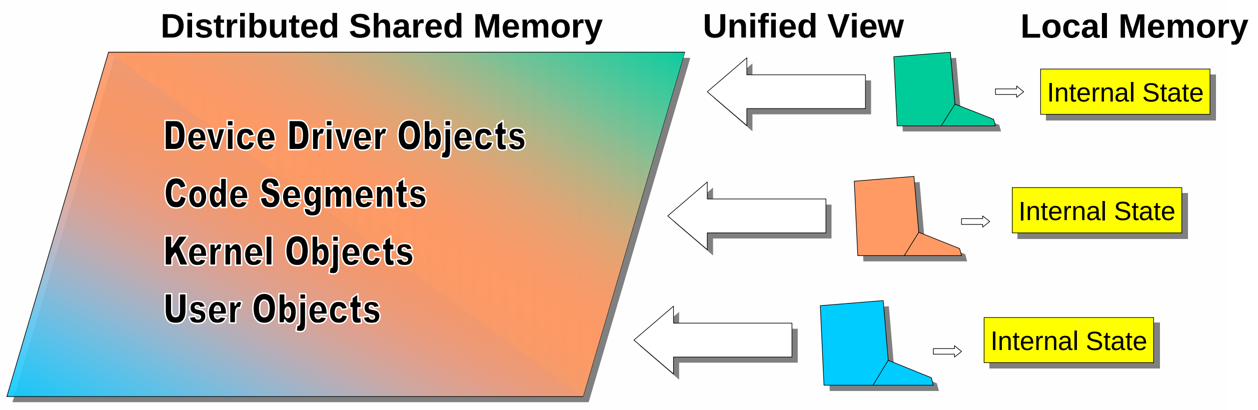
2003 IEEE International Conference on CLUSTER COMPUTING
HONG KONG

A Kernel Running in a DSM

R. Goeckelmann, M. Schoettner, S. Frenz, P. Schulthess



The Plurix project implements an object-oriented Operating System (OS) for PC clusters. Communication is achieved via shared objects in a Distributed Shared Memory (DSM). The consistency of this distributed memory is guaranteed by an optimistic synchronization scheme and restartable transactions. We contend that coupling object orientation with the DSM property allows quick system startup, simplified development of distributed applications and a type-consistent system bootstrapping procedure. The OS (including kernel and drivers) is written in Java using our proprietary Plurix Java Compiler (PJC) to translate Java source code directly into Intel machine instructions.

Plurix implements Single System Image (SSI) properties at the operating system level, using a page-based distributed shared memory.

According to the SSI concept all programs and libraries must be available on all nodes in the cluster, hence Plurix uses a global address space shared by all nodes and organized as Distributed Heap Storage (DHS) containing both data and code. Sharing the programs in the DHS reduces redundancy concerning code segments and simplifies the administration of the system.

Plurix works in a fully object oriented fashion and is entirely written in Java. The development of an operating system and its drivers requires access to device registers which is not possible in standard Java.

To support operating system and hardware level programming we have developed our own Plurix Java Compiler (PJC) with appropriate language extensions.

The compiler directly generates Intel machine instructions and initializes runtime structures and code segments in the heap. Traditional object-, symbol-, library- and exe-files are avoided. Each new program is compiled directly into the DHS and is thereby immediately available at each node.

Plurix is a lean and high speed OS and therefore able to start quickly. The start-up time of the primary node, which creates a new heap or restarts a preexisting heap from the PageServer, is less than one second (excluding BIOS). Additional nodes which join the existing heap start in approximately 250 ms. This quick-boot operation of Plurix is also crucial for achieving fast node and cluster recovery in case of (unlikely) critical errors.

The transfer of the DHS-objects between cluster nodes is managed within the page-based DSM and takes advantage of the Memory Management Unit (MMU) hardware. The MMU detects page faults, which are raised if a node requests an object on a page which is not locally present. Each page fault results in a separate network packet which contains the address of the missing page (PageRequest). This packet is broadcast to all nodes in the cluster (Fast Ethernet LAN) and the current owner of the page sends it to the requesting node.

In Plurix consistency of shared and replicated objects is synonymous to the consistency of the entire DSM. Memory consistency is achieved using a new consistency model: *transactional consistency*.

Appropriately all actions are encapsulated in transactions. At the start of a transaction, write access to all pages is prohibited. If a page is written, the system creates a shadow image of it and then enables write access. All modified pages are logged and at the end of a transaction (commit phase) these addresses are broadcasted. All partner nodes in the cluster will then invalidate these pages. If an active transaction in a partner node detects a collision with the write set of the committing transaction it aborts and restarts later. In this case all modified pages are discarded and the previous state of the node is reconstructed, using the saved shadow images. A token mechanism ensures that only one node at a time enters the commit phase.

To increase system reliability and to limit programming complexity it is preferable to pass typed objects to the OS-kernel. Plurix was therefor created as microkernel OS. Since the kernel is written in Java, a type-safe communication between the DSM applications and the OS is natural. All Java types and objects can be handed to the kernel methods. The kernel method obtains a reference and accesses the object directly, therefor data included in these objects need not be copied.

If the OS kernel runs in the DSM space, parameter passing between applications and kernel is elegant and all objects can be used as parameters. Kernel methods are called directly. There are no references between different address spaces.

Plurix

Distributed Lean
Java Transactions