A Transactional In-Memory Data Grid

Problem & Solution

- Problem: grid systems offer only message passing; data exchange and consistency management is difficult
  - No transparent data access
  - Large object structures are repeatedly passed by value
  - Need hand-written code to maintain consistency of cached data

- Solution: complement message passing by transparent remote data/object access
  - Automatic multi-consistency management
  - Peer-to-peer technologies for scalability & reliability
  - Adaptive replica control for performance and fault tolerance
  - Kaffe-based version for pointer sharing in heterogeneous setups

Target Applications

- Distributed Interactive Applications:
  - For example the multi-user virtual world Wissenheim
  - Scene graph and avatars shared using OSS
  - Game state managed using speculative transactions and weak consistency

- Number Crunching:
  - For example map & reduce
  - For cluster federations
  - Distributed caching for fast data access

Object Allocation & Replication

- Object Allocation:
  - Meta data stored in shared space
  - Object-based access detection
    - Allocating each object on a separate logical page
    - Several pages are mapped onto one page frame
  - Supports restartable object allocation within transactions

- Adaptive Replica Management:
  - Monitoring of access patterns to place replicas near accessing peers
  - And also geographically scattered for fault tolerance reasons.

Consistency Management

- Different consistency models/domains:
  - Weak and strong consistency models (within one application)
  - Stronger models can be decoupled by consistency domains

- Speculative transactions (TAs):
  - Avoiding complicated lock management and deadlocks
  - TAs are automatically restarted if a conflict is detected
  - Overlay-based commit protocols used for scalability
  - Pipelined transactions for masking commit latency
  - Transactions can call third party code

Preliminary Evaluation

- Test Applications:
  - Synthetic speculative transactions
    - Worst case: all nodes incrementing one shared variable
    - Best case: all nodes incrementing private variables, only

- Testbed:
  - Cluster with 16 nodes each with two AMD Opteron Dual Core 1.8 GHz, 2 GB RAM and Debian Linux 64 (Kernel 2.6.24.3)
  - Switched Gigabit Ethernet network
  - Average token request time 443 ms
  - Average page request time 302 ms

- Global Commit
  - Each commit synchronized globally over network

- Mixed Commit
  - Global and local commits

Future Work

- Large-scale measurements using XtreemOS testbed and Grid 5000
- Wissenheim experiments with students joining from home
- Heterogeneity addressed by a modified Kaffe version
- Optimization of scalability and fault tolerance