Parallel Ray-Tracing with a Transactional DSM

Distributed Systems Laboratory, Ulm University, Germany

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

Distributed storage consistency

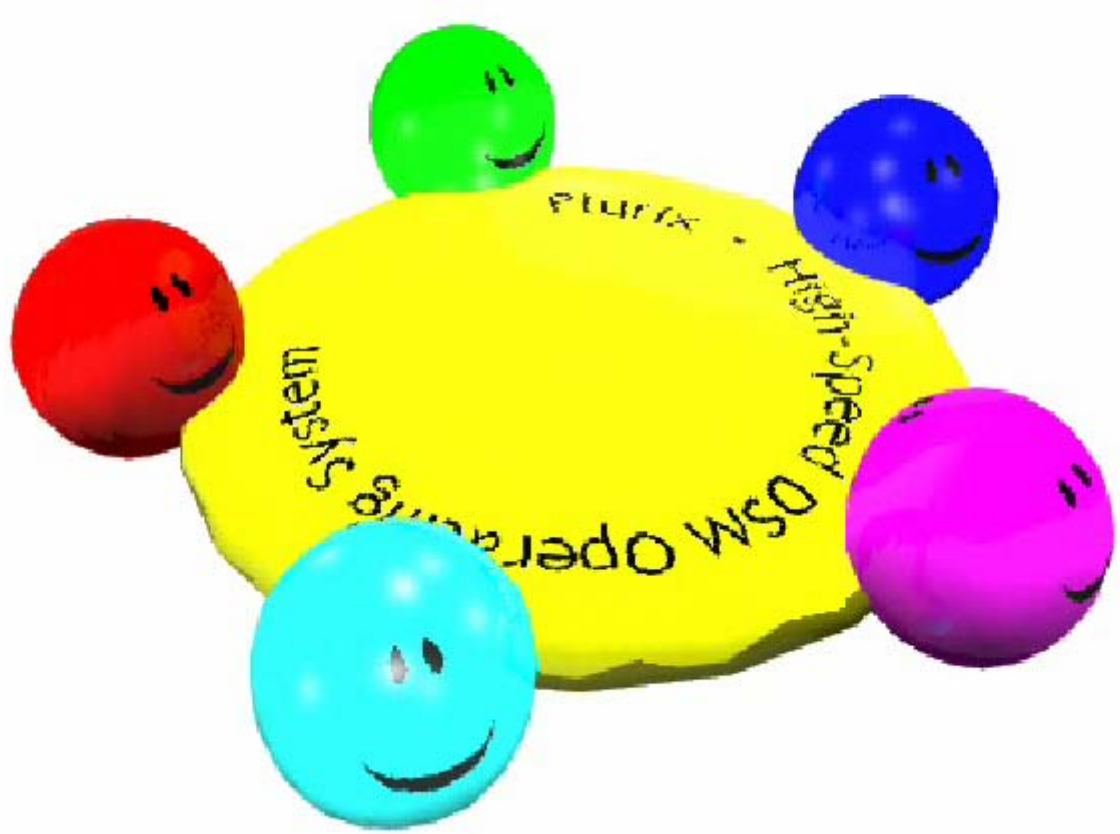
Restartable DSM transactions

Optimistic synchronization

Plurix DSM Architecture

Our ray tracing scenario

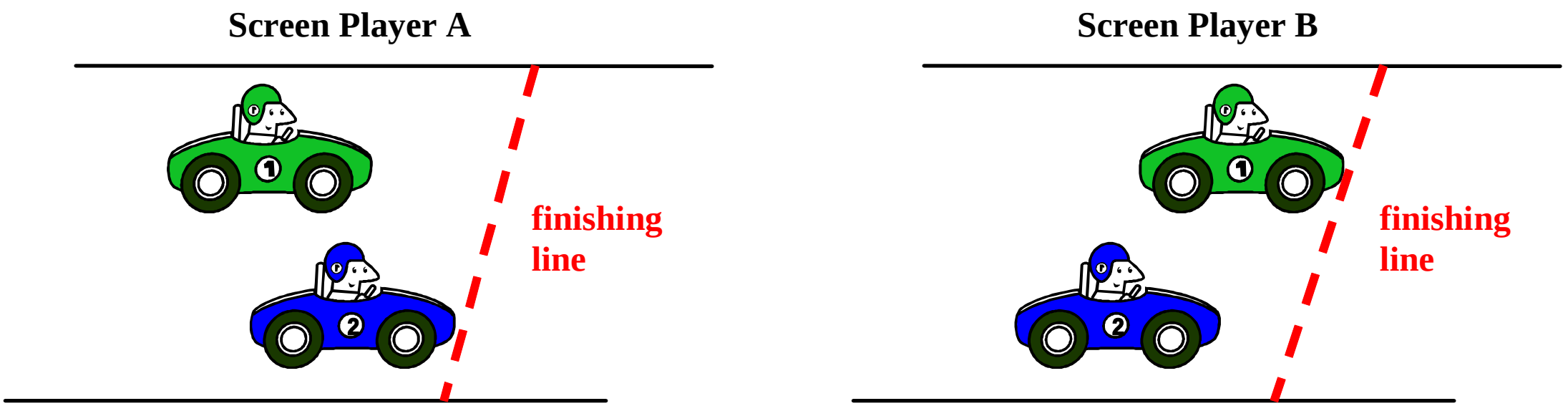
Cluster performance



Distributed storage consistency

- Coherence: How do processes see updates?
  - invalidate: a write operation invalidates all replicates on other nodes,
  - update: after a write operation all existing replicates are updated.

- Consistency: When do processes see updates?
  - contract between memory and processes.
  - All processes should have a common perspective on the distributed storage.
  - Example: multiplayer racing game → without consistency both players believe they are the winner.

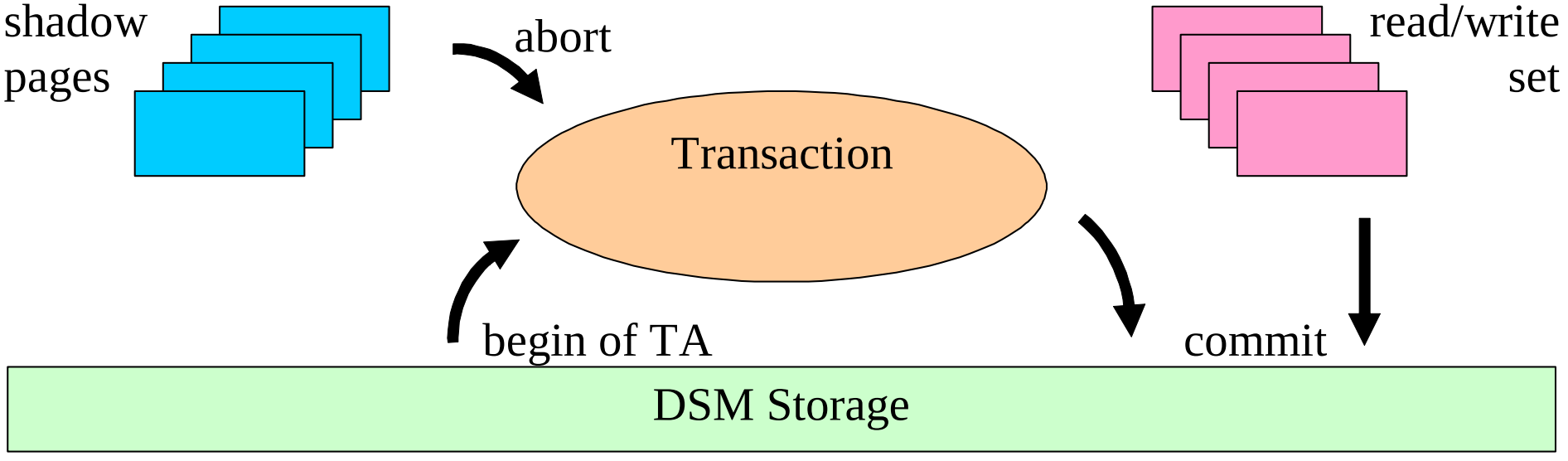


- Choosing a consistency model is always a trade off:
  - strong consistency is easy to program but less efficient,
  - weak consistency is more efficient but harder to program.

- Programmers must spend time for performance tuning of parallel programs:
  - improving concurrency and preserving correctness,
  - messages, locks, barriers, monitors, semaphores, begin/end-of-transaction …

Restartable DSM transactions

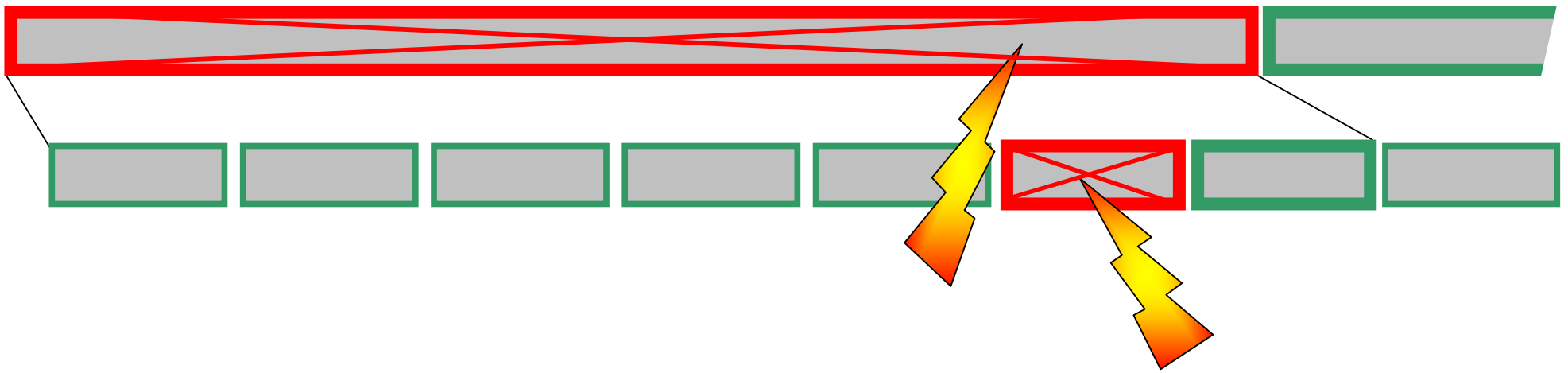
- Plurix Transactions observe the ACId principle:
  - Only after a successful commit of a transaction its modifications become visible to other stations.
  - When a transaction aborts all its modifications are undone → restartability.
- All computations are **implicitly** encapsulated into transactions.
- 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

- Assumption: read/write collisions between overlapping 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 reduce the probability of a collision.

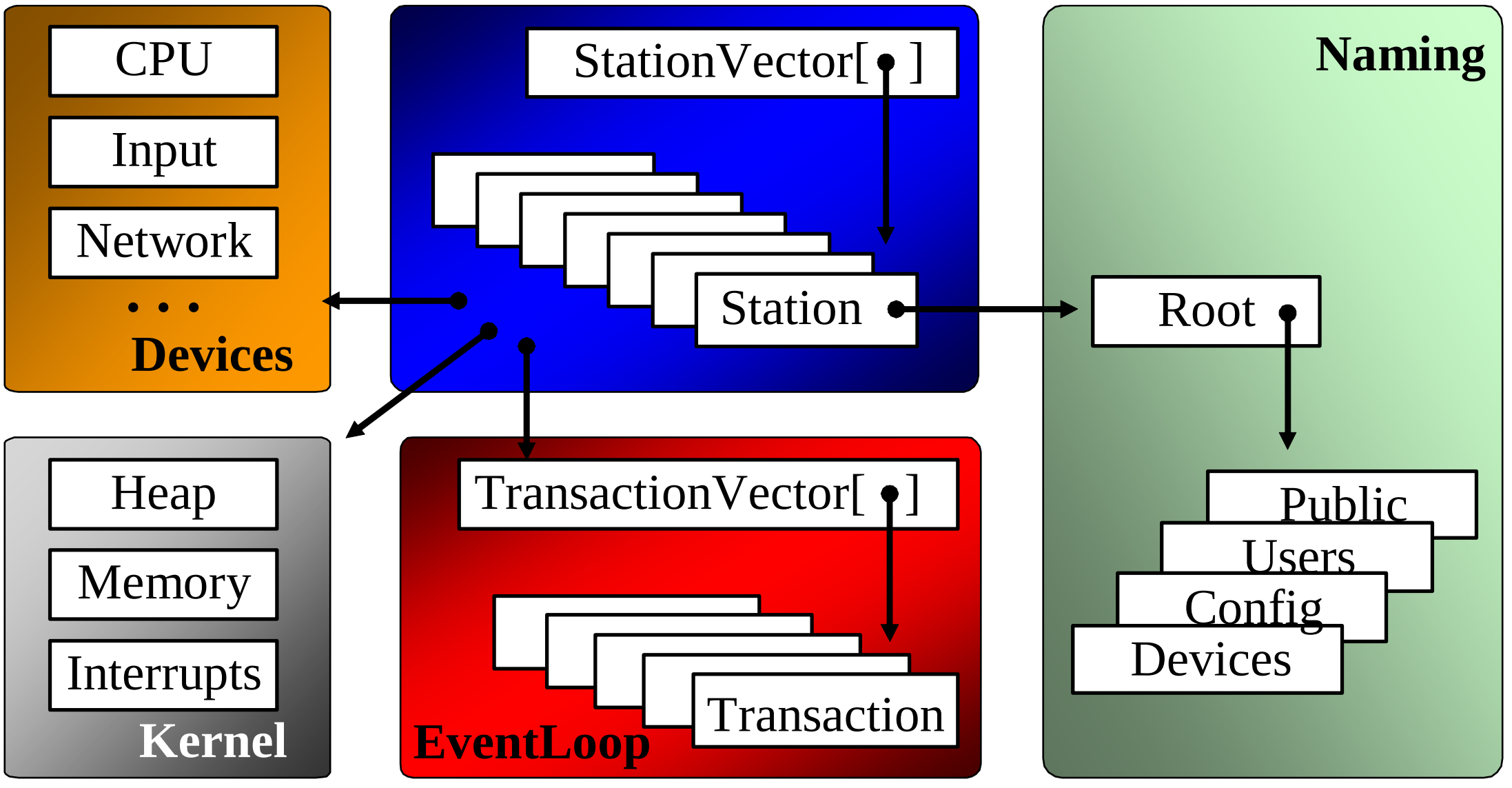
- To reduce collision cost long transactions may be **explicitly split** into smaller ones:
  - Integrated monitor facilities allow easy identification of critical hot spots.



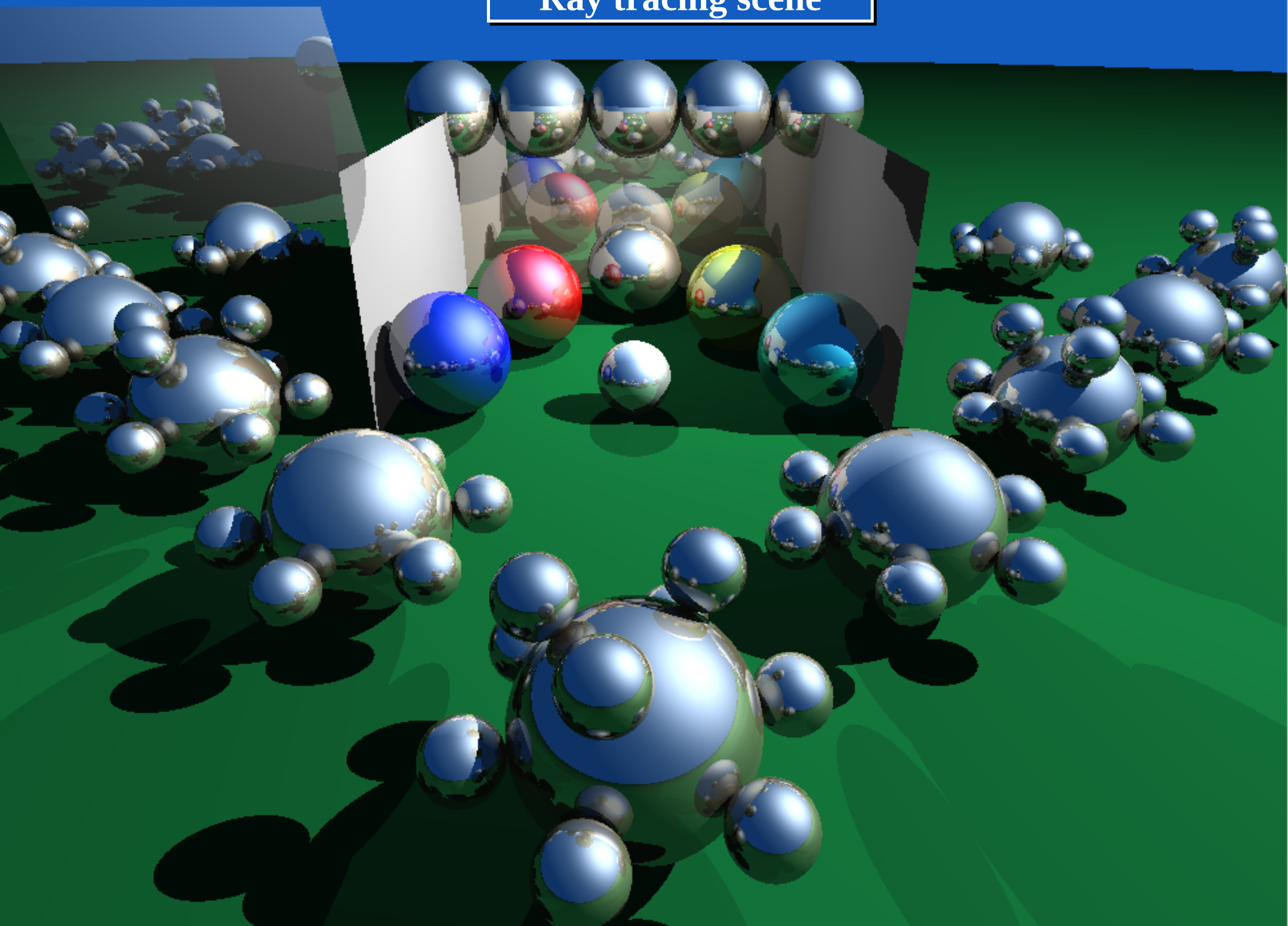
- Typical Plurix transactions are implicit and take much less than 1 second:
  - Entering a mouse click, a keystroke or a system command,
  - Compilation of a class or a program module,
  - Computation of a video frame ...

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.

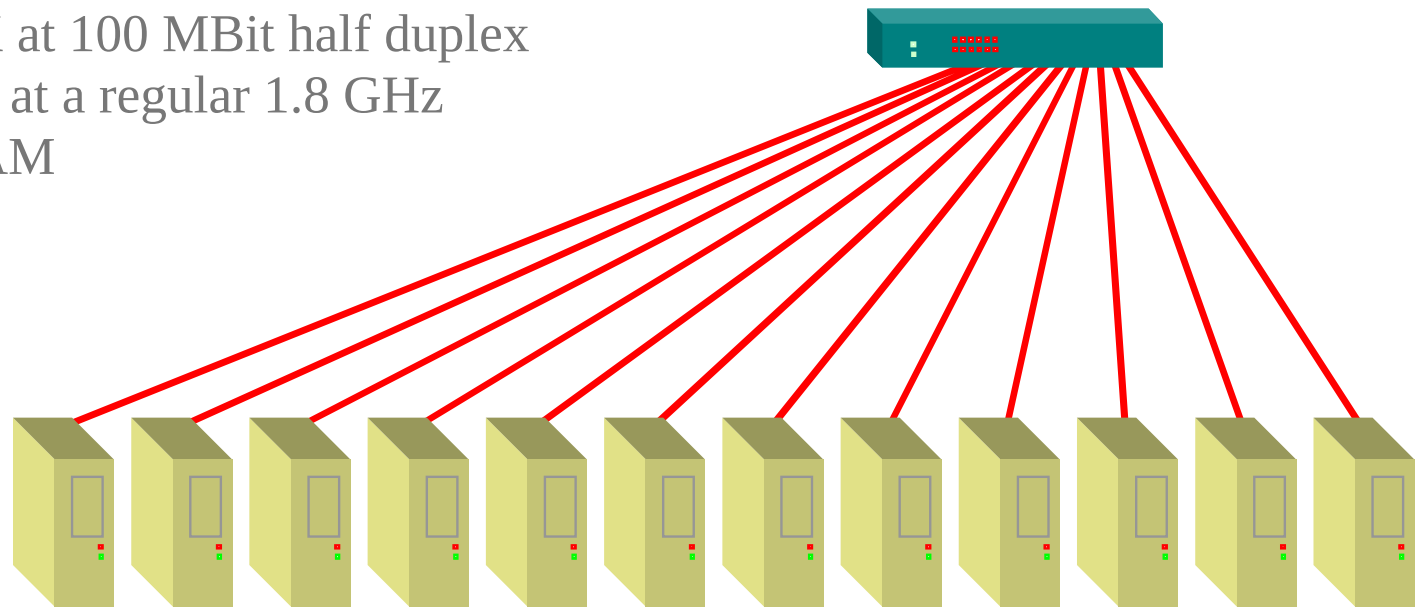


Ray tracing scene



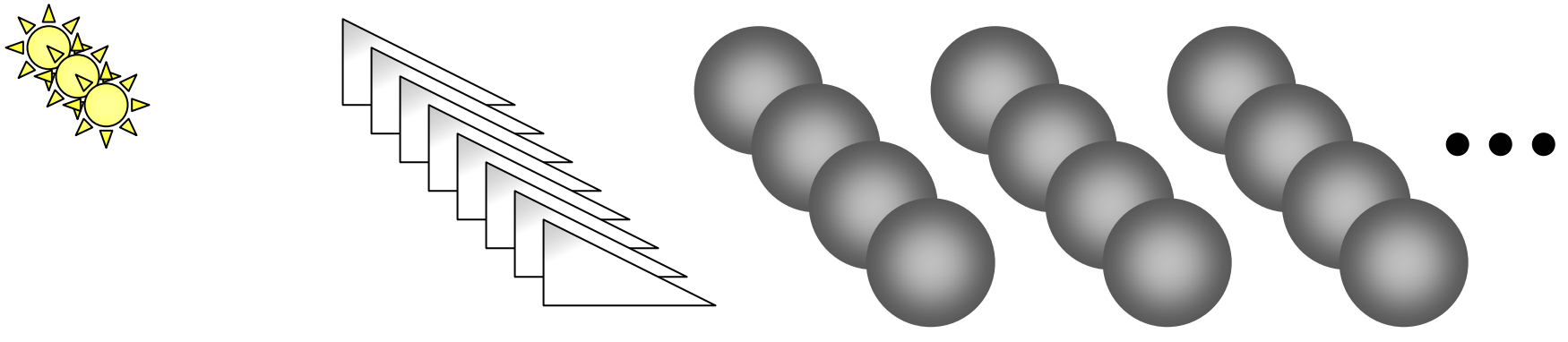
Cluster configuration

- **The Plurix 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.



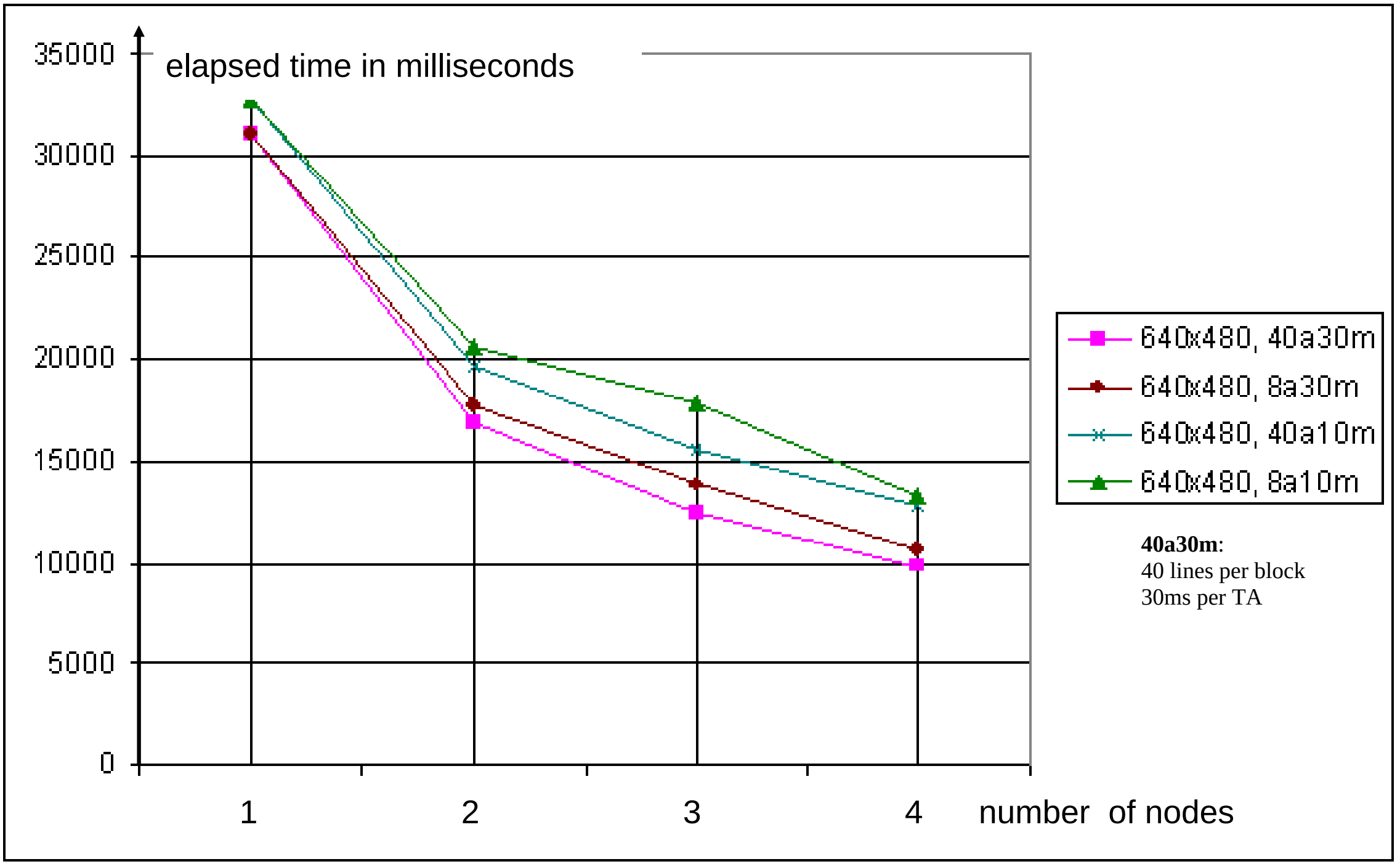
- **Ray Tracer:**
  - *m* lines per block,
  - *N* blocks per Image,
  - two phases:
    - o block allocation (short),
    - o pixel computation (longer),
  - all blocks are registered in the naming service,
  - adjustable transaction computation time interval,
  - faster nodes calculate more blocks as slower ones.

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



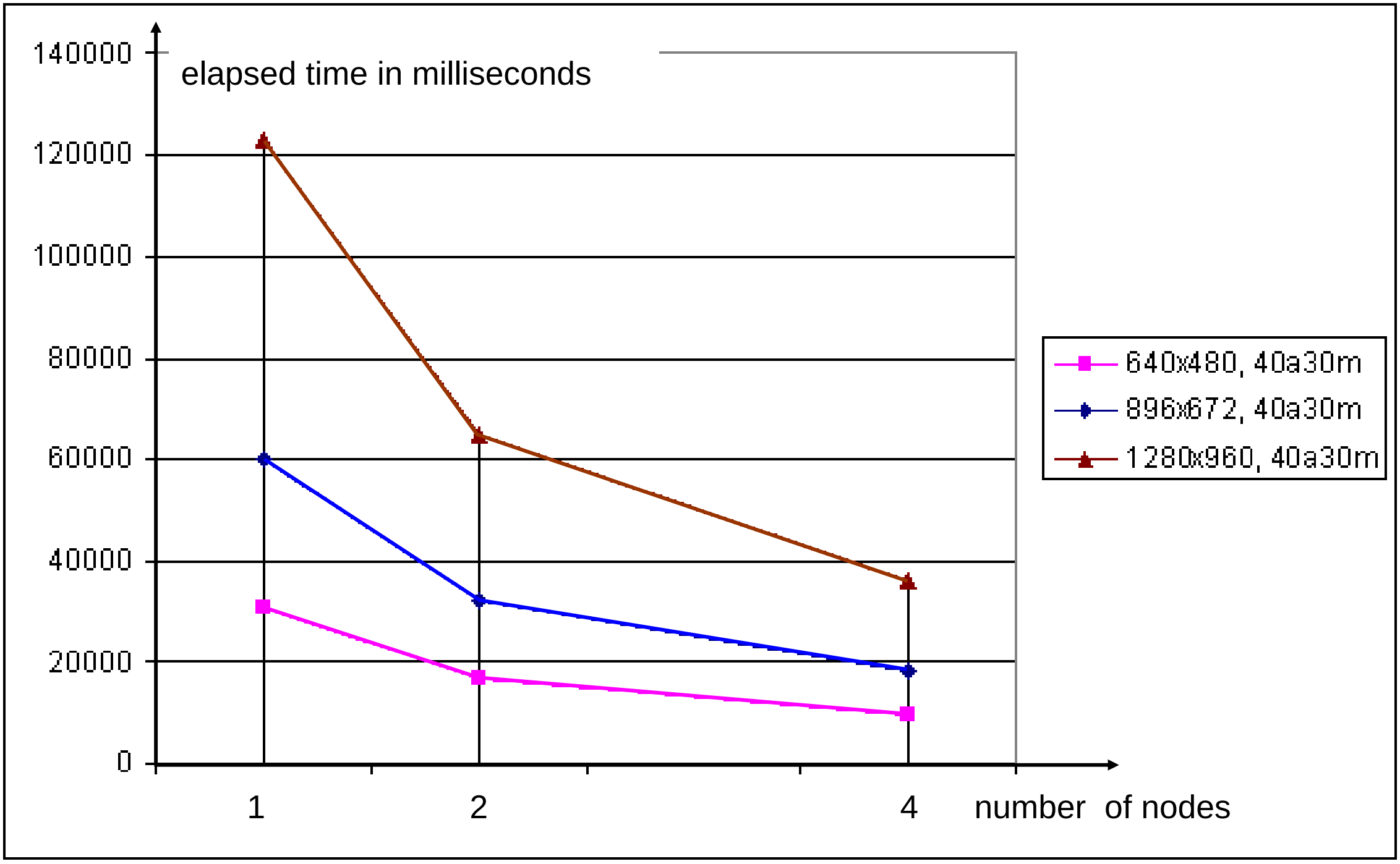
Computation time depending on blocksize and transaction time

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



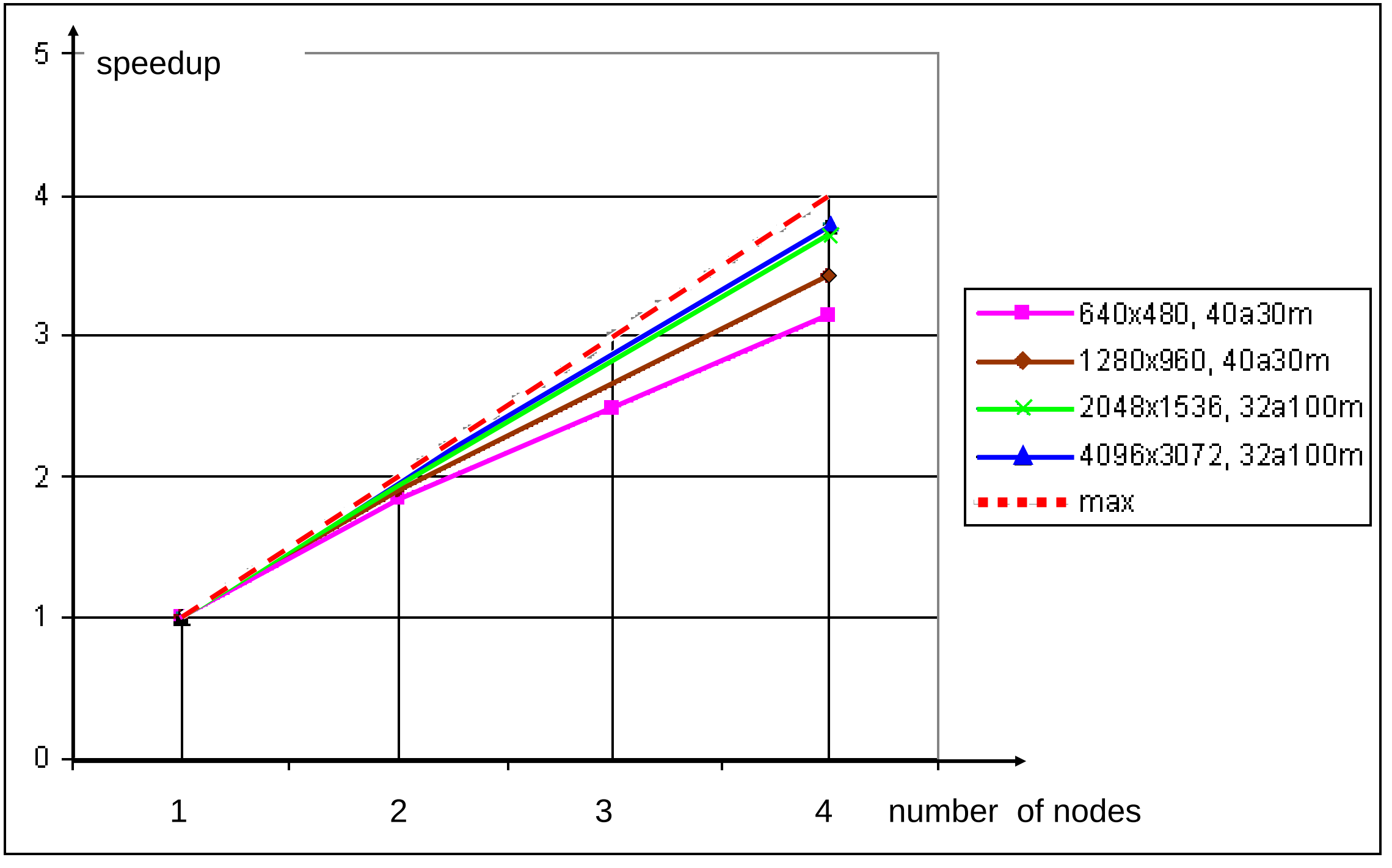
Computation time by number of nodes

- 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.



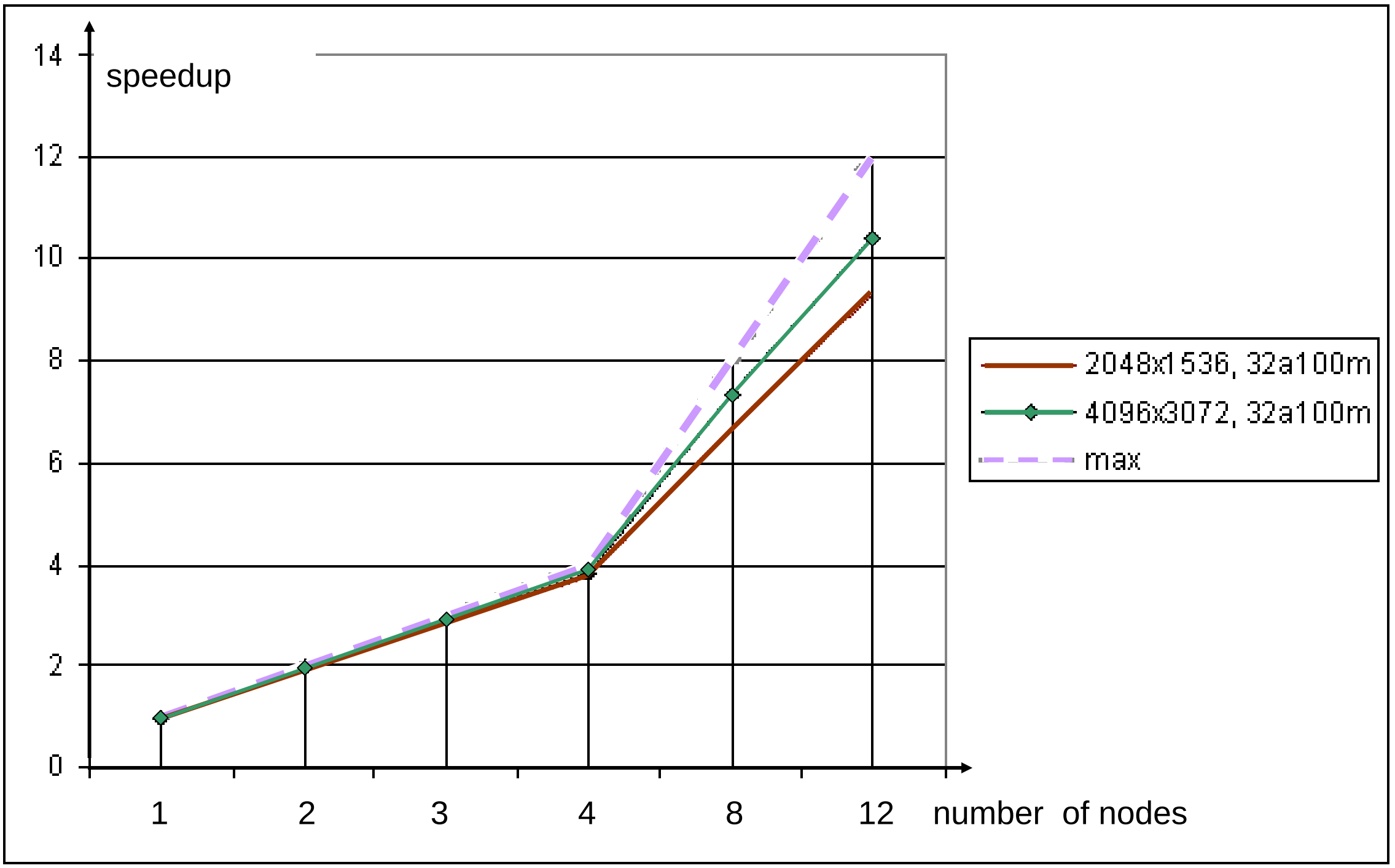
Speedup by number of nodes, varying resolution

- 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% of max. speedup).
- Very good scalability for large size images (85% of max. speedup).



Conclusion

- A transactional DSM is an alternative to message passing application frameworks.
- It provides strong consistency for all shared objects in the cluster.
- Synchronising sets of state changes (transactions) is efficient.
- 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.