**Distributed Systems (COMP9243)**

Lecture 3b: Distributed Shared Memory

---

**Slide 1**

- DSM
- Case study
- Design issues
- Implementation issues

---

**Slide 2**

**Distributed Shared Memory (DSM)**

DSM: shared memory + multicomputer

---

**Slide 3**

**Shared Address Space**

DSM consists of two components:

1. Shared address space
2. Replication and consistency of memory objects

Shared address space:

- Shared addresses are valid in all processes

---

**Slide 4**

**Transparent remote access:**

- Remote access is expensive compared to local memory access
- Individual operations can have very low overhead
- Threads can distinguish between local and remote access
Why DSM?:
- Shared memory model: easiest to program to
- Physical shared memory not possible on multicomputer
- DSM emulates shared memory

Benefits of DSM:
- Ease of programming (shared memory model)
- Eases porting of existing code
- Pointer handling
  - Shared pointers refer to shared memory
  - Share complex data (lists, etc.)
- No marshalling

DSM IMPLEMENTATIONS

Hardware:
- Multiprocessor
- Example: MIT Alewife, DASH

OS with hardware support:
- SCI network cards (SCI = Scalable Coherent Interconnect)
- SCI maps extended physical address space to remote nodes
- OS maps shared virtual address space to SCI range

OS and Virtual Memory:
- Virtual memory (page faults, paging)
- Local address space vs Large address space

Middleware:
- Library:
  - Library routines to create/access shared memory
  - Example: MPI-2, CRL
- Language
  - Shared memory encapsulated in language constructs
  - Extend language with annotations
  - Example: Orca, Linda, JavaSpaces, JavaParty, Jackal

Typical Implementation:
- Most often implemented in user space (e.g., TreadMarks, CVM)
- User space: what’s needed from the kernel?
  - User-level fault handler
    (e.g., Unix signals)
  - User-level VM page mapping and protection
    (e.g., mmap() and mprotect())
  - Message passing layer
    (e.g., socket API)
Example: two processes sharing memory pages:

Slide 9

Slide 10

Occurrence of a read fault:

Slide 11

Page migration and replication:

Slide 12

Recovery from read fault:
DSM MODELS

Shared page (coarse-grained):

- Traditional model
- Ideal page size?
- False sharing
- Examples: Ivy, TreadMarks

Shared region (fine-grained):

- More fine grained than sharing pages
- Prevent false sharing
- Not regular memory access (transparency)
- Examples: CRL (C Region Library), MPI-2 one-sided communication, Shasta

Shared variable:

- Release and Entry based consistency
- Annotations
- Fine grained
- More complex for programmer
- Examples: Munin, Midway

Shared structure:

- Encapsulate shared data
- Access only through predefined procedures (e.g., methods)
- Tightly integrated synchronisation
- Encapsulate (hide) consistency model
- Lose familiar shared memory model
- Examples: Orca (shared object), Linda (tuple space)

Tuple Space:

- A JavaSpace

LINDA EXAMPLE

```java
main() {
    ... eval("function", f()) ; eval("function", f()) ;
    ... for (i=0; i<100; i++)
        out("data", i) ;
    ...
}
f(){
    in("data", ?x) ;
    y = g(x) ;
    out("function", x, y) ;
}
```

What’s good about this?
Slide 17

APPLICATIONS OF DSM

- Scientific parallel computing
  - Bioinformatics (gene sequence analysis)
  - Simulations (climate modeling, economic modeling)
  - Data processing (physics, astronomy)
- Graphics (image processing, rendering)
- Data server (distributed FS, Web server)
- Data storage

Slide 18

DSM ENVIRONMENTS

- Multiprocessor
  - NUMA
- Multicomputer
  - Supercomputer
  - Cluster
  - Network of Workstations
  - Wide-area

Slide 19

REQUIREMENTS OF DSM

- Transparency:
  - Location, migration, replication, concurrency
- Reliability:
  - Computations depend on availability of data
- Performance:
  - Important in high-performance computing
  - Important for transparency
- Scalability:
  - Important in wide-area
  - Important for large computations

Slide 20

- Consistency:
  - Access to DSM should be consistent
  - According to a consistency model
- Programmability:
  - Easy to program
  - Communication transparency
Slide 21

**CASE STUDY**

TreadMarks:
- 1992 Rice University
- Page based DSM library
- C, C++, Java, Fortran
- Lazy release consistency model
- Heterogeneous environment

Slide 22

**DESIGN ISSUES**

- **Granularity**
  - Page based, Page size: minimum system page size
- **Replication**
  - Lazy release consistency
- **Scalability**
  - Meant for cluster or NOW (Network of Workstations)
- **Synchronisation primitives**
  - Locks (acquire and release), Barrier
- **Heterogeneity**
  - Limited (doesn’t address endianness or mismatched word sizes)

Slide 23

**USING TREADMARKS**

Compiling:
- Compile
- Link with TreadMarks libraries

Starting a TreadMarks Application:

```sh
app -- -h host1 -h host2 -h host3 -h host4
```

Slide 24

Anatomy of a TreadMarks Program:

- **Starting remote processes**
  ```c
  Tmk_startup(argc, argv);
  ```
- **Allocating and sharing memory**
  ```c
  shared = (struct shared*) Tmk_Malloc(sizeof(shared));
  Tmk_distribute(&shared, sizeof(shared));
  ```
- **Barriers**
  ```c
  Tmk_barrier(0);
  ```
- **Acquire/Release**
  ```c
  Tmk_lock_acquire(0);
  shared->sum += mySum;
  Tmk_lock_release(0);
  ```
TreadMarks Implementation

Consistency Protocol:
- Multiple writer
- Twins
- Reduce false sharing

Update Propagation:
- Modified pages invalidated at acquire
- Page is updated at access time
- Updates are transferred as diffs

Lazy Diffs:
- Normally make diffs at release time
- Lazy: make diffs only when they are requested

Communication:
- UDP/IP or AAL3/4 (ATM)
- Light-weight, user-level protocols to ensure message delivery
- Use SIGIO for message receive notification

Data Location:
- Know who has diffs because of invalidations
- Each page has a statically assigned manager

Modification Detection:
- Page Fault
- If page is read-only then do consistency protocol
- If not in local memory, get from manager

Memory Management:
- Garbage collection of diffs

Initialisation:
- Processes set up communication channels between themselves
- Register SIGIO handler for communication
- Allocate large block of memory
  - Same (virtual) address on each machine
  - Mark as non-accessible
  - Assign manager process for each page, lock, barrier (round robin)
- Register SEGV handler
READING LIST

Distributed Shared Memory: A Survey of Issues and Algorithms
An overview of DSM and key issues as well as older DSM implementations.

TreadMarks: Shared Memory Computing on Networks of Workstations
An overview of TreadMarks, design decisions and implementation.

HOMEWORK

Slide 30
Do Assignment 1!