Silenus - Meta Information

Sorcer's Integrated Local Enhanced New Users Store - Meta Information DAta Store

Max Berger <max@berger.name>

May 4th, 2005

Overview

- Introduction
- Byzantium
- Midas
- User Interface
- Daphne
- Optimizers
- Installer
- Conclusion

Introduction

- Problem Statement
- Possible users
- Objective / Approach
- Services

Problem Statement

Existing distributed file stores

- are difficult to set up
- require high maintanance
- have problems with server downtime
- don't handle topology changes
- don't provide privacy from administators
- don't scale for large nuber of nodes
- are incompatible with service-oriented architectures

Possible users

- Power User
- High performance computing lab
- Multi-student LAB
- Small Office / Workgroup
- Students rooming
- Family

Objective / Approach

Objective

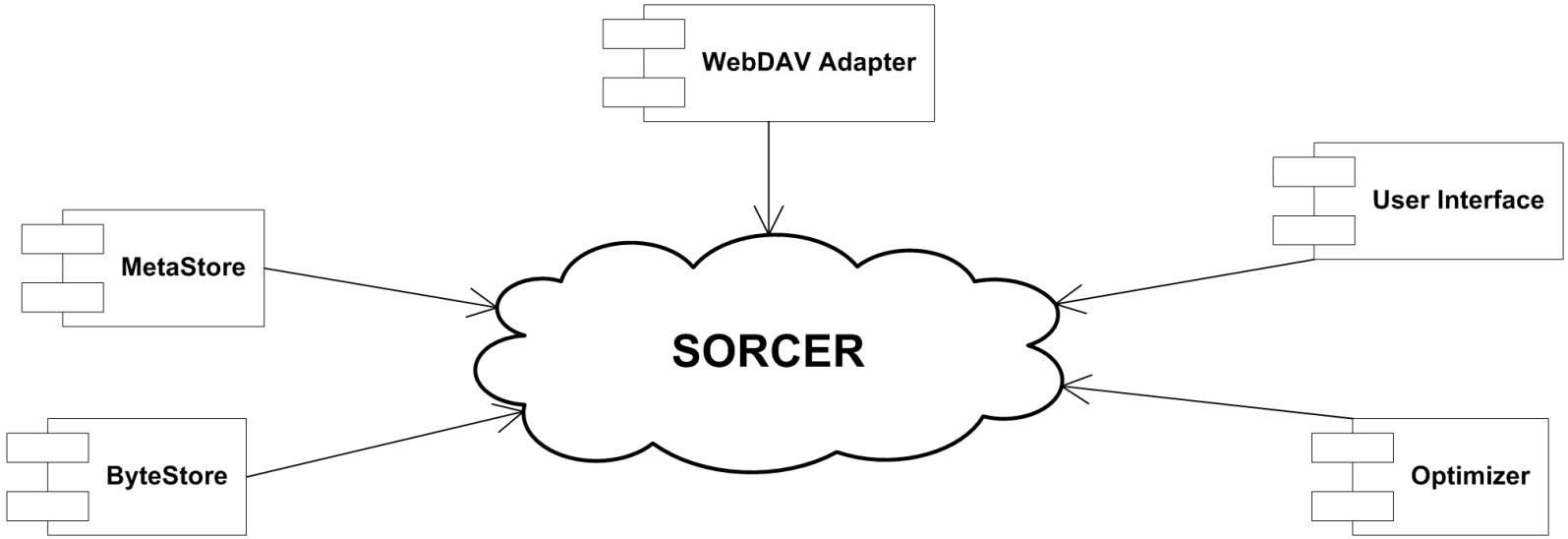
- to create a new filestore based on the SORCER environment

Approach

- Authentication for authorization
- Encryption for privacy
- Replication for availability
- Multisource download for performance

Services

Independent federated services that use SORCER for communication



Byzantium

- Introduction
- Implemented items
- Open Issues

Introduction

- Byzantium is a ByteStore implementation
- Based on Holowaa
- Files are identified uniquily by ServiceID and Entry UUID
- Does nothing but file storage

Implemented items

- Permanent FileStore
- Fast Transfer via direct TCP connection

Open Issues

- Temporary Items
- Leases
- Transfer through JINI methods only (firewalls)
- ByteStore to ByteStore transfer
- Missing support for disk space management

Midas

- Introduction
- Solved issues
- Interfaces
- Directory Structure
- Programmatical Interface
- DocumentDescriptor
- DirectoryDescriptor
- FileDescriptor
- External Interface

- Internal to Descriptors
- Internal to MetaStores
- Vector Clocks
- Internal Storage
- Security
- Versioning
- Conflict resolution
- Special meta information
- Open issues

Introduction

- Midas - Meta Information Database And Storage
- Midas is a MetaStore Implementation
- Based on Vivek's ReplicaProvider
- Stores meta information for files saved in bytestore
- Meta Information is stored in embedded database

Solved issues

- Meta Information is consistent across MetaStores
- Support for all kinds of meta information: File name, size, modified date, icon, security information, etc.
- Two parts: Service and Descriptors
- Each MetaStore should have an associated ByteStore
- Provide support for file movement / hoarding

Interfaces

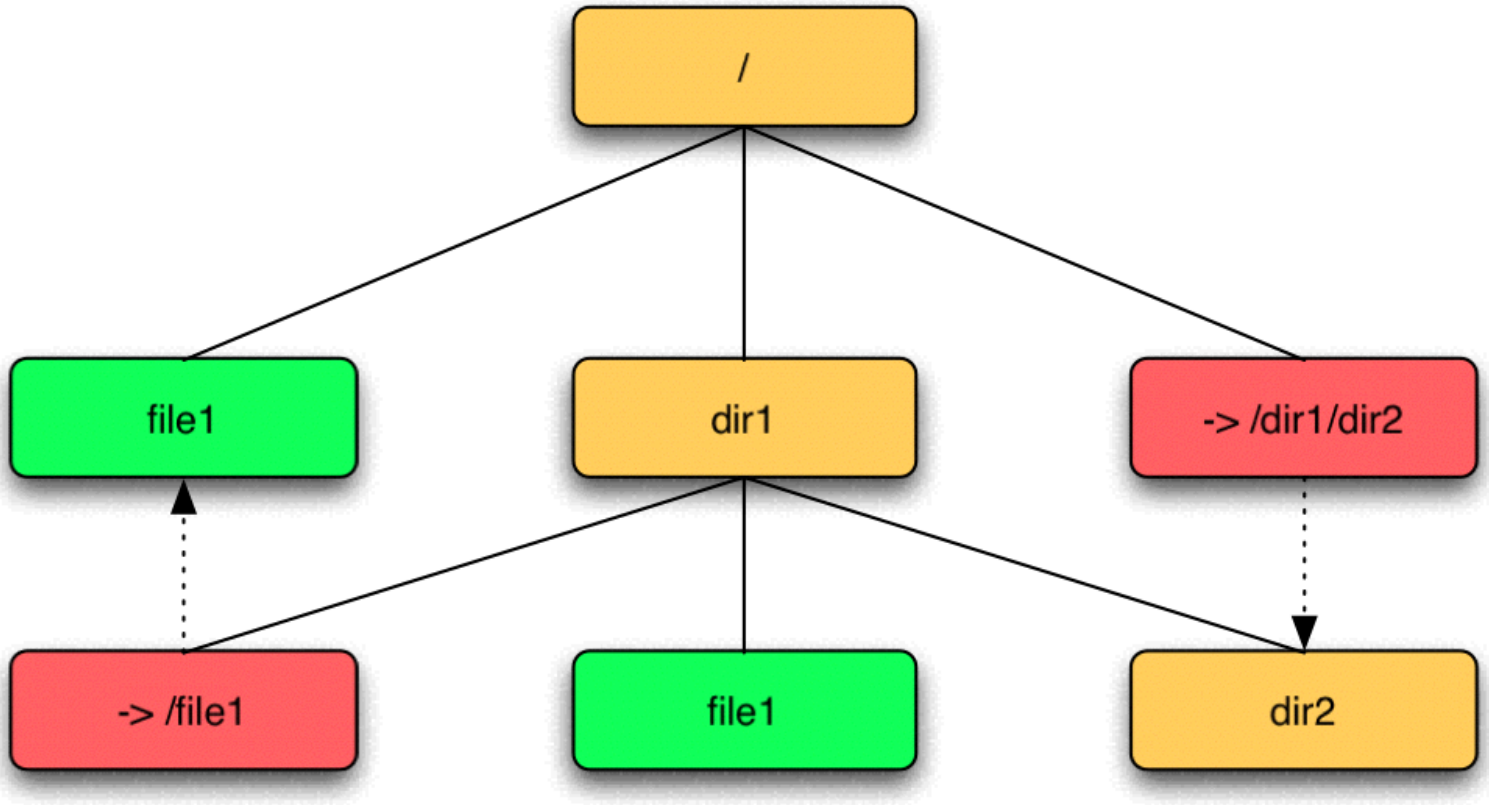
External Interface:

- Sorcer FileStore compatibility interface
- New programmatical interface
- Sorcer Service or Jini Service

Internal Interfaces:

- MetaStore-to-MetaStore
- MetaStore-to-Document Descriptors

Directory Structure



Programmatical Interface

- Tree Model similar to DOM
- Every node is a DocumentDescriptor or one of its subclasses:
- FileDescriptor
- DirectoryDescriptor
- LinkDescriptor

DocumentDescriptor

- `public String getAttribute(String key)`
- `public boolean isLeaf()`
- `public String getUID()`
- `public void putAttribute(String key, String value, boolean commit)`
- `ATTR_NAME, ATTR_TYPE, ATTR_SIZE, ATTR_CREATIONDATE`
- `MIMETYPE_DIR, MIMETYPE_LINK`

DirectoryDescriptor

- `public List getChildren()`
- `public DirectoryDescriptor createDirectory(String name) throws IOException`
- `public FileDescriptor createFile(String name) throws IOException`

FileDescriptor

- `public OutputStream getOutputStream()`
- `public OutputStream getOutputStream(boolean update)`
- `public InputStream getInputStream()`

External Interface

```
public DirectoryDescriptor
expandRootNode() throws RemoteException
```

Internal to Descriptors

- `public Map expandNodes(Collection nodes)`
- `DocumentDescriptor addChild(String parentID, String name, String type)`
- `void updateAttributes(String nodeID, Map newAttributes)`
- `void addNewVersion(String nodeID, ServiceID bytestoreID, Uuid bytestoreEntryID)`
- `public Map getLocations(String nodeID)`

Internal to MetaStores

There are currently two planned ways of sending update messages:

- Via JavaSpace (centralized)
- Via Jini Distributed Events (decentralized)

Order of messages, lost messages, etc. are neglegable, since we use Vector Clocks

Vector Clocks

- Each event increases the logical clock of that particular system by one
- Each system keeps a vector of all logical clocks it knows
- Update events include the vector clock
- Allows total odering of events, needed for synchronization
- Two clocks per node: One for internal events, another one for external events.

Internal Storage

Stored in internal McKoi database

- Parent-Child table
- Meta Info table
- Locations table

Security

- All files are stored encypted
- Decryption keys are file objects in special directory
- Decryption is done at the last node possible (in the proxy)
- Users secret key can be in FileStore or on local computer

Versioning

- Concurrency Problem is solved by versioning
- Version includes MetaStore Identifier
- Old Version can be stored for backup

Conflict resolution

Virtual duplication:

If an item A is changed on two MetaStores, it will be available 3 times:

- as A.1
- as A.2
- as A linking to either A.1 or A.2 (depending on which system)

Special meta information

- Minimum Number of copies in network
- Number of old versions to keep
- Special meta information template items

Open issues

- Not fully implemented yet

User Interface

- Introduction
- Solved issues
- Open issues

Introduction

- Provides a Jini ServiceUI compatible interface for all functions provided by the MetaStore
- Zero install: started from service browser (e.g. IncaX)

Solved issues

- design is pretty straightforward
- User UI should not provide any functionality that is not in the service!

Open issues

- No cool name yet

Daphne

- Introduction
- Solved issues
- Open issues

Introduction

- Daphne is the WebDAV Adapter for MetaStore and ByteStore

Solved issues

- WebDAV is well documented and supported in Win32, Mac OS X, Linux

Open issues

- Each OS has different quirks that need to be addressed

Optimizers

- Introduction
- Solved issues
- Open issues

Introduction

- The optimizer services will make this FileStore intelligent by doing automatic data movement, replication, indexing, ...

Solved issues

Optimizer is not a single service, but is a set of federated services:

- MetaInformationCompleter: Will read files and add missing meta information: Mime-Type, artist/title for mp3s, codec and play length for movies, ...
- Replicator: Reads the MetaInformation "Minimum copies" and ensures that there are that many copies in the network
- SpaceSaver: Saves space by reducing number of replicas in the network. Uses "Last Used" meta information.

Optimizers - Solved issues

Open issues

- No cool names yet
- Preventive Mover: Should detect that a certain file/directory is used from 9-5 at work and from 6-9 at home. Should then automagically move file from work to home between 5-6
- Other intelligence that I can't think of

Installer

- Introduction
- Solved Issues
- Unresolved Issues

Introduction

Any person should be able to install the system without much knowledge about computers!

Solved Issues

- Should be installable with as few clicks and settings as possible
- Multiple Options: Beginner, Medium, Expert

Unresolved Issues

- Technologies: WebStart, InstallAnywhere, ...
- Must run as service in Win32, Unix, Linux
- Shoudl auto-connect to WebFolder on Win32
- Should auto-connect on OS X

Conclusion

- Benefits
- Schedule

Benefits

- compehensive security
- completely self-managed
- fully transparent for the user
- adding disk space is easy
- compatible with legacy applications via WebDAV interface

Schedule

Background research	01/04 - 12/04
Initial proposal	06/04
System design	06/04 - 12/04
Byzantium	10/04 - 02/04
Midas	01/05 - 06/05
User Interface	03/05 - 11/06
Daphne	07/05 - 12/05
Optimizer	01/06 - 11/06
Installer	01/06 - 11/06
Defense	11/06