



SWG Java Technology Centre

Virtual Execution Environments: Opportunities and Challenges

Workshop on the Future of Virtual Execution Environments
September 15, 2004

Bob Blainey
Chief Java Technologist
IBM Software Group
blainey@ca.ibm.com

Caveats

- This is not meant to be an exhaustive analysis of VEE benefits and challenges
- It is based on IBM customer experience and the VEE technology we are building for Java
- I'm historically a compiler optimization guy so forgive an occasional bias toward performance 😊

Why are we here?

- **Over the last couple of years, IBM has become both excited and concerned about the divergent development of VEEs, from Java to the CLR to open source scripting environments**
- **Each of these adds unique value**
 - Examples: Java for rock-solid portability, CLR for language interoperability and multi-lingual frameworks and scripting environments for support a more dynamic and simplified programming model
 - Reinforced by the distinct communities growing in support of each of these
- **However, we see our customers increasingly using mixed environments and the pains of resource contention and interoperability are beginning to show up**
- **Our programming model, runtime and hardware roadmap is also placing new demands on the capability and performance of VEEs**
 - Examples: service-oriented architectures, aspect-oriented programming, model-driven execution, evolution in microprocessor and system design
- **We think that many of these problems and opportunities are shared and that there are some that none of us can effectively solve on our own**

Who's here?

- **Industry**

- Azul Systems, BEA, Hewlett-Packard, IBM, Microsoft, Nokia, SAP, Sun Microsystems

- **Academia**

- Princeton, Stanford, SUNY Oswego, UC Irvine, UIUC

- **Open Source Projects**

- Linux, Mozilla, Mono, Perl, Python, Eclipse

Why us?

- **The workshop was invitational because:**
 - We wanted to get some of the best minds in VEE research and practice to share their concerns and their visions for the future
 - We wanted a diverse mix of researchers and practitioners drawn from industry, academia and open source – we couldn't be more pleased with the group of our colleagues that have agreed to participate
- **I can't predict the outcome of this workshop yet but I would be satisfied if we come to a common understanding of each others concerns and goals and leave with new or strengthened relationships**
- **If nothing else, we'll be well fed and can enjoy life in the woods for a couple of days 😊**

VEE challenges and opportunities

Essential elements of a Virtual Execution Environment

- A VEE is **virtual** because its behavior and interfaces can be specified independently of any specific hosting machine or operating system.
- Programs running under a VEE can be **inspected, analyzed or transformed** by themselves or other software through the availability of rich type and structural information.
- The VEE provides a **container** for the execution of programs which can provide clients outside the VEE protection against faults and security violations.

Some observed benefits of application-level virtualization

- Application-level virtualization provides **portable, well-specified behaviour** for program execution and selected, high-level operating system interfaces.
 - Virtualization represents a factoring of the runtime into an API friendly to compilers and programmers and an implementation that can vary and be optimized for each platform
- Access to critical operating system or application resources can be controlled through VEE-enforced **security** mechanisms.
- Programmers can achieve higher levels of **productivity** through
 - VEE safety assurances
 - automated memory management
 - sophisticated analysis and debugging tools
 - automatic support for distribution and persistence
- High levels of **performance** are possible through dynamic and adaptive program transformation.
 - But we are just on the frontier of exploiting this capability

Value to Developers

- Developers spend less time authoring, debugging and supporting system-specific code in their applications. The result is ***fewer bugs and broader availability of software*** across varying platforms.
- Support for reflection enables better separation of infrastructural program elements (such as persistence and distribution) from program logic. Developers can ***focus more of their effort on core application behaviour*** and less on common infrastructure code.
- ISVs leverage their investment in software more effectively leading to ***faster time to market and increased volume and revenue opportunities***.
- A portable execution format reduces the number of artifacts that need to be produced and packaged for multi-platform software, ***reducing build and delivery cost and complexity***.

Value to Users and Customers

- ***Tolerance to and recovery from faults can be improved*** in virtual execution environments because of isolation from host operating environments and the ability to efficiently serialize VEE state
- Due to the adaptive nature of VEE performance, customers can receive (in theory, although rarely in practice) ***software which can run best on their workloads*** rather than whatever workloads were used by the developer to do tuning.
- Tools for the management and deployment of VEEs can result in systems which are ***easier to administer***, provide a ***lower total cost of ownership*** and ***improve the ability to respond to change***.

IBM's interests

- IBM is deeply invested in VEEs (mainly Java) both to enhance our server platforms (eg. via tuned virtual machines and dynamic compilers) and as an underlying portable runtime for most of our software portfolio
 - we clearly want to protect our investment and look for even more ways to benefit from VEEs
- Our WebSphere programming model (based on J2EE) is predicated on many elements of VEEs including dynamic loading, introspection, automated support for serialization and so on
- Deployment and management of software depends on high levels of VEE isolation, monitoring and control
- IBM is and always has been a very strong supporter of research in programming technologies

Some problems with current VEEs

- **Performance**

- Virtualization has a cost
- High performance JIT compilers and garbage collectors have mitigated the problem to a degree
- But problems remain due to non-uniform quality of VEE implementations and common blind spots such as large footprint and slow startup time.

- **Device support**

- Virtualization can be even more valuable in the diverse space of mobile devices and operating environments
- But VEEs are normally not designed for small and heterogeneous memories nor for relatively slow processors

- **Native embedding and extension**

- Crossing the barrier between virtual and native code remains expensive and difficult.
- VEEs are typically not designed specifically for embedding (hosted inside native applications) or extension (utilizing native code services)
- Generally poor facilities for sharing of resources such as threads and memory between native and virtual environments

More problems with VEEs

■ Interoperability

- Interoperation between different languages with VEE runtimes is rarely available.
 - The CLR is an exception but not everyone is satisfied with the CLR as a universal compilation target
- Another approach is to allow the design of a programming language and virtual machine without constraints but also allow easy, efficient, and relatively fine grained communication with other virtual machines
 - Allows more freedom to language implementors but limits expressiveness of interfaces to the intersection set of intended client languages

■ Complex behaviour

- VEEs can have very complex behaviour due to dynamic compilation and optimization, automatic memory management, multithreading and so on
- Debugging and sometimes even reproduction of failures can be very time consuming and difficult to understand
- Performance analysis and tuning can be even more difficult

Opportunities for VEEs

- **Programming language choice**

- There is currently a healthy divergence of programming language choice and the emergence of new languages
- This is good for users because they use the right tool for the job
- It also encourages innovation
- However, interoperability, sharing of system resources and the factoring of runtimes to share things like dynamic compilation and garbage collection are important practical concerns

- **Advanced debugging and availability**

- Rich type and structural information and the ability of the runtime to adapt to failure opens interesting opportunities for improving debugging capability and resiliency
- Indeed, one could argue that this is needed even more for VEEs because of their complex, dynamic behaviour

- **Service-oriented architectures and distributed programming models**

- VEEs present an opportunity to abstract and optimize potential distribution points via runtime adaptability (eg. dynamic proxy generation, dynamic interface specialization)

More opportunities for VEEs

- **Software security and DRM**

- Standards in this space can be better leveraged in a VEE due to common program representation and loading semantics and VEE safety guarantees (verification)
- Examples: copy protection, obfuscation, authentication, program integrity protection

- **Hardware innovation**

- Target-specific dynamic compilation enables greater instruction set and microarchitecture variability (eg. multicore, pipeline configurations) while ensuring more rapid utilization
- Profiling and other types of instrumentation can be used to drive code optimization or garbage collection data placement strategies

- **High performance computing**

- Large scale parallelism
- Adaptive optimization

Key messages for this workshop

- The attendee list is intentionally eclectic, drawing from industry, academia and open source communities
 - Use this opportunity to learn about these other communities and build networks
- As we jointly proceed with our research and development on VEEs, remember to draw on the experience of those that have built great VEEs in the past (LISP, Smalltalk, ML). Some of the people who built those systems are here. Get to know them.
- For the sake of making progress and making the right kinds of design decisions, we need to explore ways that VEEs can share resources and interoperate.
- BUT, we need to be vigilant about ensuring that innovation in VEEs and programming environments is supported and nurtured

Acknowledgements

- **The IBM organizing committee**
 - Paul Buck (Java Strategy)
 - John Duimovich (Virtual Machines)
 - Jim Rymarczyk (Systems Group)
 - Vivek Sarkar (Programming Technologies Research)
 - Pat Selinger (Data Management)
 - Kevin Stoodley (Compilation Technology)
 - Mark Wegman (Software Research CTO)
- **And especially ..**
 - Mike Hind (Dynamic Optimization Research)

Questions? Discussion?