Advanced Operating Systems Lecture notes

Dongho Kim
Tatyana Ryutov
University of Southern California
Information Sciences Institute

Administration

- **Class e-mail: csci555@usc.edu**
- **Office hours: SAL 234, Friday 11am to noon**
- **Reading report #1 will be posted during the week-end**
- **TAs**
  - **Chansook Lim**
    - **Office: SAL 317 ; (213) 740-6502**
    - **e-mail chansool at usc dot edu**
    - **Office Hours: Tuesday, 10 a.m. -- noon**
  - **Sunhee Yoon**
    - **Office: SAL 211 ; (213) 740-4508**
    - **e-mail sunheeyo at usc dot edu**
    - **Office Hours: Thursday, 1p.m. -- 3 p.m.**
- **Class Web page**
  http://gost.isi.edu/courses/usc_csci555.html
  - **Reading list !**

CSci555: Advanced Operating Systems

Lecture 2 – September 2, 2005

Dr. Tatyana Ryutov
Dr. Dongho Kim
University of Southern California
Information Sciences Institute

Outline: Communications Models

- **Communication Models:**
  - **General concepts.**
  - **Message passing.**
  - **Distributed shared memory (DSM).**
  - **Remote procedure call (RPC) [Birrel et al.]**
    - **Light-weight RPC [Bershad et al.]**
  - **DSM case studies**
    - **IVY [Li et al.]**
    - **Linda [Carriero et al.]**

Communication Models

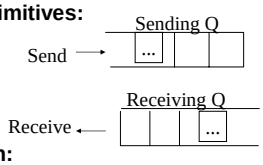
- **Support for processes to communicate among themselves.**
- **Traditional (centralized) OS's:**
  - **Provide local (within single machine) communication support.**
  - **Distributed OS's: must provide support for communication across machine boundaries.**
    - **Over LAN or WAN.**

Communication Paradigms

- **2 paradigms**
  - **Message Passing (MP)**
  - **Distributed Shared Memory (DSM)**
- **Message Passing**
  - **Processes communicate by sending messages.**
- **Distributed Shared Memory**
  - **Communication through a "virtual shared memory".**

Message Passing

- **Basic communication primitives:**
  - **Send message.**
  - **Receive message.**



- **Modes of communication:**
  - **Synchronous versus asynchronous.**
- **Semantics:**
  - **Reliable versus unreliable.**

Synchronous Communication

- **Blocking send**
  - **Blocks until message is transmitted**
  - **Blocks until message acknowledged**
- **Blocking receive**
  - **Waits for message to be received**
- **Process synchronization.**

Asynchronous Communication

- **Non-blocking send: sending process continues as soon message is queued.**
- **Blocking or non-blocking receive:**
  - **Blocking:**
    - **Timeout.**
    - **Threads.**
  - **Non-blocking: proceeds while waiting for message.**
    - **Message is queued upon arrival.**
    - **Process needs to poll or be interrupted.**

Reliability of Communication

- **Unreliable communication:**
  - **“best effort” - send and hope for the best**
  - **No ACKs or retransmissions.**
  - **Application must provide its own reliability.**
  - **Example: User Datagram Protocol (UDP)**
    - **Applications using UDP either don’t need reliability or build their own (e.g., UNIX NFS and DNS (both UDP and TCP), some audio or video applications)**

Reliability of Communication

- **Reliable communication:**
  - **Different degrees of reliability.**
  - **Processes have some guarantee that messages will be delivered.**
  - **Example: Transmission Control Protocol (TCP)**
  - **Reliability mechanisms:**
    - **Positive acknowledgments (ACKs).**
    - **Negative Acknowledgments (NACKs).**
  - **Possible to build reliability atop unreliable service (E2E argument).**

Distributed Shared Memory

- **Motivated by development of shared-memory multiprocessors which do share memory.**
- **Abstraction used for sharing data among processes running on machines that do** not **share memory.**
- **Processes think they read from and write to a “virtual shared memory”.**

DSM 2

- **Primitives: read and write.**
- **OS ensures that all processes see all updates.**
  - **Happens transparently to processes.**

DSM and MP

- **DSM is an abstraction!**
  - **Gives programmers the flavor of a centralized memory system, which is a well-known programming environment.**
  - **No need to worry about communication and synchronization.**
- **But, it is implemented atop MP.**
  - **No physically shared memory.**
  - **OS takes care of required communication.**

Caching in DSM

- **For performance, DSM caches data locally.**
  - **More efficient access (locality).**
  - **But, must keep caches consistent.**
  - **Caching of pages for of page-based DSM.**
- **Issues:**
  - **Page size.**
  - **Consistency mechanism.**

Approaches to DSM

- **Hardware-based:**
  - **Multi-processor architectures with processor-memory modules connected by high-speed LAN (E.g., Stanford's DASH).**
  - **Specialized hardware to handle reads and writes and perform required consistency mechanisms.**

Approaches to DSM

- **Page-based:**
  - **Example: IVY.**
  - **DSM implemented as region of processor's virtual memory; occupies same address space range for every participating process.**
  - **OS keeps DSM data consistency as part of page fault handling.**

Approaches to DSM

- **Library-based:**
  - **Or language-based.**
  - **Example: Linda.**
  - **Language or language extensions.**
  - **Compiler inserts appropriate library calls whenever processes access DSM items.**
  - **Library calls access local data and communicate when necessary.**

DSM Case Studies: IVY

- **Environment:"loosely coupled" multiprocessor.**
  - **Memory is physically distributed.**
  - **Memory mapping managers (OS kernel):**
    - **Map local memories to shared virtual space.**
    - **Local memory as cache of shared virtual space.**
    - **Memory reference may cause page fault; page retrieved and consistency handled.**

IVY

- **Issues:**
  - **Read-only versus writable data.**
  - **Locality of reference.**
  - **Granularity (1 Kbyte page size).**
    - **Bigger pages versus smaller pages.**

IVY

- **Memory coherence strategies:**
  - **Page synchronization**
    - **Invalidation**
    - **Write broadcast**
  - **Page ownership**
    - **Fixed: page always owned by same processor**
    - **Dynamic**

IVY Page Synchronization

- **Invalidation:**
  - **On write fault, invalidate all copies; give faulting process write access; gets copy of page if not already there.**
  - **Problem: must update page on reads.**
- **Write broadcast:**
  - **On write fault, fault handler writes to all copies.**
  - **Expensive!**

IVY Memory Coherence

- **Paper discusses approaches to memory coherence in page-based DSM.**
  - **Centralized: single manager residing on a single processor managing all pages.**
  - **Distributed: multiple managers on multiple processors managing subset of pages.**

DSM Case Studies: Linda

- **Language-based approach to DSM.**
- **Environment:**
  - **Similar to IVY, ie, loosely coupled machines connected via fast broadcast bus.**
  - **Instead of shared address space, processes make library calls inserted by compiler when accessing DSM.**
  - **Libraries access local data and communicate to maintain consistency.**

Linda

- DSM: tuple space.
- Basic operations:
  - out (data): data added to tuple space.
  - in (data): removes matching data from TS; destructive.
  - read (data): same as “in”, but tuple remains in TS (non-destructive).

Linda Primitives: Examples

- out (“P”, 5, false) : tuple (“P”, 5, false) added to TS.
  - “P” : name
  - Other components are data values.
  - Implementation reported on the paper: every node stores complete copy of TS.
  - out (data) causes data to be broadcast to every node.

Linda Primitives: Examples

- in (“P”, int I, bool b): tuple (“P”, 5, false) removed from TS.
  - If matching tuple found locally, local kernel tries to delete tuple on all nodes.

Remote Procedure Call

- Builds on MP.
- Main idea: extend traditional (local) procedure call to perform transfer of control and data across network.
- Easy to use: analogous to local calls.
- But, procedure is executed by a different process, probably on a different machine.
- Fits very well with client-server model.

RPC Mechanism

1. Invoke RPC.
2. Calling process suspends.
3. Parameters passed across network to target machine.
4. Procedure executed remotely.
5. When done, results passed back to caller.
6. Caller resumes execution.

Is this synchronous or asynchronous?

RPC Advantages

- Easy to use.
- Well-known mechanism.
- Abstract data type
  - Client-server model.
  - Server as collection of exported procedures on some shared resource.
  - Example: file server.
- Reliable.

RPC Semantics 1

- **Delivery guarantees.**
- **“Maybe call”:**
  - **Clients cannot tell for sure whether remote procedure was executed or not due to message loss, server crash, etc.**
  - **Usually not acceptable.**

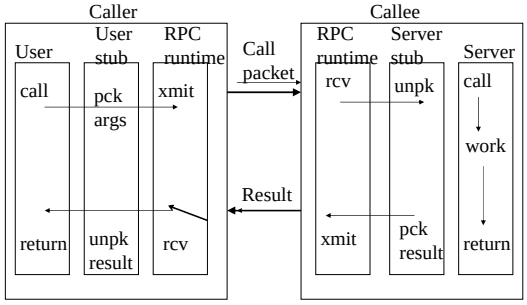
RPC Semantics 2

- **“At-least-once” call:**
  - **Remote procedure executed at least once, but maybe more than once.**
  - **Retransmissions but no duplicate filtering.**
  - **Idempotent operations OK; e.g., reading data that is read-only.**

RPC Semantics 3

- **“At-most-once” call**
  - **Most appropriate for non-idempotent operations.**
  - **Remote procedure executed 0 or 1 time, ie, exactly once or not at all.**
  - **Use of retransmissions and duplicate filtering.**
  - **Example: Birrel et al. implementation.**
    - **Use of probes to check if server crashed.**

RPC Implementation (Birrel et al.)



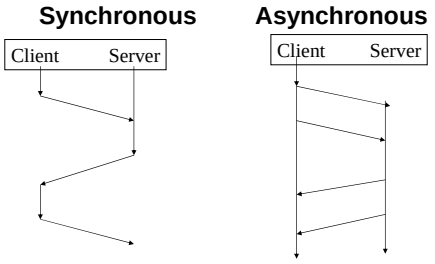
RPC Implementation 2

- **RPC runtime mechanism responsible for retransmissions, acknowledgments.**
- **Stubs responsible for data packaging and un-packaging;**
  - **AKA marshalling and un-marshalling: putting data in form suitable for transmission. Example: Sun’s XDR.**

Binding

- **How to determine where server is? Which procedure to call?**
  - **“Resource discovery” problem**
    - **Name service: advertises servers and services.**
    - **Example: Birrel et al. uses Grapevine.**
- **Early versus late binding.**
  - **Early: server address and procedure name hard-coded in client.**
  - **Late: go to name service.**

Synchronous & Asynchronous RPC



RPC Performance

- **Sources of overhead**
  - **data copying**
  - **scheduling and context switch.**
- **Light-Weight RPC**
  - **Shows that most invocations took place on a single machine.**
  - **LW-RPC: improve RPC performance for local case.**
  - **Optimizes data copying and thread scheduling for local case.**

LW-RPC 1

- **Argument copying**
  - **RPC: 4 times**
  - **copying between kernel and user space.**
  - **LW-RPC: common data area (A-stack) shared by client and server and used to pass parameters and results; access by client or server, one at a time.**

LW-RPC 2

- **A-stack avoids copying between kernel and user spaces.**
- **Client and server share the same thread: less context switch (like regular calls).**