loader APIs were minimally supported. An implementation of class loaders was made, with invasive changes to the core structures. Missing pieces of JNI functionality were also added.

2. The standard libraries that 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 enable Eclipse to use Jikes RVM as a subprocess to run Java programs. This required writing a specialized Eclipse plug-in, which was essentially an adapter that allowed Eclipse to understand how to invoke Jikes RVM.

4. A good portion of the Extreme Blue team's time was spent tracking down and fixing problems in Jikes RVM. The number of actual bugs in Jikes RVM was smaller than we expected but was substantial nonetheless. The largest single source of such bugs was the interaction of the baseline compiler and the garbage collection system; these bugs were especially problematic 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. 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 the section "Preparing for open source," Jikes RVM was distributed with non-open-source libraries. Although this might not be a problem for research users, it posed a significant problem 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.[31] 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 experience of this work, the Extreme Blue team used some of the non-core Classpath libraries to run Eclipse. In the Version 2.2.1 release of April 2003, the project completely switched to the Classpath libraries, and 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 garbage 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 the ACM conference on Programming Language Design and Implementation (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 a success in the areas of both software engineering[21] and research, where it became the platform for the first comprehensive "apples to apples" study of key memory-management algorithms,[22] as well as the basis for new algorithms.[32] Above all, MMTk is a strong example of the value of collaboration between industry and academia in developing a research infrastructure.

Building the community

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 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 encouraged users of the system to become experts in one or more areas of the system, putting 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 IBM 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 the 64-bit PowerPC platform. 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, the University of Massachusetts, and the University of New Mexico. In addition to the 64-bit port, in the period of over two 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 and a partial implementation of bytecode verification.

In June 2003, the structure of the project was formalized with the formation of 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 daily 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 implementers. 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, six 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 elements for the acceptance of an open-source project by open-source developers. First, one must release the code under a license approved by the Open Source Initiative[33] (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; that is, 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 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 the huge success of Jikes RVM 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 explain this reliance, we present an overview of the Jikes RVM build process. Further details are provided in Reference 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 the Jikes RVM, 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 virtual machine. 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 Java virtual machine 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 class loading, this is not as straightforward as it might seem. 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[34] 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 Java virtual machine, it does satisfy the wish of those who want to use only open-source tools when developing the Jikes RVM. Because of this accomplishment, a Debian** package containing the 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.

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 treated promotion of the system as an important goal. This has taken a number of forms, as described next.

Birds of a Feather Sessions. We held these sessions at the premier conferences for programming languages, when the university release was available (at PLDI'01), and when the open-source release was available (at 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 to 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 eight-hour tutorial on the complete system at the PACT (Parallel Computing Technologies) conference in Barcelona. In June and October 2002, we presented tutorials on the details of the optimizing compiler and adaptive optimization system at PLDI and OOPSLA, respectively. In October 2004, we presented a tutorial on the MMTk garbage collection toolkit at ISMM (International Symposium on Memory Management).

Online Information. The Jikes RVM home page[24] 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 CVS repository supporting browsing, 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 toward (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 implementers 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.

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: by direct access from the CVS repository or by downloading it 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 unavail-

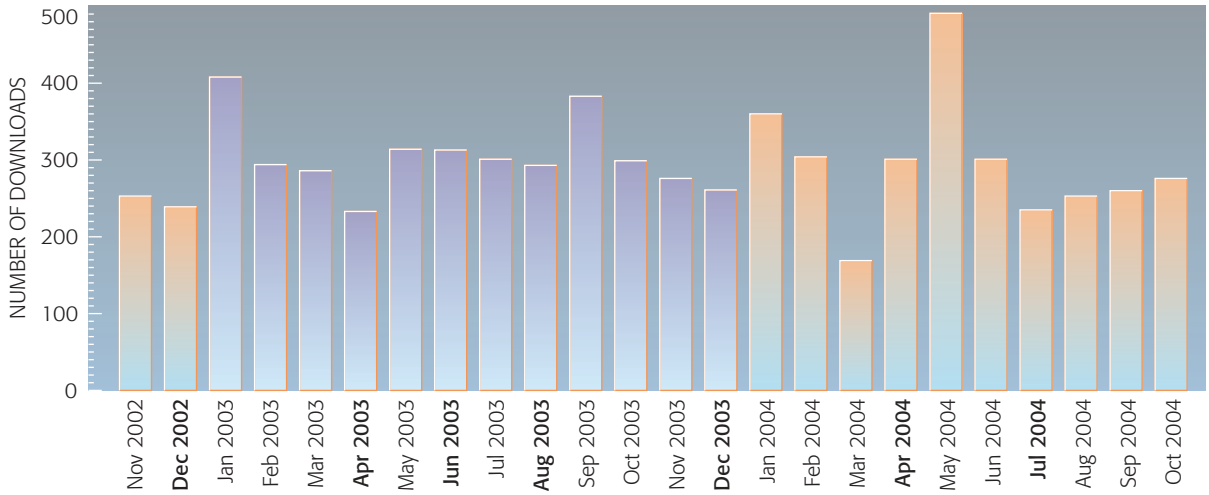

Figure 1
Downloads per month

able for two weeks. The figure shows an average of 300 downloads per month, with significant monthly variations. On the *x* axis, months when a new release was made appear in bold font, such as December 2002. Not surprisingly, the figure shows an apparent trend, wherein the month following a release has a large number of downloads, with smaller numbers in subsequent months 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 that performs its research on a locally modified version of the system. For such users, downloading a new release and then porting modifications is not always desirable, despite the improvements in functionality and robustness. Thus, some users may derive 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, and thus, they are not counted in the download metric.

Another way to measure user activity is to monitor the main mailing list, jikesrvm_researchers. ***Figure 2***

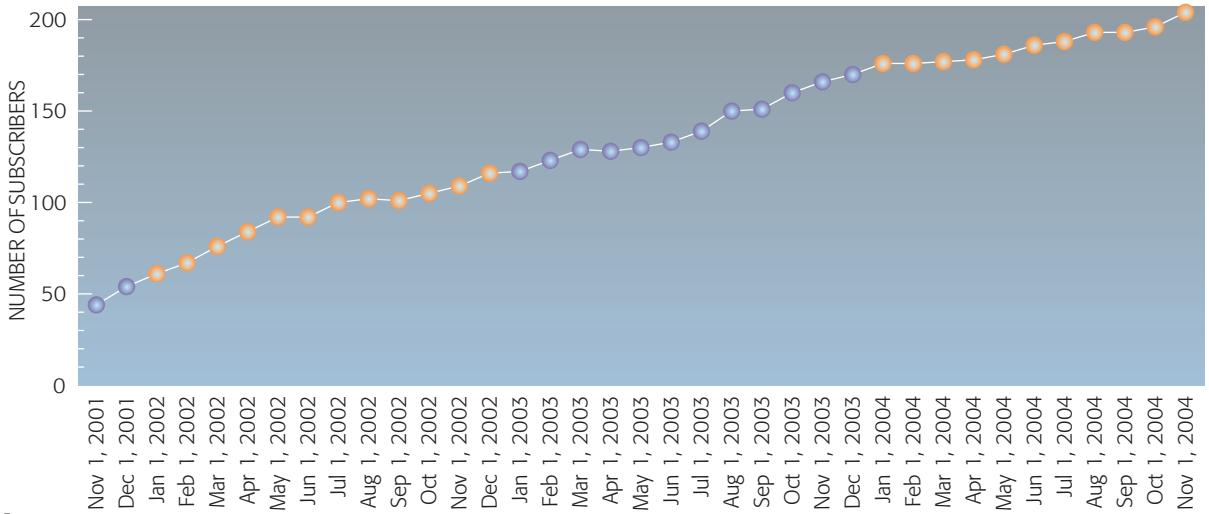

Figure 2
Mailing-list subscribers

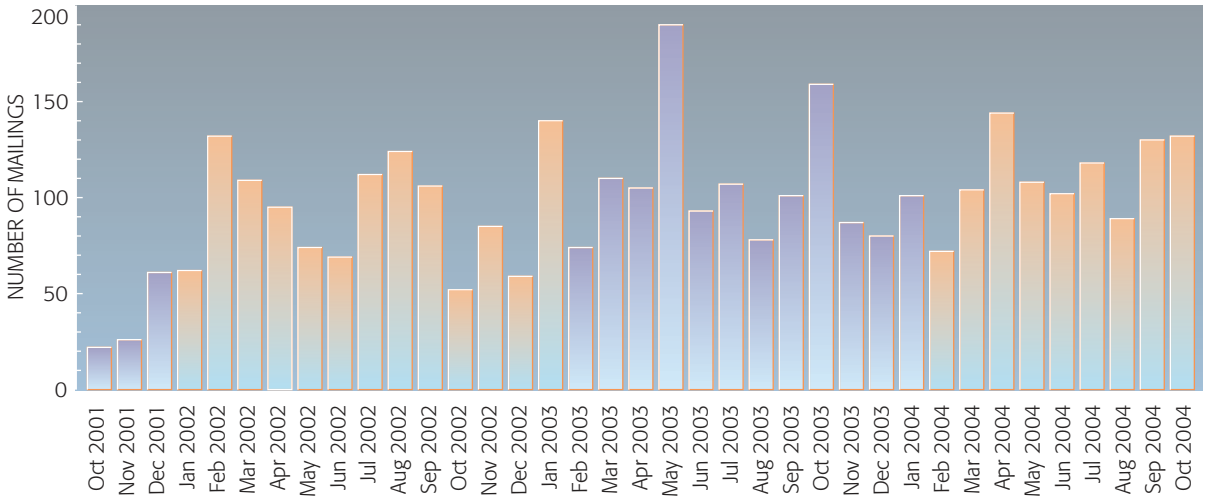

Figure 3
Mailing-list traffic

presents the number of people subscribing 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 new subscribers per year. The number of IBM employees subscribed to this list has varied from 17–19 people over the time period 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, making this metric useful but not perfect.

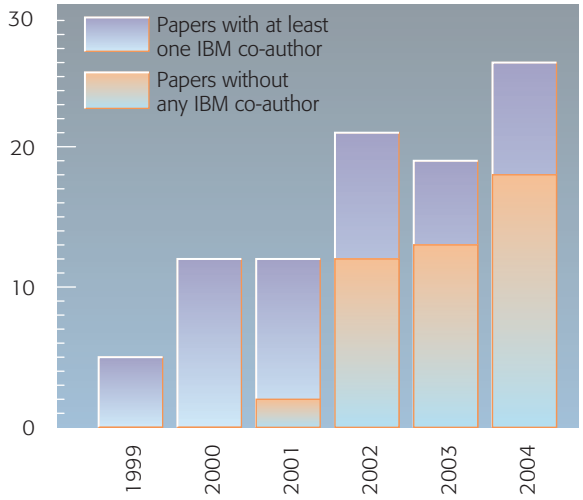

Figure 4
Publications using Jikes RVM

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 usually involves 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 of the system are succeeding in this effort. Clearly, this depends as much on the quality of the user community as on that of the system. ***Figure 4*** presents the number of publications describing research 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 understand how the system has impacted researchers outside of its initial developers and

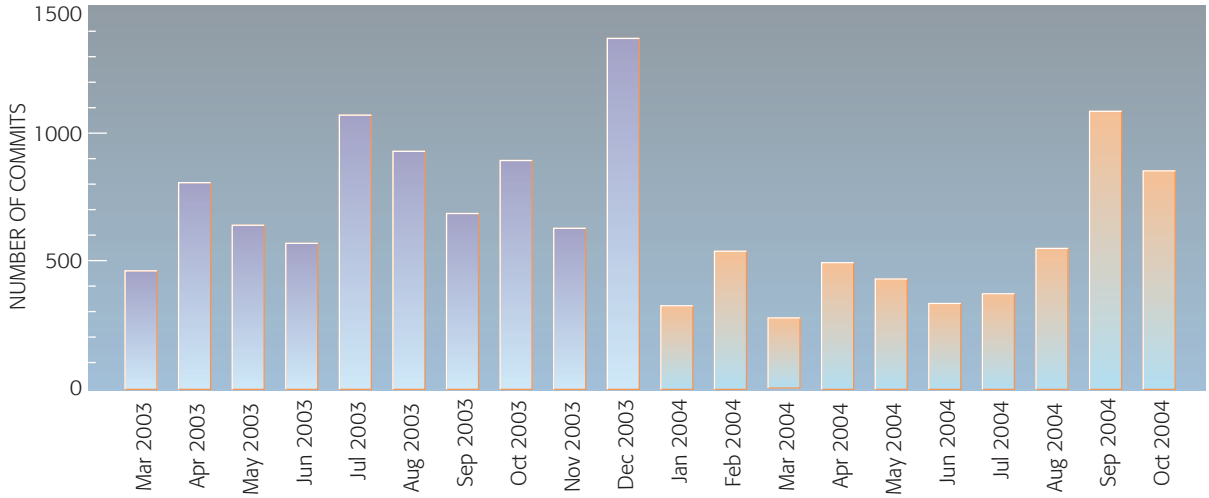

Figure 5
CVS commits

other IBM employees, the figure partitions publications into two categories: publications with at least one author affiliated with IBM, and publications with no IBM authors. This is not a perfect partition because some papers in the former category are written predominantly by non-IBM-related researchers. Thus, the latter category represents a conservative underestimate of 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. As of November 2004, 96 papers (that we are aware of) have been published describing research that used the system. A complete list of these 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 over 60 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, 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 the section "Preparing for open source," 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 Ph.D. candidate recruiting. The project has seen evidence of this phenomenon. In particular, the existence of the project has attracted dozens of graduate students interested in internships, and resulted in at least four new Ph.D. hires over the past three years.

CONCLUSIONS AND LESSONS LEARNED

This paper has described 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 has a voracious appetite for information on a 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 an outstanding

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 a high-quality infrastructure with which researchers the world over can reproduce and improve on published results. Open-source sharing makes all this possible.

The trade-offs between flexibility and maintainability present difficult problems. Software designed for research use should be flexible enough, and provide a feature set rich enough, 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 the tension between these forces and must carefully manage a system's evolution to strike a judicious balance.

When a systems software research project is started, it is prudent to assume that the infrastructure will be adopted by a large community and to manage the software development accordingly. One should 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, it may not be worth pursuing.

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 a myriad of 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 infrastructures such as Jikes RVM will change this, 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. Thus, most researchers do not have a 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 mind-set 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.

ACKNOWLEDGMENTS

Many people have contributed to the success of the Jalapeño project and the Jikes RVM open-source project. We thank Dick Attanasio, Stephen Smith, Derek Lieber, Janice Shepherd, Arvin Shepherd, Matthew Arnold, David Bacon, Peter F. Sweeney, Igor Pechtchanski, Harini Srinivasan, JongDeok Choi, Mauricio Serrano, and John Whaley for their contributions to the initial Jalapeño system and the Jikes RVM release. We thank Jim Russell, John Barton, Mark Wegman, Dan Yellin, and Alfred Spector for their support of this project. We thank Julianne Young, Sergiy Kyrylkov, Jeff Palm, and Dave Hovemeyer for their efforts in the Eclipse on Jikes RVM Extreme Blue project. We thank the GNU Classpath developers for providing the libraries that are used in Jikes RVM. We thank David R. Hanson, Christoper W. Fraser, and Todd Proebsting for making available the "iburg" tool, which we have enhanced for use in Jikes RVM. We thank John Leuner for contributing the initial port to the Classpath libraries.

We also thank Feng Qian, Martin Hirzel, and Kim Hazelwood Cettei for performing night sanity guru duties during their summer internships. Finally, we thank fellow core team members Feng Qian, Peter F. Sweeney, and Chris Hoffman for their support of the system and Laureen Treacy for feedback on earlier drafts of this paper.

CITED REFERENCES AND NOTES

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.org Main Page, 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 and 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**, No. 1, 211–238 (2000).
10. B. Alpern, C. R. Attanasio, A. Cocchi, D. Lieber, S. Smith, T. Ngo, J. J. Barton, S. F. Hummel, J. C. Sheperd, and M. Mergen, "Implementing Jalapeño in Java," *Proceedings of the 1999 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA'99), ACM SIGPLAN Notices* **34**, No. 10, 314–324, ACM, New York (1999).
11. M. G. Burke, J. Choi, S. Fink, D. Grove, M. Hind, V. Sarkar, M. J. Serrano, V. C. Sreedhar, H. Srinivasan, and J. Whaley, "The Jalapeño Dynamic Optimizing Compiler for Java," *Proceedings of the ACM 1999 Java Grande Conference*, 129–141, ACM, New York (1999).
12. M. Arnold, S. Fink, D. Grove, M. Hind, and P. F. Sweeney, "Adaptive Optimization in the Jalapeño JVM," *Proceedings of the 2000 ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA'00), ACM SIGPLAN Notices* **35**, No. 10, 47–65, ACM, New York (2000).
13. C. R. Attanasio, D. F. Bacon, A. Cocchi, and S. Smith, "A Comparative Evaluation of Parallel Garbage Collector Implementations," *Proceedings of the 14th International Workshop on Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science **2624**, 177–192, Springer-Verlag, Germany (2001).
14. A system that maps *m* Java threads to *n* operating-system threads by inserting potential yield points during the compilation of methods. When an external event, such as a timer interrupt or a garbage collection request, occurs, an executing thread will call the scheduler upon execution of the next yield point.
15. W. F. Tichy, "RCS: A System for Version Control," *Software—Practice and Experience* **15**, No. 7, 637–654 (July 1985).
16. Concurrent Versions System (CVS) Domain Home Page, http://www.cvshome.org.
17. J. Choi and H. Srinivasan, "Deterministic Replay of Java Multithreaded Applications," *Proceedings of the SIGMETRICS Symposium on Parallel and Distributed Tools*, 48–59, ACM, New York (1998).
18. M. Serrano, R. Bordawekar, S. Midkiff, and M. Gupta, "Quicksilver: A Quasistatic Compiler for Java," *Proceedings of the 2000 ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA'00), ACM SIGPLAN Notices* **35**, No. 10. 66–82 (October 2000).
19. V. C. Sreedhar, M. Burke, and J. Choi, "A Framework for Interprocedural Optimization in the Presence of Dynamic Class Loading," *Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI), ACM SIGPLAN Notices* **35**, No. 5, 196–207 (May 2000).
20. B. Alpern, A. Cocchi, and D. Grove, "Dynamic Type Checking in Jalapeño," *Proceedings of the USENIX Java Virtual Machine Research and Technology Symposium (JVM'01)*, 41–52, USENIX Advanced Computing Systems Association, Berkeley, CA (April 2001).
21. S. Blackburn, P. Cheng, and K. McKinley, "Oil and Water? High Performance Garbage Collection in Java with JMTk," *Proceedings of the Twenty-Sixth International Conference on Software Engineering*, 137–146, IEEE Computer Society, Washington, D.C. (May 2004).
22. S. Blackburn, P. Cheng, and K. McKinley, "Myths and Reality: The Performance Impact of Garbage Collection," *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, ACM SIGMETRICS Performance Evaluation Review* **32**, No. 1, 25–36, ACM, New York (June 2004).
23. B. Alpern, M. Butrico, A. Cocchi, J. Dolby, S. Fink, D. Grove, and T. Ngo, "Experiences Porting the Jikes RVM to Linux/IA32," *Proceedings of the Second USENIX Java Virtual Machine Research and Technology Symposium (JVM'02)*, 51–64, USENIX Advanced Computing Systems Association, Berkeley, CA August 2002.
24. Jikes RVM Home Page, http://jikesrvm.sourceforge.net.
25. 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**, No. 1, 175–193 (2000).
26. N. Grcevski, A. Kilstra, K. Stoodley, M. Stoodley, and V. Sundaresan, "Java Just-in-Time Compiler and Virtual Machine Improvements for Server and Middleware Applications," *Proceedings of the Third USENIX Java Virtual Machine Research and Technology Symposium (JVM'04)*, 151–162, USENIX Advanced Computing Systems Association, Berkeley, CA (May 2004).
27. An informal meeting at a conference where people with similar interests gather. This is in contrast with the regular conference program which is much more formal and of general interest.
28. M. Poletto and V. Sarkar, "Linear Scan Register Allocation," *ACM Transactions on Programming Languages and Systems* **21**, No. 5, 895–913 (September 1999).
29. *SPECjvm98 Benchmarks*, Standard Performance Evaluation Corporation, http://www.spec.org/jvm98.
30. *SPECjbb2000—Java Business Benchmark* 2000, Standard Performance Evaluation Corporation., http://www.spec.org/jbb2000.
31. GNU Classpath—GNU Project—Free Software Foundation, http://www.gnu.org/software/classpath.
32. S. M. Blackburn and K. S. McKinley, "Ulterior Reference Counting: Fast Garbage Collection without a Long Wait," *ACM SIGPLAN Notices* **38**, No. 11, 344–358 (November 2003).
33. Open Source Initiative (OSI), http://www.opensource.org.
34. Kaffe.org. Home Page, http://www.kaffe.org.

Accepted for publication November 30, 2004.
Published online April 13, 2005.

Bowen Alpern
IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (alpernb@us.ibm.com). Dr. Alpern received a B.S. degree from the University of Michigan in 1974 and a Ph.D. degree 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). Mr. Augart is a freelance software engineer. He first did free software work for the GNU project in 1983. He received a Bachelor's degree from Harvard University in 1989 and an M.S. degree from the University of California at Irvine in 1992. After a varied career including seven years at the University of Southern California 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, real-time 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). Dr. 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 University of New South Wales in 1990 and 1992, and his Ph.D. degree from the Australian National University in 1997. During postdoctoral work 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). Ms. 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 interests 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's and a Master's degree in electrical engineering from Pratt Institute (in Brooklyn, New York) and a Master's degree in computer science from New York Polytechnic Institute. On the Jikes RVM project, he worked on the garbage collector, the runtime, and the basic compilers, and did performance measurements 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 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 programming language design and implementation, virtual machines, and runtime systems. He is a member of the ACM.

Julian Dolby
IBM Research Division, Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, New York 10532 (dolby@us.ibm.com). Mr. Dolby is a research staff member in the Software Technology Department at the IBM Watson Research Center. He attended the University of Wisconsin at Madison and the University of Illinois at Urbana-Champaign. He was a member of the team that developed the 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. Watson Research Center. He received a B.S. degree from Duke University in 1992 and 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 JavaBeans. His research interests include programming-language implementation techniques, program analysis, and parallel and scientific computation. He is a member of the 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 Watson Research Center. He received a B.S. degree from Yale College in 1992, and 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 programming language design and implementation, virtual machines, and adaptive optimization. He is a member of the 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 Watson Research Center. He received a B.A. degree from the State University of New York at New Paltz in 1985 and M.S. and Ph.D. degrees from New York University in 1987 and 1991. From 1992 to 1998, Dr. Hind was a professor of computer science at the State University of New York at New Paltz. In 1998, he joined IBM to work on the Jalapeño project, particularly 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 the 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). Dr. McKinley is an associate professor at the University of Texas at Austin. She received her Ph.D. degree in 1992, Master's degree in 1990, and Bachelor's degree in 1985, all from Rice University. Her research interests include compilers, memory management, and architecture. She has received two IBM Faculty Awards (2003, 2004) and an NSF CAREER Award (1996, 2000); she is an IEEE Senior Member; she was awarded 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 was the program chair for the ACM conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) 2004, and is the program chair for 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 non-optimizing 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 on the K42 open-source, scalable, customizable operating-system 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 at 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 languages while a graduate student, and contributed the nested transaction model in his dissertation. Since then he has explored persistent object storage 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 (Jalapeño at the time). Dr. Moss received a Ph.D. degree in 1981, an M.S. degree in 1978, and a B.S. degree 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. degrees in computer science from the University of Washington, an M.S. degree in electrical engineering from the Florida Institute of Technology, and a B.S. degree in electrical engineering from the Georgia Institute of Technology. He joined the IBM Systems Product Division in 1982, the IBM Watson Research Center in 1987, and finally the Almaden Research Center in 2001. In the Jikes RVM project, he developed the debugger and the Java Native Interface and contributed to the Record/Replay tool. Currently he is with the WebFountain™ project, working on a cluster of several hundred/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 Watson Research Center, where he oversees research projects in the areas of programming models, programming tools, code optimization, and virtualized runtime systems. He received a B.Tech. degree from the Indian Institute of Technology in Kanpur in 1981, an M.S. degree from the University of Wisconsin at Madison in 1982, and a Ph.D. degree from Stanford University in 1987. From 1998 to 2000, he was manager of the Dynamic Optimization group, which was responsible for the dynamic optimizing compiler and adaptive optimization system in Jalapeño. 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, D90478 Nuremberg, 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 to 2002, he was a member of the Dynamic Optimization Group in the Software Technology Department at the IBM Watson Research Center. His main interests are the reliability of software systems and improvements in software engineering. ■