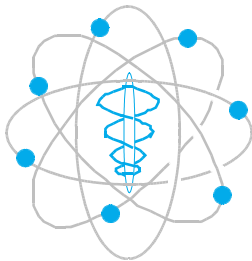


Type-Safe Object Exchange between Applications and a DSM-Kernel



University of Ulm
Germany

- Major goals of Plurix
- Type-safety in OS development
- The Plurix Project
- DHS and transactional consistency

- Where to store kernel and applications?
- Inter address space pointers
- A kernel running in the DHS

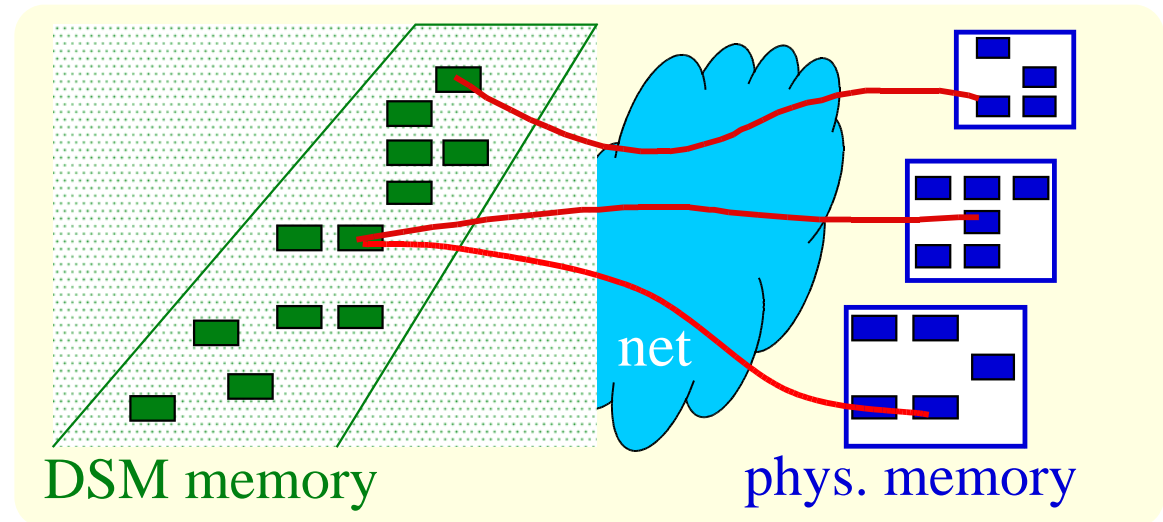
- Measurements

Speaker: Ralph Goeckelmann



Major Goals of the Plurix Project

- Distributed operating system for PC cluster
- Simplified development of distributed applications
 - implicit synchronisation
 - automatic garbage collection
- Using DSM for communication
- Lean system design
 - stand alone OS
- Object orientation



Traditional Operating Systems

- Most distributed OS implemented as Middleware
- Host OS mostly written in C / C++
- OS is not type-safe
 - no protection of code segments
 - no type-safe parameter passing
- ➔ **explicit protection of the kernel is needed**
 - using different address spaces
 - invoking kernel methods by using interrupts
- Challenging parameter passing to the kernel
 - no direct parameter passing possible
- Systems mostly very complex
 - high risk of faults and attacks (viruses / worms)



Type-Safe Operating System

- Using a type-safe language for OS development
- Implicit protection of the kernel
 - modification of pointers not possible
 - only published methods can be called
 - code segments automatically protected
- No different address spaces needed
- No explicit parameter check
- Simplified programming
- Increased security and stability



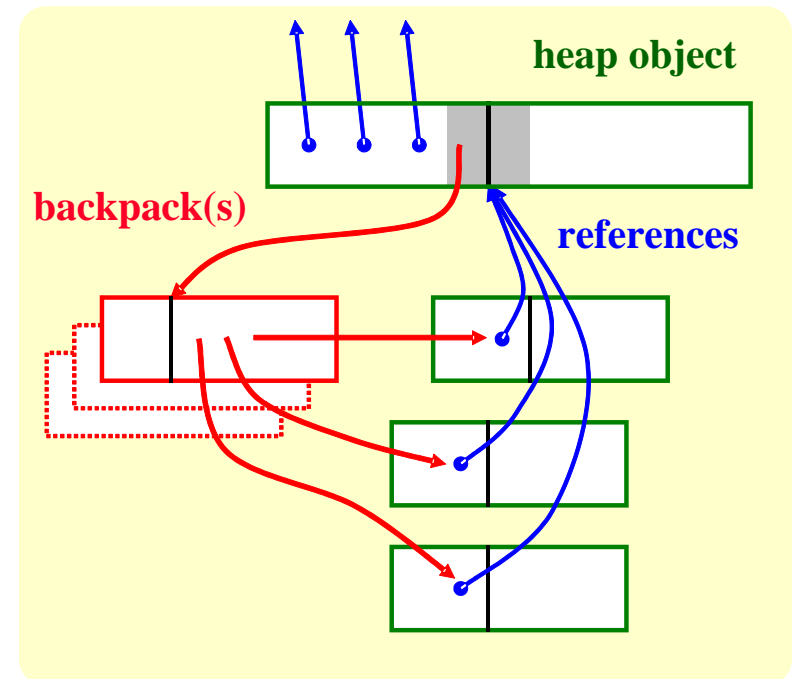
The Plurix Project

- Type-safe language: Java
- Special Java-Compiler
- Only objects in the DSM
 - DSM organized as type-safe heap
- Transactional consistency
- Persistence and fault tolerance
- Unified name service



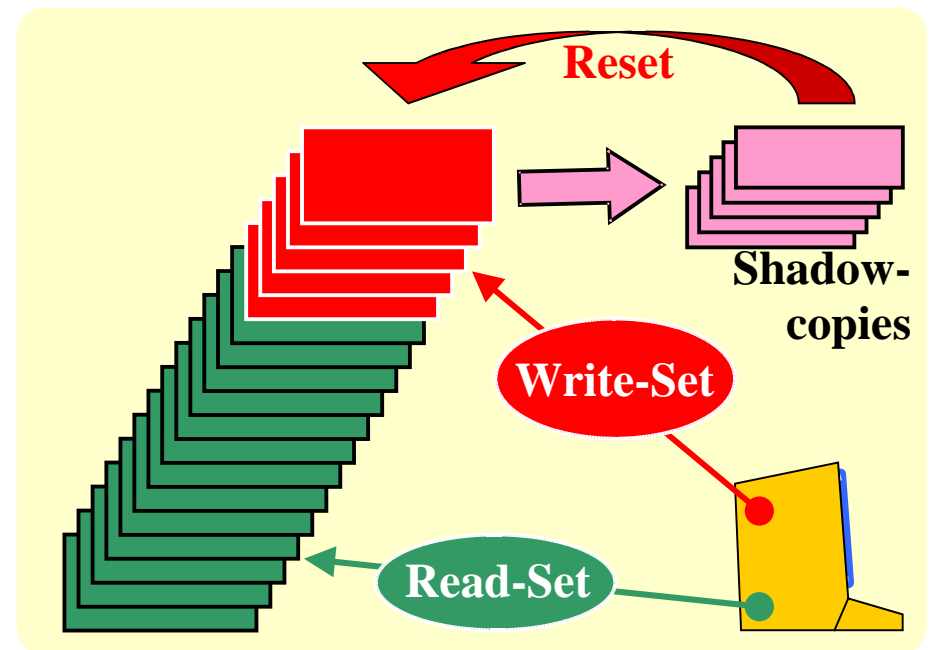
Distributed Heap Storage

- Fragmentation and False-Sharing must be handled
- ➔ **Relocation of objects necessary**
- Relocation of objects in the DHS
 - reallocate object
 - adjust all references
- Bookkeeping of references
 - Backlinks and Backpacks
- Garbage Collection using Backlinks
 - cluster wide and incremental GC
 - objects with empty Backlinks → garbage



Transactional Consistency

- Restartable transactions
 - follow the well known ACID paradigm
 - implicitly defined by the central loop
 - aborted in case of a collision
- Optimistic synchronization
 - writes initially on local copies only, shadow copies are preserved
 - separate Read- and Write-Sets are built during a transaction
 - an ending transaction broadcasts a Commit-Request,
 - **reset on collision**
- strict consistency at commit points
- sequential consistency always preserved



Where to store the applications?

- Transparency requirements for a distributed OS
 - Possible solutions: local memory or DHS
 - Applications generated or modified during runtime
 - Applications in local memory
 - separate installation on each node
 - explicit transferred to each node
 - new applications must be simultaneously installed
 - Applications in the DHS
 - Compiler is running in the DHS
 - new applications are automatically compiled into the DHS
 - Code Segments are automatically shared and distributed
- ➔ Applications should reside in the DHS



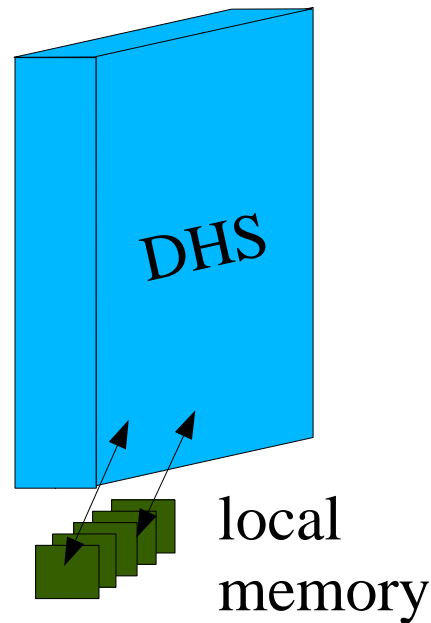
Where to store the kernel?

- Possible solutions: local memory or DHS
- Kernel in local memory
 - obvious approach
 - same requirements as for applications
 - changes must be simultaneously broadcasted to each node
- Kernel in the DHS
 - same benefits as for applications
 - but special attention needed
 - protection of kernel objects against invalidation necessary



Inter Address Space Pointers

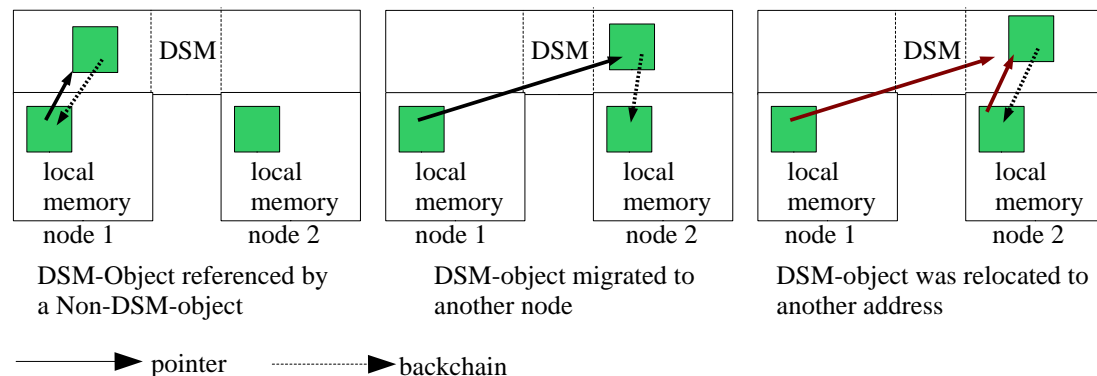
- Direct pointer from local memory into the DHS or Vice versa
- Indirect pointers / translation tables decreases performance
- Problem 1: Typedescriptor
 - typedescriptor resides in the DHS only => Kernel can not start
 - typedescriptor in local memory only => must reside at the same address
 - two sets of typedescriptors => no Java like type-check possible



Inter Address Space Pointers 2

- Problem 2: Relocation of DHS-Objects
 - addresses in local memory not unique
 - relocation of DHS-object requires pointer adjustment
 - difficult adjustment: each node must adjust at the same time
 - automatic adjustment by using the Backlinks not possible
 - relocation is prohibited

➔ Avoiding different address spaces



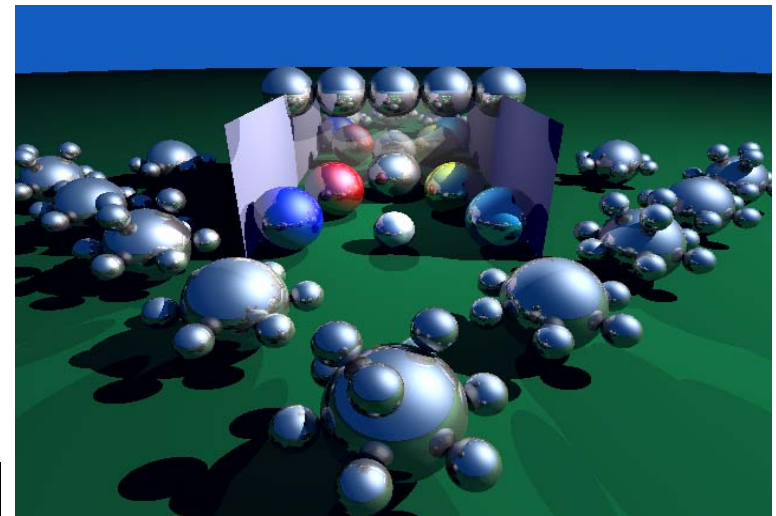
A kernel running in the DHS

- All objects reside in the DHS
- Avoiding inter address space pointers
 - normal type-check possible
 - relocation of objects not prohibited
- Elegant and type-safe parameter passing
- Current kernel version available by the DHS
 - all nodes running the same kernel version
- Ease checkpointing and transaction migration
- But: protection of system-objects needed



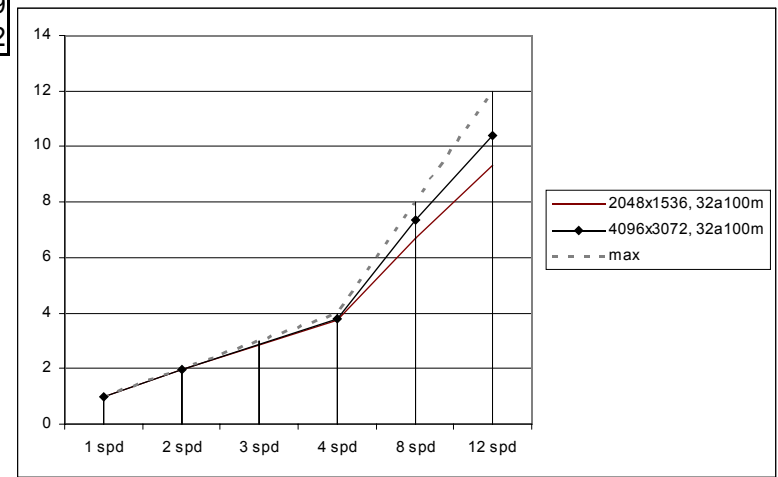
Measurements

- Using a distributed raytracer for performance evaluation
- Used scene:
 - 103 spheres
 - 3 spot lights
 - 8 triangles
- Measurements

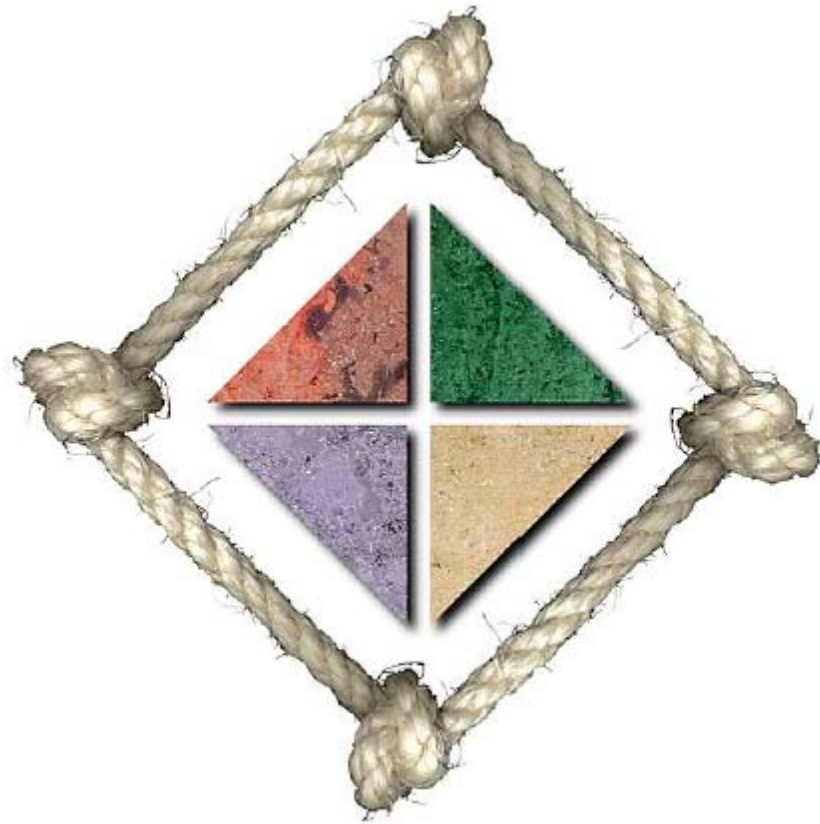


pixels	lfd	1 node	2 max	4 max	8 max	12 max	2 min	4 min	8 min	12 min
307200	1	30969	16844	9890			15941	7551		
602112	2	60265	32344	18062	10878		30762	14789	6926	
3145728	3	306599	157073	81743	45709	32955	155151	80772	41087	26719
12582912	4	1221654	623468	322903	166561	117484	620206	313608	162127	107382

lfd	2 spd	3 spd	4 spd	6 spd	8 spd	12 spd
1	1.84	2.49	3.13			
2	1.86	2.62	3.34	4.6	5.54	
3	1.95		3.75	5.36	6.71	9.3
4	1.96		3.78		7.33	10.4



The End



- <http://www.plurix.de>

