Sistem Terdistribusi

Sistem Operasi Terdistribusi
Peran sistem operasi

• Menyediakan abstraksi **kontrol sumber daya fisik** bagi user
• Manajemen **resource**
• Menyediakan **sistem call** terhadap sumber daya baik fisik / non fisik
  • Dalam bentuk API
    • Win 32 api, POSIX api, Java api, .NET api
• Jenis OS:
  – Desktop OS
  – Network OS
  – Mobile OS
  – Distributed OS
System layers

Applications, services

Middleware

OS: kernel, libraries & servers

OS1
Processes, threads, communication, ...

Computer & network hardware

Node 1

OS2
Processes, threads, communication, ...

Computer & network hardware

Node 2

Platform
OS Tasks

• Raise the **programming interface** for resources to a more useful level:
  – By providing **abstractions / encapsulation** of the basic resources such as:
    * processes, virtual memory, files, communication channels*
  – **Protection** of the resources used by applications
  – **Concurrent processing** to enable applications to complete their work with minimum interference from other applications

• **Provide** the resources needed for (distributed) services and applications to complete their task:
  – **Communication** - network access provided
  – **Scheduling** - processors scheduled at the relevant computers
Core OS functionality

- **Process manager**: Creation & operation upon processes / programs
- **Communication manager**: Communication between threads / programs on same computer
- **Thread manager**
- **Memory manager**
- **Supervisor**: Dispatching interrupts, system call, trapping
Process address space

- Memory’s regions can be shared
  - kernel code
  - libraries
  - shared data & communication
  - stack / heap
  - **copy-on-write system**

- Files can be mapped to memory
  - Virtual memory system
Copy-on-write

Process A's address space

RA

Process B's address space

RB

RB copied from RA

Kernel

A's page table

Shared frame

B's page table

a) Before write

b) After write
Copy on Write

• Region RA dan RB merupakan sharing memori dengan teknik **copy-on write** antara dua proses A dan B

• Ketika proses B hendak menulis ke share memory, maka akan terjadi **memory protection page fault**, sehingga akan mengalokasikan frame baru dari hasil duplikasi frame asli sehingga akan terdapat 2 page frame

• Data bisa dishare tanpa “merusak” data masing-masing jika ada penulis / pengupdate-an
Thread

• Thread = *Lightweight Process*
• Thread = *satuan dasar* penggunaan CPU
• Pembuatan Thread dilakukan oleh:
  – *Kernel Thread* – lebih lambat
  – *User Thread* – lebih cepat, berbasis API
• Kernel juga digunakan dalam Distributed OS
  – Menggunakan konsep *multithreading*
Single and Multithreaded Processes

- Single-threaded process
- Multithreaded process
Thread Life Cycles

- New
  - Program starts thread
  - Runnable
    - Unlock
    - Signal
    - SignalAll
    - Await
    - Sleep
    - Timed waiting
    - Interval
    - Expires
    - Terminated
    - Task completes

- Waiting
- Timed waiting
- Terminated
Client and server with threads

Thread 1 generates results

Thread 2 makes requests to server

Receipt & queuing

Requests

Input-output

N threads

The threads 'worker pool' architecture on server
Alternative server threading architectures – remote object

a. Thread-per-request

b. Thread-per-connection

c. Thread-per-object

Implemented by the server-side ORB in CORBA
Invocations between address spaces

(a) System call

(b) RPC/RMI (within one computer)

(c) RPC/RMI (between computers)
remote procedure call in OS perspective

1. Copy args
2. Trap to Kernel
3. Upcall
4. Execute procedure and copy results
5. Return (trap)
Times for serialized and concurrent invocations

Serialised invocations

- Marshal
- Send
- Process args
- Marshal
- Send
- Transmission
- Receive
- Unmarshal
- Process results
- Marshal
- Send
- Receive
- Unmarshal
- Execute request
- Marshal
- Send
- Receive
- Unmarshal
- Process results
- Send
- Time

Concurrent invocations

- Marshal
- Send
- Process args
- Marshal
- Send
- Receive
- Unmarshal
- Execute request
- Marshal
- Send
- Receive
- Unmarshal
- Marshal
- Send
- Receive
- Unmarshal
- Process results
- Marshal
- Send
- Receive
- Unmarshal
- Process results
- Send
- Time
Arsitektur Model Prosesor

- Multiprocessors
  - Tightly coupled.
  - Shared memory.

- Multicomputers.
  - Loosely coupled.
  - Private memory.
  - Autonomous.
Univprocessor vs Multiprocessors

- Univprocessor has **only one** processor for OS
- Multiprocessor
  - Memory: could be shared or be private to each CPU
- Data is sent by bus-based multiprocessor.

Diagram:

- CPU
  - Cache
- CPU
  - Cache
- CPU
  - Cache
- Memory
- Bus
Uniprocessor Operating Systems

• An OS acts as a **resource manager**
  – Manages CPU, I/O devices, memory

• OS provides a **virtual interface** that is easier to use than **hardware**

• Structure model of uniprocessor operating systems, can be:
  – **Monolithic** (e.g., MS-DOS, early UNIX)
    • One large kernel that handles everything
  – **Micro Kernel**
    • Only essential kernel function, otherwise in user space
Uniprocessor Operating Systems

Microkernel architecture

- **Small kernel**: interprocess communication, low level I/O, memory, process management & scheduling
- **user-level** implement *additional functionality*

No direct data exchange between modules
Monolithic vs Microkernel

Monolithic Kernel based Operating System

- Application
- VFS, System call
- Scheduler, Virtual Memory
- Device Drivers, Dispatcher, ...
- Hardware

Microkernel based Operating System

- Application
- Application IPC
- UNIX Server
- Device Driver
- File Server

Basic IPC, Virtual Memory, Scheduling

Hardware
Jenis OS selain Uniprocessor OS

- **Multiprocessor OS**
  - Looks like a virtual uniprocessor, contains only one copy of the operating system, communication via shared memory, single run-queue

- **Network OS**
  - Does not look like a virtual uniprocessor, contains \( n \) copies of the operating system, communication via shared files, \( n \) run-queues

- **Distributed OS**
  - Looks like a virtual uniprocessor (more or less), contains \( n \) copies of the operating system, communication via messages, \( n \) run-queues
Network OS

• Setiap host menjalankan Sistem Operasi untuk mengatur sumber daya yang dimiliki nya termasuk mengakses sumber daya di jaringan.

• It provides an environment where users are aware of the multiplicity of machines.

• Users can access remote resources by
  – logging into the remote machine OR
  – transferring data from the remote machine to their own machine.

• Users should know where the required files and directories are and mount them.

• Each machine could act like a server and a client at the same time.

• Contoh OS: Windows 2000 dan Windows NT

• Contoh implementasi:
  – NFS (Network File System)
  – Samba – implementasi protokol SMB di Win & Linux
Network Operating System

Machine A
- Network OS services
- Kernel

Machine B
- Network OS services
- Kernel

Machine C
- Network OS services
- Kernel

Distributed applications

Network
The microkernel supports middleware via subsystems

**flexibility and extensibility**
- services can be added, modified and debugged
- small kernel -> fewer bugs
- protection of services and resources is still maintained
Distributed Operating System

- Dapat memanajemen komputer-komputer dan membuat “mereka” tampak sebagai single komputer
- Dapat menjalankan proses di komputer lain tanpa mengetahui siapa yang meresponsnya
- Manages resources in a distributed system
  - transparently to the user
- Looks to the user like a centralized OS
  - But operates on multiple independent CPUs
- Provides transparency
  - Location, migration, concurrency, replication,…
- Presents users with a virtual uniprocessor
DOS infrastructure

- Processor Pool
- Workstations
- Local Area Network
- Specialized Servers (File, Database, etc.)
- Gateway
- Wide Area Network
Keterangan DOS

• Workstation atau PC mengeksekusi proses yang memerlukan interaksi dari user seperti text editor atau window manager
  – Spesific task

• Processor pool: kumpulan prosesor, tiap unitnya biasanya terdiri dari prosesor, memori lokal, dan koneksi jaringan.
  – Tiap prosesor mengerjakan satu buah proses
Distributed OS

• Presents users (and applications) with an **integrated computing** platform that **hides** the individual computers.

• Has **control over all** of the nodes (computers) in the network and allocates their resources to tasks **without** user involvement.
  – In a distributed OS, the user **doesn't know** (or care) where his programs are running.

• Examples:
  – Amoeba ([http://amoebaos.org](http://amoebaos.org))
  – EyeOS ([www.eyeos.org](http://www.eyeos.org))
More examples

Your desktop from everywhere, any device
More Examples

G.HO.ST

ejolicloud

Glide os
## Daftar web OS

<table>
<thead>
<tr>
<th>Name</th>
<th>Browser support</th>
<th>Developer</th>
<th>Engine</th>
<th>Free</th>
<th>License</th>
<th>3rd party applications</th>
<th>Productivity Suite</th>
<th>Graphical user interface</th>
<th>Downloadable to Web server</th>
</tr>
</thead>
<tbody>
<tr>
<td>DesktopTwo</td>
<td>IE7</td>
<td>Sapotek</td>
<td>Flash</td>
<td>Yes  (Eta)</td>
<td>Open Source AGPL</td>
<td>Yes</td>
<td>OpenOffice</td>
<td>Mac+Windows-like</td>
<td>No</td>
</tr>
<tr>
<td>Glide OS</td>
<td>IE7, Firefox 3, Safari, Chrome</td>
<td>TransMedia</td>
<td>Flash</td>
<td>Yes  (30 GB limit)</td>
<td>Proprietary From Glide Community</td>
<td>Yes</td>
<td>Glide Writer</td>
<td>Mac+Windows-like</td>
<td>No</td>
</tr>
<tr>
<td>eyeOS</td>
<td>IE6/7/8, Firefox2/3, Safari, Opera, Chrome</td>
<td>eyeOS Team</td>
<td>PHP + AJAX</td>
<td>Yes</td>
<td>Open Source: AGPL3</td>
<td>Yes</td>
<td>Yes</td>
<td>Customizable</td>
<td>Yes</td>
</tr>
<tr>
<td>G. ho. st</td>
<td>IE6+, Firefox2+, Safari. Partial: Chrome &amp; Opera</td>
<td>Ghost Inc (&quot;G. ho. st&quot;)</td>
<td>Flash + AJAX (mobile version is WAP)</td>
<td>No</td>
<td>Proprietary</td>
<td>Yes</td>
<td>Yes</td>
<td>Windows-like</td>
<td>No</td>
</tr>
<tr>
<td>icloud</td>
<td>IE 6/7/8, Firefox 3.5+, Mobile Version: all browsers, Chrome</td>
<td>Xcerion AB</td>
<td>AJAX</td>
<td>Yes</td>
<td>Proprietary</td>
<td>Yes</td>
<td>Yes</td>
<td>Zoho, proprietary email, calendar, IM</td>
<td>Windows-like</td>
</tr>
<tr>
<td>Netvibes</td>
<td>IE7, IE6 is supported, but not recommended, FF 1+, Safari 2+ Opera 9.02+, Google Chrome</td>
<td>Netvibes Team</td>
<td>? + Ajax</td>
<td>Yes</td>
<td>Proprietary</td>
<td>Yes</td>
<td>Yes</td>
<td>Tab-based</td>
<td>No</td>
</tr>
<tr>
<td>Online OS</td>
<td>FF 1.5 and higher, IE7</td>
<td>iCUBE Network Solutions</td>
<td>Java + Ajax</td>
<td>Yes</td>
<td>Proprietary</td>
<td>Yes</td>
<td>Yes</td>
<td>Windows-like</td>
<td>No</td>
</tr>
<tr>
<td>Spiral Universe</td>
<td>Firefox, Safari, IE, Chrome</td>
<td>Spiral Universe</td>
<td>Java + Ajax (GWT/Ext)</td>
<td>Yes</td>
<td>Proprietary</td>
<td>No</td>
<td>Proprietary email, calendar, school software</td>
<td>Mac+Windows-like</td>
<td>No</td>
</tr>
</tbody>
</table>
DOS: Transparency

- **Location** Transparency
  - Users are not aware of the positioning of the resources in the system.

- **Migration** Transparency
  - Resources can move without changing names / URL

- **Replication** Transparency
  - Users should not be aware of the presence of multiple copies of a resource

- **Failure** Transparency
  - Masking the partial failures in the system
Transparency Cont’d…

- **Performance** Transparency
  - Reconfiguring the resources to improve the performance of the system

- **Concurrency** Transparency
  - Resource sharing is automatic

- **Parallelism** transparency
  - Activities can happen in parallel without the knowledge of the user. Users sees only speedup.

- **Scaling** Transparency
  - Allowing the system to expand in scale without disrupting the activities of the users
Distributed Operating Systems

Machine A

Distributed applications

Distributed operating system services

Kernel

Machine B

Kernel

Machine C

Kernel

Network
The differences

DOS

Machine A
Machine B
Machine C

Distributed applications
Distributed operating system services
Kernel
Kernel
Kernel

Network

NOS

Machine A
Machine B
Machine C

Distributed applications
Network OS services
Network OS services
Network OS services
Kernel
Kernel
Kernel

Network

- User is not aware of the multiple CPUs.
- Each machine runs a part of the Distributed Operating System.
- The system is fault-tolerant.

- User is aware of the existence of multiple CPUs.
- Each machine has its own private Operating System.
- The system is not fault-tolerant.
What is Amoeba?

• Amoeba is a distributed operating system
• Runs on a simple micro-kernel
• Developed by Andrew Tanenbaum
• Has user transparency
  – The user logs into the system not a specific machine
  – When a program is initiated, the system decides what machine will run it.
The History of Amoeba

• Developed by Andrew Tanenbaum at the Vrije Universiteit in conjunction with Centrum voor Wiskunde en Informatica
• First prototype was released in 1983
• The last official update was in 1996
• Others have developed their own versions
  – Fireball Amoeba by Fireball Software Distribution
Goals of Amoeba

• There are four main goals
  – Distribution
    • Connecting together many machines
  – Parallelism
    • Allowing individual jobs to use multiple CPUs easily
  – Transparency
    • Having the collection of computer act like a single system
  – Performance
Key Concepts

• **Micro-kernel**
  – A simple micro-kernel is the basis for Amoeba
  – All computers in the network run this kernel
  – It handles the memory management, I/O, communication, object primitive, and basic processes

• **Remote Procedure Calls (RPC)**
  – Used for communication between client and server
  – Accessed by stubs which are created by Amoeba Interface Language
Key Concepts

• **Threads**
  - Each process has its own address space and contains multiple threads
  - These threads have their own stack and program counter, but share the global data and code of the process

• **FLIP (the protocol)**
  - Fast Local Internet Protocol
  - Developed by Andrew Tanenbaum
  - Designed to optimize the speed of RPCs
Key Concepts

• **Objects**
  – The abstract data type used by Amoeba
  – Each object has a list of operations that can be preformed

• **Capability (Protection)**
  – Store data in 128 bit value
  – Used to verify that the user has permission to access the object
  – Capabilities are encrypted
Key Concepts

• **Bullet / File Server**
  - Store files in a contiguously fashion
  - Most files can be sent in a single RPC
  - Designed to be a dedicated server

• **Directory Server**
  - Handles naming of files
  - Knows the physical location of each file
Significant Points

• The system is **free**
• It has not had an official update in over **10 years**
• Can use **older/slower CPUs** to create a powerful system
• **Micro-Kernel** allows for other file systems to be created
• Has many **UNIX like** commands and programs
• Can only hold programs as large as its physical memory
## Comparison of DOS

<table>
<thead>
<tr>
<th>Developed By</th>
<th>Cambridge</th>
<th>Amoeba</th>
<th>V Kernel</th>
<th>Eden Project</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Computing Laboratory@</td>
<td>Tanenbaum@ Vrije Universiteit-</td>
<td>David Cheriton@ Stanford</td>
<td>University of Washington-</td>
</tr>
<tr>
<td></td>
<td>Univ. of Cambridge</td>
<td>Amsterdam</td>
<td>University</td>
<td>Seattle</td>
</tr>
<tr>
<td>Communication Primitives</td>
<td>RPC</td>
<td>RPC</td>
<td>RPC</td>
<td>RPC</td>
</tr>
<tr>
<td>Naming and Protection</td>
<td>Single Name Server</td>
<td>Sparse capabilities with</td>
<td>Three-level naming mechanism</td>
<td>Capabilities without</td>
</tr>
<tr>
<td></td>
<td></td>
<td>encryption</td>
<td></td>
<td>protection</td>
</tr>
<tr>
<td>Resources</td>
<td>Processor Bank</td>
<td>Processor Pool</td>
<td>Workstation Model</td>
<td>Workstation Model</td>
</tr>
<tr>
<td>Fault tolerance</td>
<td>Small server to startup services</td>
<td>Some fault tolerance through boot server</td>
<td>No fault tolerance</td>
<td>Uses Recorder process.</td>
</tr>
<tr>
<td>File Server</td>
<td>Universal file service and Filing Machine</td>
<td>Several file services.</td>
<td>Similar to Unix</td>
<td>No file server. One process for each file</td>
</tr>
</tbody>
</table>
amoebaOS
Paper Diskusi (20%)

• Topik:
  – Cloud Computing
  – Bit Torrent System
  – Database Terdistribusi
  – VOIP dan Streamming
  – Grid computing
  – Jabber Protocol
  – JINI – Service Oriented Architecture Framework in Java
  – DCOM (distributed component)
  – Master - Slave MySQL database (Replication)
  – Konfigurasi File Sharing di Linux

• Dibuat dalam doc dan ppt dipresentasikan pada 19/10 2010

• Dikerjakan kelompok
• Referensi dari jurnal minimal 1 buah
TTS dan Presentasi dan Diskusi 1
Bahan TTS:
- Awal smp slide ini
- Bentuk soal: pil ganda dan essay
- Sifat: open 1 lembar kertas A4 bolak balik boleh diketik