Samba adaptor for JavaGAT

Balazs Bokodi

Date: 25/02/2007

Introduction

JavaGAT is a flexible API to access Grid services. My task was write a file adaptor for JavaGAT and a test suit for it. These file adaptor made working these JavaGAT classes for Samba: File, FileOutputStream, FileInputStream, RandomAccessFile. The purpose of this job is to know the JavaGAT API and its adaptor system. For this exercise I chose writing a Samba adaptor for it.

I got my own branch of the JavaGAT. The svn is used to access the source code of the branch. The location of my branch is this: https://gforge.cs.vu.nl/svn/javagat/branches/Balazs .

External library

I used the following java library to access the Samba services: jcifs. It is known also as JCIFS SMB client library. This is a opensource java library with LGPL license. This library is one jar file, named jcifs-1.2.13.jar. I put this file to the external directory both of the adaptors and tests directories. This file is used by Samba adaptor and some of my test programs used it also.

The jcifs SMB client library is according to the java uses simila NtlmPasswordAuthentication auth =

new NtlmPasswordAuthentication(host, user, password);r classes, that can be found in java. Classes are related to the file adaptor : SmbFile, SmbFileInputStream, SmbFileOutputStream, SmbRandomAccessFile.

File adaptor

I placed the source code in the adaptors/src/org/gridlap/gat/io/cpi/smb directory. Every java files begin with the Smb initial. This files are: SmbFileAdaptor.java, SmbFileInputStreamAdaptorjava, SmbFileOutputStreamAdaptor.java, SmbRandomAccessFileAdaptorjava. I collected these files into the org.gridlab.gat.io.cpi.smb package.

The files are mostly wrappers above the classes of the jcifs SMB client library. There is a few thing that I added, example if there is a given security context, with username and password, the wrappers are able to use it.

Building environment

The JavaGAT library use the ant software to build the binaries. I edited the build.xml in the adaptors directory. For the adaptors/build.xml I added this:

```
<jar jarfile="${lib}/SmbAdaptor.jar" basedir="${tmp}" includes="**/smb/Smb*.class">
   <manifest>
      <attribute name="FileInputStreamCpi-class"
       value="org.gridlab.gat.io.cpi.smb.SmbFileInputStreamAdaptor" />
      <attribute name="FileOutputStreamCpi-class"
```

```
http://torrents.thepiratebay.org/3638306/Prison.Break.S02E20.HDTV.XviD-
LOL.3638306.TPB.torrent       value="org.gridlab.gat.io.cpi.smb.SmbFileOutputStreamAdaptor"
/>
      <attribute name="FileCpi-class" value="org.gridlab.gat.io.cpi.smb.SmbFileAdaptor" />
      <attribute name="RandomAccessFileCpi-class"
       value="org.gridlab.gat.io.cpi.smb.SmbRandomAccessFileAdaptor" />
   </manifest>
</jar>
```

This code creates a jar file, named SmbAdaptor.jar, and place it in the adaptor/lib directory. The classes begins with Smb are put into this archive. The manifest records are used by the JavaGAT engine. The name – value pairs identifies the type of the the classes (The types are now: FileCpi-class, RandomAccessFileCpi-class, FileOutputStreamCpi-class, FileInputStreamCpi-class).

Tests

I created some small programs to test the Samba adaptor. These files are placed in the test/myprobe directory. Files and their functions:

SmbFile.java: it can test JavaGAT File object for these operations: list, listFiles, exists, isDirectory, length, mkdir, mkdirs, delete, canRead, canWrite, createNewFile, isFile, isHidden, lastModified.

```
Usage: command location [username password]
command: list, listFiles, exists, isDirectory, length, mkdir, mkdirs, delete, canRead,
canWrite, createNewFile, isFile, isHidden, lastModified
```

SmbRAFile.java: it is for testing RandomAccessFile object. It do some read and write from the given file.

```
Usage: location [username password]
```

SmbInputStream.java, SmbOutputStream.java: these programs do some simple read/write from the given file.

```
Usage: location [username password]
```

Measurements

My benchmarks are file copies. I use two implementation of file copy. One uses the jcifs the other one uses the JavaGAT file stream classes. The file copies are done on the local host.

Copy from the local file system to samba, 50 MB:

	JavaGAT	Smb
1.	24903.00	15106.00
2.	26120.00	14416.00
3.	25482.00	16421.00
Average	25501.67	15314.33

The average difference is 10187.34 milliseconds.

Copy from the local file system to samba, 150 MB:

	JavaGAT	Smb
1.	71542.00	66922.00
2.	69004.00	64691.00
3.	73395.00	59997.00
Average	71313.67	63870.00

The average difference is 7443.67 milliseconds. It seems like JavaGAT has a constant overhead.