Script generated by TTT

Title: Distributed_Applications (17.04.2012)

Date: Tue Apr 17 14:32:27 CEST 2012

Duration: 85:48 min

Pages: 23

Background

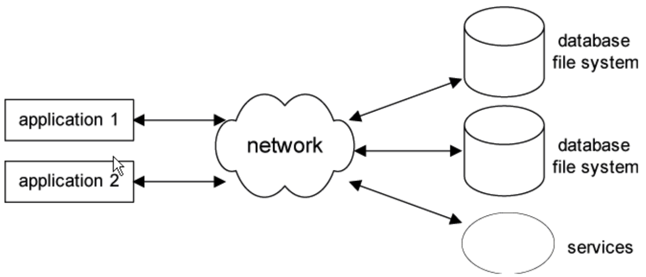
Variety of domains for distributed applications

collaborative information spaces, workflow management, telecooperation, autonomous agents

Development of computer technology

Internet computing

Enterprise Computing



Enterprise computing systems

- close, direct coupling of application programs running on multiple, heterogeneous platforms in a networked environment.

These systems must be completely integrated and very reliable, in particular

- information consistency, even in case of partial system breakdown.
- security and guaranteed privacy.
- adequate system response times.
- high tolerance in case of input and hardware/user errors (fault tolerance).
- autonomy of the individual system components.

Background

Variety of domains for distributed applications

collaborative information spaces, workflow management, telecooperation, autonomous agents

Development of computer technology

Internet computing

Enterprise Computing

Generated by Targeteam

The following factors contribute to the increasing importance of distributed systems:

- Decrease of processor and storage cost.
- High bandwidth networks
- Insufficient and often unpredictable response times of mainframe systems
- Growing number of applications with complex information management and complex graphical user interfaces.
- Growing cooperation and usage of shared resources by geographically dispersed users; caused by the globalization of markets and enterprises,
  - e.g. applying telecooperation (groupware, CSCW) and mobile communication to improve distributed teamwork.

Informal definition

Methods of distribution

In the following sections, we will focus on the latter three types of distribution, in particular on the processing distribution.

The term ***distributed system*** may be defined informally:

1. after ***Tanenbaum*** : a distributed system is a collection of independent computers which appears to the user as a single computer.
2. after ***Lamport*** : a distributed system is a system that stops you from getting any work done when a machine you've never heard of crashes.
3.

   Definition: We define a **distributed system** as one in which hardware and software components located at networked computers

   - communicate and
   - coordinate their actions mainly by passing messages.

There are five fundamental methods of distribution:

1. Hardware components.
2. Load.
3. Data.
4. Control, e.g. a distributed operating system.
5. Processing, e.g. distributed execution of an application.

Distributed systems have a number of characteristics, among them are:

1. Existence of multiple functional units (physical, logical), e.g. software services.
2. Distribution of physical and logical functional units.
3. Functional units break down independently.
4. Distributed component control: a distributed operating system controls, integrates and homogenizes the distributed functional units
5. Transparency : details irrelevant for the user (e.g. distribution of data across several computers) remains hidden in order to reduce complexity.
6. Cooperative autonomy during the interaction among the physical and logical functional units $\Rightarrow$ implies concurrency during process execution.

Challenges of distributed systems

The design of distributed systems poses a number of challenges

Heterogeneity applies to networks, computer hardware, operating systems, programming languages and implementations by different programmers.

- Use of ***middleware*** to provide a programming abstraction masking the heterogeneity of the underlying system.
- middleware provides a uniform computational model for use by the programmers of servers and distributed applications.
- middleware examples are Corba , Java RMI .

Openness requires standardized interfaces between the various resources.

Scalability : adding new resources to the overall system.

Security : for information resources.

Privacy : protect user profile information.

Examples for development frameworks

There is a high motivation to use standardized development frameworks for the design and implementation of distributed applications.

Sun Network File System (NFS) by SUN

- a distributed file system behaving like a centralized file system.

Open Network Computing (ONC) by SUN

- platform for distributed application design; it contains libraries for remote procedure call (RPC) and for external data representation (XDR) .

distributed applications in ODP (Open Distributed Processing) by ISO

- specification of the interfaces and the component behavior.

Common Object Request Broker Architecture (CORBA) by OMG

- defines a common architecture model for heterogeneous environments based on the object-oriented paradigm.

Java 2 Platform Enterprise Edition (J2EE) by Sun, e.g. RMI

- component-based Java framework providing a simple, standardized platform for distributed applications; runtime infrastructure and a set of Java API's.

.NET framework of Microsoft

- middleware platform especially for Microsoft environments
- consists of a class library and a runtime environment
- incorporates the distributed component object model (DCOM)

a ***set of cooperating, interacting functional units***

- reasons for distribution: ***parallelism*** during the execution, ***fault tolerance*** , and ***inherent distribution*** of the application domain.

Definition

Programmer's perspective

Interfaces help to establish well-defined interaction points between the components of a distributed application.

- **Interfaces of a distributed application**
- **Interface specification**

Distributed application vs. parallel program

Definition: The term **distributed application** contains three aspects.

- The application A, whose the functionality is split into a set of cooperating, interacting components A_1 , .., A_n , $n \in IN$, $n > 1$; each component has an internal state (data) and operations to manipulate the state.
- The components A_i are autonomous entities which can be assigned to different machines F_i .
- The components A_i exchange information via the network.

Distributed application

a ***set of cooperating, interacting functional units***

reasons for distribution: ***parallelism*** during the execution, ***fault tolerance*** , and ***inherent distribution*** of the application domain.

Definition

Programmer's perspective

Interfaces help to establish well-defined interaction points between the components of a distributed application.

Interfaces of a distributed application

Interface specification

Distributed application vs. parallel program

Distributed application

a ***set of cooperating, interacting functional units***

reasons for distribution: ***parallelism*** during the execution, ***fault tolerance*** , and ***inherent distribution*** of the application domain.

Definition

Programmer's perspective

Interfaces help to establish well-defined interaction points between the components of a distributed application.

Interfaces of a distributed application

Interface specification

Distributed application vs. parallel program

Generated by Targeteam

specifies component operations (names, functionality) and communication between components.

required parameters (including their types).

the results returned by the operation including arity and type.

visible side-effects caused by the component execution, for example data entry into a database.

effects of an operation on the results of subsequent operations.

constraints concerning the sequence of operations.

Distributed application vs. parallel program

Although distributed applications might look similar to parallel programs at first glance, there are still some differences.

	distributed application	**parallel program**
granularity	coarse	fine
data space	private	shared
failure handling	within the communication protocols	not considered

Introduction

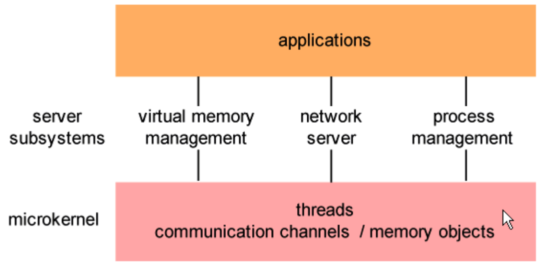
Issues

Issues of the following section

- Motivation for distributed systems and distributed applications.
- Basic terminology for distributed systems, e.g. terms like ***distributed applications*** , and ***interface*** .
- Introduction to some influential historic distributed systems, such as NFS File system, Mach and Java 2 Platform Enterprise Edition.

The process (a task) defines an execution environment that provides secured access to system resources such as virtual memory and communication channels.

A process consists of a set of threads.

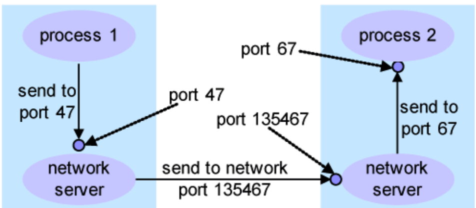


- threads as distribution unit, i.e. only entire threads are assigned to different processors.
- memory objects realize virtual storage units; shared utilization of memory objects by different processes is based on "Copy-on-Write", i.e. the memory object is copied when write operation takes place.

Mach message exchange

Processes communicate through communication channels, called ***ports*** .

- A port is realized as a message queue to which multiple senders may send messages; there is only one receiving process per queue.
- Ports are protected by capabilities.
- Mach supports network communication through network servers.



Mach

Mach is an operating system developed at Carnegie-Mellon University. It is characterized through its small kernel, called microkernel.

Goals of Mach

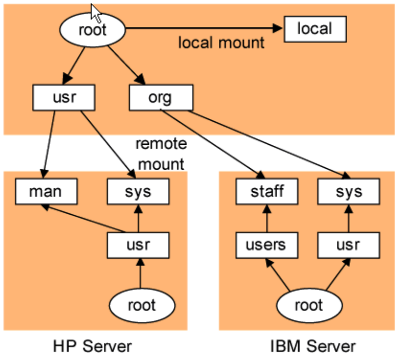
Major design goals were

- Emulation of Unix.
- transparent extension to network operation.
- portability.

Characteristics

File catalogs are exported (by server subsystems) and mounted (by the client machines).

Sun Client



Support of a ***mount service*** :

file `/etc/exports` on NFS server lists names of local filesystem available for remote mounting.

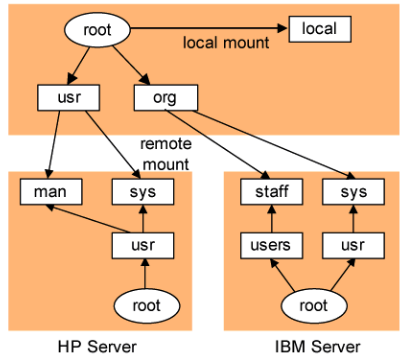
mounting request by client with: remote host, directory pathname and local name with which it is to be mounted.

automounter : dynamically mounting of a remote directory whenever an 'empty' mount point is referenced by a client.

NFS supports access transparency

Characteristics

Sun Client



Support of a ***mount service*** :

file `/etc/exports` on NFS server lists names of local filesystem available for remote mounting.

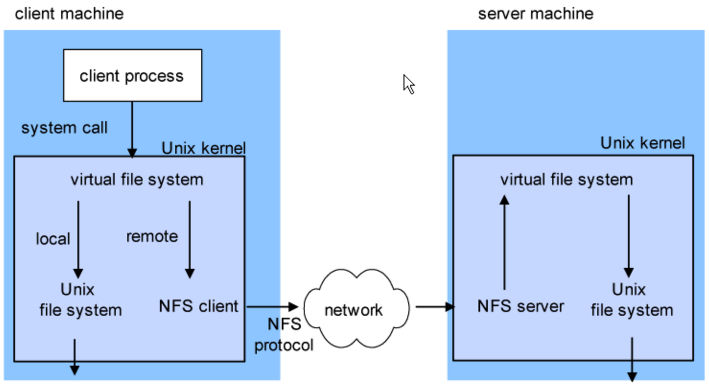
mounting request by client with: remote host, directory pathname and local name with which it is to be mounted.

automounter : dynamically mounting of a remote directory whenever an 'empty' mount point is referenced by a client.

NFS supports access transparency .

NFS implementation

NFS implementation is based on RPC calls between the involved operating systems. It can be configured to use UDP or TCP.



earlier version of NFS was a ***stateless*** file server, i.e. a server subsystem does not store state information about its clients and their past operations.

current version of NFS is a ***stateful*** file server, i.e. a server subsystem supports locking and delegation of actions to client to improve client-side caching.

network extension to Unix and other operating systems for distributed file management.