



KnowARC

KnowARC_D4.1-1

8/2/2008

GRID-BASED CONTENT CREATION AND MULTIMEDIA RETRIEVAL

– Integrated documentation and manual

Xin Zhou, Adrien Depeursinge, Henning Müller*

Contents

1	Introduction	5
1.1	README	5
1.1.1	The arcGIFT prototype	5
1.1.2	The lung image analysis prototype (Talisman)	5
1.2	Technical description	6
1.2.1	The arcGIFT prototype	6
1.2.2	The arcTextureAnalysis prototype (Talisman)	7
2	The manual of arcGIFT prototype	9
2.1	Installation	9
2.1.1	GridJM	9
2.1.2	Pre-requisite packages	9
2.1.3	GIFT 0.1.15b	9
2.1.4	ArcGIFT	10
2.2	Tutorial	10
2.2.1	Indexing an image database	10
2.2.2	Run GIFT server	11
2.2.3	Perform image retrieval	11
2.2.4	Debug and analysis scripts	12
3	The manual of arcTextureAnalysis prototype	13
3.1	Installation	13
3.2	Tutorial	13

Chapter 1

Introduction

1.1 README

This deliverable is part of the KnowARC project, work package 4, task 4.1. The goal is to provide a grid-based medical image retrieval prototype. Two retrieval applications were griddified and prototypes are available.

In this directory, two sub-directories containing the mentioned prototypes can be found that contain the following content:

- arcGIFT : prototype of a griddified GIFT;
- arcTextureAnalysis : prototype for lung image retrieval (Talisman);

All prototypes are based on the Grid Job Manager (GridJM). The scripts in the the arcGridJMtester directory in the arcGIFT folder test this environment. These scripts serve of trouble shooting for both prototypes.

1.1.1 The arcGIFT prototype

ArcGIFT is a griddified version of the GNU Image Finding Tool (GIFT). GIFT is an open source content-based images retrieval (CBIR) system. The changes of GIFT in this deliverable concern the access to grid resources by the formerly sequential feature extraction parts. GIFT is an application currently used mainly in a research context. Thus testing of parameters by repetitively indexing larger databases happens frequently. Our solution is based on the ARC (Advance Resource Connector) middleware. ARC is a popular, light-weight grid middleware.

1.1.2 The lung image analysis prototype (Talisman)

Our lung image analysis project aims at the extraction of several features from lung computed tomography (CT) images to aid diagnosis. Features are gray level histograms and texture features based on wavelet frames. Input images have to be in the medical DICOM format, and if extraction is to be limited to a region of interest this has to be supplied in an internal XML format. Example images and annotated regions are supplied for testing.

ArcTextureAnalysis is implemented in Java.

1.2 Technical description

This chapter explains technical description for the griddified image retrieval prototypes. We based the text on the text [1] to create this section.

In general, image retrieval applications consist of two large parts, an offline feature extraction and indexation and an online query process, where the user interacts with the system.

The online part needs to be extremely quick to allow for an interactive navigation in image databases (response time generally shorter than 1 second).

The most time consuming part is currently the offline part, so the extraction of visual features describing the indexed images. As the features of every image can be extracted independently and thus completely parallelizing this part is easy and is our first target for using a grid infrastructure.

There are basically two ways for adapting our application to the grid:

1. Change the algorithm in order to be able to submit independent jobs to the Grid.
2. Write an independent part to create and submit jobs to the Grid.

As there are two applications to be griddified (the GIFT and the lung texture analysis system of the Talisman project), to have an independent component for both applications is a better choice.

First tests showed that several jobs delivered no results and had to be resubmitted manually. Based on this we decided to use a job manager (GridJM) to manage the jobs of the two applications. Several parameters such as the number of resubmissions can be set. More information about GridJM can be found on the GridJM homepage*. A script to analyze the GridJM performance can be found in the ARC_GridJM_tester directory in the arcGIFT folder.

Note about GridJM: GridJM serves both arcGIFT and arcTextureAnalysis. This application addresses the following points [2]:

- set up a fault tolerance;
- avoid submission failures by using history information to mark the "bad" clusters;
- optimize the time token by job submission;
- visualize the resource usage graphically;
- automatically collecting of results.

This hides the complexity of the ARC usage and supervises the job status. For further simplification, all the parameters required by GridJM are specified in arcGIFT.conf. The job description files are produced automatically by the application. Thus, the ARC usage is completely hidden and the user can concentrate on the image processing part.

1.2.1 The arcGIFT prototype

The arcGIFT scripts perform the parallelization in the following way:

- help users to specify the important parameters;
- deal with the image transformation into PPM format;
- divide the images into packages and produce the xrsl [3] file;
- run GridJM and send xrsl file to GridJM to submit the jobs;
- wait for GridJM messages to receive the results;
- create the inverted file from the feature files;
- provide a PHP web interface for navigation in the image database.

These steps are implemented in the following way:

*<http://www.tcs.hut.fi/~aehyvari/gridjm/>

1. *The configuration file:* A configuration file is created with precise indications in the .arc directory to save the reference parameters for GIFT, GridJM and ARC. The arcGIFT scripts start by reading this file.
2. *The pre-treatment:* ImageMagick is used to convert images into the required PPM format. This part is currently not griddified.
3. *The packing and xrsl file creation:* For each job two packages are needed to be sent to cluster: the execution program and the archive of images. Size is supervised when images are packed. The job description file is automatically generated.
4. *The job submission:* arcGIFT uses GridJM for job submission. The processing can be visualized through GridJM modules.
5. *Collecting the results:* Files are downloaded by GridJM in random sub-folders in the indicated place. Completeness checks can only be performed by checking the correct number of results. These results are then put together to simplify the inverted file generation.
6. *Generate the inverted file:* This is performed using an existing executable.
7. *Web interface:* A web interface is provided. Configuration is manual.

Next steps:

- Use of the RTE to install the needed software on the clusters.
- Use a grid-based storage element to avoid large data resubmission.

1.2.2 The arcTextureAnalysis prototype (Talisman)

The arcTextureAnalysis is fully implemented in Java. The class parallelize.java is responsible for the parallelization of the feature extraction. By parallelizing we mean extracting features from several image blocks on separated computers. Indeed, as computationally intensive feature extraction algorithms are recursive (wavelets) [4, 5], parallelizing them for a single image is not trivial.

The main class parallelize.java implements the following steps:

1. All images specified in "paths_images.txt" are separated into N blocks, with N being defined by the user.
2. For each block, a .tar package is created that contains:
 - the images series in DICOM format to be analyzed and a file containing their corresponding paths,
 - text files (.txt) describing the regions of interest (ROI) from which features will be extracted,
 - a batch file (.bat), which contains the instructions to locally run the feature extraction in Java,
 - an xrsl (.xrsl) file, which describes the job to be submitted on ARC resources,
 - the java libraries (.jar) required for local feature extraction.
3. Once packages are created, each job is submitted to GridJM through a socket implemented in Perl (xrslsender.pl).
4. Finally, a "while" loop verifies that all job results are contained in the GridJM download directory.

Next steps:

- send only the images with associated ROI instead of the entire image series to reduce the data transfer.

Chapter 2

The manual of arcGIFT prototype

2.1 Installation

The installation is based on: GridJM, pre-requisite packages, and the GIFT system.

2.1.1 GridJM

GridJM will be integrated into ARC in the future but this is not yet realized. The latest version can be found in the `gridjm-0.6` sub-directory. Installation instructions can be found at: <http://www.tcs.hut.fi/~aehyvari/gridjm/>

2.1.2 Pre-requisite packages

The pre-requisite packages include several Perl modules, and the ImageMagick package. For Ubuntu, Debian and SUSE users, installing the Perl modules and ImageMagick is directly performed with the `apt-get` system by executing the following command:

```
apt-get install imagemagick libhtml-parser-perl libparse-yapp-perl libtext-iconv-perl libwww-perl
libxml-dom-perl libxml-handler-trees-perl libxml-libxml-perl libxml-parser-perl libxml-sax-expat-
perl libxml-sax-perl libxml-writer-perl libxml-xql-perl libc6 libexpat1 libgcc1 libmrm1c2a lib-
stdc++6 krmrml
```

For other Linux systems, the pre-requisite packages are provided in the `gift-prerequisites-0.11` subdirectory. Use `install-as-root.sh` or `install-in-home.sh` to do the installation.

2.1.3 GIFT 0.1.15b

When all the pre-requisite packages are installed, the GIFT system installation can be started. Three components are required:

1. GIFT in version 0.1.15b;
2. the patches for gcc 4.1 or later;
3. the web interface.

GIFT version 0.1.15b is in the `gift-0.1.15b` subdirectory. If GCC version 4.1 or higher is used, the patches in `gift-patch` need to be applied.

Install the GIFT under the directory "gift-local" under your home directory (We will fix the problem concerning the fixed path in the future, to allow users installing the prototype anywhere.) with :

```
configure --prefix=$HOME/gift-local && make && make install
```

There are several interface for the GIFT. In the `web_interface` subdirectory, we provide a `web_interface` package developed by the University & Hospitals of Geneva. PHP and Apache are needed to use this interface. Other web clients as interface for GIFT can be found at: <http://www.gnu.org/software/gift/mrml-clients.html>

Copy all the files in `web_interface` to a directory, which is accessible by Apache (eg: `/var/www/`, depending on the configuration). The entry point for the interface is `index.php`. In `config.php` several parameters can be changed.

2.1.4 ArcGIFT

After correctly installing `gift`, run `install.pl` in the `arcGIFT_scr` directory to install the `arcGIFT` scripts. These scripts are dedicated to the integration of ARC and GIFT.

2.2 Tutorial

This section explains how to use `arcGIFT`.

2.2.1 Indexing an image database

First, an image database needs to be indexed. The indexing step is the main component that is griddified. Images are separated into blocks for this indexation to process the blocks in parallel instead of sequentially.

Before start the indexation, to assure a correct functionality of the ARC client and GridJM, the following points need to be checked:

- Make sure that you have the personal certificate in the `$(HOME)/.globus` directory. There should be two files: `usercert.pem` and `userkey.pem`. Both should be in the property 400 (only readable for the owner).
- Under `$(HOME)/.arc`, you need to configure your `client.conf`. See <http://www.nordugrid.org/documents/ng-client-install.html> for more details.
- Use `$/usr/sbin/ntpdate ntp.ubuntu.com` to check if the system time is correct. If a difference was detected, use `$/sbin/hwclock -w` to reset the system time.
- Use the scripts in the `ARC_GridJM_tester` directory for testing.

Start with the script `arcgift-set-config.pl`. This script guides you to put all the preference parameters for `arcGIFT` asking questions. The input will be saved in the document `arcGIFT.conf` in `$(HOME)/.arc/` directory.

Run the `ng-gift-add-collection.pl` to add an image collection. You need to specify the following options:

```

-image-directory    the directory which contains the image collection
-collection-name    gives a unique name to the collection
-url-prefix         provides the url for showing the images in the web interface, otherwise file:/
                   URLs are used

```

The parameters for the grid configuration will use those specified in the `arcGIFT.conf` by default. If you want to use other parameters, you can add the following options:

<code>-gift-home</code>	changes the directory for the gift configuration file
<code>-feature-prefix</code>	details a directory to store the index files
<code>-thumbnail-dir</code>	gives a directory to stock the thumbnail images, which are used by the web interface
<code>-thumbnail-url-prefix</code>	gives a url to access to the thumbnail images
<code>-port</code>	to use another port to run the grid job manager
<code>-gridjm-dir</code>	changes the GridJM directory. It is used only when you want to try another version of GridJM. Otherwise, do not use it as the one specified in <code>arcGIFT.conf</code> was checked by <code>arcgift-set-config.pl</code>
<code>-workarea</code>	changes the directory where the temporal image packages and files are stored
<code>-image-in-batch</code>	changes the number of images per package. If you find that packages are often lost in you should change reduce the size of the package.
<code>-retry</code>	number of retries when the grid execution part fails. A larger number of retries can be used when packages are frequently lost.

Some options allow you to activate different modes:

<code>-debug</code>	Active debug mode. More information will be printed, and all the messages are printed in <code>gift_debug.output</code> .
<code>-timer</code>	Active timer mode. This measures the time spent for each step of the indexation (principally three: pre-treatment, feature extraction, index generation). The output is written into the file <code>gift_timer.log</code>

Here is a use example:

```
$ng-gift-add-collection.pl -image-directory $HOME/public_html/Images -collection-name c-images
  -thumbnail-dir $HOME/public_html/Images_thumbnails -url-prefix http://localhost/~xmzh/Images
```

For an execution time evaluation of the feature extraction use `"time-analyser.pl"` under `ARC_GridJM_tester` sub-directory. This script gives you the average time taken per job, the longest time per job, and the total time for the execution. The information is extracted from the log file `"gridjm.ins"`, which can be found in the directory that you specified for GridJM. A real time visualization of the execution of every package is also provided by GridJM. See Section 2.2.4 for more details.

More detail about the GridJM usage is available with option `"-h"`.

2.2.2 Run GIFT server

When the indexation is finished, you can run the server with:

```
gift -port XXX -datadir YYY
```

Pay attention, here the port needs to be different from the port you used above, which is the one used by the grid job manager! The `-datadir` is the configuration directory! These parameters should be already specified with `arcgift-set-config.pl` in `arcGIFT.conf`. If you want to use the presets, use simply the script `"gift-server-run.pl"`.

2.2.3 Perform image retrieval

There are two ways to perform image retrieval. One is by using a command line script. The other through using the web interface. The web interface is more intuitive to use. You need to set up apache and install the web interface. The installation is explained in the Section 2.1.

Then, you can simply open the `index.php` in a web browser. The port used by the gift server is required (The XXX part above). When the web interface is connected to the gift application, the navigation within the database is easy.

Performing image retrieval with a script allows you to have more flexibility for running a larger number of queries, for example for performance evaluation.

The script `runAllQueries.pl` was written for such tests in the context of the ImageCLEF benchmark. Three parameters are needed:

- `-imagelist` the file contains the query images. An example of `imagelist.txt` can be found in the example directory; you can add/change queries by separating them with an empty line
- `-collection-name` the collection you want to do the retrieval with
- `-resultdir` the results will be written in this directory

For questions concerning directly the gift, a mailing list exists: help-gift@gnu.org

2.2.4 Debug and analysis scripts

In the `ARC_GridJM_tester` sub-directory, you can find test scripts to help debugging in case of problems.

- `"configtest.pl"` is a script developed by Lubeck that can check the system time delay and other arc configuration details. This script can also be found at SVN T2.5 directory.
- `"socketSrv.pl"` is used to simulate the GridJM server on a port given (by default 12345). When there is a problem with GridJM, run it instead of the real GridJM server. The script will show all the content you sent to GridJM on that port. If your want to simulate another port, just add it as parameter.

e.g. `./socketSrv.pl 12344`

- `"xsrlsender.pl"` allows you to send a simple xsrl file to GridJM to perform a test. It requires the file name of the xsrl file

e.g. `./xsrlsender.pl -file test.xsrl`

- `"time-analyser.pl"` allows you to analyze the time taken per job in GridJM. It requires the directory containing the log file. By default it looks for a file called `"gridjm.ins"`. If the log file name was modified the file argument is required.

e.g. `./time-analyser.pl -ngLogpthr /home/xmzh/arcGIFT/log -file gridjm.ins`

- An executable `"foo3"` is provided under the directory of the GridJM/visualization. It gives a real time visualization of job execution on grid. In graphic interface, every job is represented by a timeline, and the status is represented by the colors. Blue shows a job succeeded. Yellow means a job succeeded after at least one resubmission. Green means a job status is unknown.

For any question or comments, please contact:

Xin ZHOU email: xin.zhou@sim.hcuge.ch

Chapter 3

The manual of arcTextureAnalysis prototype

3.1 Installation

This section explains how to install the arcTextureAnalysis application:

The system works based on the GridJM package, the ARC middleware, several Perl modules, and the Java Runtime Environment (JRE) newer than version 1.4. For Ubuntu, Debian and SUSE users, installing Perl modules can be done with apt-get.

When all the pre-requisite packages are installed, download and unpack the software. The arcTextureAnalysis application is under arcTextureAnalysis directory.

3.2 Tutorial

In this file we explain how to use the lung texture analysis application.

The main idea for parallelization of this applications is currently to separate the images into several packages and perform feature extraction on several grid nodes at the same time.

A personal certificate needs to be installed in the \$HOME/.globus/ directory. There need to be two files: usercert.pem and userkey.pem. Both should have property 400 (only readable for the owner).

In the ./images directory, several example images and marked regions of interest are supplied to properly test the application.

GridJM needs to be started first.

The Java command to start the application is as follows:

```
$ java -cp textureAnalysis.jar:ij.jar:imageware.jar ch.unige.parallelize arg0 arg1 arg2
```

with the arguments:

- arg0 is the number of packages to use. If your total number of images is 100, 4 will result in 4 packages containing 25 images each
- arg1 is a .txt file containing the path of the folders with your images
- arg2 is the location of the download directory of GridJM (default is /tmp/ngdownload/). It will contain the output files with the features: "feature-group0.txt", "feature-group1.txt", ...

The output after the execution is as follows:

- The files containing the features are located in the the download directory of GridJM. The names are "feature-group0.txt", "feature-group1.txt".

- The files containing the execution time are in `./measures/`

A use example:

```
$ java -cp textureAnalysis.jar:ij.jar:imageaware.jar ch.unige.parallelize 2 paths_images.txt  
/tmp/ngdownload/
```

For an execution time evaluation or real time visualization, see Section 2.2.4

For any question or comments, please contact:

Adrien Depeursinge, email: adrien.depeursinge@sim.hcuge.ch

Bibliography

- [1] S. Moeller. Implementation plan of the taverna-arc-integration. [Online]. Available: http://svn.nordugrid.org/trac/workarea/browser/T2.6/technical_description.txt
- [2] A. E. J. Hyvärinen. Gridjm-a way for client job management in arc. [Online]. Available: [http://www.tcs.hut.fi/~sim\\$ahyvari/gridjm/presentations/jobmanager.pdf](http://www.tcs.hut.fi/~sim$ahyvari/gridjm/presentations/jobmanager.pdf)
- [3] O. Smirnova, *Extended Resource Specification Language*, The NorduGrid Collaboration, nORDUGRID-MANUAL-4.
- [4] A. Depeursinge, D. Sage, A. Hidki, A. Platon, P.-A. Poletti, M. Unser, and H. Müller, “Lung tissue classification using wavelet frames,” in *Proceedings of International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Lyon, France, August 2007.
- [5] M. Unser, “Texture classification and segmentation using wavelet frames,” *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1549–1560, 1995.