Parallel Ray-Tracing with a Transactional DSM

Distributed Systems Laboratory, Ulm University, Germany

S. Frenz, M. Schöttner, R. Göckelmann, P. Schulthess

Distributed storage consistency

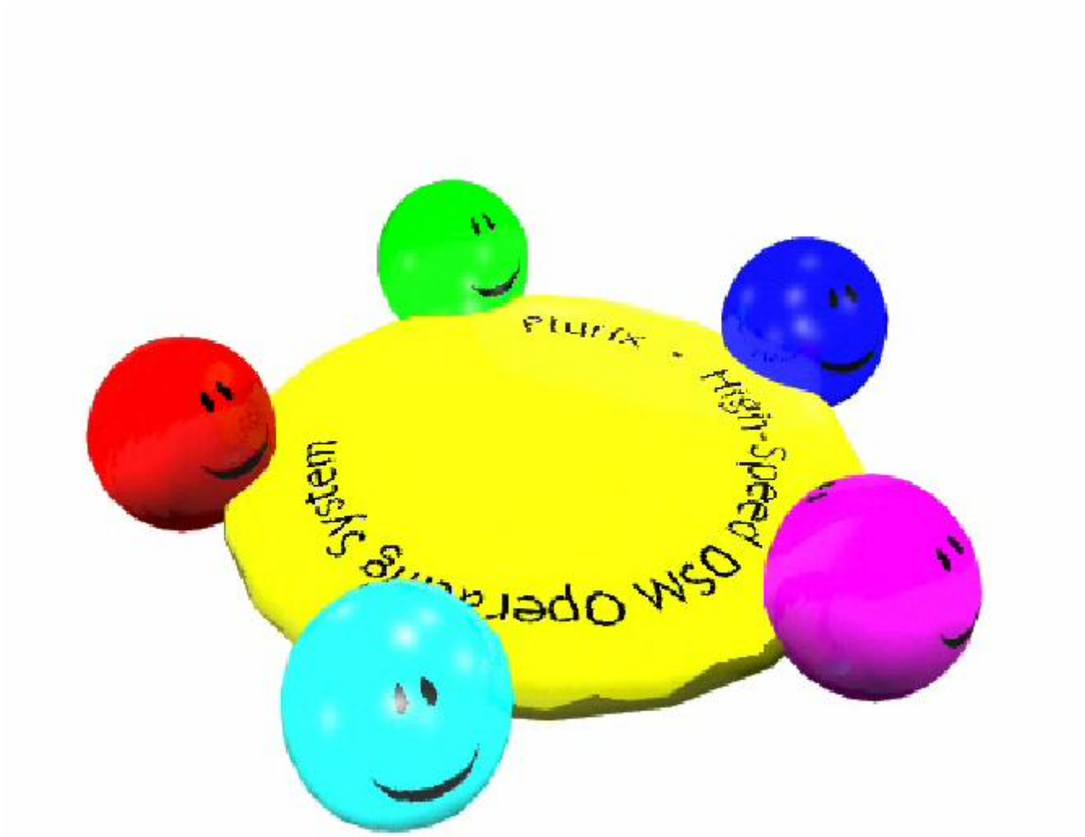
Restartable DSM transactions

Optimistic synchronization

Plurix DSM Architecture

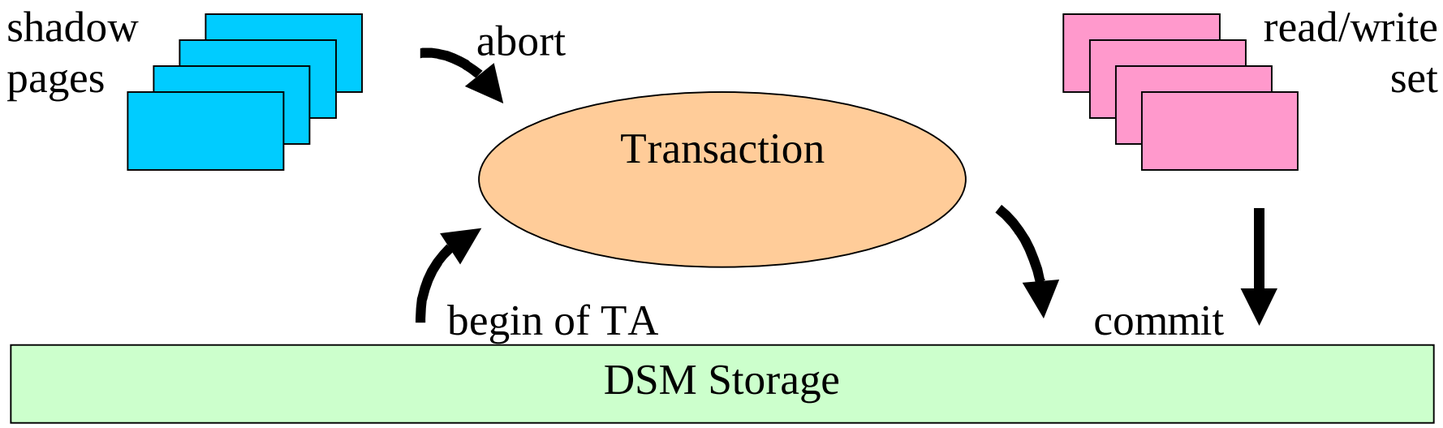
Our raytracing scenario

Cluster performance



Restartable DSM transactions

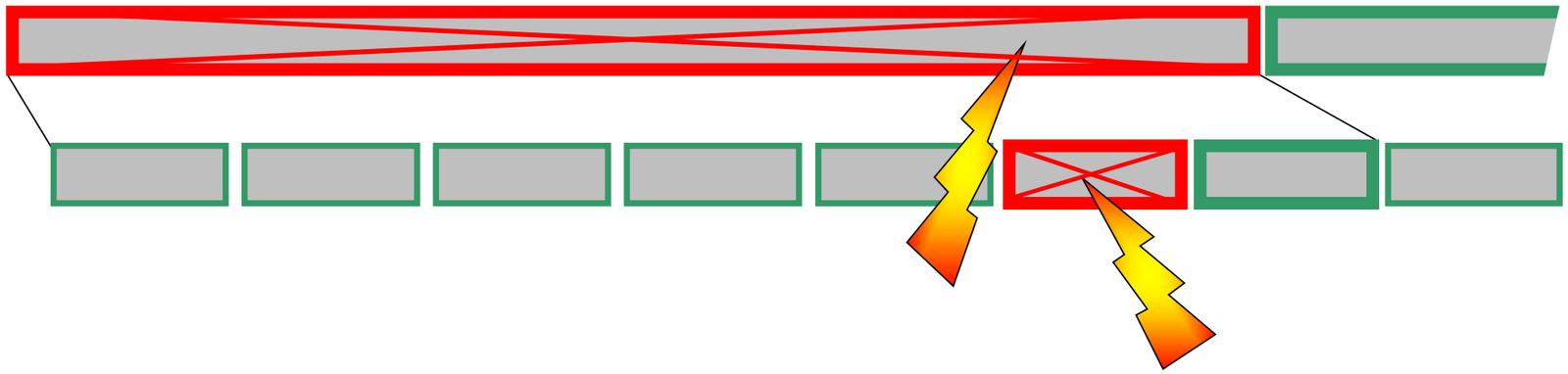
- Plurix Transactions observe the ACID principle:
  - Only after a successful commit of a transaction its modifications become visible to other stations.
  - When a transaction aborts all its modifications are undone.
- All computations are **implicitly** encapsulated into transactions.
- Read/write-sets are collected during the course of each transaction.
- The commit request broadcasts the write-set to all stations in the cluster.
- Stations will individually check whether they have to abort/restart the TA.
- Shadow copies of modified pages are created and restored when a TA aborts.



Optimistic synchronisation

- Assume that read/write collisions between competing transactions are infrequent.
- Optimistic synchronisation lets the computation proceed and masks network latency.
- Traditional locks and barriers introduce network latency and the risk of deadlock.
- Short transactions and late reading reduces the probability of a collision.

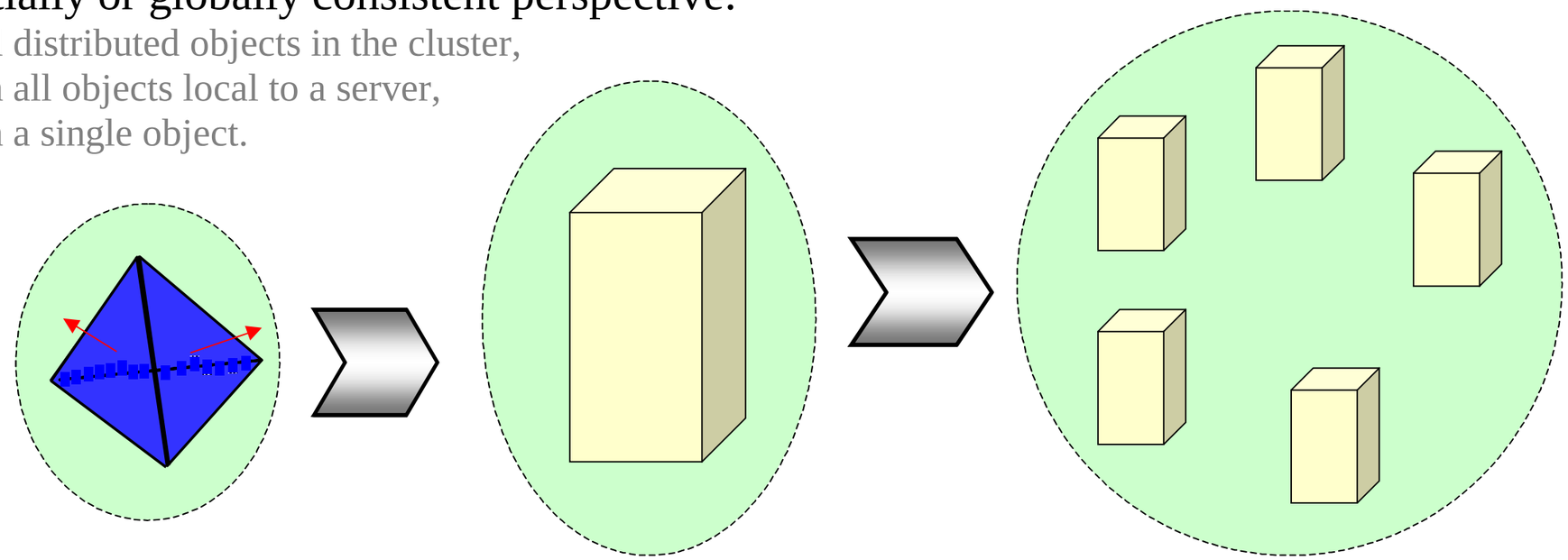
- To reduce collision cost long transactions may be **explicitly split** into smaller ones:



- Typical Plurix transactions are implicit and take much less than 1 second:
  - Entering a mouse click, a keystroke or a system command,
  - Compilation of a class or a program module,
  - Computation of a video frame ...

Distributed storage consistency

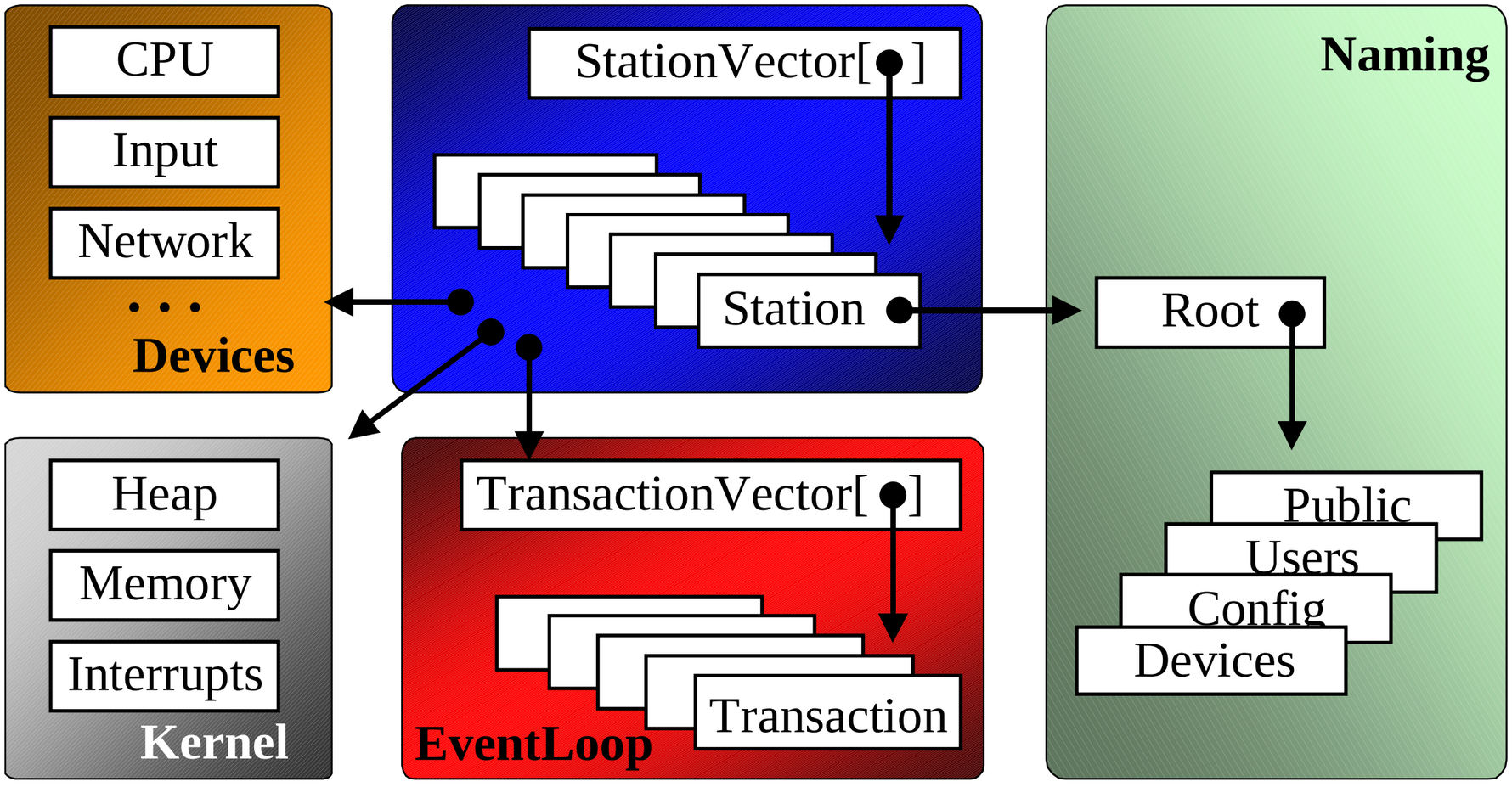
- Storage consistency requires all processes to have a common perspective on memory.
- Synchronisation tools consolidate the perspective on memory:
  - Messages, locks, barriers, monitors, semaphores, begin/end-of-transaction …
- Partially or globally consistent perspective:
  - all distributed objects in the cluster,
  - on all objects local to a server,
  - on a single object.



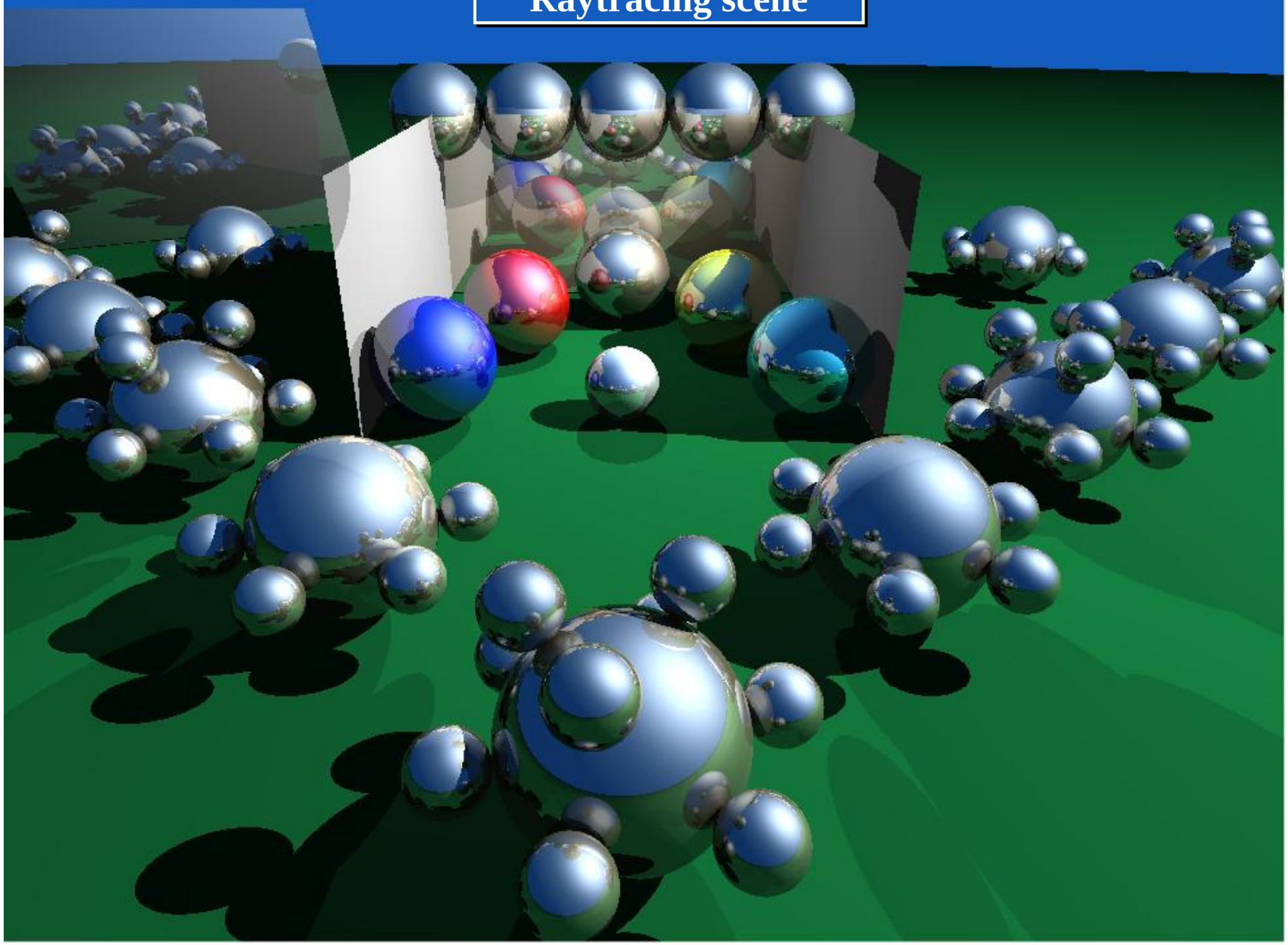
- Programmers must control application specific consistency semantics of their objects:
  - Invoking explicit synchronisation for partial consistency will improve concurrent execution,
  - Relying on built-in storage consistency models (transactions, atomic operations …).
- Consistency of distributed OS data structures is guaranteed by the Plurix DSM system:
  - Memory management, name space, queues, devices, transactions, fault recovery …

Architecture of the Plurix DSM system

- Sophisticated DSM memory management for PC clusters.
- Non-preemptive transaction loop in each station.
- Non-transactional interrupt & kernel space.
- Naming for persistent objects.
- Native Intel486 code.

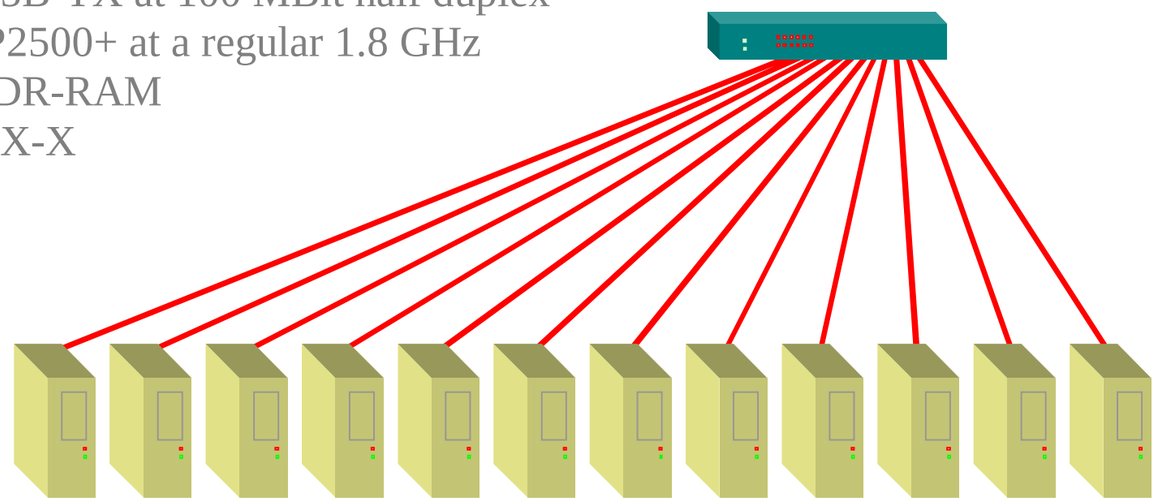


Raytracing scene

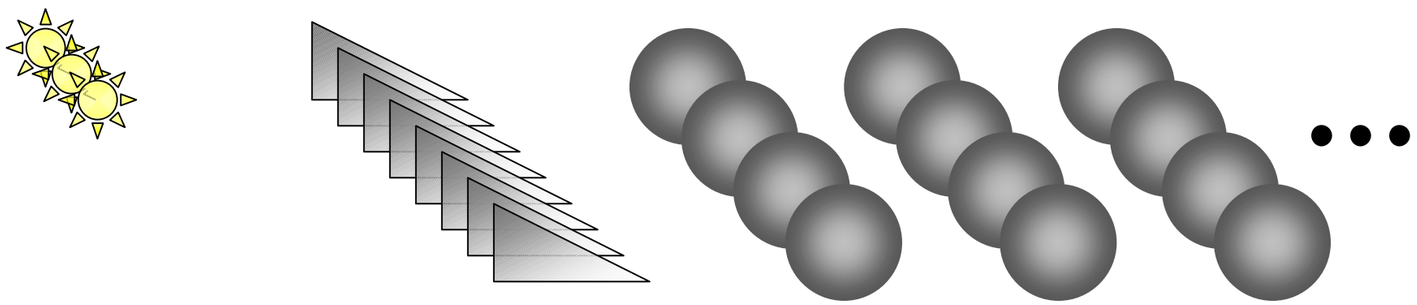


Cluster configuration

- **The PC cluster:**
  - Network adapter: 3com 905B-TX at 100 MBit half duplex
  - Main Processor: Athlon XP2500+ at a regular 1.8 GHz
  - Main memory: 512 MB DDR-RAM
  - Motherboard: Asus A7V8X-X
  - 12 nodes.

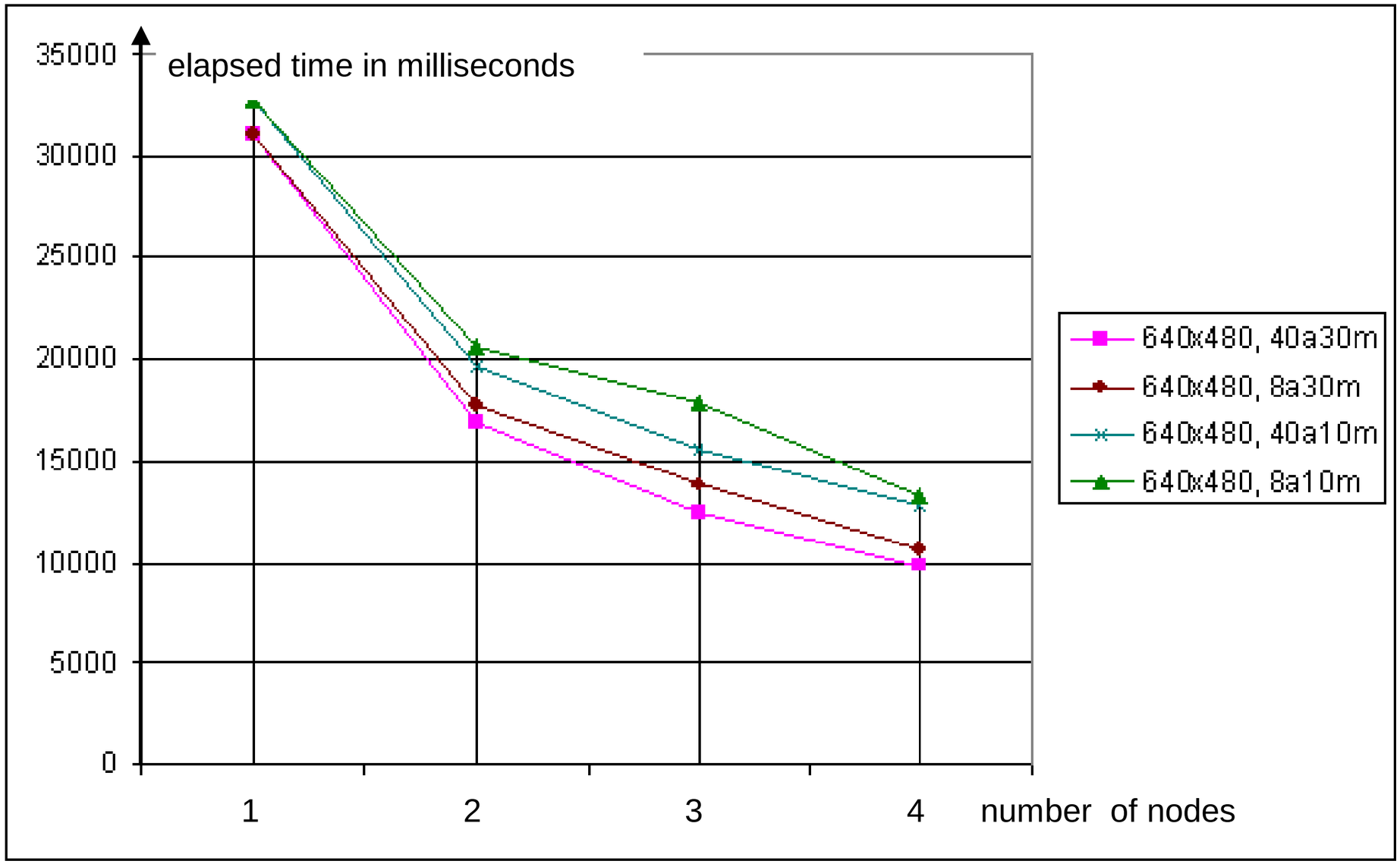


- **The Scene:** 3 light sources, 8 triangles, 104 reflecting spheres



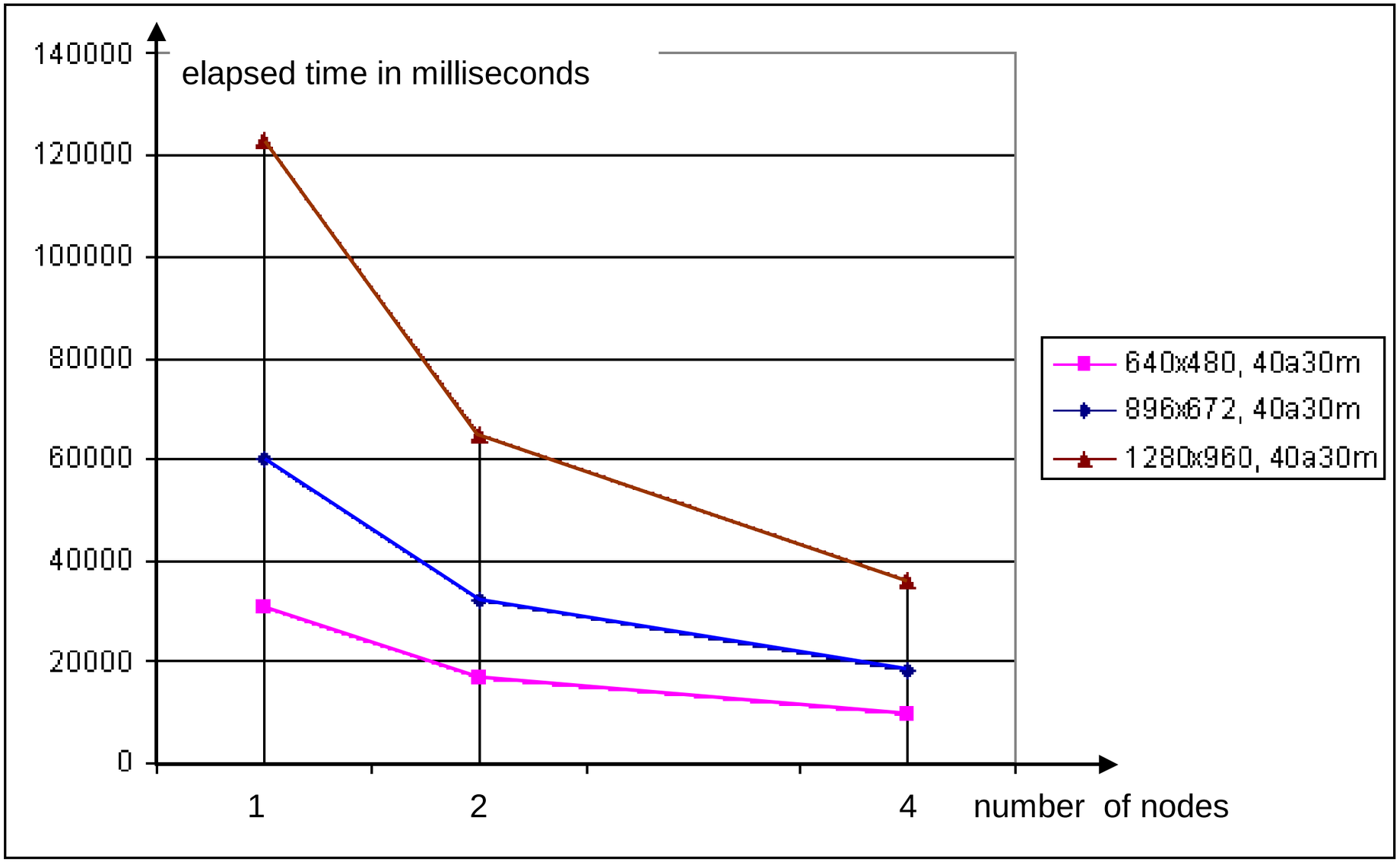
Computation time depending on blocksize and transaction time

- Larger blocksize for allocation of scanlines reduces collisions between stations.
- Longer transaction time reduces the overhead percentage (~ 300 µsec/TA).



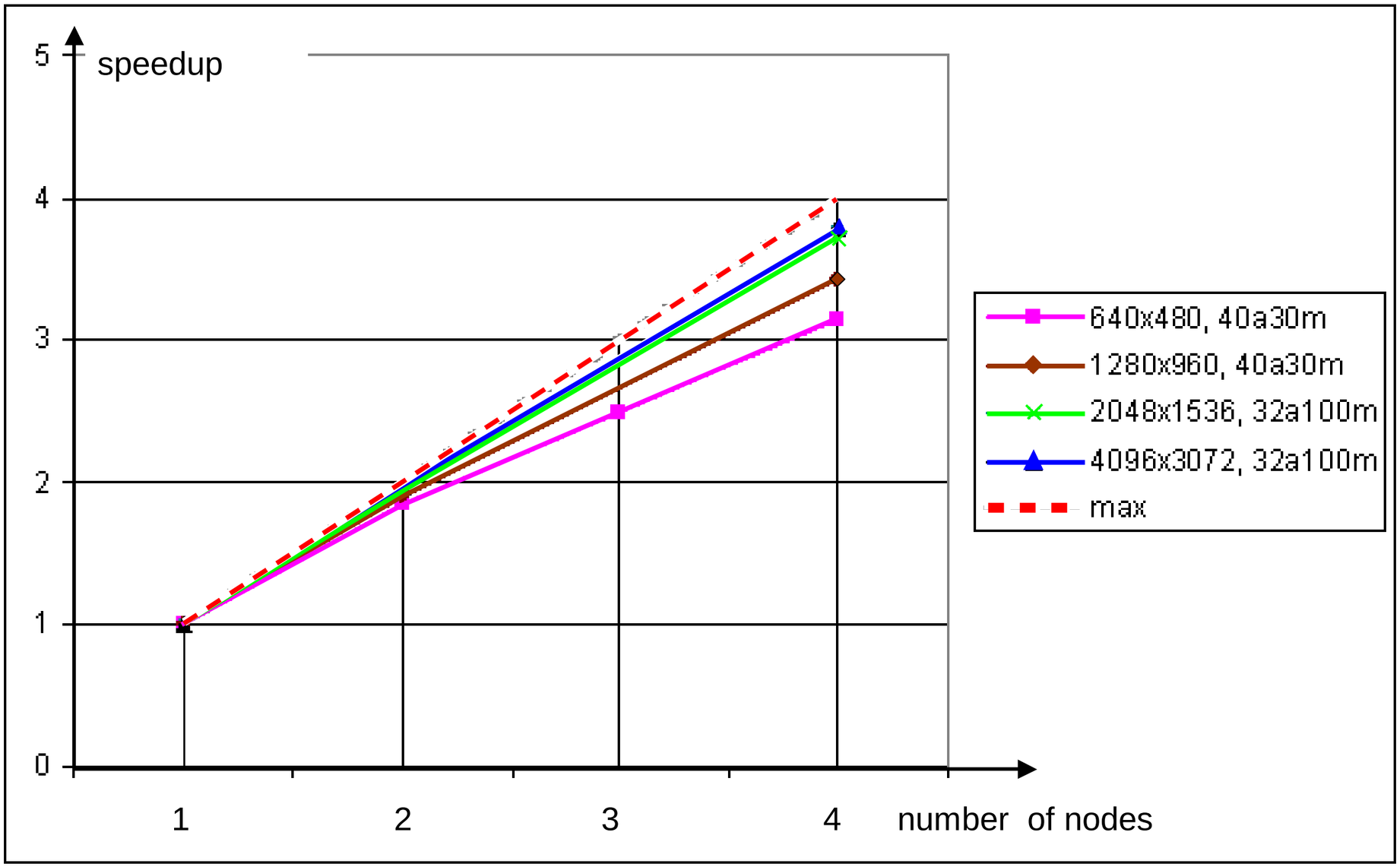
Computation time by number of nodes

- A single station still incurs transaction overhead but no paging and no collisions
- A single station takes approximately 3.5 times longer than 4 stations.
- Linear with number of pixels in the image.



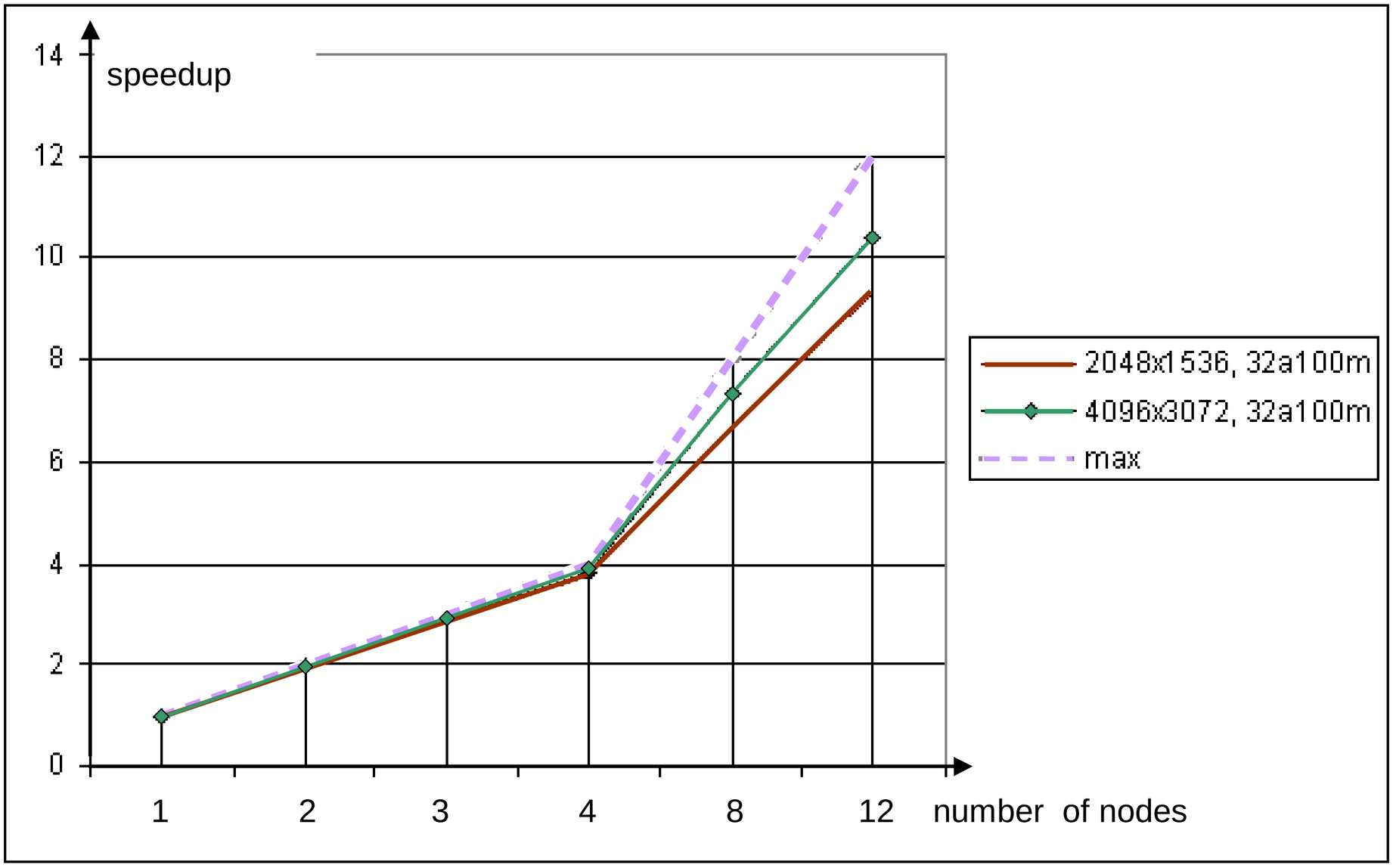
Speedup by number of nodes, varying resolution

- Large image sizes reduce the amount of collisions.
- Collisions are more severe with many cluster stations.



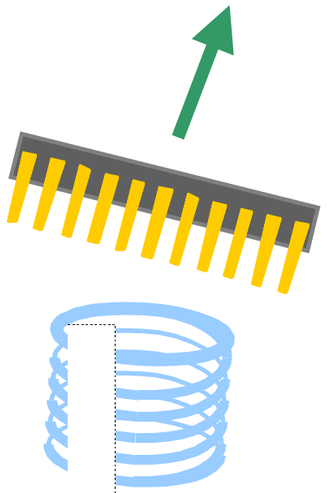
Computational speed-up

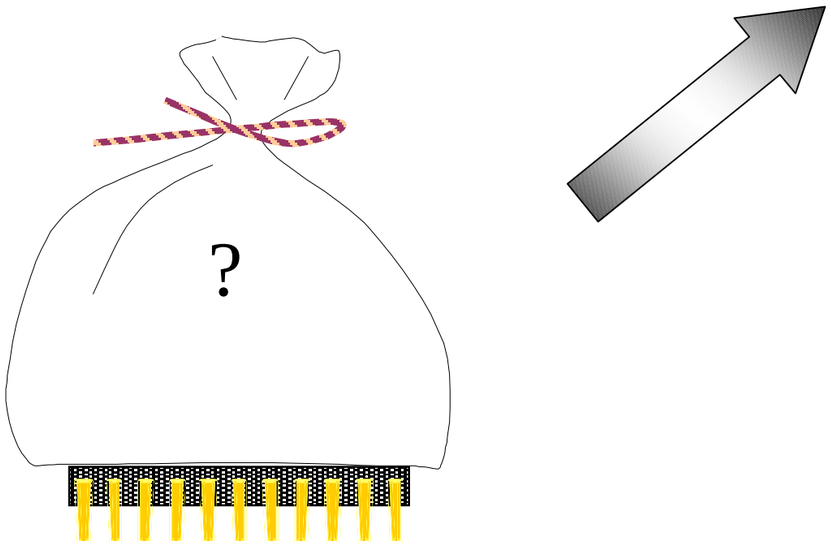
- Measured with up to 12 nodes.
- Good scalability for medium sized images (75% @ 12 stations).
- Very good scalability for large size images (85% @ 12 Stations).



Conclusion

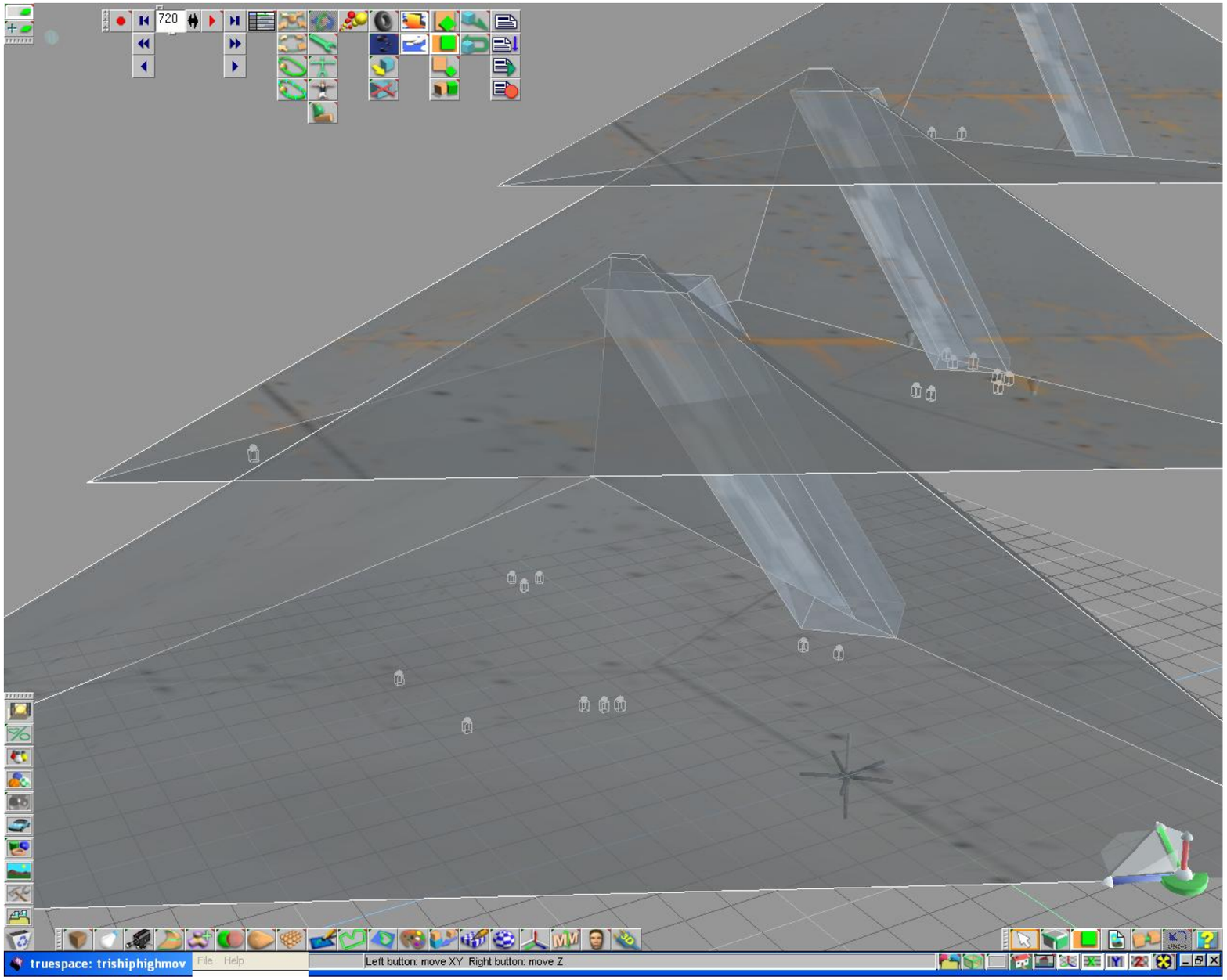
- Sometimes DSM is an alternative to message passing application frameworks.
- It can automatically provide clusterwide storage consistency.
- It simplifies the development of distributed applications.
- **It lends itself to lean & reliable implementation =>**
- Correctness of the algorithm is easily achievable.
- Concurrent performance requires tuning.
- Access patterns require special care.
- Persistent objects as a bonus.





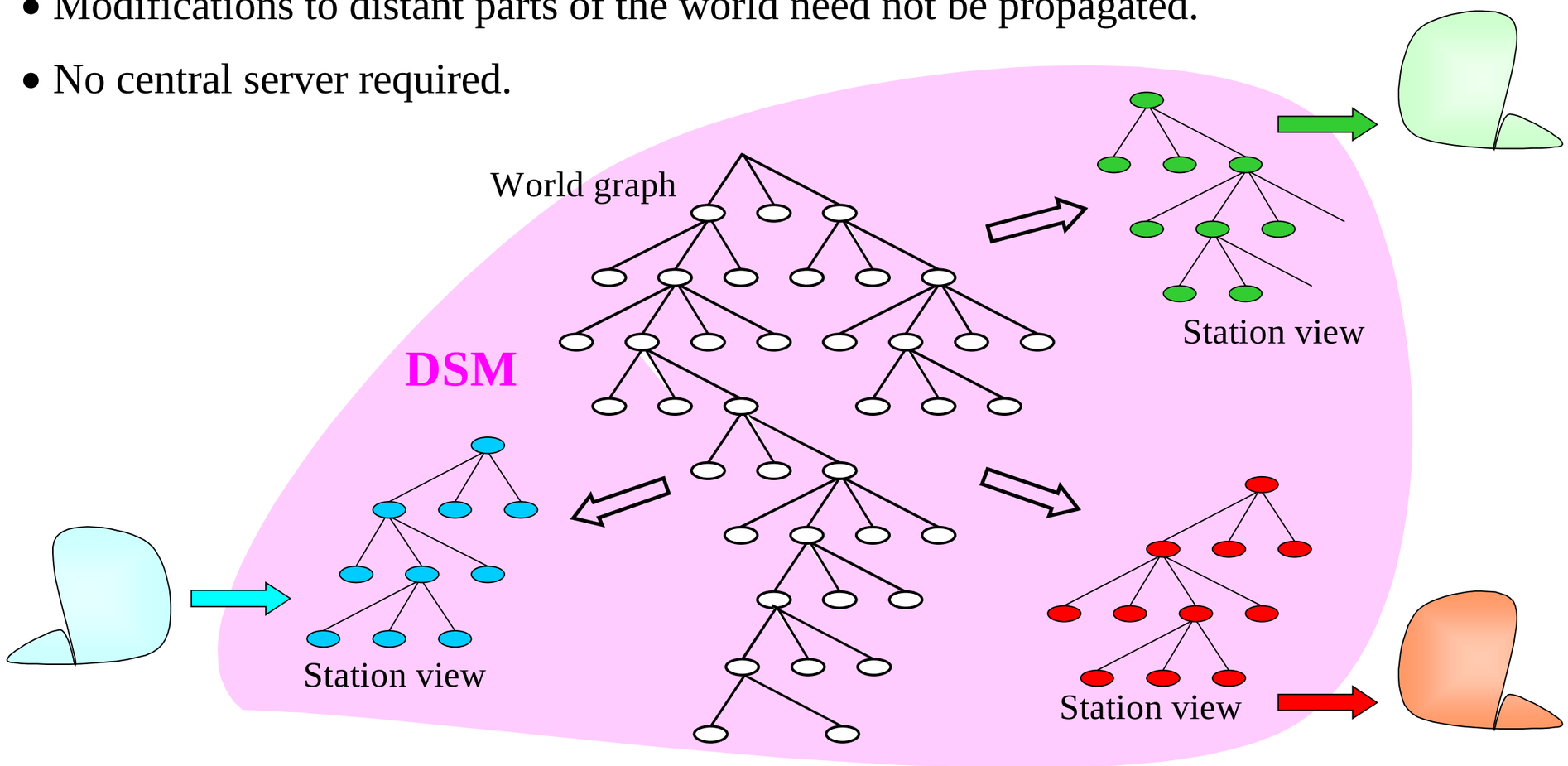
Virtual Presence Project

- Student avatars will be virtually present and can interact in a virtual spaceship.



Scene graphs in DSM storage

- The world graph is "read mostly" and separated from the views offered to each station.
- Modifications to distant parts of the world need not be propagated.
- No central server required.



- DSM storage appears suitable to host the scene for multi-player scenarios:
  - Virtual 3D worlds, multiplayer games, business meetings, collaboration …