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.
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.

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

Layer semantics

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 context and file attributes.

Block service
- accesses and allocates disk blocks for the file.

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 Codis file service.

Introduction
Layers of a distributed file service
Update of replicated files
Codis file system
**Pessimistic concurrency control**

Pessimistic concurrency control for data-critical applications, e.g., banking applications. Always acquires 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.

---

**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).

---

**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 partitions.

- 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 $\text{SUM} = \sum_{r \in K} sg(r)$.

**Definitions**

- **Multiple-reader-single-writer strategy**
- **Voting scheme variants**
Voting schemes

**Introduction**

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

Let us assume there exist REP replicas of file d.

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

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

**Definitions**

- Multiple-reader-single-writer strategy
- Voting scheme variants

Architecture

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

Venus 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.

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
Coda assigns each file a 96-bit file identifier.

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

When a file is opened, an entire copy of the file is transferred to the client; caching of the file 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 ⇒ notification to the server ⇒ invalidation message to other clients.

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

Client caching

Server replication
Codas uses an optimistic strategy for file replication. For each file version there exists a Codas 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 in all servers of AVSG update the relevant CVV entries.

Let v1 and v2 are CVVs for two versions of a file f:

\[ \text{when neither } v1 \leq v2 \text{ nor } v2 \leq v1 \Rightarrow \text{there is a conflict between the two file versions.} \]

Codas 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.

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

AVSG = {}: client is disconnected.

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

Codas version vector

---

**Distributed Applications - Verteilte Anwendungen**

- Prof. J. Schlichter
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
  - Boltzmannstr. 3, 85748 Garching
  - Email: schlchter@in.tum.de
  - Tel.: 089-289 18654
  - URL: http://www1.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
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-invalidates**
- 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.

**Issues of the section**
- implicit communication via shared memory
- what is the Linda tuple space?
- Javaspaces as modern tuple space

**Introduction**
- Programming model
- Consistency model
- Tuple space
- Object Space

The Object Management Group (OMG) 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/COM and Web Services
- .NET Framework