


## Script generated by TTT

Title: Distributed\_Applications (10.07.2012)


Date: Tue Jul 10 14:30:41 CEST 2012

Duration: 87:25 min

Pages: 27



### Introduction



When a group of programmers has the task to build a distributed application, in addition to distributed code management there is also the need for distributed file services.


[Definitions](#)

[Motivation for replicated files](#)


[Two consistency types](#)

[Replica placement](#)

Generated by Targeteam



### Introduction



A distributed file system supporting replicated files has the following characteristics:

Less network traffic and better response times.

Higher availability and fault tolerance with respect to communication and server errors.

Parallel processing of several client requests.

The key concept of a distributed file system is transparency.

User's impression: interaction with a normal, central file system.

Goal to support the following [transparency](#) types: location, access, name, replication and concurrency transparency.

Generated by Targeteam

When a group of programmers has the task to build a distributed application, in addition to distributed code management there is also the need for distributed file services.

[Definitions](#)

[Motivation for replicated files](#)

[Two consistency types](#)

[Replica placement](#)

Generated by Targeteam



## Replica placement



A major issue of distributed data store is the decision when and where to place the file replicas.

### Permanent replicas

The number and placement of replicas is decided in advance, e.g. mirroring of files at different sites.

### Server-initiated replicas

They are intended to enhance the performance of the server.

Dynamic replication to reduce the load on a server.

file replicas migrate to a server placed in the proximity of clients that issue file requests.

### Client-initiated replicas

Client-initiated replicas are more commonly known as caches.

Used only to improve access times to data.

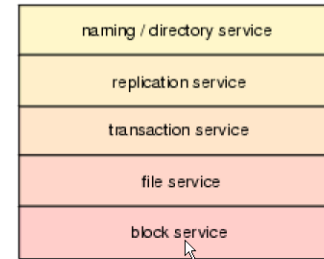
Client caches are normally placed on the same machine as its client.

Replicas are only kept for a limited time.

Generated by Targeteam



The functions of a distributed file service are usually arranged in a hierarchical way.



### [Layer semantics](#)

Generated by Targeteam



## Distributed file service



Each layer of the distributed file service has a specific task.

### Name/directory service

placement of files; file relocation for load balancing and performance improvement; localization of the server which manages the referenced file.

mapping of textual file names to file references (server name and file identifier).

### Replication service

file replication for shorter response times and increased availability.

handles data consistency and the multiple copy update problem.

### Transaction service

provides a mechanism for grouping of elementary operations so as to execute them atomically;

mechanisms for concurrency control;

Mechanisms for reboot after errors;

### File service

relates file identifiers to particular files;

performs read and write operations on the file content and file attributes.

### Block service

accesses and allocates disk blocks for the file.

Generated by Targeteam



### Issues

This section introduces schemes for replication and concurrency control in the context of distributed file services.

What are the general characteristics of a distributed file service?

How to maintain consistency of replicated files?

What are voting schemes?

Presentation of the Coda file service.

### [Introduction](#)

### [Layers of a distributed file service](#)

### [Update of replicated files](#)

### [Coda file system](#)

Generated by Targeteam



Pessimistic concurrency control for data-critical applications, e.g. banking applications. Always access to consistent data.

### Classification of pessimistic concurrency control

#### Primary site

A well-defined file copy, the primary site, serializes and synchronizes all (write) operations.

#### Token passing

Access to the replicated file (i.e. a file copy) is only permitted, if the client has the token.

#### Voting schemes

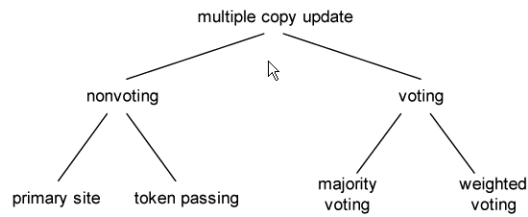
The result of the negotiation between all file replicas determines whether a file access is granted or not.

global consent is necessary, but control is decentralized.

in case of consent, the relevant file block is locked.

Examples: Majority consensus, weighted voting.

Generated by Targeteam



Generated by Targeteam



Voting schemes provide pessimistic concurrency control.

#### Introduction

Voting schemes are algorithms for maintaining mutual consistency of replicates even in situations of computer crashes and network partitionings.

Let us assume, there exist REP replicas of file d.

Let  $sg(r)$  be the weight of the vote of computer r; K be the set of all computers considered.

Let the sum of all weights be  $SUM = \sum_{r \in K} sg(r)$ .

#### Definitions

##### Multiple-reader-single-writer strategy

##### Voting scheme variants

Generated by Targeteam



**Definition:** The **votum** for a desired access of a file is defined as the sum of votes from the set of computers that have voted for the desired access.

**Definition:** The obtained votum is called successful if the sum of votes from the set of computers that have voted for the desired access is equal to or greater than a lower bound, the so-called **quorum**.

File access is permitted (positive votum), if the following holds

for read access: at least R positive votes (read quorum).

for write access: at least W positive votes (write quorum).



Generated by Targeteam



## Voting schemes



Voting schemes provide pessimistic concurrency control.

### Introduction

Voting schemes are algorithms for maintaining mutual consistency of replicates even in situations of computer crashes and network partitionings.

Let us assume, there exist  $REP$  replicas of file  $d$ .

Let  $sg(r)$  be the weight of the vote of computer  $r$ ;  $K$  be the set of all computers considered.

Let the sum of all weights be  $SUM = \sum_{r \in K} sg(r)$ .

### Definitions

[Multiple-reader-single-writer strategy](#)

[Voting scheme variants](#)

Generated by Targeteam



## Voting scheme variants



For further variants and details see the book Borghoff/Schlichter, Springer-Verlag, 2000.

[Write-All-Read-Any](#)

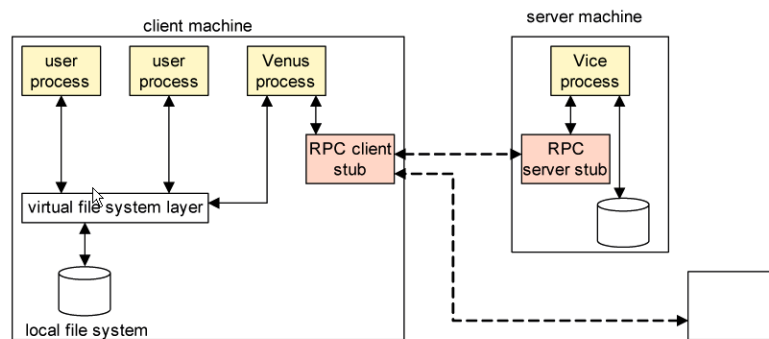
[Majority consensus](#)

[Weighted voting](#)

Generated by Targeteam



## Architecture



Venus processes provide access to files maintained by the Vice file servers.

role is similar to that of an NFS client.

responsible for allowing the client to continue operation even if access to the file servers is (temporarily) impossible.

Generated by Targeteam



## Coda file system



Coda was designed to be a scalable, secure, and highly available distributed file service.

supporting the mobile use of computers.

files are organized in volumes.

Coda relies on the replication of volumes.

[Architecture](#)

[Naming](#)

[Replication strategy](#)

[Disconnected operation](#)

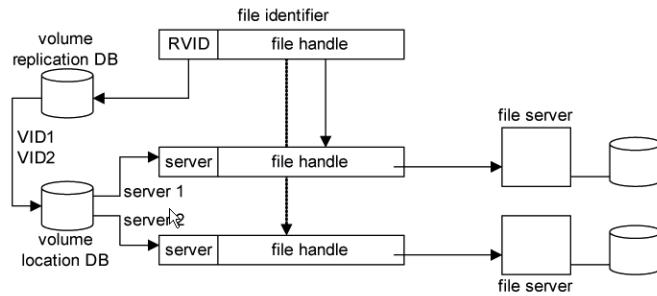
Generated by Targeteam



## Naming



Coda assigns each file a 96-bit file identifier.



Generated by Targeteam



Each file is contained in exactly one volume. Distinction between physical volumes.

logical volume (represents all replicas of a volume).

RVID (Replicated Volume Identifier): identifier of a logical volume.

VID (Volume Identifier): identifier of a physical volume.

### File identifier

Generated by Targeteam



## Replication strategy

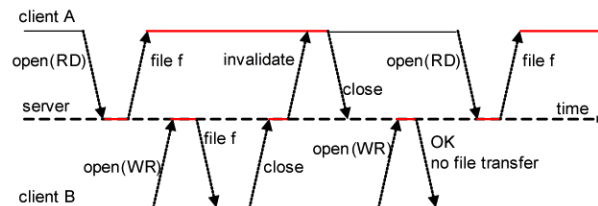


When a file is opened, an entire copy of the file is transferred to the client; caching of the file.

client becomes less dependent on the availability of the server.

Cache coherence is maintained by means of callbacks.

- Server records a **callback promise** for a client.
- update of the file by a client  $\Rightarrow$  notification to the server  $\Rightarrow$  invalidation message to other clients.



Generated by Targeteam



Coda relies on replication to achieve high availability. It distinguishes between two types of replication.

### Client caching

### Server replication

Generated by Targeteam



Coda uses an optimistic strategy for file replication. For each file version there exists a Coda version vector (CVV).

CVV is a [vector timestamp](#) with one element for each server in the relevant VSG.

CVV is initialized to [1, ..., 1].

On file close the Venus process of the client broadcasts an update message to all servers in AVSG  $\Rightarrow$  all servers of AVSG update the relevant CVV entries.

Let  $v_1$  and  $v_2$  are CVVs for two versions of a file  $f$ .

When neither  $v_1 \leq v_2$  nor  $v_2 \leq v_1 \Rightarrow$  there is a conflict between the two file versions.

Generated by Targeteam



Coda allows file server to be replicated; the unit of replication is a volume.

Volume Storage Group (VSG): collection of servers that have a copy of a volume.

client's Accessible Volume Storage Group (AVSG): list of those servers in the volume's VSG that the client can contact.

AVSG = {}: client is disconnected.

Coda uses a variant of the "read-one, write-all" update protocol.

[Coda version vector](#)

Generated by Targeteam



- Prof. J. Schlichter
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
  - Boltzmannstr. 3, 85748 Garching
  - Email: [schlichter@in.tum.de](mailto:schlichter@in.tum.de)
  - Tel.: 089-289 18654
  - URL: <http://www11.in.tum.de/>

[Overview](#)

[Introduction](#)

[Architecture of distributed systems](#)

[Remote Invocation \(RPC/RMI\)](#)

[Basic mechanisms for distributed applications](#)

[Web Services](#)

[Design of distributed applications](#)

[Distributed file service](#)

[Distributed Shared Memory](#)

[Object-based Distributed Systems](#)

[Summary](#)

Generated by Targeteam



Message passing model

variables have to be marshalled from one process, transmitted and unmarshalled into other variables at the receiving process.

**Distributed shared memory**

the involved processes access the shared variables directly; no marshalling necessary.  
processes may communicate via DSM even if they have non-overlapping lifetimes.

**Implementation approaches**

in hardware

shared memory multiprocessor architectures, e.g. NUMA architecture.

in middleware

language support such as Linda tuple space or JavaSpaces.

Generated by Targeteam



The content of DSM may be replicated by caching it at the separate computers;  
data is read from the local replica.

updates have to be propagated to the other replicas of the shared memory.

Approaches to keep the replicas consistent

#### Write-update

updates are made locally and multicast to all replicas possessing a copy of the data item.

the remote data items are modified immediately.

#### Write-invalidate

before an update takes place, a multicast message is sent to all copies to invalidate them;

acknowledgement by the remote sites before the write can take place.

other processes are prevented to access the blocked data item.

the update is propagated to all copies, and the blocking is removed.

Generated by Targeteam



#### Issues of the section

implicit communication via shared memory

what is the Linda tuple space?

Javaspace as modern tuple space

#### Introduction

#### Programming model

#### Consistency model

#### Tuple space

#### Object Space

Generated by Targeteam



The **OMG** (Object Management Group) was founded in 1989 by a number of companies to encourage the adoption of *distributed object systems* and to enable *interoperability* for heterogeneous environments (hardware, networks, operating systems and programming languages).

[Object Management Architecture - OMA](#)

[Object Request Brokers ORB](#)

[Common object services](#)

[Inter-ORB protocol](#)

[Distributed COM](#)

[Integration of Corba/DCOM and Web Services](#)

[.NET Framework](#)

Generated by Targeteam

