

Processes

3 November 2009
Lecture 3

Slide Credits: Insup Lee (UPenn)

System Architectures for Applications

- Centralized Architecture
 - Client-server model
 - Application layering
 - Multitiered architecture
- Decentralized Architecture
 - Peer-to-peer architecture
 - Distributed hash table (DHT)
 - Overlay Networks
- Hybrid Architectures
 - Edge – Server
 - Collaborative

November 3, 2009

ISE 431: Distributed Information Systems

2

Edge-Server Systems (1)

- A hybrid architecture
- An idempotent class of distributed systems
- Deployed on the Internet where servers are “at the edge” of the network (i.e. first entry to network)
- Each client connects to the Internet by means of an edge server

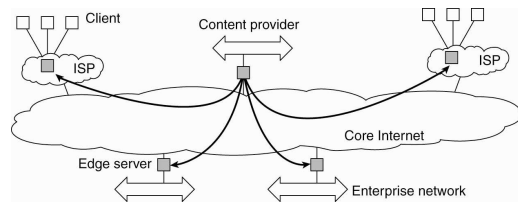
November 3, 2009

ISE 431: Distributed Information Systems

3

Edge-Server Systems (2)

- Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.



November 3, 2009

ISE 431: Distributed Information Systems

4

Collaborative Distributed Systems (1)

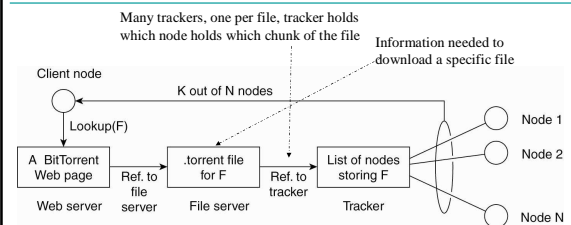
- A hybrid distributed model that is based on mutual collaboration of various systems
- Examples of Collaborative Distributed System:
 - BitTorrent: is a P2p File downloading system. It allows download of various chunks of a file from other users until the entire file is downloaded
 - Globule: A Collaborative content distribution network. It allows replication of web pages by various web servers

November 3, 2009

ISE 431: Distributed Information Systems

5

Collaborative Distributed Systems (2)



- Figure 2-14. The principal working of BitTorrent [adapted with permission from Pouwelse et al. (2004)].

November 3, 2009

ISE 431: Distributed Information Systems

6

Collaborative Distributed Systems (3)

- Components of Globule collaborative content distribution network: (replication of web pages by various users)
 - A component that can redirect client requests to other servers.
 - A component for analyzing access patterns.
 - A component for managing the replication of Web pages.

November 3, 2009

ISE 431: Distributed Information Systems

7

Collaborative Distributed Systems (4)

- Benefits:
- Example:
 - Alice has a web server; Bob has a web server
 - Alice's server can have replicated contents of the Bob's server and vice versa
- Good if your server goes down
- Good if too much traffic that your server can not handle or server gets too slow
- Better geographic diversity
- End users voluntarily provide enhanced web servers that are capable of collaborating in the replication of web pages

November 3, 2009

ISE 431: Distributed Information Systems

8

Introduction to Threads

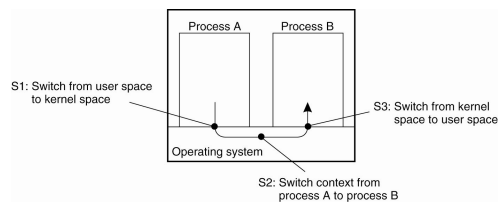
- **Basic idea:** we build **virtual processors** in software, on top of physical processors:
- **Processor:** Provides a set of instructions along with the capability of automatically executing a series of those instructions.
- **Thread:** A minimal software processor in whose **context** a series of instructions can be executed. Saving a thread context implies stopping the current execution and saving all the data needed to continue the execution at a later stage.
- **Process:** A software processor in whose context one or more threads may be executed. Executing a thread, means executing a series of instructions in the context of that thread.

November 3, 2009

ISE 431: Distributed Information Systems

9

Thread Usage in Nondistributed Systems



- Context switching as the result of IPC.

November 3, 2009

ISE 431: Distributed Information Systems

10

Context Switching (1/2)

- **Processor context:** The minimal collection of values stored in the registers of a processor used for the execution of a series of instructions (e.g., stack pointer, addressing registers, program counter).
- **Thread context:** The minimal collection of values stored in registers and memory, used for the execution of a series of instructions (i.e., processor context, state).
- **Process context:** The minimal collection of values stored in registers and memory, used for the execution of a thread (i.e., thread context, but now also at least MMU register values).

November 3, 2009

ISE 431: Distributed Information Systems

11

Context Switching (2/2)

- **Observation 1:** Threads share the same address space. Thread context switching can be done entirely independent of the operating system.
- **Observation 2:** Process switching is generally more expensive as it involves getting the OS in the loop, i.e., trapping to the kernel.
- **Observation 3:** Creating and destroying threads is much cheaper than doing so for processes.

November 3, 2009

ISE 431: Distributed Information Systems

12

Threads and Operating Systems (1/2)

Main issue: Should an OS kernel provide threads, or should they be implemented as user-level packages?

User-space solution:

- We'll have nothing to do with the kernel, so all operations can be completely handled **within a single process** → implementations can be extremely efficient.
- **All** services provided by the kernel are done **on behalf of the process in which a thread resides** → if the kernel decides to block a thread, the entire process will be blocked. Requires messy solutions.
- In practice we want to use threads when there are lots of external events: **threads block on a per-event basis** → if the kernel can't distinguish threads, how can it support signaling events to them?

November 3, 2009

ISE 431: Distributed Information Systems

13

Threads and Operating Systems (2/2)

- **Kernel solution:** The whole idea is to have the kernel contain the implementation of a thread package. This does mean that *all* operations return as system calls
 - Operations that block a thread are no longer a problem: the **kernel schedules another available thread** within the same process.
 - Handling external events is simple: the **kernel** (which catches all events) **schedules the thread associated with the event**.
 - The big problem is the **loss of efficiency** due to the fact that each thread operation requires a trap to the kernel.
- **Conclusion:** Try to mix user-level and kernel-level threads into a single concept.

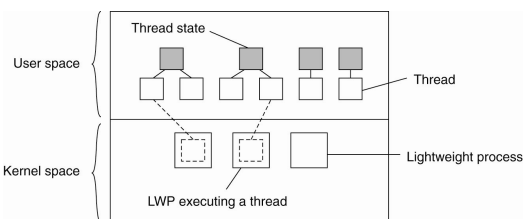
November 3, 2009

ISE 431: Distributed Information Systems

14

Solaris Threads (1/2)

- **Basic idea:** Introduce a two-level threading approach:
 - **lightweight processes** that can execute user-level threads.



November 3, 2009

ISE 431: Distributed Information Systems

15

Solaris Threads (2/2)

- When a user-level thread does a system call, **the LWP that is executing that thread, blocks**. The thread remains **bound** to the LWP.
- The kernel can simply **schedule another LWP having a runnable thread bound to it**. Note that this thread can switch to **any** other runnable thread currently in user space.
- When a thread calls a blocking user-level operation, we can simply do a context switch to a runnable thread, which is then bound to the same LWP.
- When there are no threads to schedule, an LWP may remain idle, and may even be removed (destroyed) by the kernel.
- **Note:** This concept has been virtually abandoned – it's just either user-level or kernel-level threads.

November 3, 2009

ISE 431: Distributed Information Systems

16

Threads and Distributed Systems (1/2)

- **Multithreaded clients:** Main issue is hiding network latency.
- **Multithreaded Web client:**
 - Web browser scans an incoming HTML page, and finds that **more files need to be fetched**.
 - **Each file is fetched by a separate thread**, each doing a (blocking) HTTP request.
 - As files come in, the browser displays them.
- **Multiple request-response calls to other machines (RPC):**
 - A client does several calls at the same time, each one by a different thread.
 - It then waits until all results have been returned.
 - Note: if calls are to different servers, we may have a **linear speed-up** compared to doing calls one after the other.

November 3, 2009

ISE 431: Distributed Information Systems

17

Threads and Distributed Systems (2/2)

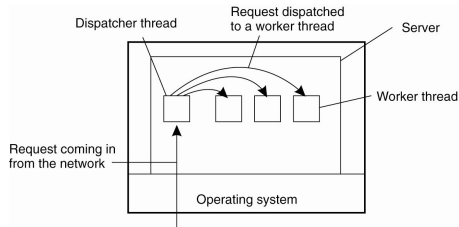
- **Multithreaded servers:** Main issue is improved performance and better structure.
- **Improve performance:**
 - Starting a thread to handle an incoming request is **much** cheaper than starting a new process.
 - Having a single-threaded server prohibits simply scaling the server to a **multiprocessor system**.
 - As with clients: **hide network latency** by reacting to next request while previous one is being replied.
- **Better structure:**
 - Most servers have high I/O demands. Using simple, **well-understood blocking calls** simplifies the overall structure.
 - Multithreaded programs tend to be **smaller and easier to understand** due to **simplified flow of control**.

November 3, 2009

ISE 431: Distributed Information Systems

18

Multithreaded Servers (1)



- A multithreaded server organized in a dispatcher/worker model.

November 3, 2009

ISE 431: Distributed Information Systems

19

Multithreaded Servers (2)

- Three ways to construct a server.

Model	Characteristics
Threads	Parallelism, blocking system calls
Single-threaded process	No parallelism, blocking system calls
Finite-state machine	Parallelism, nonblocking system calls

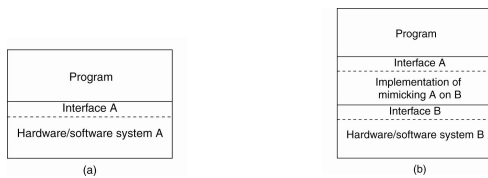
November 3, 2009

ISE 431: Distributed Information Systems

20

Virtualization

- **Observation:** Virtualization is becoming increasingly important:
 - Hardware changes faster than software
 - Ease of portability and code migration
 - Isolation of failing or attacked components



November 3, 2009

ISE 431: Distributed Information Systems

21

Virtualization

- Virtualization is to extend or replace an existing interface to mimic the behavior of another system.
- IBM 370 mainframe, VMM (Virtual Machine Monitor) - 1970s
 - Support multi users by one VM per user
 - Support different operating systems
- After 1990s,
 - To provide legacy interface on new hardware platforms
 - To provide uniformity over a heterogeneous collection of servers connected by networks
 - To provide a high degree of portability and flexibility

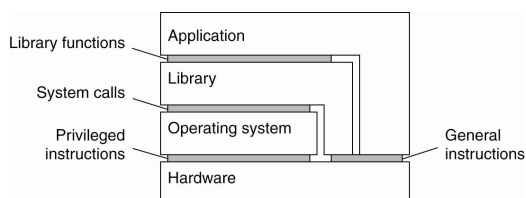
November 3, 2009

ISE 431: Distributed Information Systems

22

Architecture of VMs

- **Observation:** Virtualization can take place at very different levels, strongly depending on the interfaces as offered by various systems components:



November 3, 2009

ISE 431: Distributed Information Systems

23