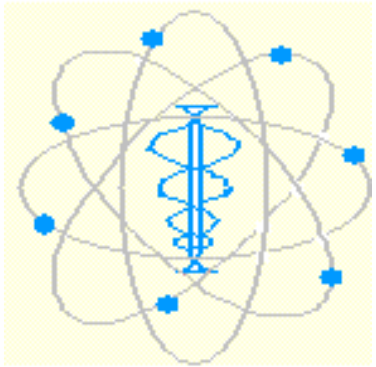
A Gaming Framework for a transactional DSM System



Ulm University
Germany

- Distributed Heap Storage
- Transactional Consistency

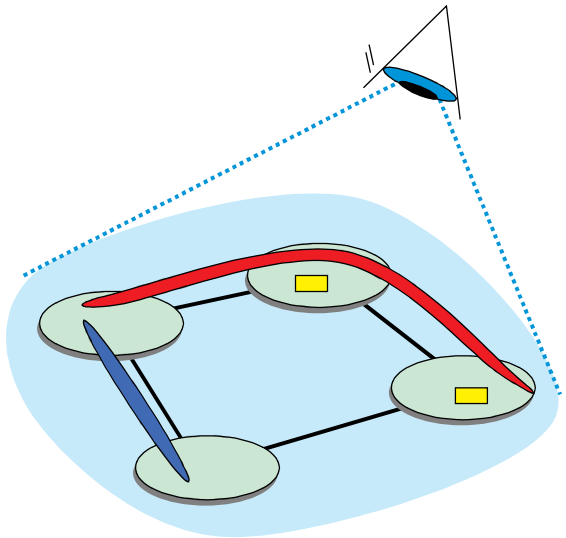
- Multiplayer Gaming Framework
- A Sample Game ‘Teletennis’
- Performance Evaluation

- Conclusions & Future Work

Speaker: Michael Schoettner

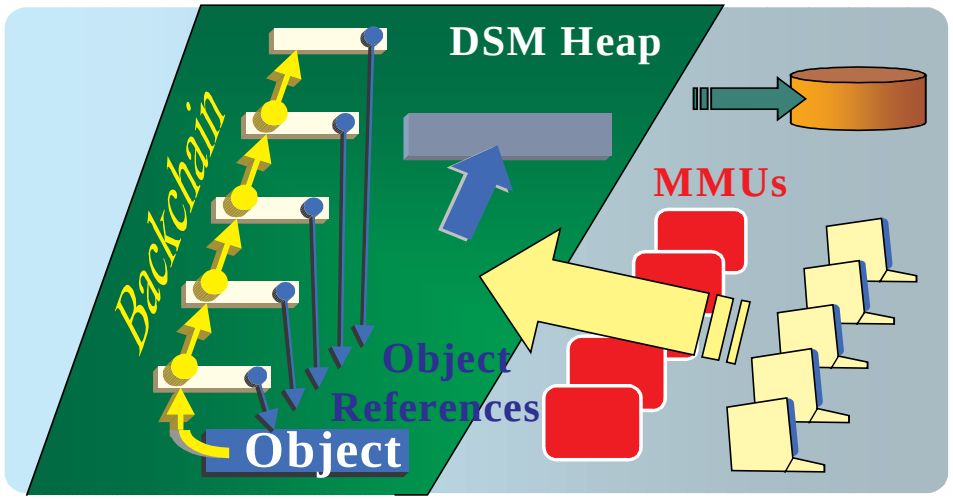
The Plurix Project

- The Plurix project:
  - PC-cluster Operating System with special Java compiler,
  - Single System Image by storing everything in the DSM (including the kernel).
- Major Goal: simplified development of distributed applications, by:
  - DSM-communication.
  - Transactional consistency.
  - Persistence & fault tolerance.
  - Distributed garbage collection.
  - Unified name service & cluster perspective.
- Applications:
  - Virtual worlds,
  - Telecooperation,
  - **Multi-player games** & edutainment,
  - (Number crunching & parallel processing).



Distributed Heap Storage

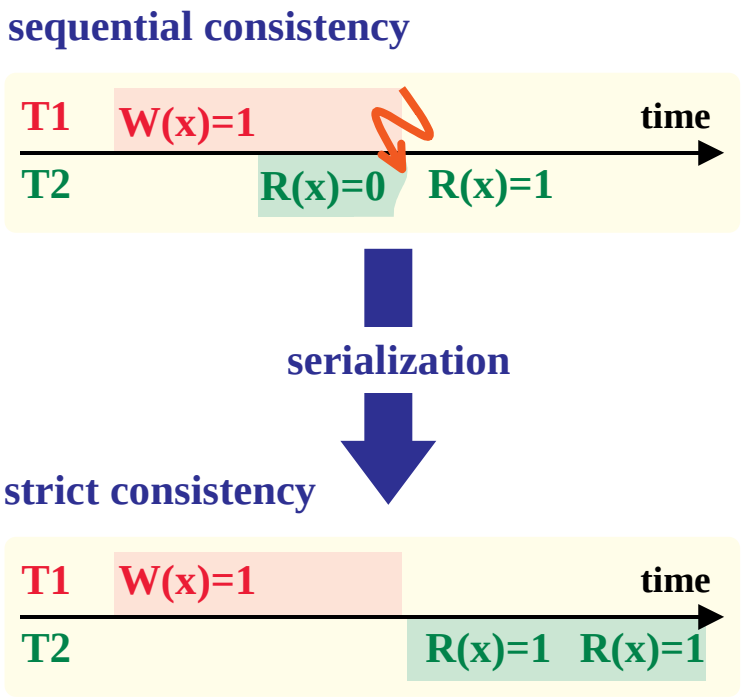
- Currently 4 Gbytes of distributed shared virtual memory.
- Garbage collection using a backchain:
  - Cluster-wide & incremental GC,
  - object references are linked together,
  - Garbage blocks have empty backchain.
- Relocation of objects in the Heap:
  - Reallocate object,
  - Adjust all references,
  - avoid fragmentation,
  - control False Sharing ...
- Persistent heap:
  - stations leave and join,
  - persistence by checkpointing,
  - PageServer(s) used for recovery.



Transactional Consistency

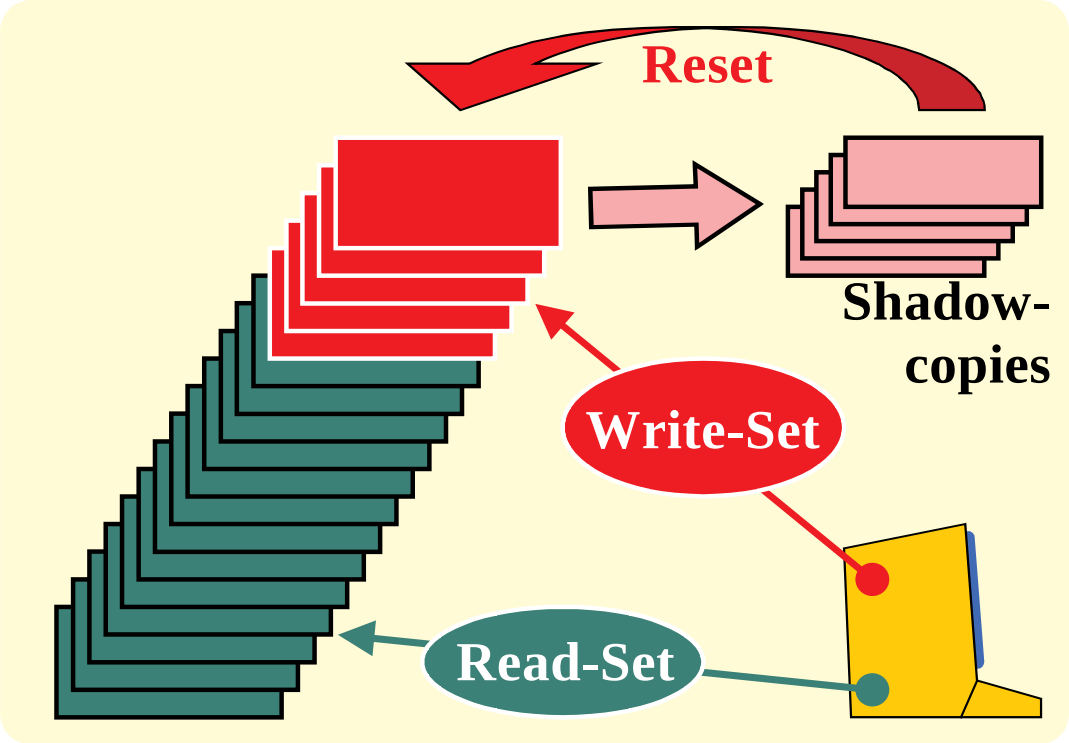
- Single memory accesses are not of interest, but only a set of operations performed within a **transaction**.
- Transactions within Plurix:
  - Follow the well-known ACID properties.
  - Implicitly defined by the command loop.
  - Are aborted in case of a collision.
  - May automatically restart.
- **Transactional Consistency** (Wende):
  - Sequential consistency always preserved.
  - Strict consistency at commit points.

→ **Reduced complexity,**
but number crunching applications may need additional weaker models.



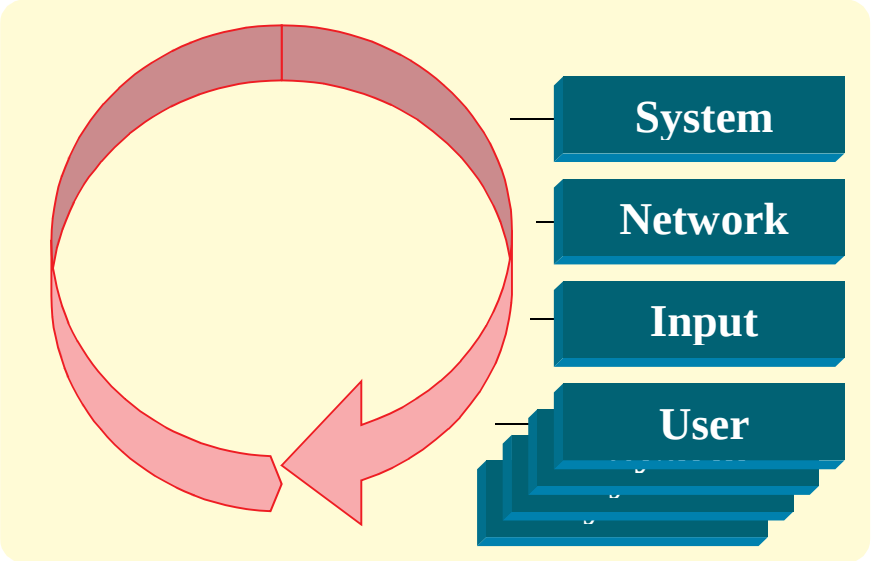
Restartable Transactions

- Optimistic synchronization between concurrent Transactions:
  - Writes will initially be on local copies only, shadow copies are preserved
  - Separate Read-Sets and Write-Sets are built during a transaction,
  - An ending transaction broadcasts a Commit-Request,
  - **Reset on collision →**
- Collision resolution:
  - short TAs preferred,
  - Forward-Validation,
  - Serializing on a token,
  - Currently „First wins“,
  - Fairness to be improved.
- Devices & Restartability:
  - SmartBuffers,
  - “undo”-Function.



Activity - Transactions

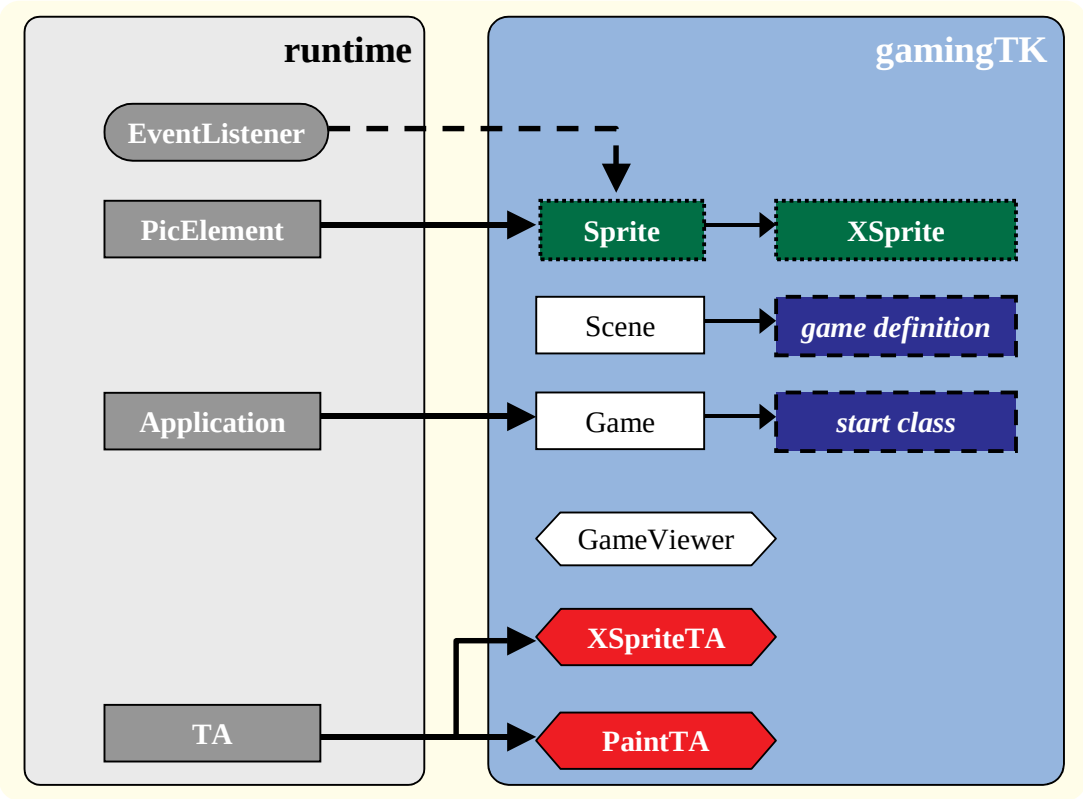
- Activity in Plurix:
  - no Processes and Threads,
  - but Transactions.
- Cooperative multitasking (Oberon-style):
  - **Central transaction loop,**
  - Garbage collection task,
  - Legacy network protocols,
  - Mouse & keyboard events,
  - User-installed transactions.



- Transactions:
  - extend the class “TA”
  - implement method “run”.
  - “run” is called by the scheduler.
  - periodically or as an event listener.
  - BOT & EOT inserted by scheduler implicitly.
  - Long running TAs need to be split up explicitly by the programmer.

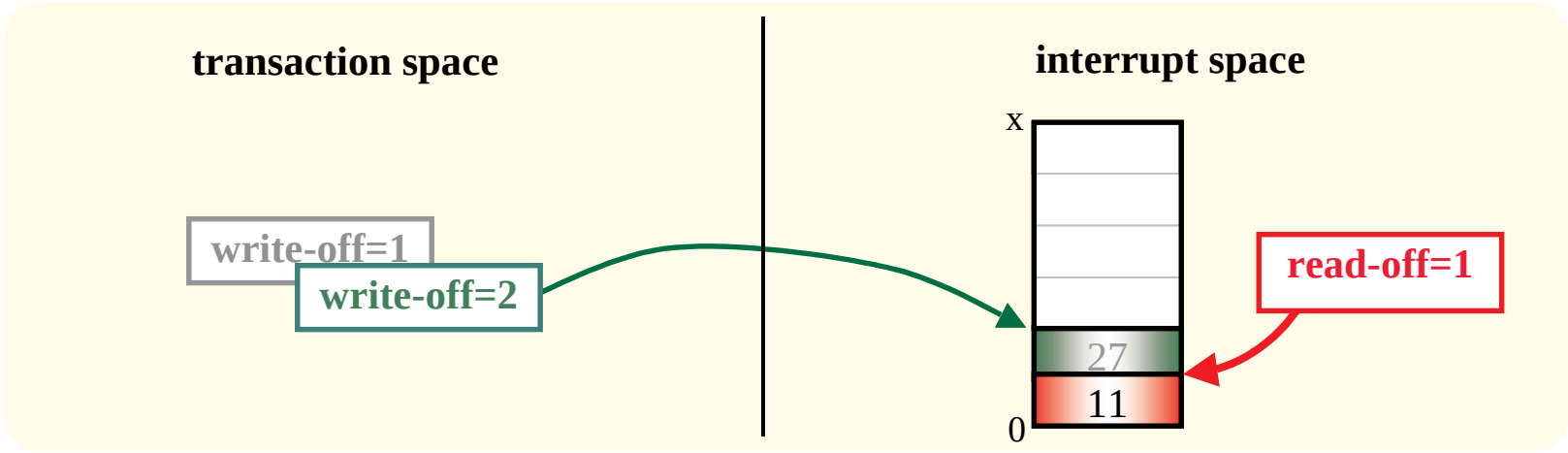
Multiplayer Gaming Framework

- Object-oriented framework in Java:
  - provides a shared scene graph with moving objects,
  - replication & consistency of scene data managed by DSM,
  - application-specific classes inherit, extend, and adapt the framework.



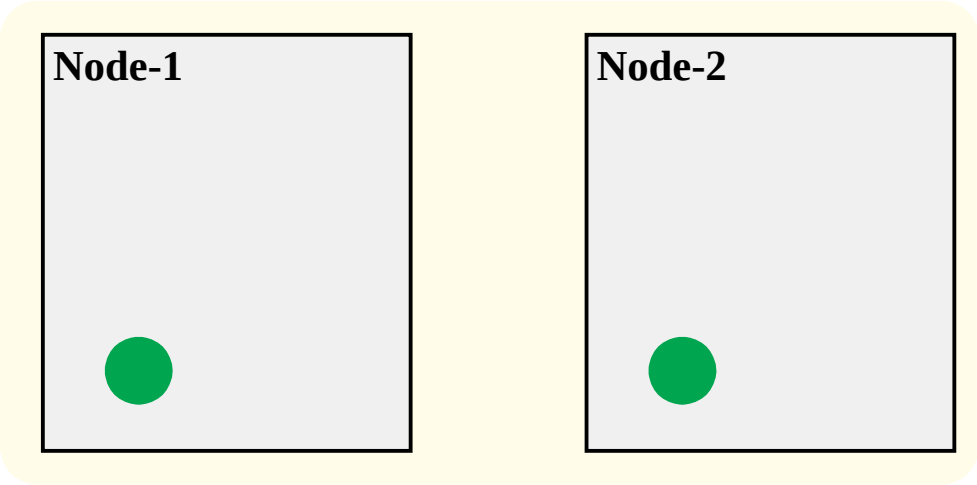
Multiplayer Gaming Framework (2)

- Shared scene graph accessible through the cluster-wide name-service.
- Each moveable sprite is registered as a periodically called transaction.
- User input to control avatars handled by Java listeners.
- **Challenge: transactional graphic output**
  - transactions rollbackable and restartable but device I/O not.
  - solution 1: implement an undo buffer in display memory
  - solution 2: execute device commands at commit time → **smart buffers**



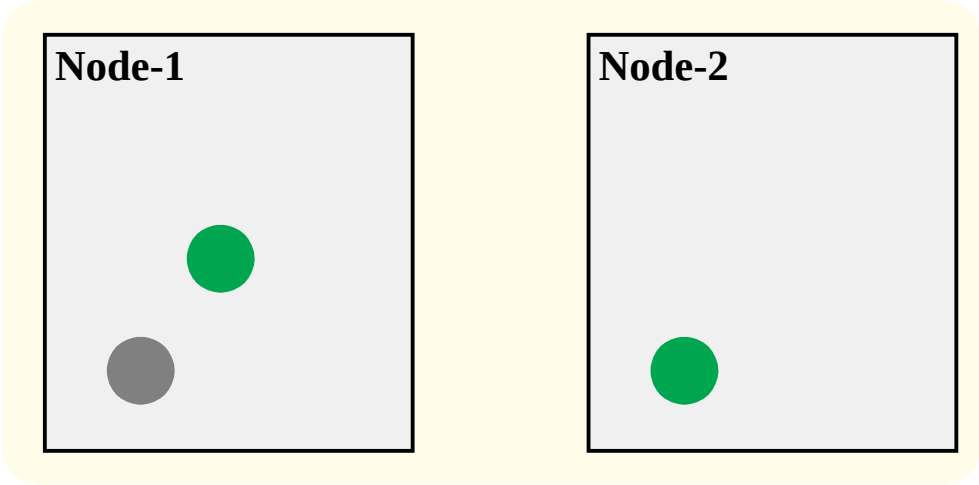
Update on Demand

- Each node executes periodically a **PaintTA** refreshing the game scene.
- Thus, changed data is automatically fetched by the underlying DSM.
- And slower nodes automatically skip object movements.
- But old positions must not be stored within the sprite.
  → better stored in the node-local viewer object



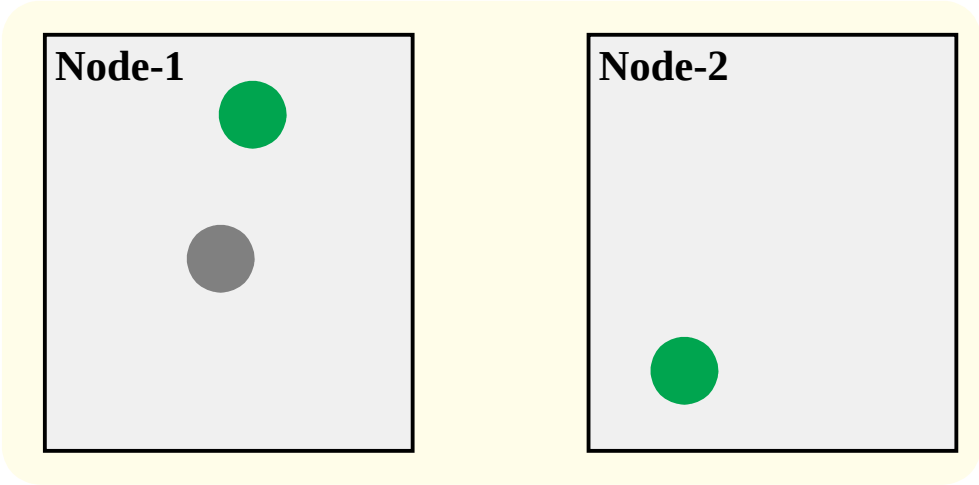
Update on Demand

- Each node executes periodically a PaintTA refreshing the game scene.
- Thus, changed data is automatically fetched by the underlying DSM.
- And slower nodes automatically skip object movements.
- But old positions must not be stored within the sprite.
  → store with the local viewer object



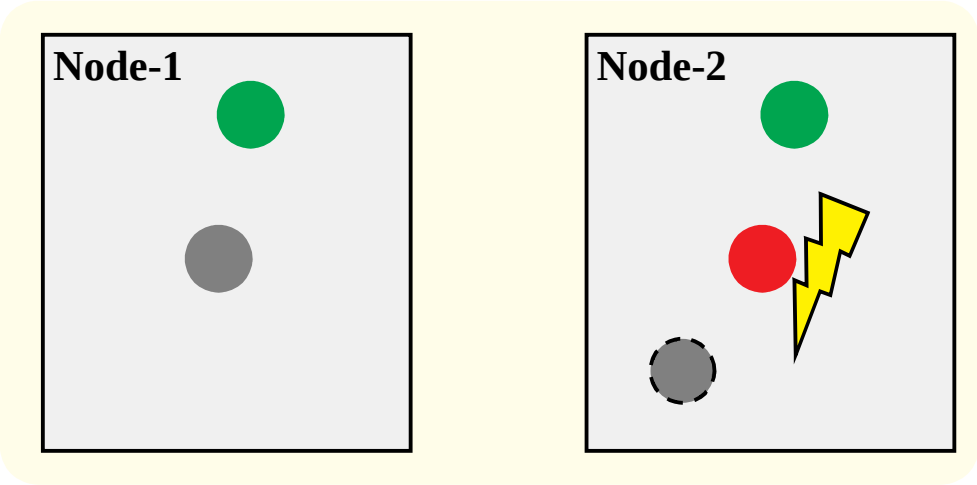
Update on Demand

- Each node executes periodically a PaintTA refreshing the game scene.
- Thus, changed data is automatically fetched by the underlying DSM.
- And slower nodes automatically skip object movements.
- But old positions must not be stored within the sprite.
  → store with the local viewer object



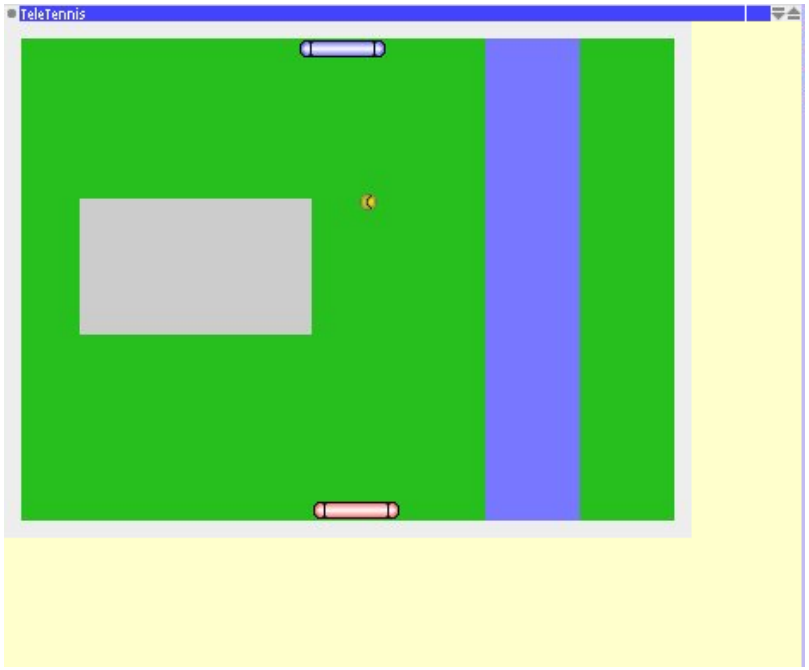
Update on Demand

- Each node executes periodically a PaintTA refreshing the game scene.
- Thus, changed data is automatically fetched by the underlying DSM.
- And slower nodes automatically skip object movements.
- But old positions must not be stored within the sprite.
  → store with the local viewer object



A Sample Game ‘Teletennis’

- 256 lines of source code.
- Support up to four players.
- Ball moved by a periodically called TA.
- Each racket is controlled by an interactive user.

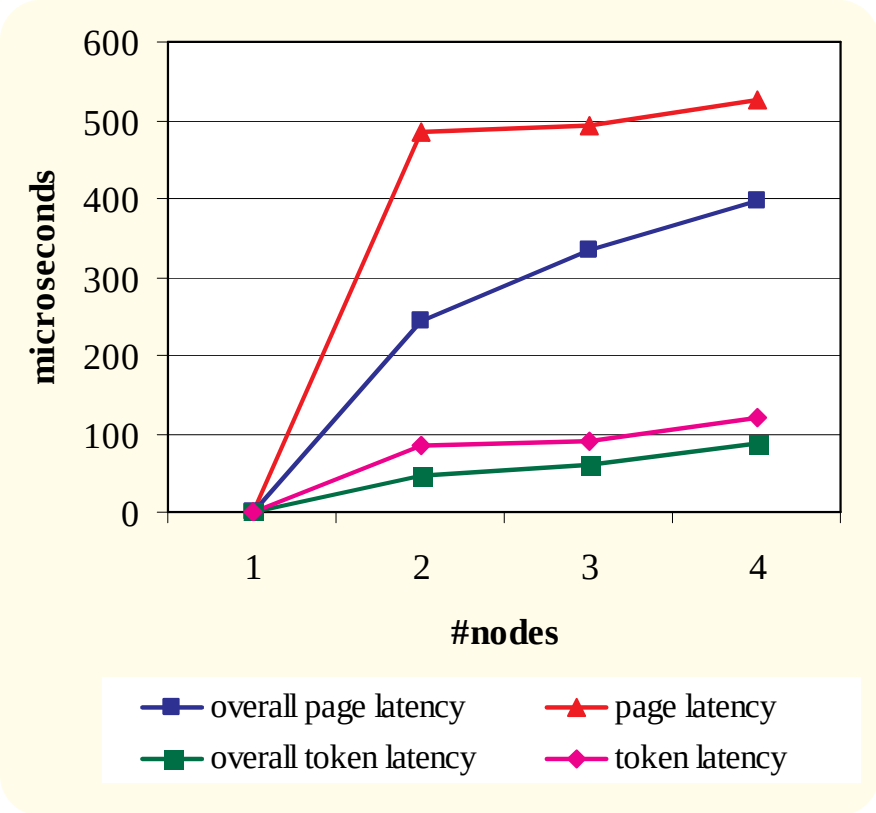


Performance Evaluation

- Measurements with four nodes:
  - Node 1: Pentium 4, 2.4 GHz, 256 MB RAM (266 Mhz).
  - Node 2: AthlonXP 2.2, 1.8 GHz, 256 MB RAM (333 Mhz).
  - Node 3: AthlonXP, 2.0 GHz, 256 MB RAM (333 Mhz).
  - Node 4: Celeron, 1.8 GHz, 256 MB RAM (266 Mhz).
  - Connected by a FastEthernet Hub.
- Measurements are integrated into the memory consistency protocol.
- Game setup:
  - Rackets are moved automatically every 50ms.
  - PaintTA is executed on each node every 40ms.
  - Ball is moved by a TA running on the first node every 50ms.

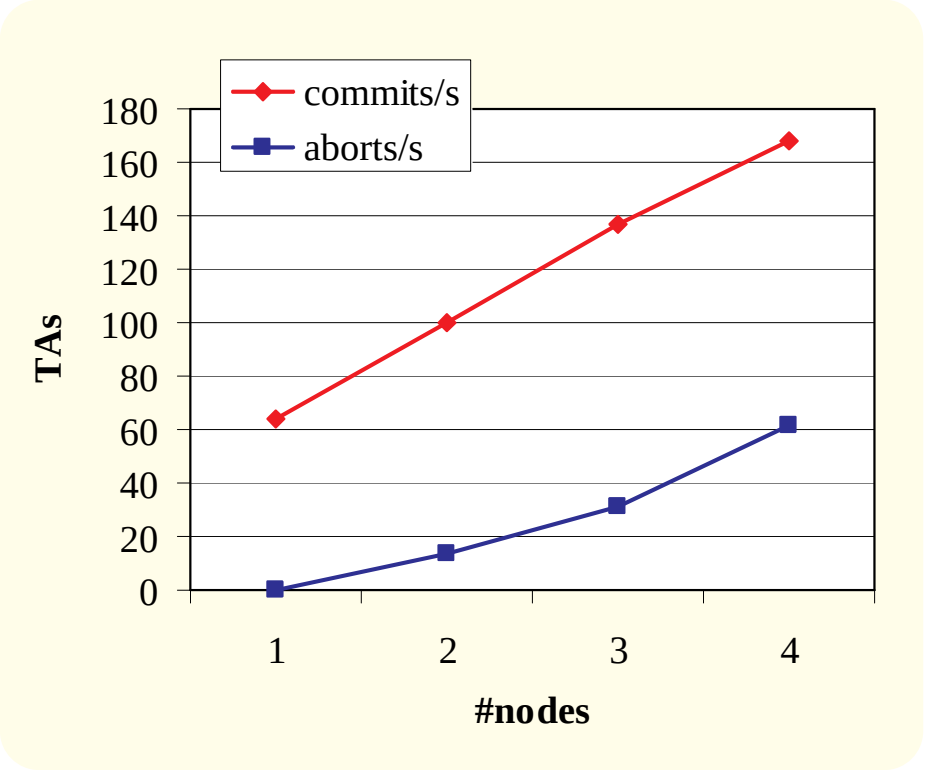
Performance Evaluation – Cluster Latencies

- Average of all page/token requests of all nodes in the cluster.



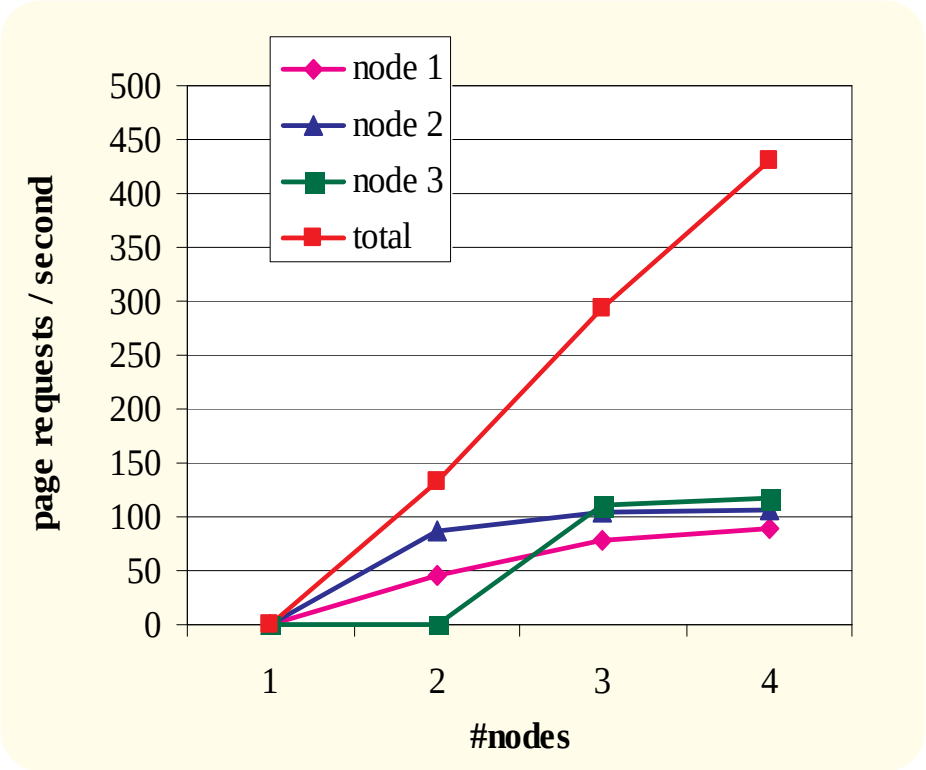
Performance Evaluation – Cluster Throughput

- Total number of commits and aborts in the cluster.



Performance Evaluation – Network Traffic

- OS can manage 2´700 page requests per second using Fast Ethernet.
- Number of page requests per second during running game:

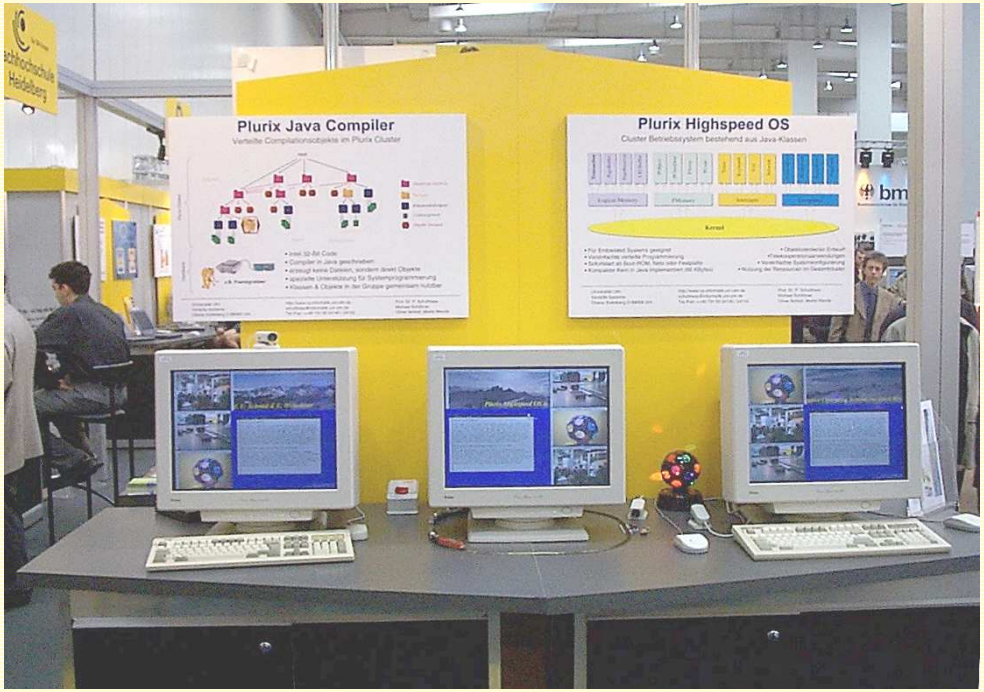


Conclusions

- Multiplayer games over new perspectives for DSM systems.
- Distributed Shared Memory:
  - simplifies distribution and replication of data and
  - ensures consistency of shared scene graphs.
- More sophisticated games are needed to evaluate scalability …
- Multi-room games and virtual worlds of special interest.

The End

CeBIT 2000 DSM-Demonstrator



http://www.plurix.de