Object Sharing Service

Heinrich-Heine University Duesseldorf
Objectives

- Simplify data exchange & consistency management
- By supporting (shared) memory-mapped files
- Allowing transparent remote data access
- Automatic consistency management
- Complement traditional message passing:
  - Eliminate hand-written code to maintain consistency of cached data
  - Avoid passing large object structures repeatedly by value
  - Avoid deep-copy of parameters

XtreemOS IP project is funded by the European Commission under contract IST-FP6-033576
Target Applications

- **Distributed interactive applications:**
  - Multi-user applications
  - E.g. virtual worlds (Wissenheim, WP4.2)
    - Test Scene graph and avatars are accessed through OSS
    - Implicit synchronization using speculative transactions

- **Number crunching:**
  - Cluster of clusters (JuxMem)?
  - Not our major goal but OSS is open source ...
Replica & Consistency Management

- **Naming / access control through XtreemFS**
  - One file contains one or many objects
  - New objects can be allocated dynamically

- **Replication Management:**
  - Shared objects are automatically replicated
  - For performance near clients accessing objects
  - For reliability reasons also farer away

- **Consistency Management:**
  - Supporting different consistency models
    - Further models can use basic operations: *push*, *pull*, *sync*, …
  - Transactional consistency of major interest (~transactional memory)
Speculative transactions defined by the programmer.
  - BOT, EOT, Abort

Write accesses to shared objects are bundled into transactions:
  - Reduce synchronization frequency
  - Smaller number of messages
  - Avoid lock management

Write sets are validated & propagated at commit time.

In case of a conflict transactions may be aborted:
  - Changes are reset using shadow pages
  - But for modified shared objects only
- Different consistency domains.

- Local commits / read-only transactions.

- Pipelined transactions:
  - Start next transaction before a commit is validated
  - Pros: Hides latency of commit
  - Cons: May result in a cascading abort

- P2P techniques (synergies with WP3.2):
  - Hierarchical network structure (super peers)
  - Distributed hash table for data search
  - P2P server network + clients

- Weak consistent objects
• Overlay network structures.

• Transaction history buffers for recovery from missed TAs
  • Avoiding a reliable overlay multicast

• Replication of shared objects

• Grid Checkpointing for severe errors.

• XtreemFS for persistence.
Heterogeneity

- Types and data structures need to be defined using a IDL
- Language-dependent mappings by a custom pre-compiler
- Conversion mechanisms
  - Pointer-swizzling to adapt pointers to local machine architectures
  - Data conversion using IDL stubs
- Memory access detection by MMU or compiler support
- Alternative: integration of OSS into a JVM (e.g. Kaffe).
False Sharing Control

Solution: one logical memory page per object
- But several objects stored on a page frame
- Allows access detection at the object level

Pros:
- Eliminates false sharing
- Without wasting physical memory

Cons:
- Pollutes TLB (not too critical in a grid)
- Consumes more logical address space (→ 64-Bit)

Object access groups:
- For adaptive access control management
- One page fault per object access group
Conclusions

- Simplify development of distributed/parallel applications.
- Automatic replica & consistency management.
- Allowing transparent remote data accesses.
- Complement traditional message passing.
- Speculative transactions for convenience and efficiency.