Implementation of a Virtual File System

Group 03

Thomas Frick (LEGI)
Matthias Ganz (04-862-850)
Philipp Rohr (04-397-030)

March 26, 2013

Contents

Abstract

The *Virtual File System* was implemented during the course *Java and C# in depth*.

This version of the document describes the project at the final state of milestone 1. and so on. . .

1. The Full Model

After we did one more experiment that is explained in the appendix I could could examine the measured values and made some observations that are described in this section. Based on those observations I built the full model that will be the basis of a mean value analysis shown further in the document.

1.1. Definitions

blablab

1.2. Observations

This is a table (table 1) measured throughout the and a footnote [1]

N	R_{puts}	$R_{retrieves}$	R_{meas}	X_{meas}	X_{calc}	R_{calc}	Z_{calc}
32	69	112	181	176.5	176.8	181	0
64	72	113	185	344.7	345.9	186	1
96	81	115	196	486.4	489.8	197	1
128	98	119	217	586.8	589.9	218	1
160	124	130	254	628.3	629.9	255	1
192	161	156	317	603.9	605.7	318	1
224	213	204	417	536.4	537.2	418	1
256	251	230	481	531.2	532.2	482	1

Table 1: Measured (on client side) and calculated data of the whole system.

1.3. The Model

and a figure 1. fancy foobar

[1] Described in the appendix **??**

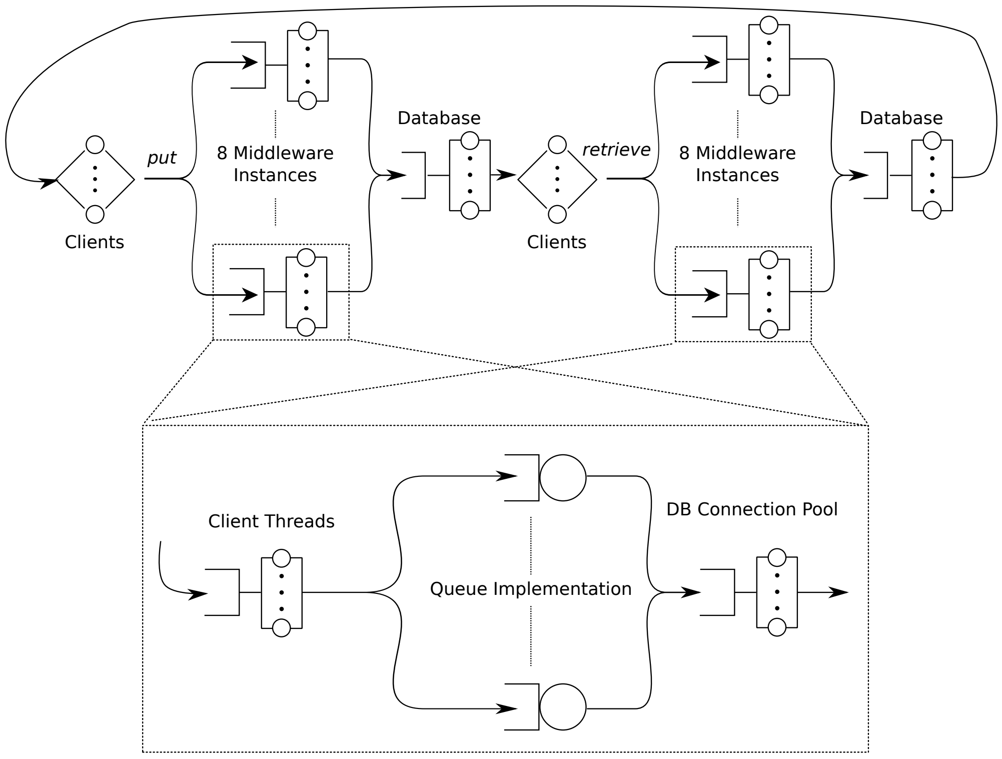

Figure 1: Full model of the system.

citation[1].

some math

$$\mu(n) = \begin{cases} n/S & \text{if } n = 1, 2, ..., m-1 \\ m/S & \text{if } n = m, m+1, ..., \infty \end{cases}$$

2. The File Format

This section describes the binary file format used by the file system inside a virtual disk. The file is separated into three major parts. The header, index and the data section. Each of them is described below.

2.1. Header Section

Name	**Lenght**	**Description**
Info	50 byte UTF-8 String	Contains something like Badger VFS 2013 V1.0
Version	10 byte UTF-8 String	Contains something like "1.0"
Compression used	20 byte UTF-8 String	null or indicates compression used for this file
Encryption used	20 byte UTF-8 String	null or indicates encryption used for this file
IndexSectionOffset	long (8 byte)	File offset where our index section starts
DataSectionOffset	long (8 byte)	File offset where our data section starts
SaltString	8 bytes	Salt used to hash username and password randomly string generated while creating this file
Password	xxx bytes	CryptoHash (SHA-whatever) of Password+SaltString

2.2. Index Section

Data in the index section are organized in a B-Tree structure.

2.3. Data Section

The data section is split into blocks where each of them is X bytes long. Each block contains some amount of data and points to a subsequent block

Block layout

Name	**Length**	**Description**
BlockHeader 0) Header-Bit (LSB) 1) Directory-Bit 2) not used 3) not used 4) not used 5) not used 6) not used 7) not used	1 byte	 If set to 1 this is the first datablock of a file. If set to 1 this is a directory, not a file
NextDataBlock	8 byte long	Points to the start address of the next Datablock (linked list). 0 if this is the last Data block of a certain file or folder.
HeaderLengthIndicator	4 byte	indicates the lenght of the DataBlock Header in bytes This field only exists if Block-Header Bit is set to 1
Header	n byte	Header Informationen creation date, modification date, file name. (May be encrypted/compressed) This field only exists if Block-Header Bit is set to 1
DataLenghtIndicator	4 byte	Indicates the number of data saved on this DataBlock
Data	n byte	user data (may be encrypted/-compressed)

A. Glossary

VFS core The main Java library, that handles all the interaction with virtual disks and importing/exporting/storing files. It is used by the command line client and the gui.

Virtual Disk A virtual disk denotes a container file that is stored on the host file system. A virtual disk can be opened with the software that is developed during this project and stores the actual files. The file extension of the virtual disk is "*.bfs".

B. Command line client

The command line client allows the usage of the VFS core and is mainly intended to test the basic functionalities. The console runs either in management mode or in filesystem mode. The management mode is entered automatically when starting the command line client. It allows creating and disposing virtual disks. The filesystem mode is entered as soon as a virtual disk is opened.

TODO: DISCUSSION: sollen ganze ordner importiert und exportiert werden können? wird dies von der client-seite gehandelt?

B.1. startup

The command line client can be started as follows:

```
java -jar VFSCore.jar ch.eth.jcd.badgers.vfs.ui.VFSConsole
```

B.2. commands

Following commands can be used with the command line client in management mode:

- **create c:\path\to\disk.bfs 1024** creates virtual disk with a maximum quota of 1024 megabytes on the host system. The file may grow up to 1024 megabytes. **TODO**: more parameters are needed (encryption, compression, password if there is encryption)

- **dispose c:\path\to\disk.bfs** deletes the given virtual disk
- **open c:\path\to\disk.bfs** opens filesystem mode for the given virtual disk
- **exit** exits the console program

follwing commands can be used in filesystem mode:

- **ls** lists the contents of the current directory
- **cd dst** changes current directory to *dst* which must be either a child directory of the current path or ".."
- **mkdir dirName** creates a new directory *dirName* in the current path
- **mkfile fileName** creates a new empty file *fileName* in the current path - this is rather not usefull, as the "import" creates a file with content
- **rm file** deletes the entry denoted as *file*, it must be a child of the current path
- **cp src dst** copies the *src* file to *dst* as a child of the current path
- **mv src dst** moves the *src* file to *dst*
- **import ext_dst** imports a *ext_src* from the host system to *dst*
- **export src ext_src** exports a *src* file to the host system *ext_dst*
- **find searchString** lists all filesystem entries below the current entry containing *searchString*
- **close** closes the filesystem mode, from now on management mode commands can be executed

List of Figures

List of Tables

References

[1] Raj Jain. *The Art of Computer Systems Performance Analysis.* John Wiley and Sons, Inc., 1991.