The SAP VM Container is an application server framework that uses a new paradigm of Process Attachable Virtual Machines (PAVMs) to combine high scalability with strong isolation between user sessions. In this short paper, we present the design and implementation of the PAVM paradigm for Java virtual machines.

1 Introduction

When it comes to mission critical large enterprise server environments, robustness and scalability become main concerns for server virtual machines.

From a most general point of view, enterprise servers can be characterized as request processing engines, serving large numbers of (typically small) user requests that belong to user sessions. The actual request processing involves running user code (e.g. Servlets, EJBs) in a virtual machine. Request throughput is the main objective for scalability, which is traditionally achieved with thread pool based architectures.

System robustness necessarily requires strong isolation between user sessions, something that is often hard to achieve if a large number of user sessions are run within a single virtual machine.

On the other hand, operating systems provide near perfect isolation for processes - even a crashing process does not affect any other process nor does it leave behind resource leaks. Why not just provide each user session with a virtual machine and an OS process of its own? The answer is obvious: resource consumption by hogging the CPU. Threads and monitors in user code deserve some special attention. Native OS threads cannot easily be persisted to shared memory, so PAVMs can only provide "green threads" functionality to user code. All data structures related to thread management and scheduling (including the threads' call stacks, mutexes and condition variables for Java monitors) are kept directly in session memory. This includes both the Java stacks and the C stacks used by the VM implementation, e.g. for the JNI implementation of dynamic method invocation. OS coroutines (see [2] for Unix) are employed to control the location of C stack memory.

Thread scheduling is coroutine-based and thus non-preemptive, which seems a drawback at first glance. But it turns out that for the special case of a request processing engine as described above, preemptive scheduling is actually less desirable than one might expect. In order to maximize request throughput (as opposed to fairness), a batch processing strategy is employed instead: within each PAVM, threads yield cooperatively to the thread scheduler when entering wait state (i.e. blocking on I/O or a Java monitor). The coordination between blocking I/O calls and the thread scheduler is part of the I/O redirection mecha-

---

1 Very coarse preemption (implemented with timers and signals) is additionally used to prevent user sessions from monopolizing a work process by hogging the CPU.
The work of Dillenberger et al. [5], PAVMs could be initialized “master” PAVM at system startup. This can be achieved by copying the session memory, which is easily obtained by initializing each PAVM’s static initializers in system classes). This can be avoided by initializing each PAVM’s session memory from a suitable “template” image, which is easily obtained by copying the session memory of a freshly initialized “master” PAVM at system startup.

This approach can be taken even further. Following the work of Dillenberger et al. [5], PAVMs could be implemented to be **serially reusable** after the corresponding user session ends. This is particularly effective for very short user sessions or stateless user requests.

An API for Java isolation is currently being defined by JSR 121 [6]. We expect that the PAVM paradigm can assist in the implementation of JSR 121 for server environments.

### 6 Conclusion and work in progress

We have presented a short overview of the PAVM paradigm within the SAP VM Container, its benefits and restrictions and outlined possible optimizations with respect to related works.

The PAVM paradigm is currently being implemented for Java VMs to be part of the SAP Web Application Server, where the SAP VM Container technology has been used quite successfully for more than 10 years as the foundation for SAP’s R/3 system, an application server running a comprehensive suite of business applications by means of a SAP specific programming language and virtual machine named ABAP (Advanced Business Application Programming).

Sun’s CVM codebase from J2ME is used as the starting point for SAP’s PAVM implementation. A working prototype running Servlets and JSPs demonstrates feasibility (5/2002), but does not yet allow any performance measurements and still lacks the possible optimizations outlined above.

### References


