9. Distributed Shared Memory

Provide the usual programming model of shared memory in a – generally loosely coupled – distributed environment.

- Shared Memory
  - Easy to program
  - Difficult to build
    - Tight coupling in hardware
- Distributed Memory
  - Scales well
  - Difficult to program
    - Message passing or RPC based

- Solution: DSM (Distributed Shared Memory)
  - Logically shared memory
  - Physically distributed local memories
  - Page based
    - Shared pages
    - Demand paging between nodes (Li)
  - Variable based
    - Shared Variables
  - Object based (Linda, Orca etc.)
    - Shared encapsulation units
    - Easier to optimize
9.1. Non Uniform Memory Access (NUMA)

- NUMA (Non Uniform Memory Access)
  - The access (mainly its time) to different memory locations depends on their physical location
  - Interconnection network – e.g. a bus or a LAN
  - Granularity – maybe a word, a page, a segment
  - The access is supported by hardware – apart from speed it is transparent

- A generic NUMA architecture:
• **Multiprocessor Architecture**
  - A global, common memory exists
  - The memory blocks maybe cached in local memories
  - Connection is fast and tight (common bus, switch etc.)
  - Main goal is *performance*, based on the *locality* principle

• **Distributed Shared Memory Architecture**
  - No global memory
  - The local memories form a global, virtual memory
  - Connection is relatively slow and loose (LAN, MAN …)
  - *Scalability*
  - *Transparency* of
    - information sharing
    - communication
9.2. Page-based DSM

- The IVY system (Li and Hudak, 1986 …)
- No remote memory access (NORMA)
  - In (true) NUMA machines the reference to remote memories is supported by the hardware
  - In NORMA systems the computers see only the local memory – everything else must be done in software
  - Can be implemented on top of any LAN
  - With the help of the normal Memory Management Unit
  - Multiprocessor programs over multicomputer systems
    - Sequential consistency must be provided
    - Relaxing the model requires rewrite of the programs
    - Multicomputer systems cost typically less
  - Sequential consistency is no problem without replication
- Granularity
  - Cannot be smaller than a page
    - The MMU works with pages
  - Page
    - Maybe an integral number of the basic page size
      - The communication time for a large page is not much greater than that of a small page
      - Increases the probability of *false sharing*
9.2.1. Basic Design

- Emulate multiprocessor cache by MMU + operating system
- On a missing page
  - Page fault is generated
  - The missing page is fetched by the DSM software

![Diagram of Shared Global Virtual Address Space]

Ref. to P7
9.2.2. Replication + Sequential Consistency

- Easy for read-only pages
  - E.g. pages containing only code
- Difficult for writable pages
- If a page is required a copy is made
  - We don’t know whether this is read-only or not
- Replicated pages are marked as write protected (read-only)
  - A write access causes a trap
  - All copies must be invalidated
  - An update of individual words would be too expensive
    - The software does not know the individual words
    - To emulate a common address bus would be very costly
- Invalidation protocol
  - Every page has an owner – the process that most recently wrote the page
  - Apply the implementations of the consistency models (consistency protocols) as described in the course on replication and consistency.
9.2.3. Further Administration

Finding the Owner

• Send a broadcast request
  ➢ Piggyback additional info about the sender’s wish
    ▪ Read / write, needs a copy etc.
  ➢ Causes network load and interrupts n-1 processors
• Page manager (central component)
  ➢ The page manager knows the owner of all pages
    1. Send a request to the page manager
    2. The page manager replies the number of the owner
    3. Send a request to the owner
    4. The owner sends a reply
  ➢ An enhancement of the protocol (2 and 3 in one step)
    ▪ The page manager forwards the request to the owner
• Distributed algorithm
  ▪ Every node keeps track of the probable owner
  ▪ The probable owner may forward the request further

Finding the Copies for Invalidation

• Send a broadcast request
  ➢ Must be fully reliable
• The owner or the page manager maintains a copy set

Page Replacement

• Known techniques like LRU can be used
• Spontaneous invalidation may effect the selection

Synchronization

• Ping-pong with S-variables must be avoided
• Central manager is better
9.3. Shared-Variable DSM

- Only a limited number of variables are shared
- False sharing can be excluded
- The MMU can be used if we place each shared variable on an own page
- Release consistency can be applied easily
- Classes of variables
  - Ordinary variables
    - Unshared
  - Shared variables
    - Release consistency must apply
  - S-variables
    - Sequential consistency must apply

```plaintext
Lock (L);
A:= 1;
B:= 2;
C:= 3;
Unlock(L);

A, B, C
A, B, C
```
9.4. Object-based DSM

- The DSM consists of objects which are globally visible
  - Globally known identity
  - Name
  - Moving
  - Replication

- Linda (David Gelernter)
  - Tuple-space
  - Language extension

- Orca (Andrew S. Tanenbaum + Henry Bal)
  - Own language + distributed environment