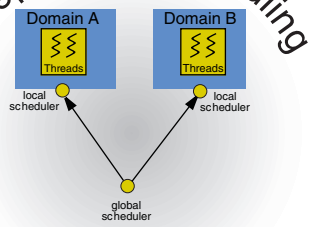
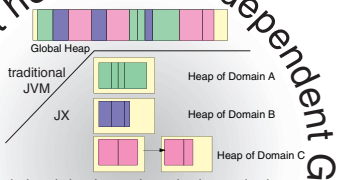
JX - A flexible Java Operating System

independent scheduling



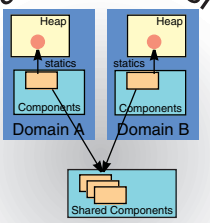
Threads of one domain can be scheduled independently of other domains. It is possible to use domain-specific schedulers that are written in Java.

independent heaps and independent GC



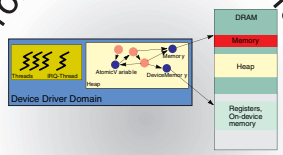
Each domain has its own heap that is completely independent from the heaps of other domains. This is in contrast to namespace-based domains of traditional Java systems and allows independent garbage collection runs, domain-specific garbage collection implementations, and domain termination with immediate resource reclamation.

transparent code sharing

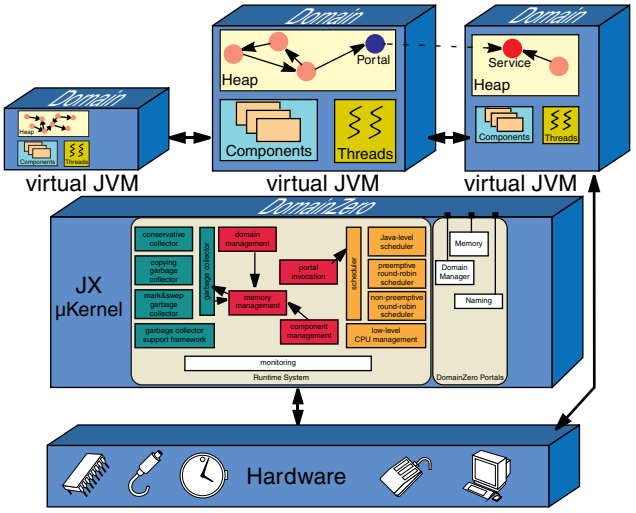


Immutable state can be shared between domains. This includes compiled code and string constants. Mutable state, such as static variables, is created per domain which makes code sharing completely transparent.

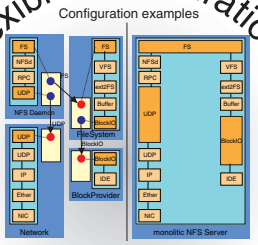
robust device drivers



All device drivers are written in Java and benefit from the robustness that is provided by type safety. Robustness is further supported by special portal interfaces that allow drivers to access device registers and interrupt handler threads to synchronize with other threads.

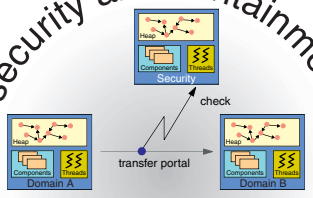


flexible configuration



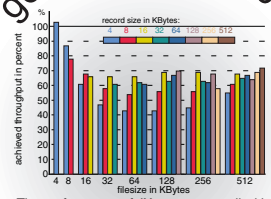
The protection domain structure is independent of the component structure. Components can be placed into the same or in separate domains without changing them.

security and containment



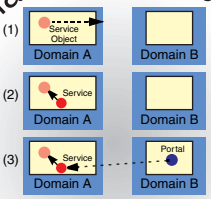
Domains are isolated and can communicate only via dedicated channels: the portal objects. The JX security system can control the propagation of all portals as well as their use. The security system that contains the security policy is separated from the domains and can be exchanged without changing the application domains.

good performance



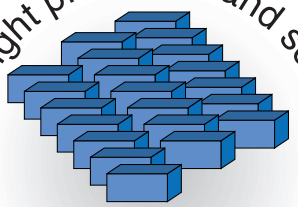
The performance of JX compares well with a traditional UNIX system, such as Linux. Our ext2-compatible file system, which is completely written in Java, achieves a throughput of 50 to 100 percent of Linux in the IOZONE re-read benchmark.

automatic service creation



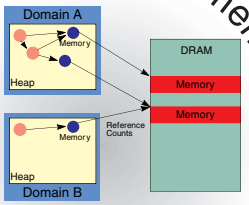
A service is automatically created when a service object is passed to another domain. This includes the creation of a new service thread or the addition of the service to an existing thread pool. The other domain receives a portal to the service.

light-weight protection and scalability



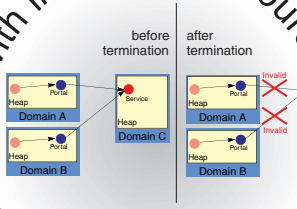
A protection domain is a complete Java execution environment but has a very low memory footprint. Creating 1000 domains that concurrently print "Hello World!" in a loop requires about 35 MBytes, creating 1000 domains that concurrently serve 1000 TCP connection needs 105 MBytes.

automatic management of memory areas



JX provides a special abstraction – memory portals – to cope with large memory or special memory areas, such as memory-mapped device registers. Memory portals can be shared between domains and are garbage-collected.

termination with immediate resource reclamation



Except device driver domains all domains can be terminated immediately and their resources can be released completely.

© 2002 JX-Team

University of Erlangen-Nuremberg
Department of Computer Science
Michael Golm • Meik Felser • Christian Wawersich • Jürgen Kleinöder